Entwicklung eines Eclipse-Plug-Ins zur Erstellung von HTML/Script-Dokumenten

Anhang zur Diplomarbeit
im Studiengang Medieninformatik
der Technischen Fachhochschule Berlin

Vorgelegt von
Karsten Thiemann
im WS 2003/2004

Betreuerin: Prof. Dr. Fanny-Michaela Reisin
Gutachterin: Prof. Dr. Heike Ripphausen-Lipa

Inhaltsverzeichnis

A Verwendete Werkzeuge

Name	*Version*	*Hersteller/Vertrieb*	*Web*
Eclipse	2.1.0	eclipse.org	http://www.eclipse.org
Eclipse UML	1.2.1	Omondo	http://www.eclipseuml.com
Java SDK	1.4.1_01	SUN©	http://java.sun.com
Visual Thought	1.4	Confluent Inc.	http://www.confluent.com
Windows XP	Home, SP1	Microsoft©	http://www.microsoft.com

B Test des Scripteditor-Plug-Ins

B.I Referenzdokument

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>    <!-- Kommentar -->
<head>
     <title>Testdokument</title>
     <style>
          td{border: 2px solid red;}
     </style>
     <script language="javascript">
                                        /* Mehrzeiliger
                                           Kommentar  */
          function hideInnerTable(){
               var table = document.getElementById('tabelle_innen');
               if(table.style.visibility == "hidden"){
                    table.style.visibility = "visible";
               } else {
                    table.style.visibility = "hidden";
               }
          }
                                        // einzeiliger Kommentar
          function checkForm(){
               if (!document.forms.form1.check1.checked ){
                    alert("Checkbox 1 markieren!");
                    return false;
               }
               return true;
          }
     </script>
</head>
<body>
     <ul>
          <li>Punkt 1</li>
          <li>Punkt 2</li>
          <li>Punkt 3</li>
     </ul>
     <table id="tabelle_aussen">
          <tr>
               <td>TD AUSSEN 1</td>
               <td>TD AUSSEN 2</td>
          </tr>
          <tr>
               <td>TD AUSSEN 3</td>
               <td>
                    <table id="tabelle_innen">
                         <tr>
                              <td>TD INNEN 1</td>
                              <td>TD INNEN 2</td>
                         </tr>
                    </table>
               </td>
          </tr>
     </table>
     <br>
     <form name="form1" onsubmit="return checkForm()">
          <input name="check1" type="checkbox" value="check1">Checkbox 1<br>
          <input name="check2" type="checkbox" value="check2">Checkbox 2<br><br>
          <input type="submit">
     </form>
     <br>
     <a href="javascript:hideInnerTable()">Innere Tabelle ein/ausblenden</a>
</body>
</html>
```

B.II Testvorschrift

1 HTML

- Datei öffnen, Highlighting kontrollieren
- Outline View Vergleich mit Screenshot Nr.1
- Outline View filtern (Standardeinstellungen kontrollieren) -> Vergleich mit Screenshot Nr.2
- am Ende von Body
 einfügen -> Outline view darf sich nicht ändern
- Filter entfernen ->
 muss in Outline enthalten sein
- (wenn Preview vorhanden) Vergleich mit Screenshot Nr.3
- (wenn Preview vorhanden) Test des Links (Scripttest)
- Script auslagern durchführen (Kontextmenü), danach erneut Preview
- Externe CSS-Datei (vorgegeben) über Navigator verbinden, danach Preview
- Bild einfügen (über Kontextmenü und über Button), danach Preview

Codevervollständigung

- innerhalb des Body-Tags -> Attribute, 'o' eingeben und Filterung der Vorschläge testen
- vor dem Body-Tag -> HTML-Tags, '<B' eingeben und Filterung der Vorschläge testen
- am Ende des Dokumentes -> HTML-Tags, '<B' eingeben und Filterung der Vorschläge testen

Codebibliothek

- Bereich markieren und im Kontextmenü zur Bibliothek hinzufügen
- an anderer Stelle über Kontextmenü einfügen
- Schließen aller Editoren, Datei neu öfnnen (Bibliothek wird neu eingelesen)
- Prüfen ob neuer Eintrag vorhanden ist
- Menüeintrag über Kontextmenü entfernen und prüfen ob tatsächlich entfernt

HTML Tidy

- HTML Tidy durchführen (Kontextmenü)
- Formatierung des Dokumentes ändern und HTML Tidy erneut ausführen
- Präfernz groß/klein ändern und Format Tags ausführen

2 JavaScript

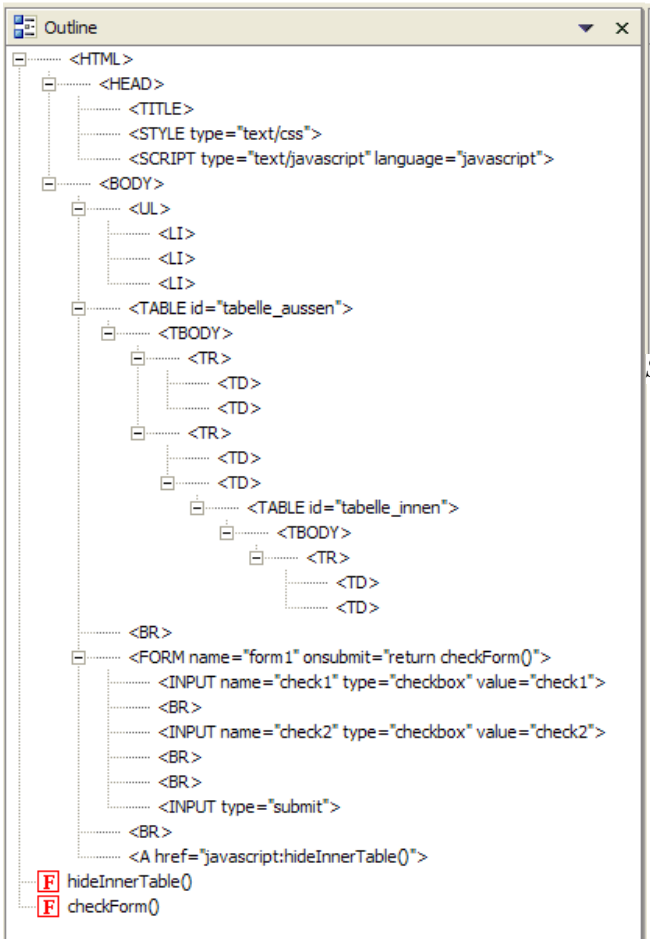
- Ausgelagerte Scriptdatei öffnen, Highlighting kontrollieren
- Outline View Vergleich mit Screenshot
- Außerhalb der Funktionen eine Variable einfügen
- Outline View kontrollieren (Variable muss erscheinen)
- Innerhalb einer Funktion eine Variable einfügen
- Outline View kontrollieren (Variable darf nicht erscheinen)

3 Hilfe

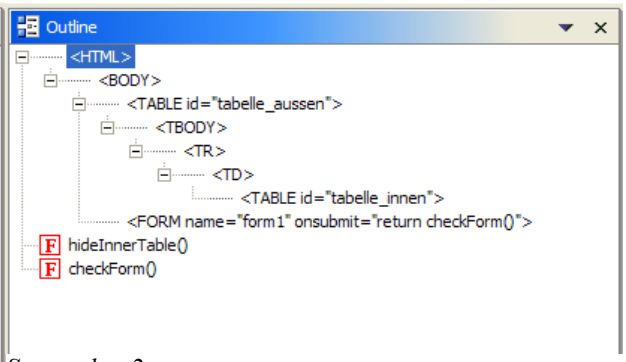
- Cursor in Editor und F1 drücken, Editorhilfe muss erscheinen

- Cursor in Outline View und F1 drücken, Editorhilfe muss erscheinen

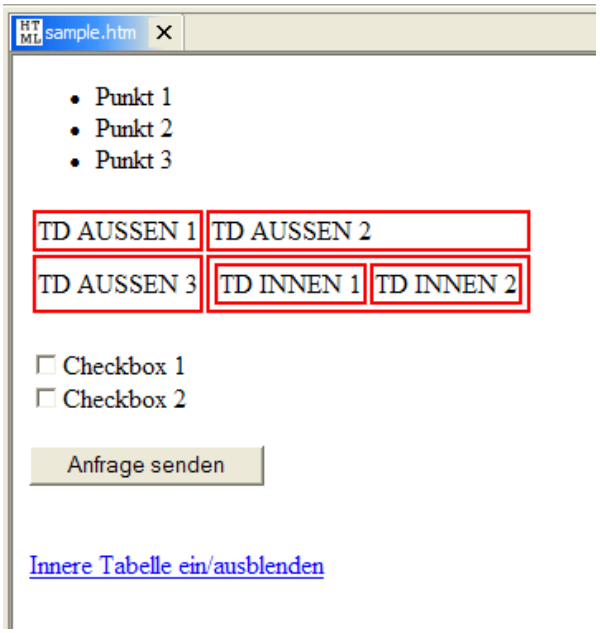
B. III Vergleichs-Screenshots



Screenshot 1



Screenshot 2



Screenshot 3

B. IV Testprotokoll

Plattform

Betriebssystem:

Prozessor:

Hauptspeicher:

Eclipse Version:

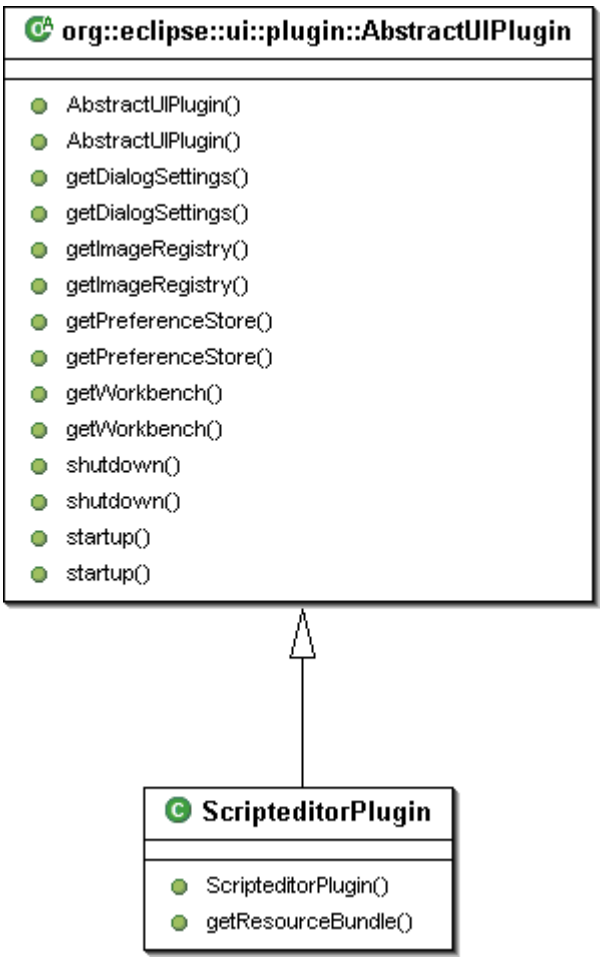
Scripteditor-Plug-In Version:

Vorschau-Plug-In Name/Version:

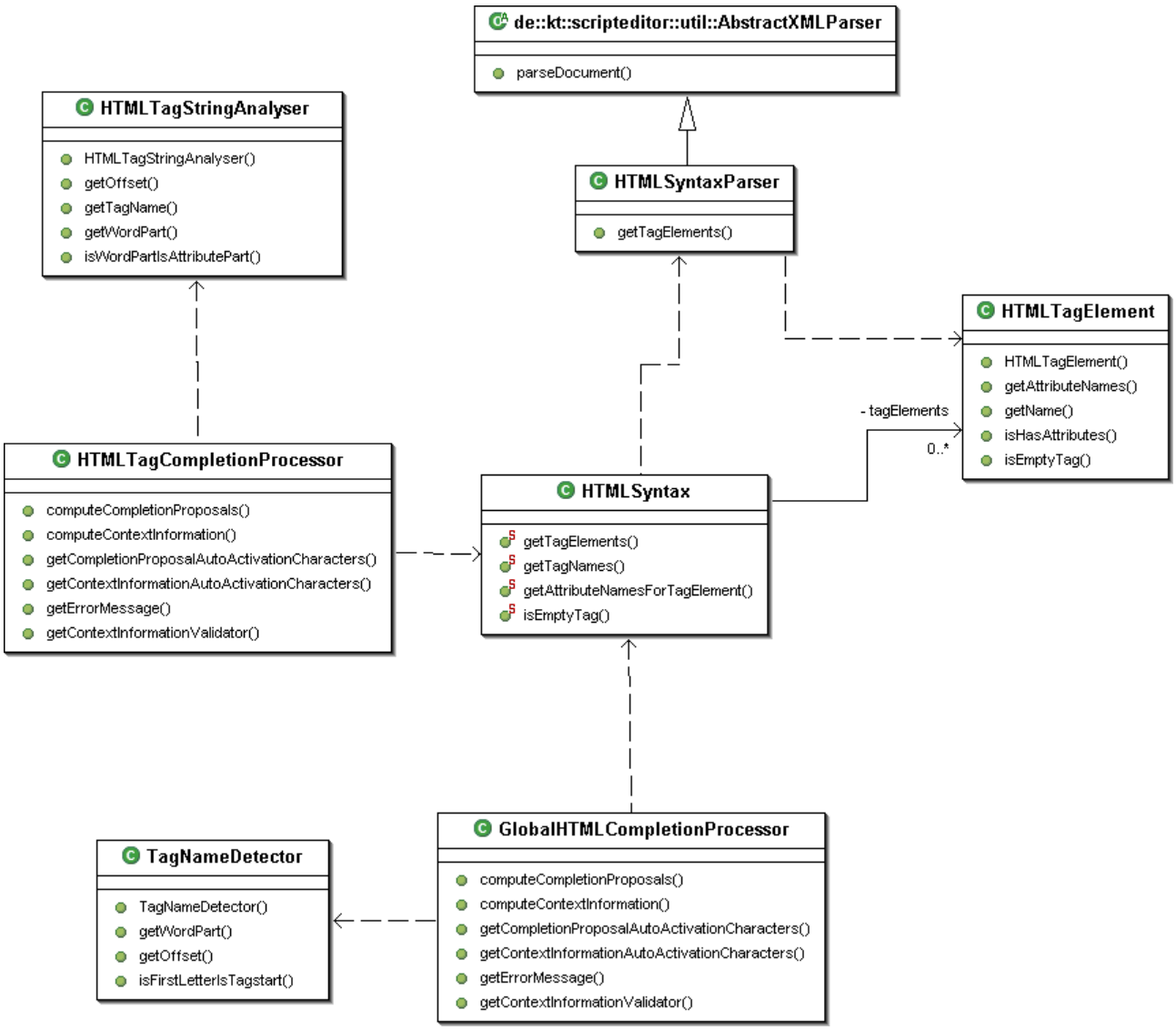
Fehlerbeschreibungen

C Klassendiagramme

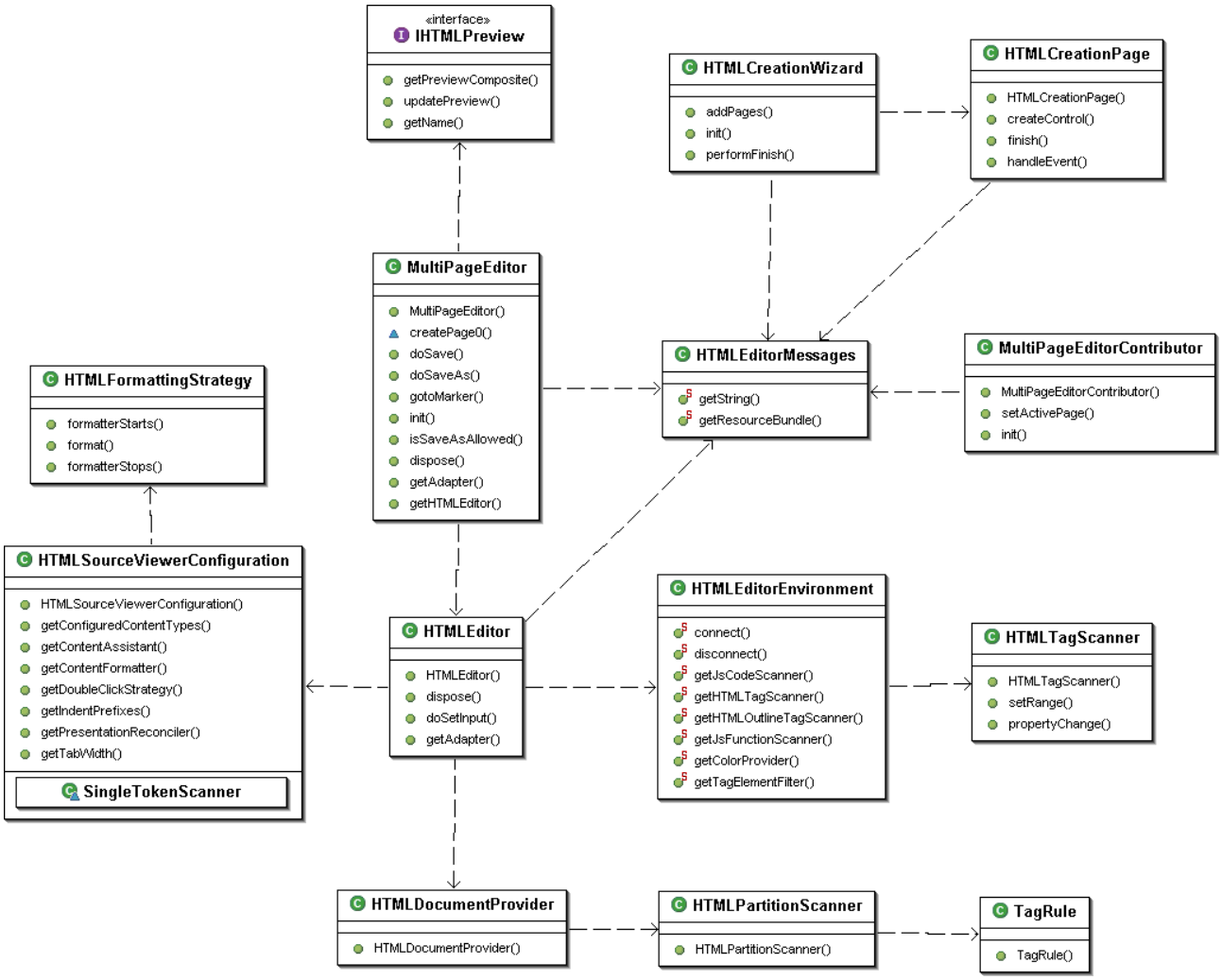
C.I de.kt.scripteditor



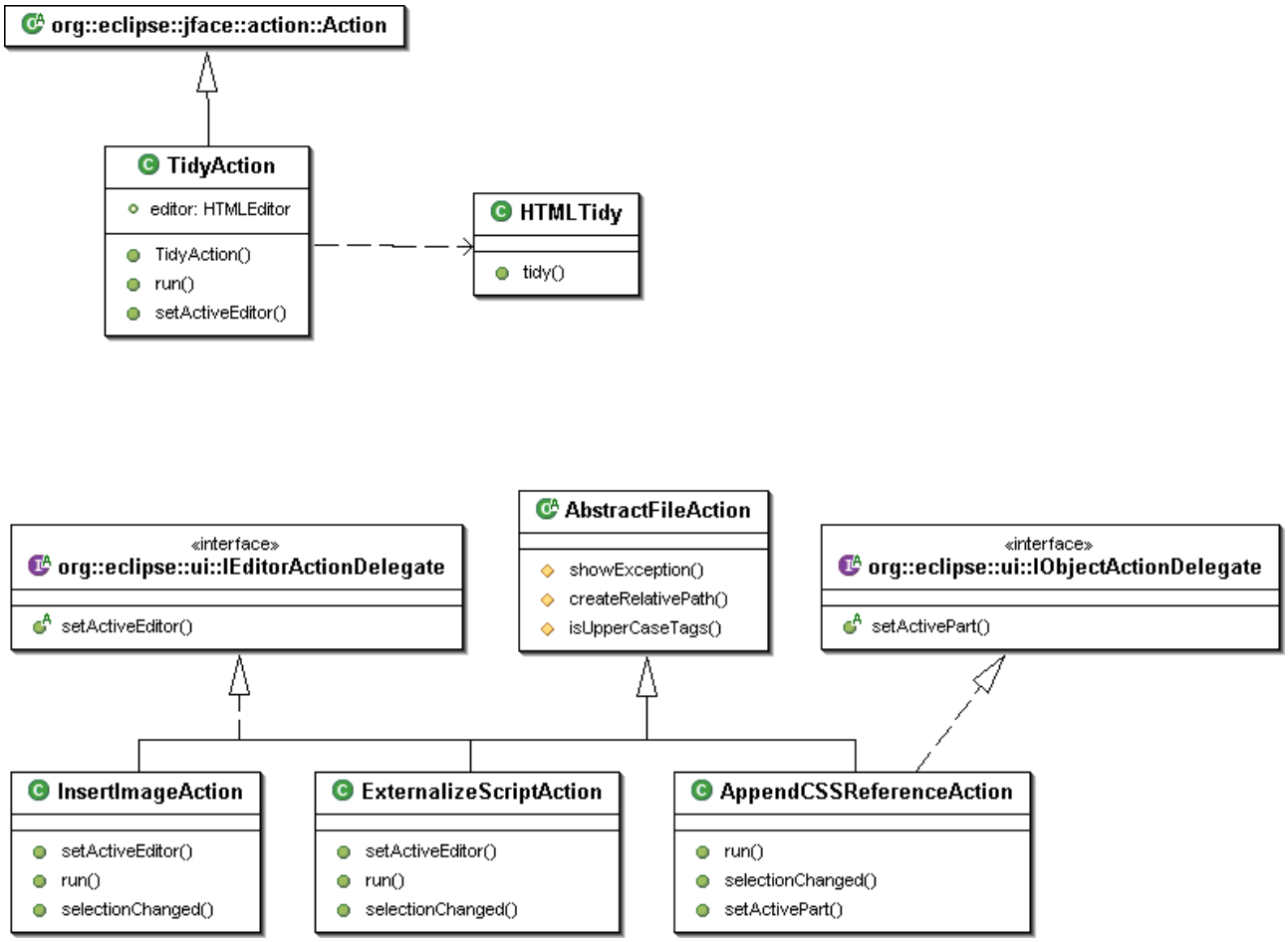
C.II de.kt.scripteditor.html



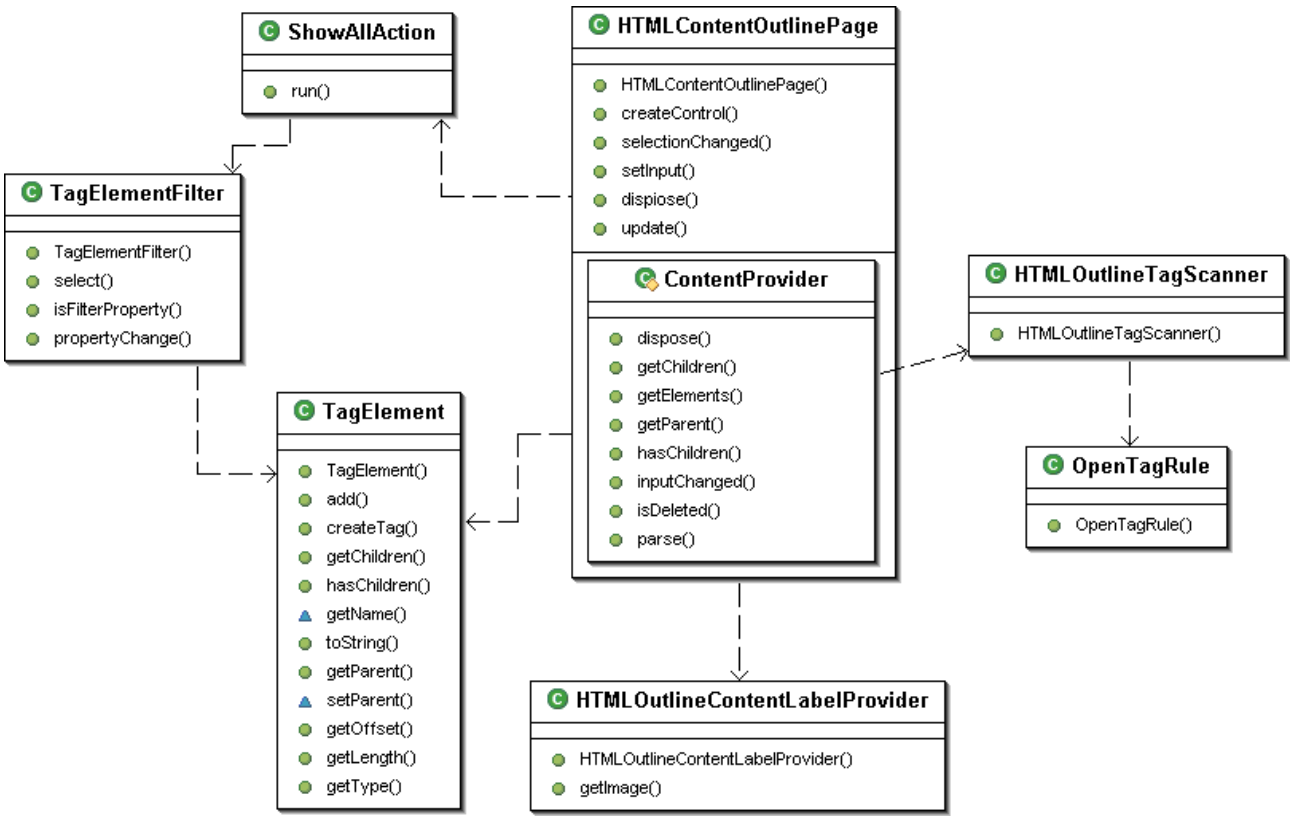
C.III de.kt.scripteditor.htmleditor



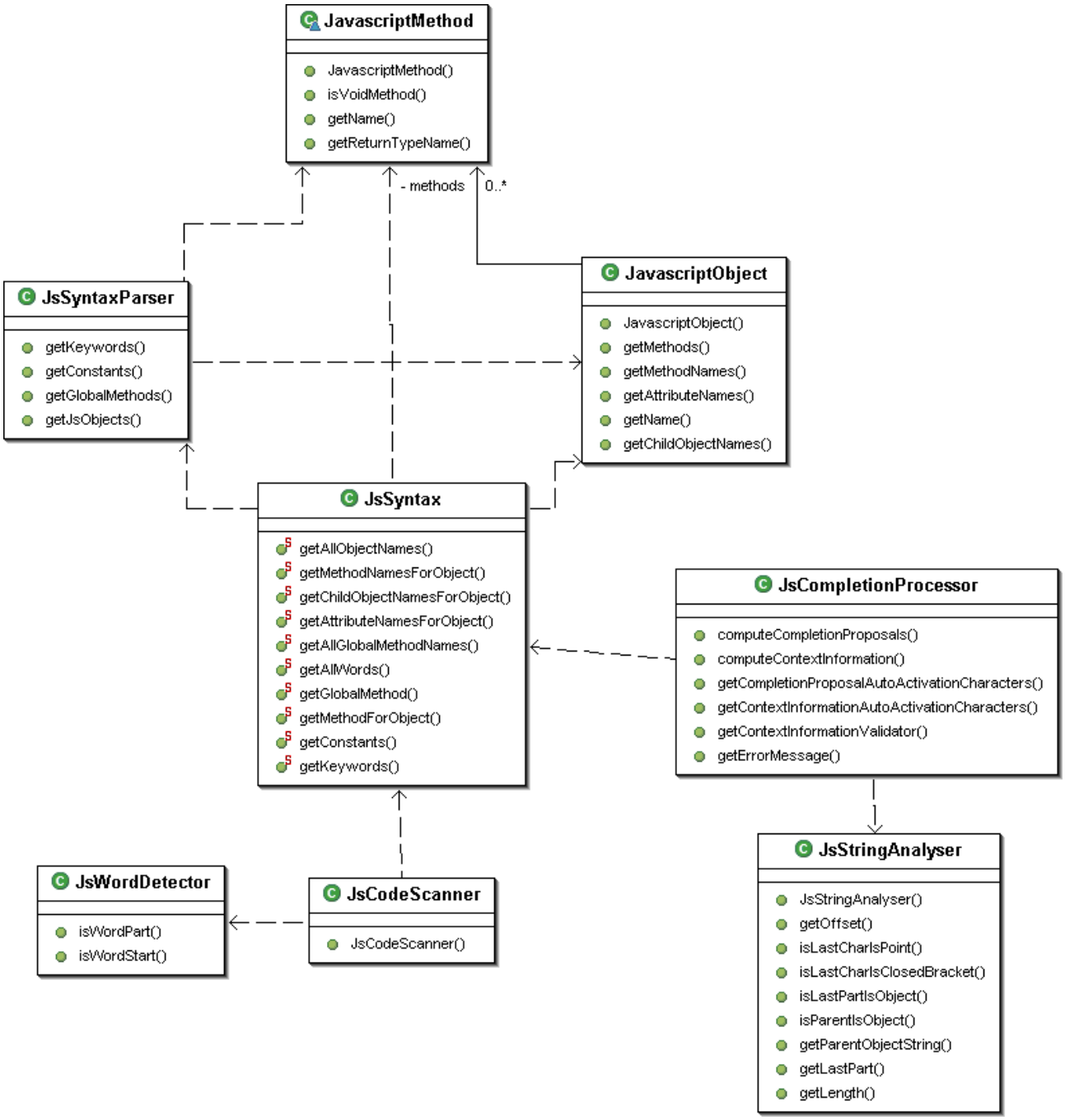
C.IV de.kt.scripteditor.htmleditor.actions



C.V de.kt.scripteditor.htmleditor.outline

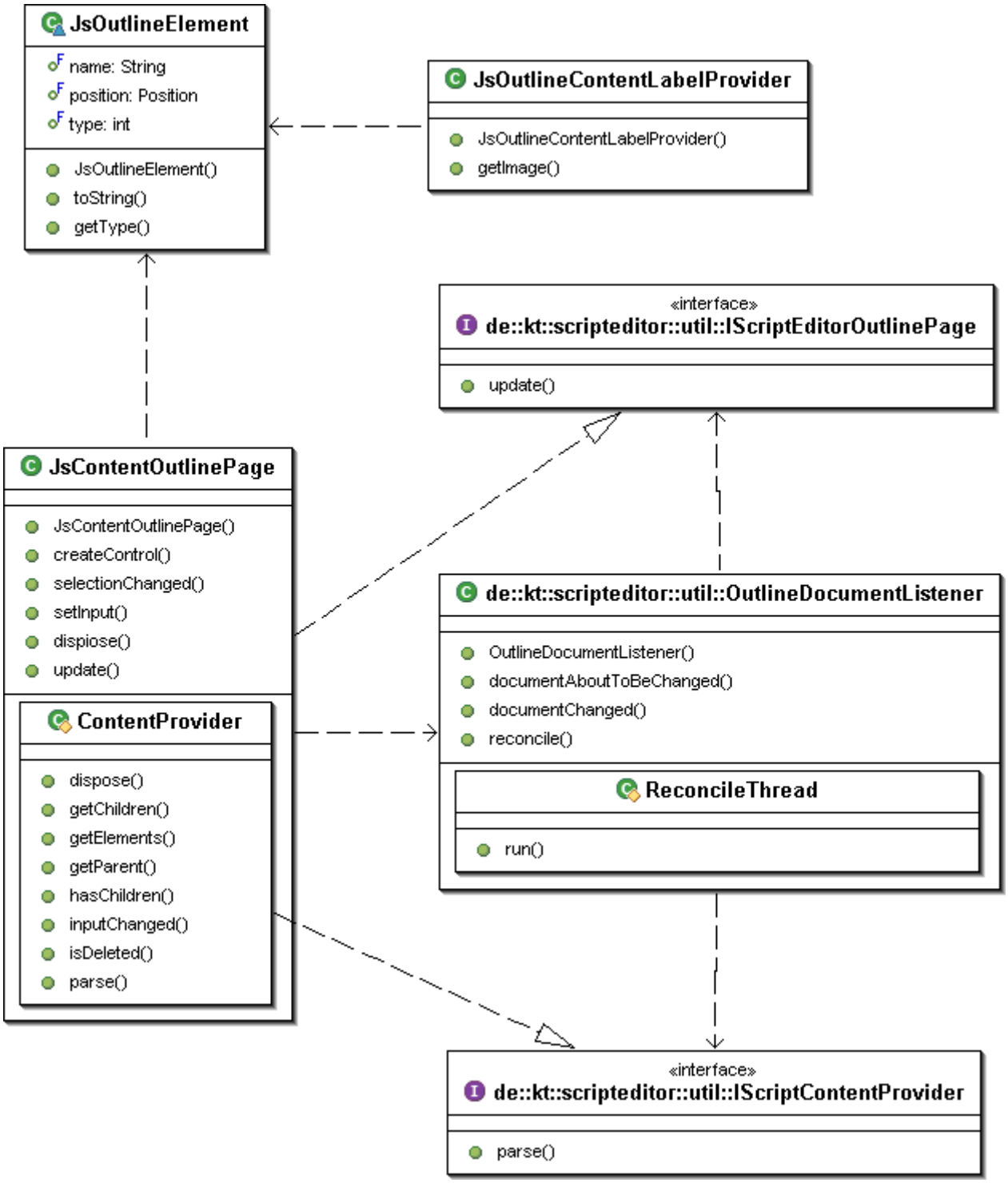


C.VI de.kt.scripteditor.javascript

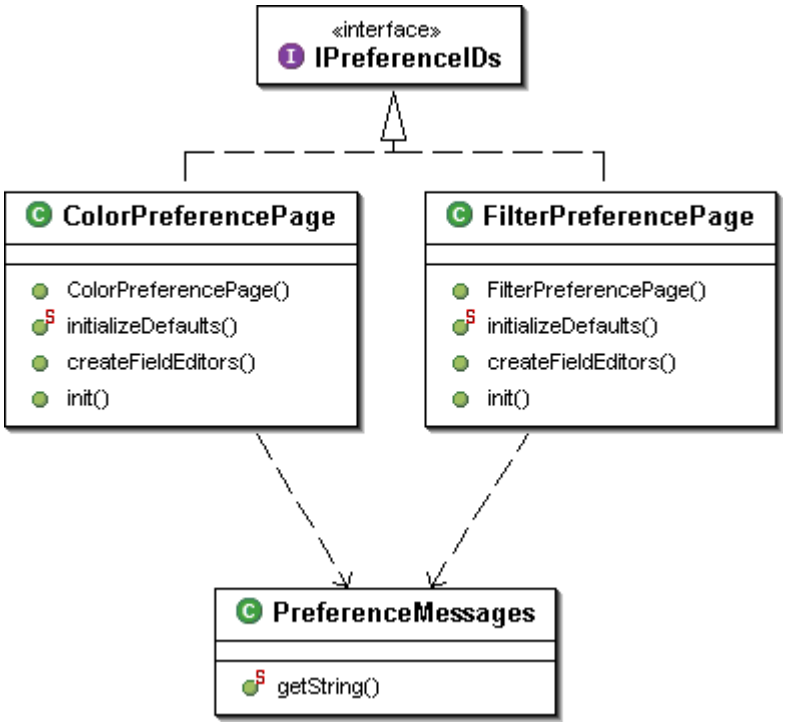


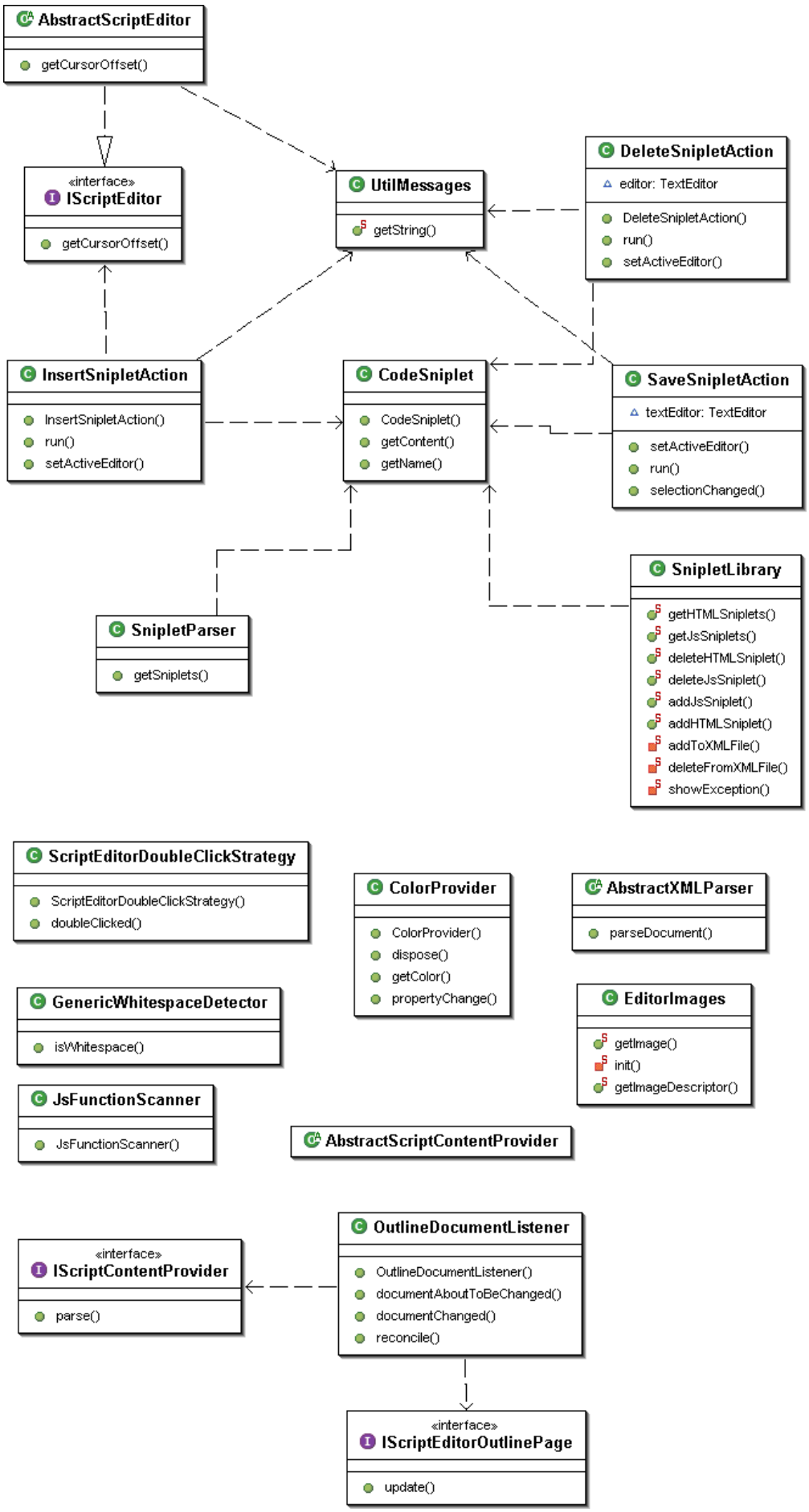
C.VII de.kt.scripteditor.javascripteditor

C.VIII de.kt.scripteditor.javascripteditor.outline

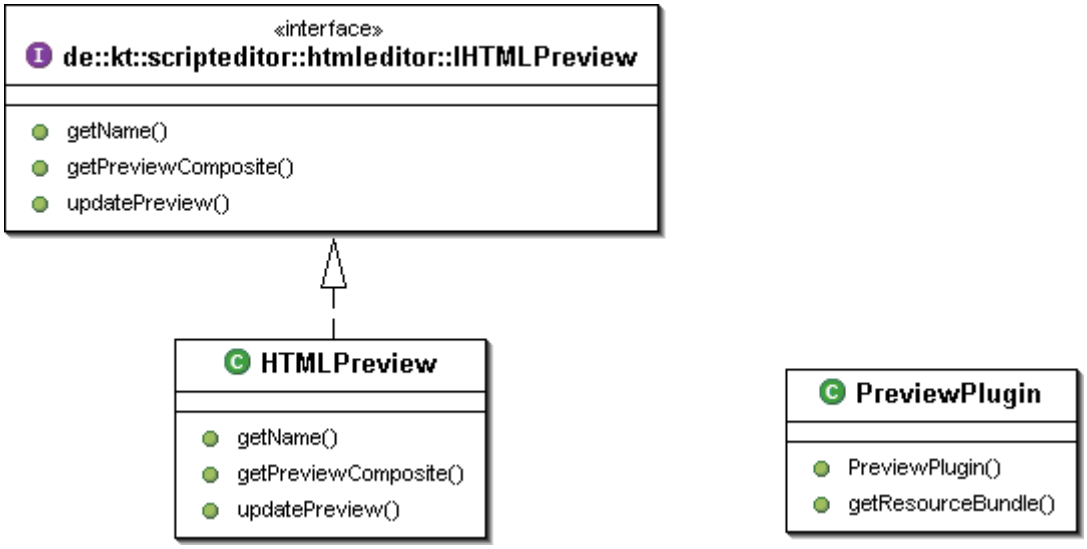


C.IX de.kt.scripteditor.preferences




AbstractScriptEditor
getCursorOffset()
«interface»
IScriptEditor
getCursorOffset()
UtilMessages
getString()
DeleteSnipletAction
editor: TextEditor
DeleteSnipletAction()
run()
setActiveEditor()
InsertSnipletAction
InsertSnipletAction()
run()
setActiveEditor()
CodeSniplet
CodeSniplet()
getContent()
getName()
SaveSnipletAction
textEditor: TextEditor
setActiveEditor()
run()
selectionChanged()
SnipletLibrary
getHTMLSniplets()
getJsSniplets()
deleteHTMLSniplet()
deleteJsSniplet()
addJsSniplet()
addHTMLSniplet()
addToXMLFile()
deleteFromXMLFile()
showException()
SnipletParser
getSniplets()
ScriptEditorDoubleClickStrategy
ScriptEditorDoubleClickStrategy()
doubleClicked()
ColorProvider
ColorProvider()
dispose()
getColor()
propertyChange()
AbstractXMLParser
parseDocument()
GenericWhitespaceDetector
isWhitespace()
EditorImages
getImage()
init()
getImageDescriptor()
JsFunctionScanner
JsFunctionScanner()
AbstractScriptContentProvider
«interface»
IScriptContentProvider
parse()
OutlineDocumentListener
OutlineDocumentListener()
documentAboutToBeChanged()
documentChanged()
reconcile()
«interface»
IScriptEditorOutlinePage
update()

C.XI de.kt.scripteditor.preview (Vorschau-Plug-In)



D Quelltexte ScriptEditor-Plug-In

D.I de.kt.scripteditor

D.I.1 ScripteditorPlugin.java

```
package de.kt.scripteditor;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

import org.eclipse.core.resources.IWorkspace;
import org.eclipse.core.resources.ResourcesPlugin;
import org.eclipse.core.runtime.IPluginDescriptor;
import org.eclipse.ui.plugin.AbstractUIPlugin;

import de.kt.scripteditor.preferences.ColorPreferencePage;
import de.kt.scripteditor.preferences.FilterPreferencePage;

/**
 * The main plugin class to be used in the desktop.
 */
public class ScripteditorPlugin extends AbstractUIPlugin {
	//The shared instance.
	private static ScripteditorPlugin plugin;
	//Resource bundle.
	private ResourceBundle resourceBundle;

	/**
	 * The constructor.
	 */
	public ScripteditorPlugin(IPluginDescriptor descriptor) {
		super(descriptor);
		plugin = this;
		try {
			resourceBundle = ResourceBundle
			.getBundle("de.kt.scripteditor.JseditorPluginResources");
				//$NON-NLS-1$
		} catch (MissingResourceException x) {
			resourceBundle = null;
		}
	}

	/**
	 * Returns the shared instance.
	 */
	public static ScripteditorPlugin getDefault() {
		return plugin;
	}

	/**
	 * Returns the workspace instance.
	 */
	public static IWorkspace getWorkspace() {
		return ResourcesPlugin.getWorkspace();
	}

	/**
	 * Returns the string from the plugin's resource bundle,
	 * or 'key' if not found.
	 */
	public static String getResourceString(String key) {
		ResourceBundle bundle= ScripteditorPlugin
			.getDefault().getResourceBundle();
		try {
			return bundle.getString(key);
```

```
            } catch (MissingResourceException e) {
                  return key;
            }
      }

      /**
       * Returns the plugin's resource bundle.
       */
      public ResourceBundle getResourceBundle() {
            return resourceBundle;
      }

      /* (non-Javadoc)
       * @see org.eclipse.core.runtime.Plugin#initializeDefaultPluginPreferences
       */
      protected void initializeDefaultPluginPreferences(){
            //ensure that the default values are set -> if no Preferences stored
            FilterPreferencePage.initializeDefaults();
            ColorPreferencePage.initializeDefaults();
      }
}
```

D.II de.kt.scripteditor.html

D.II.1 GlobalHTMLCompletionProcessor.java

```
package de.kt.scripteditor.html;

import java.util.Iterator;
import java.util.Vector;

import org.eclipse.jface.text.ITextViewer;
import org.eclipse.jface.text.contentassist.CompletionProposal;
import org.eclipse.jface.text.contentassist.ICompletionProposal;
import org.eclipse.jface.text.contentassist.IContentAssistProcessor;
import org.eclipse.jface.text.contentassist.IContextInformation;
import org.eclipse.jface.text.contentassist.IContextInformationValidator;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.preferences.IPreferenceIDs;

/**
 * Computes the CompletionProposals for the default document area.
 * @author Karsten Thiemann
 */
public class GlobalHTMLCompletionProcessor
      implements IContentAssistProcessor, IPreferenceIDs {

      protected final Vector proposalList = new Vector();
      private boolean closeTags = false;
      private boolean upperCaseTags = false;
      private boolean isClosingTag = false;


      /* (non-Javadoc)
       * @see org.eclipse.jface.text.contentassist.IContentAssist
       * Processor#computeCompletionProposals
       * (org.eclipse.jface.text.ITextViewer, int)
       */
      public ICompletionProposal[] computeCompletionProposals(
            ITextViewer viewer,
            int documentOffset) {

            closeTags =
                  ScripteditorPlugin.getDefault()
                  .getPreferenceStore()
                  .getBoolean(CLOSE_TAG);
            upperCaseTags =
                  ScripteditorPlugin.getDefault()
                  .getPreferenceStore()
                  .getBoolean(
                        TAGS_TO_UPPERCASE);

            final TagNameDetector analyser =
                  new TagNameDetector(viewer, documentOffset);

            final String[] list = HTMLSyntax.getTagNames();
            String wordPartToCompare = analyser.getWordPart().toUpperCase();
            if (wordPartToCompare.startsWith("/")) { //$NON-NLS-1$
                  //closing tag
                  closeTags = false;
                  isClosingTag = true;
                  wordPartToCompare =
                        wordPartToCompare.substring(1,
                              wordPartToCompare.length());
            } else {
                  isClosingTag = false;
            }
            for (int y = 0; y < list.length; y++) {
```

```
                if (list[y].toUpperCase().startsWith(wordPartToCompare)) {
                      if (upperCaseTags) {
                            proposalList.add(list[y].toUpperCase());
                      } else {
                            proposalList.add(list[y].toLowerCase());
                      }
                }
          }

          return turnProposalVectorIntoAdaptedArray(analyser);

    }
    /*
     * Turns the vector into an Array of ICompletionProposal objects
     */
    protected ICompletionProposal[] turnProposalVectorIntoAdaptedArray
          (TagNameDetector analyser) {
          ICompletionProposal[] result =
                new ICompletionProposal[proposalList.size()];

          int index = 0;

          for (Iterator i = proposalList.iterator(); i.hasNext();) {
                String keyWord = (String) i.next();
                int documentOffset = analyser.getOffset();
                int replaceLength = analyser.getWordPart().length();
                boolean isEmptyTag = HTMLSyntax.isEmptyTag(keyWord);
                if(analyser.isFirstLetterIsTagstart()){
                      replaceLength++;
                      documentOffset--;
                }
                if (isClosingTag) {
                      keyWord = "/" + keyWord; //$NON-NLS-1$
                }
                String displayString = "<" + keyWord + ">"; //$NON-NLS-1$
                      //$NON-NLS-2$
                String closedTag = "</" + keyWord + ">"; //$NON-NLS-1$
                      //$NON-NLS-2$
                keyWord = displayString;
                int cursorOffset = keyWord.length();
                if (closeTags && !isEmptyTag) {
                      keyWord += closedTag;
                }
                //Creates a new completion proposal.
                result[index] =
                      new CompletionProposal(
                            keyWord,
                            documentOffset,
                            replaceLength,
                            cursorOffset,
                            null,
                            displayString,
                            null,
                            null);
                index++;
          }

          proposalList.removeAllElements();
          return result;
    }

    /* (non-Javadoc)
     * Method declared on IContentAssistProcessor
     */
    public IContextInformation[] computeContextInformation(
          ITextViewer viewer,
          int documentOffset) {
```

```
		return null;
	}

	/**
	 * The CompletionProposal is activated by the char '<' (tag start).
	 * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
	 * #getCompletionProposalAutoActivationCharacters()
	 */
	public char[] getCompletionProposalAutoActivationCharacters() {
		return new char[] { '<' };
	}

	/**
	 * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
	 * #getContextInformationAutoActivationCharacters()
	 */
	public char[] getContextInformationAutoActivationCharacters() {
		return null;
	}

	/**
	 * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
	 * #getErrorMessage()
	 */
	public String getErrorMessage() {
		return null;
	}

	/**
	 * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
	 * #getContextInformationValidator()
	 */
	public IContextInformationValidator getContextInformationValidator() {
		return null;
	}

}
```

D.II.2 HTMLSyntax.java

```
package de.kt.scripteditor.html;

import de.kt.scripteditor.ScripteditorPlugin;

/**
 * Represents the syntax of HTML. Provides information about tag names,
 * attributes and empty tags.
 * @author Karsten Thiemann
 */
public class HTMLSyntax {

	private static HTMLTagElement[] tagElements;

	static {
		final HTMLSyntaxParser parser = new HTMLSyntaxParser();

		parser.parseDocument(
			ScripteditorPlugin.getDefault()
				.getDescriptor().getInstallURL()
				+ "HTMLSyntax.xml"); //$NON-NLS-1$

		tagElements = parser.getTagElements();

	}

	/**
	 * Returns all HTMLTagElements.
	 * @return HTMLTagElement[] all tag elements
	 */
	public static HTMLTagElement[] getTagElements() {
		return tagElements;
	}

	/**
	 * Returns all tag names.
	 * @return String[] all tag names
	 */
	public static String[] getTagNames() {
		final String[] tagNames = new String[tagElements.length];
		for (int i = 0; i < tagNames.length; i++) {
			tagNames[i] = ((HTMLTagElement) tagElements[i]).getName();
		}
		return tagNames;
	}

	/**
	 * Returns all Attibutes for a given tag.
	 * @param tagName the name of the tag
	 * @return String[] all attributes of the given tag
	 */
	public static String[] getAttributeNamesForTagElement(String tagName) {
		for (int i = 0; i < tagElements.length; i++) {
			if(tagElements[i].getName().equalsIgnoreCase(tagName)){
				return tagElements[i].getAttributeNames();
			}
		}
		return new String[0];
	}
	/**
	 * Returns true if the given tag is an empty tag, false otherwise.
	 * @param tagName the name of the tag
	 * @return boolean true if the given tag is an empty tag, false otherwise
	 */
	public static boolean isEmptyTag(String tagName) {
```

```
            for (int i = 0; i < tagElements.length; i++) {
                  if(tagElements[i].getName().equalsIgnoreCase(tagName)){
                        return tagElements[i].isEmptyTag();
                  }
            }
            return false;
      }
}
```

D.II.3 HTMLSyntaxParser.java

```
package de.kt.scripteditor.html;

import java.util.Iterator;
import java.util.Vector;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import de.kt.scripteditor.util.AbstractXMLParser;

/**
 * Parses a HTMLSyntax-XMLFile and generates an array of HTMLTagElements.
 * @author Karsten
 */
public class HTMLSyntaxParser extends AbstractXMLParser{

	private final Vector tags = new Vector();

	/**
	 * Fills the Vector with HTMLTagElements.
	 * @param d the Document
	 */
	protected void doContent(Document d) {
		final NodeList nodeList = d.getElementsByTagName("tag");
			//$NON-NLS-1$
		for (int i = 0; i < nodeList.getLength(); i++) {
			final Element e = (Element) nodeList.item(i);
			String tagName = ""; //$NON-NLS-1$
			if (e.hasAttribute("name")) { //$NON-NLS-1$
				tagName = e.getAttribute("name"); //$NON-NLS-1$
			}
			boolean isEmptyTag = false;
			if (e.hasAttribute("isEmptyTag")) { //$NON-NLS-1$
				isEmptyTag = (e.getAttribute("isEmptyTag")
					.equalsIgnoreCase("yes"))
					?true
					:false; //$NON-NLS-1$ //$NON-NLS-2$
			}
			final Vector attributeNames = new Vector();
			final NodeList attributeNodeList = nodeList.item(i)
				.getChildNodes();
			for (int j = 0; j < attributeNodeList.getLength(); j++) {
				if (attributeNodeList
					.item(j)
					.getNodeName()
					.equalsIgnoreCase("attribute")) { //$NON-NLS-1$
					Element el = (Element) attributeNodeList.item(j);

					if (el.hasAttribute("name")) { //$NON-NLS-1$
						attributeNames.add(el.getAttribute("name"));
							//$NON-NLS-1$
					}
				}
			}
			final String[] attributeNamesArray = new String
				[attributeNames.size()];
			final Iterator it = attributeNames.iterator();
			final boolean hasAttributes = it.hasNext();
			if (hasAttributes) {
				for (int j = 0; j < attributeNamesArray.length; j++) {
					attributeNamesArray[j] =
						(String) attributeNames.elementAt(j);
				}
				tags.add(
```

```
                                    new HTMLTagElement(tagName, attributeNamesArray,
                                          true, isEmptyTag));
                              } else {
                                    tags.add(
                                    new HTMLTagElement(tagName, attributeNamesArray,
                                          false, isEmptyTag));
                              }
                        }
                  }

                  /**
                   * Returns all HTMLTagElements, found in the XML-file.
                   * @return HTMLTagElement[] all HTMLTagElements
                   */
                  public HTMLTagElement[] getTagElements() {
                        final HTMLTagElement[] tagsArray = new HTMLTagElement[tags.size()];
                        for (int i = 0; i < tagsArray.length; i++) {
                              tagsArray[i] = (HTMLTagElement) tags.elementAt(i);
                        }
                        return tagsArray;
                  }
}
```

D.II.4 HTMLTagCompletionProcessor.java

```
package de.kt.scripteditor.html;

import java.util.Iterator;
import java.util.Vector;

import org.eclipse.jface.text.ITextViewer;
import org.eclipse.jface.text.contentassist.CompletionProposal;
import org.eclipse.jface.text.contentassist.ICompletionProposal;
import org.eclipse.jface.text.contentassist.IContentAssistProcessor;
import org.eclipse.jface.text.contentassist.IContextInformation;
import org.eclipse.jface.text.contentassist.IContextInformationValidator;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.preferences.IPreferenceIDs;

/**
 * The HTML content assist processor. Computes the CompletionProposals
 * within HTML-tags.
 * @author Karsten Thiemann
 */
public class HTMLTagCompletionProcessor
      implements IContentAssistProcessor, IPreferenceIDs {

      /**
       * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
       * #computeCompletionProposals(ITextViewer, int)
       */
      protected final Vector proposalList = new Vector();
      private boolean closeTags = false;
      private boolean isClosingTag = false;
      private boolean upperCaseTags = false;

      /**
       * This method returns a list of completion proposals as
       * ICompletionProposal
       * objects. The proposals are based on the word at the offset in the
       * document where the cursor is positioned, either the tags name or an
       * attribute of the tag.
       * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
       * #computeCompletionProposals(ITextViewer, int)
       */

      public ICompletionProposal[] computeCompletionProposals(
            ITextViewer viewer,
            int documentOffset) {

            closeTags =
                  ScripteditorPlugin.getDefault().getPreferenceStore()
                        .getBoolean(CLOSE_TAG);
            upperCaseTags =
                  ScripteditorPlugin.getDefault().getPreferenceStore()
                        .getBoolean(TAGS_TO_UPPERCASE);

            HTMLTagStringAnalyser analyser =
                  new HTMLTagStringAnalyser(viewer, documentOffset);

            String[] list;
            boolean isAttribute = analyser.isWordPartIsAttributePart();
            if (isAttribute) {
                  //compare wordPart with all attributes of the tag
                  list =
                        HTMLSyntax.getAttributeNamesForTagElement(
                              analyser.getTagName());
            } else {
                  //compare wordPart with all tagnames
```

```
                list = HTMLSyntax.getTagNames();
        }

        String wordPartToCompare = analyser.getWordPart().toUpperCase();
        if (wordPartToCompare.startsWith("/")) { //$NON-NLS-1$
                //closing tag
                closeTags = false;
                isClosingTag = true;
                wordPartToCompare =
                        wordPartToCompare.substring(1,
                                wordPartToCompare.length());
        } else {
                isClosingTag = false;
        }

        for (int y = 0; y < list.length; y++) {
                if (list[y].toUpperCase().startsWith(wordPartToCompare)) {
                        if (upperCaseTags && !isAttribute) {
                                proposalList.add(list[y].toUpperCase());
                        } else {
                                proposalList.add(list[y].toLowerCase());
                        }
                }
        }
        return turnProposalVectorIntoAdaptedArray(analyser);

}
/*
 * Turns the vector into an Array of ICompletionProposal objects
 */
protected ICompletionProposal[] turnProposalVectorIntoAdaptedArray
        (HTMLTagStringAnalyser analyser) {
        final ICompletionProposal[] result =
                new ICompletionProposal[proposalList.size()];

        int index = 0;

        for (Iterator i = proposalList.iterator(); i.hasNext();) {
                String keyWord = (String) i.next();
                String displayString = keyWord;
                int cursorOffset = keyWord.length();
                boolean isEmptyTag = HTMLSyntax.isEmptyTag(keyWord);
                if (analyser.isWordPartIsAttributePart()) {
                        keyWord += "=\"\""; //$NON-NLS-1$
                        cursorOffset += 2;
                } else {
                        if (isClosingTag) {
                                keyWord = "/" + keyWord; //$NON-NLS-1$
                                cursorOffset++;
                        }
                        displayString = "<" + keyWord + ">"; //$NON-NLS-1$
                                //$NON-NLS-2$
                        if (closeTags && !isEmptyTag) {
                                keyWord = keyWord + ">" + "</" + keyWord + ">";
                                        //$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$
                        } else {
                                keyWord += ">"; //$NON-NLS-1$
                        }
                        cursorOffset++;
                }
                //Creates a new completion proposal.
                result[index] =
                        new CompletionProposal(
                                keyWord,
                                analyser.getOffset(),
                                analyser.getWordPart().length(),
                                cursorOffset,
```

```
                              null,
                              displayString,
                              null,
                              null);
                        index++;
                  }
                  //für nächsten Aufruf Vector leeren!
                  proposalList.removeAllElements();
                  return result;
            }

            /* (non-Javadoc)
             * Method declared on IContentAssistProcessor
             */
            public IContextInformation[] computeContextInformation(
                  ITextViewer viewer,
                  int documentOffset) {
                  return null;
            }

            /**
             * The CompletionProposal is activated by the char '<' (tag start).
             * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
             * #getCompletionProposalAutoActivationCharacters()
             */
            public char[] getCompletionProposalAutoActivationCharacters() {
                  return new char[] { '<' };
            }

            /**
             * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
             * #getContextInformationAutoActivationCharacters()
             */
            public char[] getContextInformationAutoActivationCharacters() {
                  return null;
            }

            /**
             * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
             * #getErrorMessage()
             */
            public String getErrorMessage() {
                  return null;
            }

            /**
             * @see org.eclipse.jface.text.contentassist.IContentAssistProcessor
             * #getContextInformationValidator()
             */
            public IContextInformationValidator getContextInformationValidator() {
                  return null;
            }

}
```

D.II.5 HTMLTagElement.java

```
package de.kt.scripteditor.html;

/**
 * A HTMLTagElement holds the information about a HTML tag.
 * @author Karsten Thiemann
 */
public class HTMLTagElement {
	private final String name;
	private final String[] attributeNames;
	private final boolean hasAttributes;
	private final boolean emptyTag;

	/**
	 * @param name the tags name
	 * @param attributeNames the names of all allowed attributes
	 * @param hasAttributes true if the tag has attributes
	 * @param emptyTag true if the tag is an empty tag
	 */
	public HTMLTagElement(
		String name,
		String[] attributeNames,
		boolean hasAttributes,
		boolean emptyTag) {
		this.name = name;
		this.attributeNames = attributeNames;
		this.hasAttributes = hasAttributes;
		this.emptyTag = emptyTag;
	}

	/**
	 * @return String[] the names of the attributes
	 */
	public String[] getAttributeNames() {
		return attributeNames;
	}
	/**
	 * @return String the tags name
	 */
	public String getName() {
		return name;
	}
	/**
	 * @return boolean true if the tag has attributes
	 */
	public boolean isHasAttributes() {
		return hasAttributes;
	}

	/**
	 * @return boolean true if the tag is an empty tag
	 */
	public boolean isEmptyTag() {
		return emptyTag;
	}

}
```

D.II.6 HTMLTagStringAnalyser.java

```
package de.kt.scripteditor.html;

import java.util.StringTokenizer;
import java.util.Vector;

import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.ITextViewer;

/**
 * The HTMLTagStringAnalyser analyses the chars left of the cursor.
 * It determinates the type of the wordPart (tagname or attributename).
 * @author Karsten Thiemann
 */
public class HTMLTagStringAnalyser {
	private String tagName;
	private String wordPart;
	private int documentOffset;
	private final Vector elements;
	private boolean wordPartIsAttributePart = false;
	private boolean lastCharacterIsSpace = false;

	/**
	 * Constructor.
	 * @param viewer is a text viewer
	 * @param documentOffset the cursors offset
	 */
	public HTMLTagStringAnalyser(ITextViewer viewer, int offset) {
		elements = new Vector();
		documentOffset = offset;
		int docOffset = documentOffset - 1;
		try {
			if (Character.isSpace(viewer.getDocument()
				.getChar(docOffset))) {
				lastCharacterIsSpace = true;
			}
			StringBuffer buffer = new StringBuffer();
			//TODO solution for < in inner strings
			//we are inside a HTML-Tag!
			while (((docOffset) >= viewer.getTopIndexStartOffset())
				&& (viewer.getDocument().getChar(docOffset)) != '<') {
				buffer.append(viewer.getDocument().getChar(docOffset));
				docOffset--;
			}
			wordPart = buffer.reverse().toString();
			generateStringVector(wordPart);
			analyse();
		} catch (BadLocationException e) {
			// do nothing
		}
	}

	/**
	 * Determinates the type of the wordPart (tagname or attributename).
	 */
	private void analyse() {
		if (!elements.isEmpty()) {
			tagName = (String) elements.firstElement();
			wordPart = (String) elements.lastElement();
		}
		if (elements.size() >= 2) {
			wordPartIsAttributePart = true;
			//mindestens zwei Strings im Tag, erster ist tagName
		}
		if (lastCharacterIsSpace) {
			wordPartIsAttributePart = true;
```

```
                wordPart = ""; //$NON-NLS-1$
            }

        }

        private void generateStringVector(String wordPart) {
            final StringTokenizer tokenizer =
                new StringTokenizer(wordPart, " "); //$NON-NLS-1$
            while (tokenizer.hasMoreTokens()) {
                elements.add(tokenizer.nextToken());
            }
        }

        /**
         * Returns the offset of the wordPart.
         * @return int the offset of the wordPart.
         */
        public int getOffset() {
            return documentOffset - wordPart.length();
        }

        /**
         * Returns the name of the tag.
         * @return String the tags name
         */
        public String getTagName() {
            return tagName;
        }

        /**
         * Returns the word part, left of the cursors position.
         * @return String the word part
         */
        public String getWordPart() {
            return wordPart;
        }

        /**
         * Returns the type of the word part.
         * @return boolean true if the word part is an attribute,
         * false if it is a tagname
         */
        public boolean isWordPartIsAttributePart() {
            return wordPartIsAttributePart;
        }

}
```

D.II.7 TagNameDetector.java

```
package de.kt.scripteditor.html;

import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.ITextViewer;

/**
 * Detects the HTML-Tagname left from the given documentOffset.
 * @author Karsten Thiemann
 */
public class TagNameDetector {
	private String tagName = ""; //$NON-NLS-1$
	private int docOffset;
	private boolean firstLetterIsTagstart = false;

	/**
	 * Constructor.
	 * @param viewer is a text viewer
	 * @param documentOffset into the document
	 */
	public TagNameDetector(ITextViewer viewer, int documentOffset) {
		docOffset = documentOffset - 1;
		try {
			while (((docOffset) >= viewer.getTopIndexStartOffset())
				&& (Character
					.isLetterOrDigit(viewer.getDocument().getChar
						(docOffset))
					|| viewer.getDocument().getChar(docOffset) == '/'
			|| viewer.getDocument().getChar(docOffset) == '<')) {
				if(viewer.getDocument().getChar(docOffset) == '<'){
					firstLetterIsTagstart = true;
					break;
				}
				docOffset--;
			}
			//we've been one step too far : increase the offset
			docOffset++;
			tagName =
				viewer.getDocument().get(docOffset, documentOffset -
					docOffset);
		} catch (BadLocationException e) {
			// do nothing
		}
	}

	/**
	 * @return String the word part
	 */
	public String getWordPart() {
		return tagName;
	}

	/**
	 * @return int the offset of the word part
	 */
	public int getOffset() {
		return docOffset;
	}

/**
* @return boolean true if the first char left of the cursor position is '<'
*/
	public boolean isFirstLetterIsTagstart() {
		return firstLetterIsTagstart;
	}
}
```

D.III de.kt.scripteditor.htmleditor

D.III.1 HTMLCreationPage.java

```
package de.kt.scripteditor.htmleditor;

import java.io.ByteArrayInputStream;
import java.io.InputStream;

import org.eclipse.core.resources.IFile;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Combo;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Event;
import org.eclipse.swt.widgets.FileDialog;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Text;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.dialogs.WizardNewFileCreationPage;

/**
 * This page allows the user to set a title for the new document.
 * Additionally, the user can choose a html-version, set the documents
 * title and add an external CSS-file.
 * @author Karsten Thiemann
 */
public class HTMLCreationPage extends WizardNewFileCreationPage {
	private final IWorkbench workbench;
	private Text title;
	private Text cssFileName;
	private Button showFileChooserButton;
	private Combo version;
	private FileDialog cssDialog;

	private static int nameCounter = 1;
	/**
	 * Creates the page for the html creation wizard.
	 * @param workbench  the workbench on which the page should be created
	 * @param selection  the current selection
	 */
	public HTMLCreationPage(
		IWorkbench workbench,
		IStructuredSelection selection) {
		super("CreateHTMLPage1", selection); //$NON-NLS-1$
		this.setTitle(HTMLEditorMessages.getString
			("HTMLCreationPage.Create_HTML_File")); //$NON-NLS-1$
		this.setDescription(HTMLEditorMessages.getString
		("HTMLCreationPage.Create_HTML_File_description")); //$NON-NLS-1$
		this.workbench = workbench;
	}
	/** (non-Javadoc)
	 * Method declared on IDialogPage.
	 */
	public void createControl(Composite parent) {
		// inherit default container and name specification widgets
		super.createControl(parent);
		Composite composite = (Composite) getControl();
```

```
        this.setFileName("sample" + nameCounter + ".htm"); //$NON-NLS-1$
              //$NON-NLS-2$

        // generates a new section
        Group group = new Group(composite, SWT.NONE);
        group.setLayout(new GridLayout());
        group.setText(HTMLEditorMessages.getString
              ("HTMLCreationPage.HTML_Settings")); //$NON-NLS-1$
        group.setLayoutData(
              new GridData(
                    GridData.GRAB_HORIZONTAL |
                          GridData.HORIZONTAL_ALIGN_FILL));

        title = createText(group, "page-title"); //$NON-NLS-1$
        version =
              createCombo(
                    group,
                    new String[] {
                          "html_strict", //$NON-NLS-1$
                          "html_transitional", //$NON-NLS-1$
                          "html_frameset" }); //$NON-NLS-1$
        version.select(1);
        cssFileName = createText(group, ""); //$NON-NLS-1$
        showFileChooserButton = createButton(group,
              HTMLEditorMessages.getString
                    ("HTMLCreationPage.Select_CSS_File")); //$NON-NLS-1$
        showFileChooserButton.addSelectionListener(new SelectionAdapter() {
              public void widgetSelected(SelectionEvent event) {
                    String filename = cssDialog.open();
                    if (filename != null) {
                          cssFileName.setText(filename);
                    }
              }
        });
        cssDialog = new FileDialog(getShell(), SWT.OPEN);
        cssDialog.setFilterExtensions(new String[] { "*.css" });
              //$NON-NLS-1$
        setPageComplete(validatePage());

  }

  /**
   * Creates a new file resource with a standard HTML-content based on
   * the users input.
   * @return whether creation was successful
   */
  public boolean finish() {
        // create the new file resource
        IFile newFile = createNewFile();
        if (newFile == null)
              return false; // ie.- creation was unsuccessful

        // Since the file resource was created fine, open it for editing
        try {
              IWorkbenchWindow dwindow = workbench
                    .getActiveWorkbenchWindow();
              IWorkbenchPage page = dwindow.getActivePage();
              if (page != null)
                    page.openEditor(newFile);
        } catch (PartInitException e) {
              e.printStackTrace();
              return false;
        }
        nameCounter++;
        return true;
  }
  /**
```

```
	 * Generates a standard HTML-Page based on
	 * the users input.
	 */
	protected InputStream getInitialContents() {

		StringBuffer sb = new StringBuffer();
		if (version.getSelectionIndex() != -1) {
			sb.append(HTMLEditorMessages.getString
				("HTMLCreationPage."+version.getText()));
		}
		sb.append(HTMLEditorMessages.getString
			("HTMLCreationPage.HTML_PRE_TITLE")); //$NON-NLS-1$
		sb.append(title.getText());
		sb.append(HTMLEditorMessages.getString
			("HTMLCreationPage.HTML_CLOSE_TITLE")); //$NON-NLS-1$
		if (cssFileName.getText() != "") { //$NON-NLS-1$
			sb.append("\t<link rel=\"stylesheet\" type=\"text/css\"
				href=\""); //$NON-NLS-1$
			sb.append(cssFileName.getText());
			sb.append("\">\n"); //$NON-NLS-1$
		}
		sb.append(HTMLEditorMessages.getString
			("HTMLCreationPage.HTML_POST_TITLE")); //$NON-NLS-1$
		return new ByteArrayInputStream(sb.toString().getBytes());
	}
	/** (non-Javadoc)
	 * Method declared on WizardNewFileCreationPage.
	 */
	protected String getNewFileLabel() {
		return HTMLEditorMessages.getString
			("HTMLCreationPage.HTML_file_name"); //$NON-NLS-1$
	}
	/** (non-Javadoc)
	 * Method declared on WizardNewFileCreationPage.
	 */
	public void handleEvent(Event e) {
		super.handleEvent(e);
	}
	/**
	 * Utility method that creates a text widget instance
	 * and sets the default layout data.
	 *
	 * @param parent  the parent for the new text widget
	 * @param text  the text for the new text widget
	 * @return the new text widget
	 */
	private Text createText(Composite parent, String text) {
		Text textInfo = new Text(parent, SWT.SINGLE | SWT.BORDER);
		textInfo.setText(text);
		GridData data = new GridData(GridData.FILL_HORIZONTAL);
		textInfo.setLayoutData(data);
		return textInfo;
	}

	/**
	 * Utility method that creates a combo widget instance
	 * and sets the default layout data.
	 * @param group the parent for the new combo widget
	 * @param items the items of the new combo
	 * @return the new combo widget
	 */
	private Combo createCombo(Group group, String[] items) {
		Combo combo = new Combo(group, SWT.DROP_DOWN | SWT.READ_ONLY);
		combo.setItems(items);
		GridData data = new GridData(GridData.FILL_HORIZONTAL);
		combo.setLayoutData(data);
		return combo;
```

```
	}

	/**
	 * Utility method that creates a button with the given text.
	 * @param group the parent for the new button
	 * @param string the buttons text
	 * @return the new button
	 */
	private Button createButton(Group group, String string) {
		Button button = new Button(group, SWT.PUSH | SWT.RIGHT);
		button.setText(string);
		return button;
	}
}
```

D.III.2 HTMLCreationWizard.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.wizard.Wizard;
import org.eclipse.ui.INewWizard;
import org.eclipse.ui.IWorkbench;

/**
 * This wizard creates html files.
 * @author Karsten Thiemann
 */
public class HTMLCreationWizard extends Wizard implements INewWizard {
	private IStructuredSelection selection;
	private IWorkbench workbench;
	private HTMLCreationPage mainPage;

	/** (non-Javadoc)
	 * Method declared on Wizard.
	 */
	public void addPages() {
		mainPage = new HTMLCreationPage(workbench, selection);
		addPage(mainPage);
	}

	/** (non-Javadoc)
	 * Method declared on INewWizard
	 */
	public void init(IWorkbench workbench, IStructuredSelection selection) {
		this.workbench = workbench;
		this.selection = selection;
		setWindowTitle(HTMLEditorMessages.getString
			("HTMLCreationWizard.New_HTML_File")); //$NON-NLS-1$
	}

	/** (non-Javadoc)
	 * Method declared on IWizard
	 */
	public boolean performFinish() {
		return mainPage.finish();
	}

}
```

D.III.3 HTMLDocumentProvider.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.IDocumentPartitioner;
import org.eclipse.jface.text.rules.DefaultPartitioner;
import org.eclipse.ui.editors.text.FileDocumentProvider;


/**
 * The document provider for *.htm(l) files.
 * @author Karsten Thiemann
 */
public class HTMLDocumentProvider extends FileDocumentProvider {
	/** legal content types */
	private final static String[] TYPES =
		new String[] {
			HTMLPartitionScanner.HTML_COMMENT,
			HTMLPartitionScanner.HTML_TAG,
			HTMLPartitionScanner.JAVASCRIPT};
	private static HTMLPartitionScanner scanner = null;

	public HTMLDocumentProvider() {
		super();
	}

	/* (non-Javadoc)
	 * Method declared on AbstractDocumentProvider
	 */
	protected IDocument createDocument(Object element) throws CoreException {
		final IDocument document = super.createDocument(element);
		if (document != null) {
			IDocumentPartitioner partitioner = createHTMLPartitioner();
			document.setDocumentPartitioner(partitioner);
			partitioner.connect(document);
		}
		return document;
	}

	/**
	 * Return a partitioner for .html files.
	 */
	private IDocumentPartitioner createHTMLPartitioner() {
		return new DefaultPartitioner(getHTMLPartitionScanner(), TYPES);
	}

	/**
	 * Return a scanner for creating html partitions.
	 */
	private HTMLPartitionScanner getHTMLPartitionScanner() {
		if (scanner == null) {
			scanner = new HTMLPartitionScanner();
		}
		return scanner;
	}
}
```

D.III.4 HTMLEditor.java

```
package de.kt.scripteditor.htmleditor;

import java.util.ResourceBundle;

import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.Separator;
import org.eclipse.jface.text.source.ISourceViewer;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.texteditor.ITextEditorActionDefinitionIds;
import org.eclipse.ui.texteditor.TextOperationAction;
import org.eclipse.ui.views.contentoutline.IContentOutlinePage;

import de.kt.scripteditor.htmleditor.actions.TidyAction;
import de.kt.scripteditor.htmleditor.outline.HTMLContentOutlinePage;
import de.kt.scripteditor.util.AbstractScriptEditor;
import de.kt.scripteditor.util.SnipletLibrary;

/**
 * The editor for *.htm(l) files.
 * @author Karsten Thiemann
 * @since 11.10.2003
 */
public class HTMLEditor extends AbstractScriptEditor{
	/** The outline page */
	private HTMLContentOutlinePage outlinePage;
	private IAction formatAction;
	private IAction tidyAction;

	/**
	 * Constructor.
	 */
	public HTMLEditor() {
		super();
	}

	/**
	 * Sets the input of the outline page to null.
	 * @see org.eclipse.ui.IWorkbenchPart#dispose()
	 */
	public void dispose() {
		HTMLEditorEnvironment.disconnect(this);
		if (outlinePage != null)
			outlinePage.setInput(null);
		super.dispose();
	}

	/**
	 * Sets the input of the outline page.
	 * @see org.eclipse.ui.IWorkbenchPart#doSetInput()
	 */
	public void doSetInput(IEditorInput input) throws CoreException {
		super.doSetInput(input);
		if (outlinePage != null)
			outlinePage.setInput(input);
	}

	/**
	 * Returns the HTMLContentOutlinePage if request is for a an
	 * outline page.
	 */
	public Object getAdapter(Class required) {
		if (IContentOutlinePage.class.equals(required)) {
```

```
            if (outlinePage == null) {
                  outlinePage =
                        new HTMLContentOutlinePage(getDocumentProvider(),
                              this);
                  if (getEditorInput() != null)
                        outlinePage.setInput(getEditorInput());
            }
            return outlinePage;
      }
      return super.getAdapter(required);
}

/* (non-Javadoc)
 * Method declared on AbstractTextEditor
 */
protected void initializeEditor() {
      super.initializeEditor();
      //Singeltons erzeugen
      HTMLEditorEnvironment.connect(this);
      //DocumentProvider setzen
      setDocumentProvider(new HTMLDocumentProvider());
      //SourceViewerConfiguration setzen
      setSourceViewerConfiguration(new HTMLSourceViewerConfiguration());
      setEditorContextMenuId("#HTMLEditorContext"); //$NON-NLS-1$
      setRulerContextMenuId("#HTMLRulerContext"); //$NON-NLS-1$
      setHelpContextId("de.kt.scripteditor.editor_context"); //$NON-NLS-1$
}


/** (non-Javadoc)
* Method declared on AbstractTextEditor
*/
protected void editorContextMenuAboutToShow(IMenuManager parentMenu) {
      //format/tidy action dem Kontextmenue hinzufügen
      super.editorContextMenuAboutToShow(parentMenu);
      parentMenu.add(new Separator());
      parentMenu.add(formatAction);
      parentMenu.add(tidyAction);
}

/* (non-Javadoc)
 * @see org.eclipse.ui.texteditor.AbstractTextEditor#createActions()
 */
protected void createActions() {
      super.createActions();
      final ResourceBundle bundle = HTMLEditorMessages.getResourceBundle
            ();
      //Content Assist action erzeugen damit sie automatisch mit
      //strg+space aktiviert werden kann
      final IAction a =
            new TextOperationAction(
                  bundle,
                  "ContentAssistProposal.", //$NON-NLS-1$
                  this,
                  ISourceViewer.CONTENTASSIST_PROPOSALS);
      a.setActionDefinitionId(
            ITextEditorActionDefinitionIds.CONTENT_ASSIST_PROPOSALS);
      setAction("ContentAssistProposal", a); //$NON-NLS-1$
      //Content Format action erzeugen, damit sie in
      //MultiPageEditorContributor zugänglich ist
      formatAction =
            new TextOperationAction(
                  bundle,
                  "ContentFormatProposal.", //$NON-NLS-1$
                  this,
                  ISourceViewer.FORMAT);
      setAction("ContentFormatProposal", formatAction); //$NON-NLS-1$
```

```
            tidyAction = new TidyAction();
            ((TidyAction) tidyAction).setActiveEditor(this);
      }

      /* (non-Javadoc)
       * @see de.kt.jseditor.util.AbstractScriptEditor#updateSniplets()
       */
      protected void updateSniplets() {
            sniplets = SnipletLibrary.getHTMLSniplets();
      }


}
```

D.III.5 HTMLEditorEnvironment.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.jface.text.rules.RuleBasedScanner;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.htmleditor.outline.HTMLOutlineTagScanner;
import de.kt.scripteditor.htmleditor.outline.TagElementFilter;
import de.kt.scripteditor.javascript.JsCodeScanner;
import de.kt.scripteditor.util.ColorProvider;
import de.kt.scripteditor.util.JsFunctionScanner;

/**
 * The HTMLEditorEnvironment maintains singletons used by the html editor.
 * @author Karsten Thiemann
 */
public class HTMLEditorEnvironment {

	private static ColorProvider colorProvider;
	private static HTMLTagScanner htmlTagScanner;
	private static JsCodeScanner jsCodeScanner;
	private static JsFunctionScanner functionScanner;
	private static HTMLOutlineTagScanner outlineTagScanner;
	private static TagElementFilter tagElementFilter;

	private static int refCount = 0;

	/**
	 * Initializes the receiver if it is the first activation.
	 */
	public static void connect(Object client) {
		if (++refCount == 1) {
			colorProvider = new ColorProvider();
			htmlTagScanner = new HTMLTagScanner(colorProvider);
			jsCodeScanner = new JsCodeScanner(colorProvider);
			functionScanner = new JsFunctionScanner();
			outlineTagScanner = new HTMLOutlineTagScanner();
			tagElementFilter = new TagElementFilter();
			ScripteditorPlugin
				.getDefault()
				.getPreferenceStore()
				.addPropertyChangeListener(
				colorProvider);
			ScripteditorPlugin
				.getDefault()
				.getPreferenceStore()
				.addPropertyChangeListener(
				htmlTagScanner);
			ScripteditorPlugin
				.getDefault()
				.getPreferenceStore()
				.addPropertyChangeListener(
				tagElementFilter);
		}

	}

	/**
	 * A disconnection has occured - clear the receiver if it is the last
	 * deactivation.
	 */
	public static void disconnect(Object client) {
		if (--refCount == 0) {
			ScripteditorPlugin
				.getDefault()
				.getPreferenceStore()
```

```
                        .removePropertyChangeListener(
                        htmlTagScanner);
                ScripteditorPlugin
                        .getDefault()
                        .getPreferenceStore()
                        .removePropertyChangeListener(
                        colorProvider);
                ScripteditorPlugin
                        .getDefault()
                        .getPreferenceStore()
                        .removePropertyChangeListener(
                        tagElementFilter);
                htmlTagScanner = null;
                colorProvider.dispose();
                colorProvider = null;
                functionScanner = null;
                outlineTagScanner = null;
                tagElementFilter = null;
        }
    }

    /**
     * Returns the singleton js scanner.
     */
    public static RuleBasedScanner getJsCodeScanner() {
        return jsCodeScanner;
    }

    /**
     * Returns the singleton html scanner.
     */
    public static RuleBasedScanner getHTMLTagScanner() {
        return htmlTagScanner;

    }

    /**
     * Returns the singleton html outline scanner.
     */
    public static RuleBasedScanner getHTMLOutlineTagScanner() {
        return outlineTagScanner;

    }

    /**
     * Returns the singleton function scanner.
     */
    public static RuleBasedScanner getJsFunctionScanner() {
        return functionScanner;
    }

    /**
     * Returns the singleton color provider.
     */
    public static ColorProvider getColorProvider() {
        return colorProvider;
    }

    /**
     * Returns the singleton TagElementFilter.
     */
    public static TagElementFilter getTagElementFilter(){
        return tagElementFilter;
    }

}
```

D.III.6 HTMLEditorMessages.java

```
package de.kt.scripteditor.htmleditor;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

/**
 * The HTMLEditorMessages manages the access to the editors ressource bundle.
 * @author Karsten Thiemann
 */
public class HTMLEditorMessages {

	private static final String RESOURCE_BUNDLE =
		"de.kt.scripteditor.htmleditor.HTMLEditorMessages"; //$NON-NLS-1$

	private static ResourceBundle fgResourceBundle =
		ResourceBundle.getBundle(RESOURCE_BUNDLE);

	/**
	 * private constructor to avoid instantiation.
	 */
	private HTMLEditorMessages() {
	}

	/**
	 * @param key the key of the stored string
	 * @return String the string with the given key
	 */
	public static String getString(String key) {
		try {
			return fgResourceBundle.getString(key);
		} catch (MissingResourceException e) {
			return "!" + key + "!"; //$NON-NLS-2$ //$NON-NLS-1$
		}
	}

	public static ResourceBundle getResourceBundle() {
		return fgResourceBundle;
	}
}
```

D.III.7 HTMLFormattingStrategy.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.jface.text.formatter.IFormattingStrategy;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.html.HTMLSyntax;
import de.kt.scripteditor.preferences.IPreferenceIDs;

/**
 * This FormattingStrategy turns all HTML-tags to the prefered style
 * (uppercase or lowercase).
 * @author Karsten Thiemann
 */
public class HTMLFormattingStrategy
      implements IFormattingStrategy, IPreferenceIDs {

      /* (non-Javadoc)
       * @see
       * org.eclipse.jface.text.formatter.IFormattingStrategy#formatterStarts
       * (java.lang.String)
       */
      public void formatterStarts(String initialIndentation) {
            //do nothing
      }

      /* (non-Javadoc)
       * @see org.eclipse.jface.text.formatter.IFormattingStrategy#format
       * (java.lang.String, boolean, java.lang.String, int[])
       */
      public String format(
            String content,
            boolean isLineStart,
            String indentation,
            int[] positions) {
            //Problem: wenn die Position von Tokens verändert wird muss das
            //mitgeteilt werden (wie auch immer...)
            //als Formatter daher nur TAGS klein/gross

            return toConsistantCase(content);
      }

      /**
       * Turns the given tag into upper or lowercase, as specified in the
       * preference page.
       * @param content the tag to format
       * @return
       */
      private String toConsistantCase(String content) {
            final String[] tagNames = HTMLSyntax.getTagNames();
            final boolean uppercase =
                  ScripteditorPlugin.getDefault().getPreferenceStore().
                        getBoolean(TAGS_TO_UPPERCASE);

            if (content.toUpperCase().startsWith("<SCRIPT")) { //$NON-NLS-1$
                  if(uppercase){
                        content = "<SCRIPT" + content.substring(7,
                              content.length()-9) + "</SCRIPT>";
                                    //$NON-NLS-1$ //$NON-NLS-2$
                  }else{
                        content = "<script" + content.substring(7,
                              content.length()-9) + "</script>";
                                    //$NON-NLS-1$ //$NON-NLS-2$
                  }
                  return content;
            }
```

```
            for (int i = 0; i < tagNames.length; i++) {
                  final String replaceName =
                        (uppercase)
                              ? tagNames[i].toUpperCase()
                              : tagNames[i].toLowerCase();
                  if (content
                        .toUpperCase()
                        .startsWith("<" + tagNames[i].toUpperCase())) {
                              //$NON-NLS-1$
                        content =
                              "<" //$NON-NLS-1$
                                    + replaceName
                                    + content.substring(
                                          tagNames[i].length() + 1,
                                          content.length());
                  } else if (
                        content.toUpperCase().startsWith(
                              "</" + tagNames[i].toUpperCase())) { //$NON-NLS-1$
                        content =
                              "</" //$NON-NLS-1$
                                    + replaceName
                                    + content.substring(
                                          tagNames[i].length() + 2,
                                          content.length());
                  }
            }
            return content;
      }

      /* (non-Javadoc)
       * @see
       * org.eclipse.jface.text.formatter.IFormattingStrategy#formatterStops()
       */
      public void formatterStops() {
            //do nothing
      }

}
```

D.III.8 HTMLPartitionScanner.java

```
package de.kt.scripteditor.htmleditor;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.text.rules.IPredicateRule;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;
import org.eclipse.jface.text.rules.RuleBasedPartitionScanner;
import org.eclipse.jface.text.rules.SingleLineRule;
import org.eclipse.jface.text.rules.Token;

/**
 * Partition scanner for htm(l) files. Specifies the different partitions
 * of the document (tags, comments, scripts and default).
 * @author Karsten Thiemann
 */
public class HTMLPartitionScanner extends RuleBasedPartitionScanner {

	public final static String HTML_COMMENT = "__html_comment"; //$NON-NLS-1$
	public final static String HTML_TAG = "__html_tag"; //$NON-NLS-1$
	public final static String JAVASCRIPT = "__javascript"; //$NON-NLS-1$

	public HTMLPartitionScanner() {
		super();
		final List rules = new ArrayList();

		final IToken htmlComment = new Token(HTML_COMMENT);
		final IToken tag = new Token(HTML_TAG);
		final IToken javascript = new Token(JAVASCRIPT);

		// Add rule for strings and character constants.
		rules.add(new SingleLineRule("\"", "\"", Token.UNDEFINED, '\\'));
			//$NON-NLS-1$ //$NON-NLS-2$
		rules.add(new SingleLineRule("'", "'", Token.UNDEFINED, '\\'));
			//$NON-NLS-1$ //$NON-NLS-2$

		//Add rule for xml(html) comments.
		rules.add(new MultiLineRule("<!--", "-->", htmlComment));
			//$NON-NLS-1$ //$NON-NLS-2$

		//Add rule for html-tags excluding <script ...></script>
		rules.add(new TagRule(tag));

		//Add rule for javascript area.
		rules.add(new MultiLineRule("<script", "</script>", javascript));
			//$NON-NLS-1$ //$NON-NLS-2$
		rules.add(new MultiLineRule("<SCRIPT", "</SCRIPT>", javascript));
			//$NON-NLS-1$ //$NON-NLS-2$
		rules.add(new MultiLineRule("<SCRIPT", "</script>", javascript));
			//$NON-NLS-1$ //$NON-NLS-2$
		rules.add(new MultiLineRule("<script", "</SCRIPT>", javascript));
			//$NON-NLS-1$ //$NON-NLS-2$

		IPredicateRule[] result = new IPredicateRule[rules.size()];
		rules.toArray(result);
		setPredicateRules(result);
	}
}
```

D.III.9 HTMLSourceViewerConfiguration.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.ITextDoubleClickStrategy;
import org.eclipse.jface.text.TextAttribute;
import org.eclipse.jface.text.contentassist.ContentAssistant;
import org.eclipse.jface.text.contentassist.IContentAssistant;
import org.eclipse.jface.text.formatter.ContentFormatter;
import org.eclipse.jface.text.formatter.IContentFormatter;
import org.eclipse.jface.text.presentation.IPresentationReconciler;
import org.eclipse.jface.text.presentation.PresentationReconciler;
import org.eclipse.jface.text.rules.BufferedRuleBasedScanner;
import org.eclipse.jface.text.rules.DefaultDamagerRepairer;
import org.eclipse.jface.text.rules.Token;
import org.eclipse.jface.text.source.ISourceViewer;
import org.eclipse.jface.text.source.SourceViewerConfiguration;
import org.eclipse.swt.graphics.RGB;

import de.kt.scripteditor.html.GlobalHTMLCompletionProcessor;
import de.kt.scripteditor.html.HTMLTagCompletionProcessor;
import de.kt.scripteditor.javascript.JsCompletionProcessor;
import de.kt.scripteditor.util.ColorProvider;
import de.kt.scripteditor.util.ScriptEditorDoubleClickStrategy;

/**
 * SourceViewerConfiguration used by the HTMLEditor.
 * @author Karsten Thiemann
 */
public class HTMLSourceViewerConfiguration extends SourceViewerConfiguration {

	/**
	 * Default constructor.
	 */
	public HTMLSourceViewerConfiguration() {
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public String[] getConfiguredContentTypes(ISourceViewer sourceViewer) {
		return new String[] {
			IDocument.DEFAULT_CONTENT_TYPE,
			HTMLPartitionScanner.HTML_COMMENT,
			HTMLPartitionScanner.HTML_TAG,
			HTMLPartitionScanner.JAVASCRIPT };
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public IContentAssistant getContentAssistant(ISourceViewer sourceViewer) {
		final ContentAssistant assistant = new ContentAssistant();
		assistant.setContentAssistProcessor(
			new JsCompletionProcessor(),
			HTMLPartitionScanner.JAVASCRIPT);
		assistant.setContentAssistProcessor(
			new GlobalHTMLCompletionProcessor(),
			IDocument.DEFAULT_CONTENT_TYPE);
		assistant.setContentAssistProcessor(
			new HTMLTagCompletionProcessor(),
			HTMLPartitionScanner.HTML_TAG);

		assistant.enableAutoActivation(true);
		assistant.setAutoActivationDelay(500);
```

```
            assistant.setProposalPopupOrientation(
                  IContentAssistant.PROPOSAL_OVERLAY);
            assistant.setContextInformationPopupOrientation(
                  IContentAssistant.CONTEXT_INFO_ABOVE);
            assistant.setContextInformationPopupBackground(
                  HTMLEditorEnvironment.getColorProvider().getColor(
                        new RGB(150, 150, 0)));

            return assistant;
      }

/* (non-Javadoc)
*@see
*org.eclipse.jface.text.source.SourceViewerConfiguration#getContentFormatter
*org.eclipse.jface.text.source.ISourceViewer)
*/
      public IContentFormatter getContentFormatter(ISourceViewer sourceViewer) {
            final ContentFormatter c = new ContentFormatter();
            c.enablePartitionAwareFormatting(true);
            //formats TagNames to upper or lower case
            final HTMLFormattingStrategy strategy =
                  new HTMLFormattingStrategy();
            c.setFormattingStrategy(strategy, HTMLPartitionScanner.HTML_TAG);
            c.setFormattingStrategy(strategy, HTMLPartitionScanner.JAVASCRIPT);
            return c;
      }

      /* (non-Javadoc)
       * Method declared on SourceViewerConfiguration
       */
      public ITextDoubleClickStrategy getDoubleClickStrategy(
            ISourceViewer sourceViewer,
            String contentType) {
            return new ScriptEditorDoubleClickStrategy();
      }

      /* (non-Javadoc)
       * Method declared on SourceViewerConfiguration
       */
      public String[] getIndentPrefixes(
            ISourceViewer sourceViewer,
            String contentType) {
            return new String[] { "\t", "    " }; //$NON-NLS-1$ //$NON-NLS-2$
      }

      /* (non-Javadoc)
       * Method declared on SourceViewerConfiguration
       */
      public IPresentationReconciler getPresentationReconciler(ISourceViewer
            sourceViewer) {

            final ColorProvider provider =
                  HTMLEditorEnvironment.getColorProvider();
            final PresentationReconciler reconciler = new PresentationReconciler
                  ();

            DefaultDamagerRepairer dr =
                  new DefaultDamagerRepairer(
                        HTMLEditorEnvironment.getJsCodeScanner());
            reconciler.setDamager(dr, HTMLPartitionScanner.JAVASCRIPT);
            reconciler.setRepairer(dr, HTMLPartitionScanner.JAVASCRIPT);

            dr =
                  new DefaultDamagerRepairer(
                        new SingleTokenScanner(
                              new TextAttribute(
                                    provider.getColor
                                          (ColorProvider.HTML_COMMENT))));
```

```
            reconciler.setDamager(dr, HTMLPartitionScanner.HTML_COMMENT);
            reconciler.setRepairer(dr, HTMLPartitionScanner.HTML_COMMENT);

            dr =
                  new DefaultDamagerRepairer(
                        HTMLEditorEnvironment.getHTMLTagScanner());
            reconciler.setDamager(dr, HTMLPartitionScanner.HTML_TAG);
            reconciler.setRepairer(dr, HTMLPartitionScanner.HTML_TAG);

            return reconciler;
      }

      /* (non-Javadoc)
       * Method declared on SourceViewerConfiguration
       */
      public int getTabWidth(ISourceViewer sourceViewer) {
            return 4;
      }

      /**
       * Single token scanner.
       */
      static class SingleTokenScanner extends BufferedRuleBasedScanner {
            //TODO als Listener anmelden und Farbupdate vornehmen...
            public SingleTokenScanner(TextAttribute attribute) {
                  setDefaultReturnToken(new Token(attribute));
            }
      };

}
```

D.III.10 HTMLTagScanner.java

```
package de.kt.scripteditor.htmleditor;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.resource.StringConverter;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.TextAttribute;
import org.eclipse.jface.text.rules.IRule;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.RuleBasedScanner;
import org.eclipse.jface.text.rules.SingleLineRule;
import org.eclipse.jface.text.rules.Token;
import org.eclipse.jface.text.rules.WhitespaceRule;
import org.eclipse.jface.util.IPropertyChangeListener;
import org.eclipse.jface.util.PropertyChangeEvent;
import org.eclipse.swt.graphics.RGB;

import de.kt.scripteditor.preferences.IPreferenceIDs;
import de.kt.scripteditor.util.ColorProvider;
import de.kt.scripteditor.util.GenericWhitespaceDetector;

/**
 * Scanner, used inside of HTML-tags. Defines Strings.
 * @author Karsten Thiemann
 */
public class HTMLTagScanner
      extends RuleBasedScanner
      implements IPropertyChangeListener, IPreferenceIDs {

      private IDocument document;
      private IToken string;
      private IToken htmlTag;
      private ColorProvider colorProvider;

      public HTMLTagScanner(ColorProvider colorProvider) {
            this.colorProvider = colorProvider;
            //setzt die Textfarbe für das Standardtoken
            htmlTag =
                  new Token(
                        new TextAttribute(
                              colorProvider.getColor(ColorProvider.HTML_TAG)));
            string =
                  new Token(
                        new TextAttribute(
                              colorProvider.getColor(ColorProvider.STRING)));
            initRules();
      }

      private void initRules() {
            setDefaultReturnToken(htmlTag);
            List rules = new ArrayList();
            // Add rule for double quotes
            rules.add(new SingleLineRule("\"", "\"", string, '\\'));
                  //$NON-NLS-1$ //$NON-NLS-2$
            // Add a rule for single quotes
            rules.add(new SingleLineRule("'", "'", string, '\\'));
                  //$NON-NLS-1$ //$NON-NLS-2$
            // Add generic whitespace rule.
            rules.add(new WhitespaceRule(new GenericWhitespaceDetector()));
            IRule[] result = new IRule[rules.size()];
            rules.toArray(result);
            setRules(result);
      }
```

```
	/* (non-Javadoc)
	 * @see org.eclipse.jface.text.rules.ITokenScanner#setRange
	 * (org.eclipse.jface.text.IDocument, int, int)
	 */
	public void setRange(IDocument document, int offset, int length) {
		super.setRange(document, offset, length);
		this.document = document;
	}

	/* (non-Javadoc)
	 * @see org.eclipse.jface.util.IPropertyChangeListener#propertyChange
	 * (org.eclipse.jface.util.PropertyChangeEvent)
	 */
	public void propertyChange(PropertyChangeEvent event) {
		if (event.getProperty().equals(STRING_COLOR)) {
			updateToken(event, string);
		} else if (event.getProperty().equals(HTML_TAG_COLOR)) {
			updateToken(event, htmlTag);
		}

	}

	/**
	 * Updates the color of the scanners token.
	 * @param event the PropertyChangeEvent, contains the new color
	 * @param token the token to change
	 */
	private void updateToken(PropertyChangeEvent event, IToken token) {
		RGB rgb = null;
		Object value = event.getNewValue();
		if (value instanceof RGB)
			rgb = (RGB) value;
		else if (value instanceof String)
			rgb = StringConverter.asRGB((String) value);
		token = new Token(new TextAttribute(colorProvider.getColor(rgb)));
		updateColor();
	}

	private void updateColor() {
		initRules();
		document.setDocumentPartitioner(document.getDocumentPartitioner());
		//update von mehreren Dokumenten (aus Vector) scheitert wegen
		//gleichzeitigem Zugriff
	}

}
```

D.III.11 IHTMLPreview.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.swt.widgets.Composite;

/**
 * @author Karsten Thiemann
 * @since 26.10.2003
 */
public interface IHTMLPreview {
	/**
	 * Returns the Composite that is used by the MultiPageEditor to view
	 * the rendered HTML.
	 * @return Composite the Composite that is used by the MultiPageEditor
	 * to view the rendered HTML
	 */
	public Composite getPreviewComposite(Composite parent);

	/**
	 * Updates the preview with the given URI.
	 * @param uri the URI of the HTML-page to show
	 */
	public void updatePreview(String uri);
	/**
	 * Returns the name.
	 * @return String the name is shown on the editors tab.
	 */
	public String getName();
}
```

D.III.12 MultiPageEditor.java

```
package de.kt.scripteditor.htmleditor;

import java.io.StringBufferInputStream;
import java.util.Iterator;
import java.util.Vector;

import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IMarker;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IWorkspaceRoot;
import org.eclipse.core.resources.ResourcesPlugin;
import org.eclipse.core.runtime.CoreException;
import org.eclipse.core.runtime.IConfigurationElement;
import org.eclipse.core.runtime.IExtension;
import org.eclipse.core.runtime.IExtensionPoint;
import org.eclipse.core.runtime.IPath;
import org.eclipse.core.runtime.IPluginRegistry;
import org.eclipse.core.runtime.IProgressMonitor;
import org.eclipse.core.runtime.Platform;
import org.eclipse.jface.dialogs.ErrorDialog;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.IEditorSite;
import org.eclipse.ui.IFileEditorInput;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.part.MultiPageEditorPart;

/**
 * The MultiPageEditor holds the HTML-Editor and preview pages.The preview pages
 * are defined by plugins extending the de.kt.jseditor.preview extension-point.
 * @author Karsten Thiemann
 */
public class MultiPageEditor extends MultiPageEditorPart {

	/** The HTMLEditor used in page 0. */
	private HTMLEditor editor;
	private Vector previews;
	private IFile tempFile = null;

	/**
	 * Creates a multi-page editor.
	 */
	public MultiPageEditor() {
		super();
		previews = new Vector();
	}
	/**
	 * Creates page 0 of the multi-page editor,
	 * which contains a HTMLEditor.
	 */
	void createPage0() {
		try {
			editor = new HTMLEditor();
			int index = addPage(editor, getEditorInput());
			//sets the title of the tab
			setPageText(index, HTMLEditorMessages.getString
				("MultiPageEditor.Source-View")); //$NON-NLS-1$
			//sets the title of the multiPageEditor
			setTitle(editor.getTitle());
		} catch (PartInitException e) {
			ErrorDialog.openError(getSite().getShell(),
				HTMLEditorMessages.getString
				("MultiPageEditor.Error_creating_nested_text_editor"),
```

```
                        //$NON-NLS-1$
                  null, e.getStatus());
            }
      }

      /**
       * Creates the pages of the multi-page editor.
       */
      protected void createPages() {
            createPage0();
            createPreviewPages();
      }

      /**
       * Find Preview-Plugins and add their previews.
       */
      private void createPreviewPages() {
            final IPluginRegistry registry = Platform.getPluginRegistry();
            //get the preview extensionPoint
            final IExtensionPoint extensionPoint =
                  registry.getExtensionPoint("de.kt.scripteditor.preview");
            //get all plugins
            if (extensionPoint != null) {
                  final IExtension[] previewPlugins =
                        extensionPoint.getExtensions();
                  for (int i = 0; i < previewPlugins.length; i++) {
                        final IExtension plugin = previewPlugins[i];
                        //get all previews of the current preview plugin
                        IConfigurationElement[] configurations =
                              plugin.getConfigurationElements();
                        for (int j = 0; j < configurations.length; j++) {
                              try {
                                    //try to instantiate the IHTMLPreview
                                    IHTMLPreview preview =
                                          (
                                                IHTMLPreview) configurations[j]
                                                      .createExecutableExtension(
                                                "class");
                                    previews.add(preview);
                                    int index =
                                          addPage(
                                                preview.getPreviewComposite
                                                      (getContainer()));
                                    setPageText(index, preview.getName());
                              } catch (CoreException e) {
                              // do nothing, the preview just can´t be added
                              }
                        }
                  }
            }
      }

      /**
       * Saves the multi-page editor's document.
       */
      public void doSave(IProgressMonitor monitor) {
            getEditor(0).doSave(monitor);
      }

      /**
       * Saves the multi-page editor's document as another file.
       * Also updates the text for page 0's tab, and updates this multi-page
       * editor's input
       * to correspond to the nested editor's.
       */
      public void doSaveAs() {
            final IEditorPart editor = getEditor(0);
            editor.doSaveAs();
```

```
		setPageText(0, editor.getTitle());
		setInput(editor.getEditorInput());
	}

	/* (non-Javadoc)
	 * Method declared on IEditorPart
	 */
	public void gotoMarker(IMarker marker) {
		setActivePage(0);
		getEditor(0).gotoMarker(marker);
	}

	/**
	 * The implementation of this method
	 * checks that the input is an instance of IFileEditorInput.
	 */
	public void init(IEditorSite site, IEditorInput editorInput)
		throws PartInitException {
		if (!(editorInput instanceof IFileEditorInput))
			throw new PartInitException(HTMLEditorMessages.getString
			("MultiPageEditor.Invalid_Input__Must_be_IFileEditorInput"));
				//$NON-NLS-1$
		super.init(site, editorInput);
	}

	/* (non-Javadoc)
	 * Method declared on IEditorPart.
	 */
	public boolean isSaveAsAllowed() {
		return true;
	}

	/**
	 * Updates the preview or deletes the temporary file.
	 * @see org.eclipse.ui.part.MultiPageEditorPart#pageChange(int)
	 */
	protected void pageChange(int newPageIndex) {
		super.pageChange(newPageIndex);
		if (newPageIndex != 0) {
			updatePreviews();

		} else {
			if (tempFile != null && tempFile.exists()) {
				try {
					tempFile.delete(true, null);
				} catch (CoreException e) {
					showException(e);
				}
			}
		}
	}

	/**
	 * Updates the preview-page. Generates a temporary
	 * file if the editor is dirty.
	 */
	private void updatePreviews() {
		String url =
			((IFileEditorInput) editor.getEditorInput())
				.getFile()
				.getLocation()
				.toOSString();

		if (editor.isDirty()) {
			try {
				final IWorkspaceRoot myWorkspaceRoot =
					ResourcesPlugin.getWorkspace().getRoot();
				final IFile actualFile =
```

```
                        ((IFileEditorInput) editor.getEditorInput()).
                              getFile();
                  final IProject actualProject = actualFile.getProject();
                  final IPath pathToFile =
                        actualFile.getLocation().removeLastSegments(1);
                  // open if necessary
                  if (actualProject.exists() && !actualProject.isOpen())
                        actualProject.open(null);

                  final String actualContent =
                        editor
                              .getDocumentProvider()
                              .getDocument(editor.getEditorInput())
                              .get();
                  final StringBufferInputStream inputStream =
                        new StringBufferInputStream(actualContent);
                  final String folderPath =
                        getFolderPath(actualProject.getLocation(),
                              pathToFile);
                  tempFile = actualProject.getFile(folderPath+"tmp_" +
                        getTitle()); //$NON-NLS-1$
                  if (tempFile.exists()) {
                        tempFile.delete(true, null);
                  }
                  tempFile.create(inputStream, false, null);
                  // create closes the file stream, so no worries.
                  url = tempFile.getLocation().toOSString();
            } catch (CoreException e) {
                  showException(e);
            }
      }
      Iterator iter = previews.iterator();
      while (iter.hasNext()) {
            IHTMLPreview element = (IHTMLPreview) iter.next();
            element.updatePreview(url);
      }

}

/**
 * @param path the path of the actual project
 * @param pathToFile the path of the actual file
 * @return Sting relative Path to the file from the workspaceRoot
 */
private String getFolderPath(IPath projectPath, IPath pathToFile) {
      String path =
            pathToFile
                  .removeFirstSegments(
                        pathToFile.matchingFirstSegments(projectPath))
                  .toOSString();
      //remove device
      if (path.indexOf(":") >= 0) { //$NON-NLS-1$
            path = path.substring(path.indexOf(":") + 1, //$NON-NLS-1$
            path.length());
      }
      if (path.length()>0){
            path += System.getProperty("file.separator");
      }
      return path;
}

/* (non-Javadoc)
 * @see org.eclipse.ui.IWorkbenchPart#dispose()
 */
public void dispose() {
      if (tempFile != null && tempFile.exists()) {
            try {
```

```
                        tempFile.delete(true, null);
                    } catch (CoreException e) {
                        showException(e);
                    }
                }
        }

        /**
         * Returns the HTMLContentOutlinePage of the HTMLEditor
         */
        public Object getAdapter(Class required) {
                return editor.getAdapter(required);
        }

        private void showException(Exception e) {
                MessageDialog.openError(new Shell(), e.getMessage(),
                        e.getMessage());
        }

        /**
         * Returns the HTMLEditor.
         * @return HTMLEditor the HTMLEditor
         */
        public HTMLEditor getHTMLEditor() {
                return editor;
        }
}
```

D.III.13 MultiPageEditorContributor.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.Separator;
import org.eclipse.ui.IActionBars;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.IWorkbenchActionConstants;
import org.eclipse.ui.part.MultiPageEditorActionBarContributor;
import org.eclipse.ui.texteditor.ITextEditor;
import org.eclipse.ui.texteditor.ITextEditorActionConstants;
import org.eclipse.ui.texteditor.ITextEditorActionDefinitionIds;
import org.eclipse.ui.texteditor.RetargetTextEditorAction;

/**
 * Manages the installation/deinstallation of global actions for
 * multi-page editors.
 * Responsible for the redirection of global actions to the active editor.
 * Multi-page contributor replaces the contributors for the
 * individual editors in the multi-page editor.
 */
public class MultiPageEditorContributor
      extends MultiPageEditorActionBarContributor {
      private IEditorPart activeEditorPart;
      private final RetargetTextEditorAction contentAssistProposal;
      protected final RetargetTextEditorAction contentFormatProposal;

      /**
       * Creates a multi-page contributor.
       */
      public MultiPageEditorContributor() {
            super();
            contentAssistProposal = new RetargetTextEditorAction
                  (HTMLEditorMessages.getResourceBundle(),
                  "ContentAssistProposal."); //$NON-NLS-1$
            contentAssistProposal.setActionDefinitionId(
                  ITextEditorActionDefinitionIds.CONTENT_ASSIST_PROPOSALS);
            contentFormatProposal =
                  new RetargetTextEditorAction(
                        HTMLEditorMessages.getResourceBundle(),
                        "ContentFormatProposal."); //$NON-NLS-1$
      }

      /**
       * Returns the action registed with the given text editor.
       * @return IAction or null if editor is null.
       */
      protected IAction getAction(ITextEditor editor, String actionID) {
            return (editor == null ? null : editor.getAction(actionID));
      }

      /* (non-JavaDoc)
       * Method declared in AbstractMultiPageEditorActionBarContributor.
       */
      public void setActivePage(IEditorPart part) {
            if (activeEditorPart == part)
                  return;

            activeEditorPart = part;

            final IActionBars actionBars = getActionBars();
            if (actionBars != null) {

                  ITextEditor editor =
                        (part instanceof ITextEditor) ?
```

```
                        (ITextEditor) part : null;

                if (editor != null) {
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.DELETE,
                                getAction(editor,
                                        ITextEditorActionConstants.DELETE));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.UNDO,
                                getAction(editor,
                                        ITextEditorActionConstants.UNDO));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.REDO,
                                getAction(editor,
                                        ITextEditorActionConstants.REDO));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.CUT,
                                getAction(editor,
                                        ITextEditorActionConstants.CUT));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.COPY,
                                getAction(editor,
                                        ITextEditorActionConstants.COPY));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.PASTE,
                                getAction(editor,
                                        ITextEditorActionConstants.PASTE));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.SELECT_ALL,
                                getAction(editor,
                                        ITextEditorActionConstants.SELECT_ALL));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.FIND,
                                getAction(editor,
                                        ITextEditorActionConstants.FIND));
                        actionBars.setGlobalActionHandler(
                                IWorkbenchActionConstants.BOOKMARK,
                                getAction(editor,
                                        ITextEditorActionConstants.BOOKMARK));

                //die action für den menupunkt setzen,
                //sonst wird er zwar angezeigt, macht aber nichts
                        contentAssistProposal.setAction(getAction(editor,
                                "ContentAssistProposal")); //$NON-NLS-1$
                        contentFormatProposal.setAction(
                                getAction(editor, "ContentFormatProposal"));
                                        //$NON-NLS-1$
                        contentAssistProposal.setEnabled(true);
                        contentFormatProposal.setEnabled(true);

                        actionBars.updateActionBars();
                } else { //IE-Tab
                        contentAssistProposal.setEnabled(false);
                        contentFormatProposal.setEnabled(false);
                }
        }
}

/*
 * @see IEditorActionBarContributor#init(IActionBars)
 */
public void init(IActionBars bars) {
        super.init(bars);

        IMenuManager menuManager = bars.getMenuManager();
        IMenuManager editMenu =
                menuManager.findMenuUsingPath
                        (IWorkbenchActionConstants.M_EDIT);
```

```
            if (editMenu != null) {
                editMenu.add(new Separator());
                //fügt die beiden Actions dem edit-menu hinzu
                editMenu.add(contentAssistProposal);
                editMenu.add(contentFormatProposal);
            }
        }

}
```

D.III.14 TagRule.java

```
package de.kt.scripteditor.htmleditor;

import org.eclipse.jface.text.rules.ICharacterScanner;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;

/**
 * A MultiLineRule to detect xml/html-tags but not <script>.
 * @author Karsten Thiemann
 */
public class TagRule extends MultiLineRule {

	public TagRule(IToken token) {
	//Problem am Dateiende: wird neues Tag geöffnet,
	//so kann es noch nicht als solches erkannt werden
		super("<", ">", token); //$NON-NLS-1$ //$NON-NLS-2$
	}

	protected boolean sequenceDetected(
			ICharacterScanner scanner,
			char[] sequence,
			boolean eofAllowed) {
			int c = scanner.read();
			if (sequence[0] == '<') {
				if (c == '?') {
					// processing instruction - abort
					scanner.unread();
					return false;
				}
				if (c == '!') {
					scanner.unread();
					// comment - abort
					return false;
				}
				if (c == 's' || c == 'S') {
					c = scanner.read();
					if (c == 'c' || c == 'C') {
						scanner.unread();
						// script-tag - abort
						scanner.unread();
						return false;
					}
					scanner.unread();
				}
				if (c == '/') {
					c = scanner.read();
					if (c == 's' || c == 'S') {
						c = scanner.read();
						if (c == 'c' || c == 'C') {
							scanner.unread();
							// script-tag - abort
							scanner.unread();
							scanner.unread();
							return false;
						}
						scanner.unread();
					}
					scanner.unread();
				}
			} else if (sequence[0] == '>') {
				scanner.unread();
			}
			return super.sequenceDetected(scanner, sequence, eofAllowed);
	}
}
```

D.III.15 HTMLEditorMessages.properties

```
# Actions

ContentAssistProposal.label=Content Assist@Ctrl+SPACE
ContentAssistProposal.tooltip=Content Assist
ContentAssistProposal.description=Content Assist
ContentAssistProposal.image=

ContentFormatProposal.label=Format Tags

HTMLCreationWizard.New_HTML_File=New HTML File

HTMLCreationPage.HTML_file_name=Insert the name
HTMLCreationPage.Create_HTML_File=Create a new HTML-File
HTMLCreationPage.Create_HTML_File_description= This creates a new HTML-File
HTMLCreationPage.HTML_Settings=HTML Settings
HTMLCreationPage.html_strict=<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">\n
HTMLCreationPage.html_transitional=<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">\n
HTMLCreationPage.html_frameset=<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">\n
HTMLCreationPage.HTML_PRE_TITLE=<html>\n\n<head>\n\t<title>
HTMLCreationPage.HTML_CLOSE_TITLE=</title>\n
HTMLCreationPage.HTML_POST_TITLE=</head>\n\n<body>\n\n</body>\n</html>
HTMLCreationPage.Select_CSS_File=Select CSS File

MultiPageEditor.Source-View=Source-View
MultiPageEditor.Error_creating_nested_text_editor=Error creating nested text
editor
MultiPageEditor.IE-Preview=IE-Preview
MultiPageEditor.Invalid_Input__Must_be_IFileEditorInput=Invalid Input: Must be
IFileEditorInput

ExternalizeScriptAction.wrong_directory=wrong directory
ExternalizeScriptAction.Select_file_in_the_actual_project=\tPlease select a file
in the actual project\!

HTMLContentOutlinePage.Filter_Tags=Filter Tags
HTMLContentOutlinePage.Show_All_Tags=Show All Tags
```

D.III.16 HTMLEditorMessages_de_DE.properties

```
# Actions

ContentAssistProposal.label=Content Assist@Ctrl+SPACE
ContentAssistProposal.tooltip=Content Assist
ContentAssistProposal.description=Content Assist
ContentAssistProposal.image=

ContentFormatProposal.label=Formatiere Tags

HTMLCreationWizard.New_HTML_File=Neues HTML-Dokument

HTMLCreationPage.HTML_file_name=Namen eingeben
HTMLCreationPage.Create_HTML_File=Erzeugt eine neues HTML-Dokument
HTMLCreationPage.Create_HTML_File_description= Dieser Wizard erzeugt eine neues
HTML-Dokument
HTMLCreationPage.HTML_Settings=HTML Einstellungen
HTMLCreationPage.html_strict=<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">\n
HTMLCreationPage.html_transitional=<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">\n
HTMLCreationPage.html_frameset=<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">\n
HTMLCreationPage.HTML_PRE_TITLE=<html>\n\n<head>\n\t<title>
HTMLCreationPage.HTML_CLOSE_TITLE=</title>\n
HTMLCreationPage.HTML_POST_TITLE=</head>\n\n<body>\n\n</body>\n</html>
HTMLCreationPage.Select_CSS_File=Wählen Sie ein CSS-Dokument

MultiPageEditor.Source-View=Quelltext
MultiPageEditor.Error_creating_nested_text_editor=Fehler bei der Erzeugung des
eingebetteten Editors
MultiPageEditor.IE-Preview=IE-Vorschau
MultiPageEditor.Invalid_Input__Must_be_IFileEditorInput=Invalid Input: Must be
IFileEditorInput

ExternalizeScriptAction.wrong_directory=Falsches Verzeichnis
ExternalizeScriptAction.Select_file_in_the_actual_project=\tBitte Datei aus dem
aktuellen Projekt wählen\!

HTMLContentOutlinePage.Filter_Tags=Tags filtern
HTMLContentOutlinePage.Show_All_Tags=Zeige alle Tags
```

D.IV de.kt.scripteditor.htmleditor.actions

D.IV.1 AbstractFileAction.java

```
package de.kt.scripteditor.htmleditor.actions;

import org.eclipse.core.runtime.IPath;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.widgets.Shell;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.preferences.IPreferenceIDs;

/**
 * This class is used as superclass for all actions that deal
 * with relative pathes.
 * @author Karsten Thiemann
 */
public abstract class AbstractFileAction implements IPreferenceIDs{

	/**
	 * Shows the given exception in a popup-dialog.
	 * @param e the exception to show
	 */
	protected void showException(Exception e) {
		MessageDialog.openError(new Shell(), e.getMessage(),
			e.getMessage());
	}

/**
 * Creates a relative path from the actualFilePath to the externalFilePath.
 * @param externalFilePath the path of the external file (e.g. an image to link)
 * @param actualFilePath the path of the actual file
 * (e.g. the file opened in the editor)
 * @return String the relative path from the actualFilePath
 * to the externalFilePath
 */
	protected String createRelativePath(IPath externalFilePath, IPath
		actualFilePath) {

		if (externalFilePath.matchingFirstSegments(actualFilePath)
			== actualFilePath.segmentCount() - 1) {
				final IPath tmpFilePath =
				externalFilePath.removeFirstSegments(
					externalFilePath.matchingFirstSegments
						(actualFilePath));
			String relativePath = tmpFilePath.toOSString();
			//remove device
			if (relativePath.indexOf(":") >= 0) { //$NON-NLS-1$
				relativePath =
					relativePath.substring(
						relativePath.indexOf(":") + 1, //$NON-NLS-1$
						relativePath.length());
			}
			return relativePath;
		} else {
			//Oberverzeichnis
			final IPath tmpFilePath =
				actualFilePath.removeFirstSegments(
					actualFilePath.matchingFirstSegments
						(externalFilePath));
			externalFilePath =
				externalFilePath.removeFirstSegments(
					externalFilePath.matchingFirstSegments
						(actualFilePath));
			String prefix = ""; //$NON-NLS-1$
```

```
                for (int i = 0; i < tmpFilePath.segmentCount() - 1; i++) {
                      prefix += "..\\"; //$NON-NLS-1$
                }
                String relativePath = externalFilePath.toOSString();
                //remove device
                if (relativePath.indexOf(":") >= 0) { //$NON-NLS-1$
                      relativePath =
                            relativePath.substring(
                                  relativePath.indexOf(":") + 1, //$NON-NLS-1$
                                  relativePath.length());
                }
                return prefix + relativePath;
          }
    }

    /**
     * @return boolean the user-preference for uppercase of HTML-tags.
     */
    protected boolean isUpperCaseTags(){
          return ScripteditorPlugin.getDefault().getPreferenceStore().
                getBoolean(TAGS_TO_UPPERCASE);
    }
}
```

D.IV.2 AppendCSSReferenceAction.java

```
package de.kt.scripteditor.htmleditor.actions;

import java.io.InputStream;
import java.io.StringBufferInputStream;
import java.util.Iterator;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.runtime.IPath;
import org.eclipse.core.runtime.Path;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.StructuredSelection;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.FileDialog;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.IFileEditorInput;
import org.eclipse.ui.IObjectActionDelegate;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.ui.IWorkbenchPart;

import de.kt.scripteditor.htmleditor.HTMLEditor;
import de.kt.scripteditor.htmleditor.MultiPageEditor;

/**
 * This Actiondelegate appends a relative link to a choosen CSS-file to
 * the HTML-file. The link is inserted into the HTML-files head.
 * @author Karsten Thiemann
 */
public class AppendCSSReferenceAction
      extends AbstractFileAction
      implements IObjectActionDelegate {

      private IWorkbenchPart part;
      private StructuredSelection selection;

      /* (non-Javadoc)
       * @see org.eclipse.ui.IActionDelegate#run
       * (org.eclipse.jface.action.IAction)
       */
      public void run(IAction action) {
            final FileDialog dialog = new FileDialog(new Shell(), SWT.OPEN);
            dialog.open();
            String filename = dialog.getFileName();
            if (filename != null && !filename.equalsIgnoreCase("")) {
                  //$NON-NLS-1$
                  IPath cssPath = new Path(dialog.getFilterPath() + "\\"
                        + filename); //$NON-NLS-1$
                  Iterator iter = selection.iterator();
                  while (iter.hasNext()) {
                        final Object element = (Object) iter.next();
                        if (element instanceof IFile) {

//Zur besseren Lesbarkeit nach links gerückt
try {
      final Vector positions = new Vector();
      final IFile file = (IFile) element;
      final IProject actualProject = file.getProject();
      final IPath filePath = file.getLocation();
```

```
//relativen Pfad bauen...
filename = createRelativePath(cssPath, filePath);
String content = ""; //$NON-NLS-1$
StringBuffer fileText = new StringBuffer();
IDocument document = null;
HTMLEditor editor = null;
boolean isDirtyFile = false;

//test for dirty editor
if (part != null) {
      final IWorkbenchPage[] pages = part.getSite().
      getWorkbenchWindow().getPages();
      for (int i = 0; i < pages.length; i++) {
            final IWorkbenchPage page = pages[i];
            final IEditorPart[] parts = page.getDirtyEditors();
            for (int j = 0; j < parts.length; j++) {
                  final IEditorPart part = parts[j];
                  if (part instanceof MultiPageEditor) {
                        editor = ((MultiPageEditor)part).getHTMLEditor();
                        editor.getDocumentProvider().getDocument(null);
                        final IEditorInput input = part.getEditorInput();
                        if (input instanceof IFileEditorInput) {
                              final IFileEditorInput fileEditorInput =
                                    (IFileEditorInput) input;
                              final IFile dirtyFile =
                                    fileEditorInput.getFile();
                              if (dirtyFile.equals(file)) {
                                    //take actual content from document
                                    document =
                                          editor.getDocumentProvider().
                                                getDocument(input);
                                    content = document.get();
                                    fileText.append(content);
                                    isDirtyFile = true;
                              }
                        }
                  }
            }
      }
}
//if no dirty editor was found, take content from file
if (!isDirtyFile) {
      final InputStream inputstream =file.getContents(true);
      int c = 0;
      while ((c = inputstream.read()) != -1) {
            fileText.append((char) c);
      }
      content = fileText.toString();
}
Pattern pattern = Pattern.compile(
      "<link\\s+rel=(.*)?type=(.*)?text/css(.*)?href=(.*)?>" );
            //$NON-NLS-1$
Matcher matcher = pattern.matcher(content.toLowerCase());

while (matcher.find()) {
      positions.add(new CSSPosition(matcher.start(), matcher.end()));
}

//alle alten CSS-Verweise entfernen
for (int i = positions.size() - 1; i >= 0; i--) {
      final CSSPosition position = (CSSPosition) positions.elementAt(i);
      if (position != null) {
            fileText = fileText.delete( position.start, position.end);
      }
}
//neuen Verweis im head einfügen
pattern = Pattern.compile("</head>"); //$NON-NLS-1$
```

```
		matcher = pattern.matcher(fileText.toString().toLowerCase());
		int insertPosition = 0;
		while (matcher.find()) {
			insertPosition = matcher.start();
		}
		final String link =
			(isUpperCaseTags()) ? "LINK" : "link"; //$NON-NLS-1$ //$NON-NLS-2$
		fileText.insert(insertPosition,"\t<" //$NON-NLS-1$
			+ link
			+ " rel=\"stylesheet\" Type=\"text/css\" href=\"" //$NON-NLS-1$
			+ filename
			+ "\">\n"); //$NON-NLS-1$

		//update document and editor
		if (isDirtyFile && document != null) {
			document.set(fileText.toString());
			document.setDocumentPartitioner(
			document.getDocumentPartitioner());
			if (editor != null) {
				editor.setHighlightRange(insertPosition, 0, true);
			}
		} else {
			//write to file
			final StringBufferInputStream bufferInputStream =
				new StringBufferInputStream(fileText.toString());
			file.setContents(bufferInputStream, true, true, null);
		}
	} catch (Exception e) {
		showException(e);
	}

				}
			}
		}
	}

	/**
	 * @see org.eclipse.ui.IActionDelegate#selectionChanged(IAction,
	 * ISelection)
	 */
	public void selectionChanged(IAction action, ISelection selection) {
		if (selection instanceof StructuredSelection) {
			this.selection = (StructuredSelection) selection;
		}
	}

	/* (non-Javadoc)
	 * @see org.eclipse.ui.IObjectActionDelegate#setActivePart
	 * (org.eclipse.jface.action.IAction, org.eclipse.ui.IWorkbenchPart)
	 */
	public void setActivePart(IAction action, IWorkbenchPart targetPart) {
		this.part = targetPart;
	}

	/**
	 * Defines the position of a CSS-link-tag in the document.
	 * @author Karsten
	 */
	protected class CSSPosition {
		protected int start;
		protected int end;
		/**
		 * Constructor for CSSPosition.
		 * @param start offset of the tags start
		 * @param end offset of the tags end
		 */
		public CSSPosition(int start, int end) {
```

```
                this.start = start;
                this.end = end;
            }
        }

}
```

D.IV.3 ExternalizeScriptAction.java

```
package de.kt.scripteditor.htmleditor.actions;

import java.io.StringBufferInputStream;
import java.util.Vector;

import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IWorkspaceRoot;
import org.eclipse.core.resources.ResourcesPlugin;
import org.eclipse.core.runtime.IPath;
import org.eclipse.core.runtime.Path;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.RuleBasedScanner;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.FileDialog;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IEditorActionDelegate;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.IFileEditorInput;
import org.eclipse.ui.part.FileEditorInput;

import de.kt.scripteditor.htmleditor.HTMLEditor;
import de.kt.scripteditor.htmleditor.HTMLEditorEnvironment;
import de.kt.scripteditor.htmleditor.HTMLEditorMessages;
import de.kt.scripteditor.htmleditor.outline.HTMLOutlineTagScanner;

/**
 * The ExternalizeScriptAction asks the user for a filename,
 * analyses the html-file and writes the script-parts to the given file.
 * It also sets a script-reference to the (new) script-file. If a file with
 * the given name exists, the content is
 * appended.
 * @author Karsten Thiemann
 */
public class ExternalizeScriptAction
      extends AbstractFileAction
      implements IEditorActionDelegate {

      private HTMLEditor editor;

      /**
       * Saves a reference to the current active editor
       * @see org.eclipse.ui.IEditorActionDelegate#setActiveEditor(IAction,
       * IEditorPart)
       */
      public void setActiveEditor(IAction action, IEditorPart targetEditor) {
            editor = (HTMLEditor) targetEditor;
      }

      /* (non-Javadoc)
       * @see org.eclipse.ui.IActionDelegate#run
       * (org.eclipse.jface.action.IAction)
       */
      public void run(IAction action) {
            final FileDialog dialog = new FileDialog(new Shell(), SWT.OPEN);
            dialog.open();
            final String filename = dialog.getFileName();

            if (filename != null && !filename.equalsIgnoreCase("")) {
                  //$NON-NLS-1$
                  IPath scriptPath =
```

```
                        new Path(dialog.getFilterPath() + "\\" + filename);
                              //$NON-NLS-1$
      //Testen ob gewählter Ordner ein Unterordner des aktuellen Projektes ist
                  final IWorkspaceRoot myWorkspaceRoot =
                        ResourcesPlugin.getWorkspace().getRoot();
                  //öffnet das aktuelle Projekt!
                  final IProject actualProject =
                        ((IFileEditorInput) editor.getEditorInput())
                              .getFile()
                              .getProject();

                  final int matchingSegments =
                        actualProject.getLocation()
                              .matchingFirstSegments(scriptPath);
                  if (matchingSegments
                        != actualProject.getLocation().segmentCount()) {
                        MessageDialog.openError(
                              new Shell(),
                              HTMLEditorMessages.getString
                                    ("ExternalizeScriptAction.wrong_directory"),
                                          //$NON-NLS-1$
                              HTMLEditorMessages.getString
            ("ExternalizeScriptAction.Select_file_in_the_actual_project"));
                                          //$NON-NLS-1$
                        return;
                  }
                  final String relativeScriptPath =
                        createRelativePath(
                              scriptPath,
                              ((FileEditorInput) editor.getEditorInput())
                                    .getFile()
                                    .getLocation());
                  scriptPath =
                        scriptPath.removeFirstSegments(matchingSegments)
                              .makeRelative();

                  final IDocument document =
                        editor.getDocumentProvider().getDocument(
                              editor.getEditorInput());
                  final RuleBasedScanner scanner =
                        HTMLEditorEnvironment.getHTMLOutlineTagScanner();
                  scanner.setRange(document, 0, document.getLength());

//Zur besseren Lesbarkeit nach links gerückt
try {
      final StringBuffer scriptcontent = new StringBuffer();
      boolean openScriptTagFound = false;
      int scriptContentStartPosition = 0;
      int scriptTagStartPosition = 0;
      int insertPosition = 0;
      final Vector positions = new Vector();
      IToken token = scanner.nextToken();
      while (!token.isEOF()) {
            if (token
                  .equals(HTMLOutlineTagScanner.OUTLINE_OPEN_TAG_TOKEN)) {

                  String tagname = document.get(scanner.getTokenOffset(),
                        scanner.getTokenLength()).toUpperCase();
            // if tag is a script-tag, but not a link to an external script
                  if (tagname.startsWith("<SCRIPT") //$NON-NLS-1$
                        && (tagname.indexOf("SRC") == -1 //$NON-NLS-1$
                        || tagname.indexOf("SRC=\"" //$NON-NLS-1$
                              + relativeScriptPath.toUpperCase()
                              + "\"") //$NON-NLS-1$
                              != -1
                        || tagname.indexOf("SRC='" //$NON-NLS-1$
                              + relativeScriptPath.toUpperCase()
                              + "'") //$NON-NLS-1$
```

```
                        != -1)) {
                  openScriptTagFound = true;
                  scriptTagStartPosition = scanner.getTokenOffset();
                  scriptContentStartPosition =
                        scanner.getTokenOffset()
                        + scanner.getTokenLength();
            } else if (tagname.toUpperCase().startsWith("<HEAD")) {
                  //$NON-NLS-1$
                  insertPosition = scanner.getTokenOffset()
                        + scanner.getTokenLength();
            }
      } else if (openScriptTagFound && token.equals(HTMLOutlineTagScanner
                  .OUTLINE_CLOSE_TAG_TOKEN)) {
            String tagname = document.get(scanner.getTokenOffset(),
                              scanner.getTokenLength())
                              .toUpperCase();
            if (tagname.startsWith("</SCRIPT")) { //$NON-NLS-1$
                  scriptcontent.append(document.get(
                        scriptContentStartPosition,
                        scanner.getTokenOffset()
                        - scriptContentStartPosition));
                  positions.add(new TagPosition(scriptTagStartPosition,
                        (scanner.getTokenOffset()
                        + scanner.getTokenLength()
                        - scriptTagStartPosition)));
                  openScriptTagFound = false;
            }
      }
      token = scanner.nextToken();
}
if (scriptcontent.length() > 0) {
      //scriptteile aus document entfernen, beginnend vom Dokumentenende
      for (int i = positions.size() - 1; i >= 0; i--) {
            final TagPosition position = (TagPosition) positions
                  .elementAt(i);
            if (position != null) {
                  document.replace(position.offset, position.length, "");
                        //$NON-NLS-1$
            }
      }
      //Verweis auf externe Scriptdatei setzen
      final String script = (isUpperCaseTags()) ? "SCRIPT" : "script";
            //$NON-NLS-1$ //$NON-NLS-2$
      final StringBuffer buffer = new StringBuffer();
      buffer.append(document.get(0, insertPosition));
      buffer.append("<"); //$NON-NLS-1$
      buffer.append(script);
      buffer.append(" src=\""); //$NON-NLS-1$
      buffer.append(relativeScriptPath);
      buffer.append("\" type=\"text/javascript\"></"); //$NON-NLS-1$
      buffer.append(script);
      buffer.append(">"); //$NON-NLS-1$
      buffer.append(document.get(insertPosition,
            document.getLength() - insertPosition));
      document.set(buffer.toString());
      //Refresh des Editors! WORKAROUND!
      document.setDocumentPartitioner(document.getDocumentPartitioner());
      //reset cursor
      editor.setHighlightRange(insertPosition, 0, true);

      // open project if necessary
      if (actualProject.exists() && !actualProject.isOpen()) {
            actualProject.open(null);
      }
      //write external file
      final StringBufferInputStream inputStream =
            new StringBufferInputStream(scriptcontent.toString());
      final IFile scriptfile = actualProject.getFile(scriptPath);
```

```
                if (scriptfile.exists()) {
                        //neue Scriptteile anhängen!
                        scriptfile.appendContents(inputStream, true, true, null);
                } else {
                        scriptfile.create(inputStream, false, null);
                }
        }
} catch (Exception e) {
        showException(e);
}
                }
        }

        /**
         * @see org.eclipse.ui.IActionDelegate#selectionChanged(IAction,
         * ISelection)
         */
        public void selectionChanged(IAction action, ISelection selection) {
        }

        /**
         * Defines the position of a tag in the document.
         * @author Karsten Thiemann
         */
        protected class TagPosition {
                protected int offset;
                protected int length;
                /**
                 * Constructor for TagPosition.
                 * @param offset the offset of the tags start
                 * @param length the length of the tag
                 */
                public TagPosition(int offset, int length) {
                        this.offset = offset;
                        this.length = length;
                }
        }
}
```

D.IV.4 HTMLTidy.java

```
package de.kt.scripteditor.htmleditor.actions;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringBufferInputStream;

import org.w3c.tidy.Tidy;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.preferences.IPreferenceIDs;


/**
 * HTMLTidy uses org.w3c.tidy.Tidy to clean the given (html)string.
 * @author Karsten Thiemann
 */
public class HTMLTidy implements IPreferenceIDs{

	/**
	 * Returns the tidied html-string.
	 * @param content the content to tidy
	 * @return String the tidied html-string
	 * @throws IOException is thrown if parsing errors occure
	 */
	public String tidy(String content) throws IOException {
		final Tidy tidy = new Tidy();
		tidy.setSmartIndent(true);
		tidy.setUpperCaseTags(ScripteditorPlugin.getDefault()
				.getPreferenceStore().getBoolean(TAGS_TO_UPPERCASE));
		final InputStream in = new StringBufferInputStream(content);
		final ByteArrayOutputStream out = new ByteArrayOutputStream();
		tidy.parse(in, out);
		byte[] byteArray = out.toByteArray();
		if (tidy.getParseErrors() > 0) {
			throw new IOException("HTMLTidy: Parsing Errors");
				//$NON-NLS-1$
		}
		return new String(byteArray);
	}
}
```

D.IV.5 InsertImageAction.java

```
package de.kt.scripteditor.htmleditor.actions;

import org.eclipse.core.runtime.IPath;
import org.eclipse.core.runtime.Path;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.FileDialog;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IEditorActionDelegate;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.part.FileEditorInput;

import de.kt.scripteditor.htmleditor.HTMLEditor;
import de.kt.scripteditor.htmleditor.MultiPageEditor;

/**
 * The InsertImageAction asks the user for a imagefile, and inserts a img-tag
 * with a relative path to the given file at the currant cursor-position.
 * @author Karsten Thiemann
 */
public class InsertImageAction
      extends AbstractFileAction
      implements IEditorActionDelegate {
      private HTMLEditor editor;

      /**
       * Saves a reference to the current active editor
       * @see org.eclipse.ui.IEditorActionDelegate#setActiveEditor(IAction,
       * IEditorPart)
       */
      public void setActiveEditor(IAction action, IEditorPart targetEditor) {
            if (targetEditor instanceof HTMLEditor) {
                  editor = (HTMLEditor) targetEditor;
            } else if(targetEditor instanceof MultiPageEditor){
                  editor = ((MultiPageEditor)targetEditor).getHTMLEditor();
            }
      }

      /* (non-Javadoc)
       * @see org.eclipse.ui.IActionDelegate#run
       * (org.eclipse.jface.action.IAction)
       */
      public void run(IAction action) {
            final FileDialog dialog = new FileDialog(new Shell(), SWT.OPEN);
            dialog.open();
            final String filename = dialog.getFileName();
            if (filename != null && !filename.equalsIgnoreCase("")) {
                  //$NON-NLS-1$
                  try {
                        final IPath scriptPath = new Path(dialog
                              .getFilterPath() + "\\" + filename); //$NON-NLS-1$
                        final String relativeImagePath =
                              createRelativePath(
                                    scriptPath,
                                    ((FileEditorInput) editor.getEditorInput())
                                          .getFile()
                                          .getLocation());
                        final IDocument document =
                              editor.getDocumentProvider().getDocument(
                                    editor.getEditorInput());

                        //Image-Tag einfügen
```

```
                        final String img = (isUpperCaseTags()) ? "IMG" : "img";
                              //$NON-NLS-1$ //$NON-NLS-2$
                        final StringBuffer buffer = new StringBuffer(100);
                        final int cursorOffset = editor.getCursorOffset();
                        buffer.append(document.get(0, cursorOffset));
                        buffer.append("<"); //$NON-NLS-1$
                        buffer.append(img);
                        buffer.append(" src=\""); //$NON-NLS-1$
                        buffer.append(relativeImagePath);
                        buffer.append("\" border=\"0\">"); //$NON-NLS-1$
                        buffer.append(
                              document.get(
                                    cursorOffset,
                                    document.getLength() - cursorOffset));
                        document.replace(0, document.getLength(),
                              buffer.toString());

                        //refresh des Editors! WORKAROUND!
                        document.setDocumentPartitioner(
                              document.getDocumentPartitioner());
                        editor.setHighlightRange(cursorOffset, 0, true);
                  } catch (BadLocationException e) {
                        showException(e);
                  }
            }
      }

      /**
       * @see org.eclipse.ui.IActionDelegate#selectionChanged(IAction,
       * ISelection)
       */
      public void selectionChanged(IAction action, ISelection selection) {
      }

}
```

D.IV.6 TidyAction.java

```
package de.kt.scripteditor.htmleditor.actions;

import java.io.IOException;

import org.eclipse.jface.action.Action;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.IDocument;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.IFileEditorInput;

import de.kt.scripteditor.htmleditor.HTMLEditor;

/**
 * This action is used to tidy the html document.
 * @author Karsten Thiemann
 */
public class TidyAction extends Action {
	public HTMLEditor editor;

	public TidyAction() {
		super();
		this.setText("&Tidy"); //$NON-NLS-1$
		this.setId("#Tidy"); //$NON-NLS-1$
	}

	/**
	 * @see org.eclipse.jface.action.IAction#run()
	 */
	public void run() {
		final IEditorInput input = editor.getEditorInput();

		if (input instanceof IFileEditorInput) {
			final IFileEditorInput fileEditorInput = (IFileEditorInput)
				input;
			final IDocument document =
				editor.getDocumentProvider().getDocument(
					fileEditorInput);
			final String oldText = document.get();
			final HTMLTidy tidy = new HTMLTidy();
			try {
				int oldOffset = editor.getCursorOffset();
				document.replace(0,oldText.length(),tidy.tidy(oldText));

				//refresh des Editors! WORKAROUND!
				document.setDocumentPartitioner
					(document.getDocumentPartitioner());
				editor.setHighlightRange(oldOffset,0,true);
			} catch (IOException e) {
				showException(e);
			} catch (BadLocationException e) {
				showException(e);
			}
		}
	}

	/**
	 * Sets the active HTMLEditor.
	 * @param editor the HTMLEditor
	 */
	public void setActiveEditor(HTMLEditor editor) {
		this.editor = editor;
	}
```

```
	private void showException(Exception e) {
		MessageDialog.openError(new Shell(),
			e.getMessage(), e.getMessage());
	}
}
```

D.V de.kt.scripteditor.htmleditor.outline

D.V.1 HTMLContentOutlinePage.java

```
package de.kt.scripteditor.htmleditor.outline;

import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.BadPositionCategoryException;
import org.eclipse.jface.text.DefaultPositionUpdater;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.IDocumentListener;
import org.eclipse.jface.text.IPositionUpdater;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.RuleBasedScanner;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.viewers.ITreeContentProvider;
import org.eclipse.jface.viewers.SelectionChangedEvent;
import org.eclipse.jface.viewers.TreeViewer;
import org.eclipse.jface.viewers.Viewer;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.ui.help.WorkbenchHelp;
import org.eclipse.ui.texteditor.IDocumentProvider;
import org.eclipse.ui.texteditor.ITextEditor;
import org.eclipse.ui.views.contentoutline.ContentOutlinePage;

import de.kt.scripteditor.htmleditor.HTMLEditorEnvironment;
import de.kt.scripteditor.htmleditor.HTMLEditorMessages;
import de.kt.scripteditor.util.AbstractScriptContentProvider;
import de.kt.scripteditor.util.IScriptContentProvider;
import de.kt.scripteditor.util.IScriptEditorOutlinePage;
import de.kt.scripteditor.util.JsFunctionScanner;
import de.kt.scripteditor.util.OutlineDocumentListener;

/**
 * The outline page of the HTMLEditor.
 * @author Karsten Thiemann
 */
public class HTMLContentOutlinePage
      extends ContentOutlinePage
      implements IScriptEditorOutlinePage {

      /**
       * Parses the editors input.
       */
      protected class ContentProvider extends AbstractScriptContentProvider
            implements ITreeContentProvider, IScriptContentProvider {

            protected final static String JS_FUNCTIONS =
                  "__javascript_functions"; //$NON-NLS-1$
            protected final IPositionUpdater positionUpdater =
                  new DefaultPositionUpdater(JS_FUNCTIONS);
            protected TagElement tagElement = new TagElement("ROOT", null, 0, 0,
                  TagElement.HTML_TAG); //$NON-NLS-1$

            /* (non-Javadoc)
             * @see de.kt.jseditor.util.IScriptContentProvider#parse
             * (org.eclipse.jface.text.IDocument)
             */
            public void parse(IDocument document) {
                  RuleBasedScanner scanner =
                        HTMLEditorEnvironment.getHTMLOutlineTagScanner();
                  scanner.setRange(document, 0, document.getLength());
```

```
                IToken token = scanner.nextToken();

//Zur besseren Lesbarkeit nach links gerückt
try {
      TagElement parent = tagElement;
      while (!token.isEOF()) {
            if (token.equals(HTMLOutlineTagScanner.OUTLINE_OPEN_TAG_TOKEN)) {

                  int offset = scanner.getTokenOffset();
                  int length = scanner.getTokenLength();
                  final String tagName = document.get(offset, length);

                  if (!isEmptyTag(tagName)) {
                        parent = parent.createTag(
                              tagName,
                              offset,
                              length,
                              TagElement.HTML_TAG);
                  } else {
                        parent.createTag(
                              tagName,
                              offset,
                              length,
                              TagElement.HTML_TAG);
                  }
            } else if (
                  token.equals(
                        HTMLOutlineTagScanner.OUTLINE_CLOSE_TAG_TOKEN)) {
                  final Object closeParent = parent.getParent();
                  if (closeParent != null) {
                        parent = (TagElement) parent.getParent();
                  }
            }

            token = scanner.nextToken();
      }
} catch (BadLocationException e) {
      //do nothing
}

                  scanner = HTMLEditorEnvironment.getJsFunctionScanner();
                  scanner.setRange(document, 0, document.getLength());
                  token = scanner.nextToken();
                  int lastFunctionOffset = 0;

//Zur besseren Lesbarkeit nach links gerückt
try {
      while (!token.isEOF()) {
            if (token.equals(JsFunctionScanner.JS_FUNCTION_TOKEN)) {

                  int offset = scanner.getTokenOffset();
                  int length = scanner.getTokenLength();
                  final String funcName = getFunctionName(document.get(offset,
                        length));
                  tagElement.createTag(
                        funcName,
                        offset,
                        length,
                        TagElement.JS_FUNCTION);
                  lastFunctionOffset = offset;
            } else if (
                  token.equals(JsFunctionScanner.JS_FIELD_TOKEN)) {
                  int offset = scanner.getTokenOffset();
                  if (isGlobalField(document.get(
                        lastFunctionOffset,
                        offset - lastFunctionOffset))) {
```

```
                        int length = scanner.getTokenLength();
                        final String varName = getVarName(document.get(offset,
                              length));
                        tagElement.createTag(
                              varName,
                              offset,
                              length,
                              TagElement.JS_FIELD);
                  }
            }
            token = scanner.nextToken();
      }
} catch (BadLocationException e) {
      //do nothing
}

            }

            private boolean isEmptyTag(String tag) {
                  final String tagName = tag.toUpperCase();
                  return (tagName.startsWith("<BR") //$NON-NLS-1$
                  || tagName.startsWith("<IMG") //$NON-NLS-1$
                  || tagName.startsWith("<META") //$NON-NLS-1$
                  || (tagName.startsWith("<COL") //$NON-NLS-1$
                  && !tagName.startsWith("<COLGROUP")) //$NON-NLS-1$
                  || (tagName.startsWith("<FRAME") //$NON-NLS-1$
                  && !tagName.startsWith("<FRAMESET")) //$NON-NLS-1$
                  || tagName.startsWith("<HR") //$NON-NLS-1$
                  || tagName.startsWith("<INPUT") //$NON-NLS-1$
                  || tagName.startsWith("<ISINDEX") //$NON-NLS-1$
                  || tagName.startsWith("<LINK") //$NON-NLS-1$
                  || tagName.startsWith("<AREA") //$NON-NLS-1$
                  || tagName.startsWith("<PARAM") //$NON-NLS-1$
                  || tagName.startsWith("<BASE")); //$NON-NLS-1$
            }

            /*
             * @see IContentProvider#inputChanged(Viewer, Object, Object)
             */
            public void inputChanged(
                  Viewer viewer,
                  Object oldInput,
                  Object newInput) {
                  if (oldInput != null) {
                        final IDocument document =
                              documentProvider.getDocument(oldInput);
                        if (document != null) {
                              try {
                                    document.removePositionCategory
                                          (JS_FUNCTIONS);
                              } catch (BadPositionCategoryException x) {
                                    //do nothing
                              }
                              document.removePositionUpdater(positionUpdater);
                              document.removeDocumentListener(listener);
                        }
                  }
                  tagElement = new TagElement(
                        "ROOT",
                        null,
                        0,
                        0,
                        TagElement.HTML_TAG); //$NON-NLS-1$
                  if (newInput != null) {
                        IDocument document = documentProvider.getDocument
                              (newInput);
                        if (document != null) {
                              document.addPositionCategory(JS_FUNCTIONS);
```

```
                        document.addPositionUpdater(positionUpdater);
                        document.addDocumentListener(listener);
                        parse(document);
                  }
            }

      }

      /*
       * @see IContentProvider#dispose
       */
      public void dispose() {
            if (tagElement != null) {
                  tagElement = null;
            }
      }

      /*
       * @see IContentProvider#isDeleted(Object)
       */
      public boolean isDeleted(Object element) {
            return false;
      }

      /*
       * @see IStructuredContentProvider#getElements(Object)
       */
      public Object[] getElements(Object element) {
            return tagElement.getChildren();
      }

      /*
       * @see ITreeContentProvider#hasChildren(Object)
       */
      public boolean hasChildren(Object element) {
            if (element instanceof TagElement) {
                  ((TagElement) element).hasChildren();
            }
            return false;
      }

      /*
       * @see ITreeContentProvider#getParent(Object)
       */
      public Object getParent(Object element) {
            if (element instanceof TagElement)
                  return ((TagElement) element).getParent();
            return null;
      }

      /*
       * @see ITreeContentProvider#getChildren(Object)
       */
      public Object[] getChildren(Object element) {
            if (element instanceof TagElement)
                  return ((TagElement) element).getChildren();
            return new Object[0];
      }

};

protected Object input;
protected final IDocumentProvider documentProvider;
protected final ITextEditor textEditor;
protected final ContentProvider contentProvider;
protected final IDocumentListener listener;
private ShowAllAction showAllAction;
```

```
/**
 * Creates a content outline page using the given provider
 * and the given editor.
 */
public HTMLContentOutlinePage(
      IDocumentProvider provider,
      ITextEditor editor) {
      super();
      documentProvider = provider;
      textEditor = editor;
      contentProvider = new ContentProvider();
      listener =
            new OutlineDocumentListener(contentProvider, textEditor,
                  this);
}

/* (non-Javadoc)
 * Method declared on ContentOutlinePage
 */
public void createControl(Composite parent) {

      super.createControl(parent);

      final TreeViewer viewer = getTreeViewer();
      viewer.setContentProvider(contentProvider);
      viewer.setLabelProvider(new HTMLOutlineContentLabelProvider());
      viewer.addSelectionChangedListener(this);

      if (input != null) {
            viewer.setInput(input);
            documentProvider.getDocument(input).addDocumentListener
                  (listener);
      }
      viewer.expandAll();
      makeActions();
      addFilterSortAction();

      WorkbenchHelp.setHelp(getControl(),
            "de.kt.scripteditor.outline_context"); //$NON-NLS-1$
}

/**
 * Adds a filter action to the outline views action bar.
 */
private void addFilterSortAction() {
      final IMenuManager menu = getSite().getActionBars()
            .getMenuManager();
      menu.add(showAllAction);
}

/**
 * Creates a filter action.
 */
private void makeActions() {
      //show all action
      showAllAction = new ShowAllAction(getTreeViewer());
      showAllAction.setText(HTMLEditorMessages.getString
            ("HTMLContentOutlinePage.Filter_Tags")); //$NON-NLS-1$
      showAllAction.setToolTipText(HTMLEditorMessages.getString
            ("HTMLContentOutlinePage.Show_All_Tags")); //$NON-NLS-1$

}

/* (non-Javadoc)
 * Method declared on ContentOutlinePage
 */
public void selectionChanged(SelectionChangedEvent event) {
```

```
            super.selectionChanged(event);

            final ISelection selection = event.getSelection();
            if (selection.isEmpty())
                  textEditor.resetHighlightRange();
            else {
                  final TagElement tagElement =
                        (TagElement) ((IStructuredSelection) selection)
                              .getFirstElement();
                  int start = tagElement.getOffset();
                  int length = tagElement.getLength();
                  try {
                        textEditor.setHighlightRange(start, length, true);
                  } catch (IllegalArgumentException x) {
                        textEditor.resetHighlightRange();
                  }
            }
      }

      /**
       * Sets the input of the outline page
       */
      public void setInput(Object newInput) {
            input = newInput;
            update();
      }

      /**
       * The dispose of the HTMLOutlinePage removes the
       * document listener from the
       * actual document.
       */
      public void dispiose() {
            if (input != null) {
                  documentProvider.getDocument(input).removeDocumentListener(
                        listener);
            }
            super.dispose();
      }

      /**
       * Updates the outline page.
       */
      public void update() {
            TreeViewer viewer = getTreeViewer();

            if (viewer != null) {
                  final Control control = viewer.getControl();
                  if (control != null && !control.isDisposed()) {
                        control.setRedraw(false);
                        viewer.setInput(input);
                        viewer.expandAll();
                        control.setRedraw(true);
                  }
            }
      }
}
```

D.V.2 HTMLOutlineContentLabelProvider.java

```
package de.kt.scripteditor.htmleditor.outline;

import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Image;

import de.kt.scripteditor.util.EditorImages;

/**
 * Manages the images for JavaScript-functions, -fields and HTML-tags.
 * @author Karsten Thiemann
 * @since 02.11.2003
 */
public class HTMLOutlineContentLabelProvider extends LabelProvider {

	public HTMLOutlineContentLabelProvider() {
	}

	public Image getImage(Object element) {
		if (element instanceof TagElement) {
			int t = ((TagElement) element).getType();
			if (t == TagElement.JS_FIELD) {
				return EditorImages.getImage(EditorImages.ICON_FIELD);
			}
			if (t == TagElement.JS_FUNCTION) {
				return EditorImages.getImage
(EditorImages.ICON_FUNCTION);
			}
			if (t == TagElement.HTML_TAG) {
				return EditorImages.getImage(EditorImages.ICON_TAG);
			}
		}
		return null;
	}

}
```

D.V.3 HTMLOutlineTagScanner.java

```
package de.kt.scripteditor.htmleditor.outline;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.text.rules.IPredicateRule;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;
import org.eclipse.jface.text.rules.RuleBasedPartitionScanner;
import org.eclipse.jface.text.rules.Token;


/**
 * A scanner, used to create the tree of the HTMLOutlineView.
 * It defines opened and closed tags.
 * @author Karsten Thiemann
 */
public class HTMLOutlineTagScanner extends RuleBasedPartitionScanner {

	public final static String HTMLOutlineOpenTag = "__html_outline_open_tag";
		//$NON-NLS-1$
	public final static IToken OUTLINE_OPEN_TAG_TOKEN = new Token
		(HTMLOutlineOpenTag);
	public final static String HTMLOutlineCloseTag =
		"__html_outline_close_tag"; //$NON-NLS-1$
	public final static IToken OUTLINE_CLOSE_TAG_TOKEN = new Token
		(HTMLOutlineCloseTag);

	public HTMLOutlineTagScanner() {
		super();
		final List rules = new ArrayList();

		rules.add(new OpenTagRule(OUTLINE_OPEN_TAG_TOKEN));
		rules.add(new MultiLineRule("</", ">", OUTLINE_CLOSE_TAG_TOKEN));
			//$NON-NLS-1$ //$NON-NLS-2$

		final IPredicateRule[] result = new IPredicateRule[rules.size()];
		rules.toArray(result);
		setPredicateRules(result);
	}
}
```

D.V.4 OpenTagRule.java

```
package de.kt.scripteditor.htmleditor.outline;

import org.eclipse.jface.text.rules.ICharacterScanner;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;

/**
 * Matches open tags only. Is used by the outline view.
 * @author Karsten Thiemann
 */
public class OpenTagRule extends MultiLineRule {


	public OpenTagRule(IToken token) {
		super("<", ">", token); //$NON-NLS-1$ //$NON-NLS-2$
	}
	protected boolean sequenceDetected(
			ICharacterScanner scanner,
			char[] sequence,
			boolean eofAllowed) {
			int c = scanner.read();
			if (sequence[0] == '<') {
				if (c == '?') {
					// processing instruction - abort
					scanner.unread();
					return false;
				}
				if (c == '!') {
					scanner.unread();
					// comment - abort
					return false;
				}
				if (c == '/') {
					scanner.unread();
					// closed tag - abort
					return false;
				}
			} else if (sequence[0] == '>') {
				scanner.unread();
			}
			return super.sequenceDetected(scanner, sequence, eofAllowed);
	}
}
```

D.V.5 ShowAllAction.java

```
package de.kt.scripteditor.htmleditor.outline;

import org.eclipse.jface.action.Action;
import org.eclipse.jface.viewers.TreeViewer;

import de.kt.scripteditor.htmleditor.HTMLEditorEnvironment;


/**
 * This action toggles between showing all items and adding
 * a filter that removes less important tags from the tree.
 * @author Karsten Thiemann
 */

public class ShowAllAction extends Action {
	private final TreeViewer viewer;
	private final TagElementFilter filter;
	private boolean showAll;

	/**
	 * @param viewer
	 * @param name
	 */
	protected ShowAllAction(TreeViewer viewer) {
		this.viewer = viewer;
		filter = HTMLEditorEnvironment.getTagElementFilter();
		showAll = true;
	}

	/**
	 * Toggles adding/removing the filter.
	 * @see org.eclipse.jface.action.Action#run()
	 */
	public void run() {
		showAll = !showAll;
		if (showAll)
			viewer.removeFilter(filter);
		else
			viewer.addFilter(filter);
		((TreeViewer) viewer).expandAll();
	}

}
```

D.V.6 TagElement.java

```
package de.kt.scripteditor.htmleditor.outline;

import java.util.Vector;

/**
 * A TagElement is used as a treenode in the HTMLOutlineView.
 * @author Karsten Thiemann
 */
public class TagElement {
	public static final int JS_FUNCTION = 1;
	public static final int JS_FIELD = 2;
	public static final int HTML_TAG = 3;

	private TagElement parent;
	private final String name;
	private final int offset;
	private final int length;
	private final int type;
	private Vector tags;


	/**
	 * Constructor.
	 * @param name the name
	 * @param parent the parent, null if none
	 * @param offset the offset of the corresponding tag in the document
	 * @param length the length of the corresponding tag in the document
	 */
	public TagElement(
		String name,
		TagElement parent,
		int offset,
		int length,
		int type) {

		this.name = name;
		this.parent = parent;
		this.offset = offset;
		this.length = length;
		this.type = type;
	}

	/**
	 * Adds an TagElement as child to this TagElement.
	 * @param tag the TagElement to add
	 */
	public void add(TagElement tag) {
		getTags().add(tag);
		// synchronizes backpointer of userGroup: defensive
		tag.setParent(this);
	}

	/**
	 * Creates a new TagElement nested in this TagElement
	 * @param name the name of the sub TagElement
	 */
	public TagElement createTag(String name,int offset,int length,int type) {
		TagElement newTag = new TagElement(name, this, offset, length,
			type);
		add(newTag);
		return newTag;
	}

	/**
	 * Returns all children of this TagElement.
```

```
     * @return Object[] all children of this TagElement
     */
    public Object[] getChildren() {
          return getTags().toArray();
    }

    /**
     * @return boolean true if this TagElement has children
     */
    public boolean hasChildren(){
          return getTags().size()>0;
    }


    /**
     * Returns all nested TagElements.
     * @return Vector all nested TagElements
     */
    private Vector getTags() {
          if (tags == null)
                tags = new Vector();
          return tags;
    }

    /**
     * Returns the name.
     * @return String the name
     */
    String getName() {
          return name;
    }

    /**
     * Returns the name.
     * @return String the name
     */
    public String toString() {
          return name;
    }

    /**
     * Returns the parent.
     * @return Object the TagsElements parent
     */
    public Object getParent() {
          return parent;
    }

    /**
     * Sets this instance's parent back pointer.
     * @param the parent to set.
     */
    void setParent(TagElement newParent) {
          parent = newParent;
    }

    /**
     * Returns the offset.
     * @return int the offset
     */
    public int getOffset() {
          return offset;
    }

    /**
     * Returns the length.
     * @return int the length
```

```
         */
        public int getLength() {
              return length;
        }

        /**
         * Returns the type of the TagElement.
         * @return int the type
         */
        public int getType() {
              return type;
        }

}
```

D.V.7 TagElementFilter.java

```
package de.kt.scripteditor.htmleditor.outline;

import java.util.Iterator;
import java.util.Vector;
import org.eclipse.jface.preference.IPreferenceStore;
import org.eclipse.jface.util.IPropertyChangeListener;
import org.eclipse.jface.util.PropertyChangeEvent;
import org.eclipse.jface.viewers.Viewer;
import org.eclipse.jface.viewers.ViewerFilter;
import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.preferences.IPreferenceIDs;

/**
  * Filters out TagElement objects, based on their tagname.
  * @author Karsten Thiemann
 */
public class TagElementFilter
      extends ViewerFilter
      implements IPropertyChangeListener, IPreferenceIDs {

      private final Vector tagNamesToShow;

      public TagElementFilter() {
            tagNamesToShow = new Vector();
            initTagNamesToShow();
      }

      /**
       * Inits the settings for all tags.
       */
      private void initTagNamesToShow() {

            final String[] tagNames =
                  new String[] {
                        A,
                        BODY,
                        BUTTON,
                        DIV,
                        FORM,
                        FRAME,
                        FRAMESET,
                        H1,
                        IFRAME,
                        IMG,
                        INPUT,
                        LI,
                        LINK,
                        MAP,
                        OBJECT,
                        OL,
                        OPTION,
                        P,
                        SELECT,
                        SPAN,
                        STYLE,
                        TABLE,
                        TD,
                        TEXTAREA,
                        TR,
                        UL };
            final IPreferenceStore store =
                  ScripteditorPlugin.getDefault().getPreferenceStore();
            for (int i = 0; i < tagNames.length; i++) {
                  if (store.getBoolean(tagNames[i])) {
                        tagNamesToShow.add(new String("<" + tagNames[i]
                              .toUpperCase())); //$NON-NLS-1$
```

```
                }
        }
}

/**
 * Returns true for the tags: html and all other tags set in the
 * preferences. And for all other Tags that surrounds one of these.
 */
public boolean select(Viewer viewer, Object parentElement, Object element)
{
        if (element instanceof TagElement) {
                final TagElement tag = (TagElement) element;
                final String tagName = tag.getName().toUpperCase();
                final int type = tag.getType();
                return (
                        tagName.startsWith("<ROOT") //$NON-NLS-1$
                                || tagName.startsWith("<HTML") //$NON-NLS-1$
                                || type == TagElement.JS_FIELD
                                || type == TagElement.JS_FUNCTION
                                || isTagNameToShow(tagName)
                                || includesRelevantTag((TagElement) element));
        }
        return true;
}

/**
 * Returns true if the given tag has to appear in the outlines tag tree.
 * @param tagName the given tag name
 * @return boolean
 */
private boolean isTagNameToShow(String tagName) {
        final Iterator iter = tagNamesToShow.iterator();
        while (iter.hasNext()) {
                final String s = (String) iter.next();
                if (tagName.startsWith(s)) {
                        return true;
                }
        }
        return false;
}

/**
 * Returns true if the given tag includes one of the following tags:
 * html, body, table, form, div, a.
 * @param element the TagElement
 * @return boolean  true if the given tag includes a relevant tag
 */
private boolean includesRelevantTag(TagElement element) {
        if (element.hasChildren()) {
                final Object[] children = element.getChildren();
                for (int i = 0; i < children.length; i++) {
                        if (children[i] instanceof TagElement) {
                                final TagElement child = (TagElement) children[i];
                                final String tagName = child.getName()
                                        .toUpperCase();
                                if (isTagNameToShow(tagName)) {
                                        return true;
                                } else if(includesRelevantTag(child)){
                                        return true;
                                }
                        }
                }
        }
        return false;
}
```

```
	/**
	 * Returns true if the given element is a TagElement.
	 * @see org.eclipse.jface.viewers.ViewerFilter#isFilterProperty(Object,
	 * String)
	 */
	public boolean isFilterProperty(Object element, String property) {
		return (element instanceof TagElement);
	}

	/* (non-Javadoc)
	 * @see org.eclipse.jface.util.IPropertyChangeListener#propertyChange
	 * (org.eclipse.jface.util.PropertyChangeEvent)
	 */
	public void propertyChange(PropertyChangeEvent event) {

		if (event.getProperty().equalsIgnoreCase(A)
				|| event.getProperty().equalsIgnoreCase(BODY)
				|| event.getProperty().equalsIgnoreCase(BUTTON)
				|| event.getProperty().equalsIgnoreCase(DIV)
				|| event.getProperty().equalsIgnoreCase(FORM)
				|| event.getProperty().equalsIgnoreCase(FRAME)
				|| event.getProperty().equalsIgnoreCase(FRAMESET)
				|| event.getProperty().equalsIgnoreCase(H1)
				|| event.getProperty().equalsIgnoreCase(IFRAME)
				|| event.getProperty().equalsIgnoreCase(IMG)
				|| event.getProperty().equalsIgnoreCase(INPUT)
				|| event.getProperty().equalsIgnoreCase(LI)
				|| event.getProperty().equalsIgnoreCase(LINK)
				|| event.getProperty().equalsIgnoreCase(MAP)
				|| event.getProperty().equalsIgnoreCase(OBJECT)
				|| event.getProperty().equalsIgnoreCase(OL)
				|| event.getProperty().equalsIgnoreCase(OPTION)
				|| event.getProperty().equalsIgnoreCase(P)
				|| event.getProperty().equalsIgnoreCase(SELECT)
				|| event.getProperty().equalsIgnoreCase(SPAN)
				|| event.getProperty().equalsIgnoreCase(STYLE)
				|| event.getProperty().equalsIgnoreCase(TABLE)
				|| event.getProperty().equalsIgnoreCase(TD)
				|| event.getProperty().equalsIgnoreCase(TEXTAREA)
				|| event.getProperty().equalsIgnoreCase(TR)
				|| event.getProperty().equalsIgnoreCase(UL)) {
			final Object old = event.getOldValue();
			if (old instanceof Boolean) {
				if (!((Boolean) old).booleanValue()) {
					tagNamesToShow.add(
							new String("<" + event.getProperty()
									.toUpperCase())); //$NON-NLS-1$
				} else {
					final Iterator iter = tagNamesToShow.iterator();
					final String value = new String("<" + event
							.getProperty().toUpperCase()); //$NON-NLS-1$
					while (iter.hasNext()) {
						final String s = (String) iter.next();
						if (value.equalsIgnoreCase(s)) {
							tagNamesToShow.remove(s);
							break;
						}
					}
				}
			} else if (old == null) {
				final Object newValue = event.getNewValue();
				if (newValue instanceof String) {
					if (((String)newValue).equalsIgnoreCase("true")) {
						//$NON-NLS-1$
						tagNamesToShow.add(
								new String("<" + event.getProperty()
										.toUpperCase())); //$NON-NLS-1$
}	}	}	}	}	}
```

D.VI de.kt.scripteditor.javascript

D.VI.1 JavascriptMethod.java

```
package de.kt.scripteditor.javascript;

/**
 * Holds the information about a Javascript method.
 * @author Karsten Thiemann
 */
class JavascriptMethod {
	private final String name;
	private final String returnTypeName;
	private final boolean isVoid;

	/**
	 * Constructor.
	 * @param name the name
	 * @param returnTypeName the type of the methods return value
	 * @param isVoid true if it´s a void method
	 */
	public JavascriptMethod(
		String name,
		String returnTypeName,
		boolean isVoid) {
		this.name = name;
		this.returnTypeName = returnTypeName;
		this.isVoid = isVoid;
	}

	/**
	 * @return boolean true if it´s a void function
	 */
	public boolean isVoidMethod() {
		return isVoid;
	}
	/**
	 * Returns the name.
	 * @return String the name
	 */
	public String getName() {
		return name;
	}

	/**
	 *  Returns the type of the methods return value.
	 * @return String the type of the methods return value
	 */
	public String getReturnTypeName() {
		return returnTypeName;
	}

}
```

D.VI.2 JavascriptObject.java

```
package de.kt.scripteditor.javascript;

/**
 * Holds the information about a Javascript object.
 * @author Karsten Thiemann
 */
public class JavascriptObject {
	private final String name;
	private final JavascriptMethod[] methods;
	private final String[] childObjectNames;
	private final String[] attributeNames;

	/**
	 * Constructor.
	 * @param name the name
	 * @param methods all methods
	 * @param childObjectNames the names of all child Javascript objects
	 * @param attributeNames the names of all attributes
	 */
	public JavascriptObject(
			String name,
			JavascriptMethod[] methods,
			String[] childObjectNames,
			String[] attributeNames) {
			this.name = name;
			this.methods = methods;
			this.childObjectNames = childObjectNames;
			this.attributeNames = attributeNames;
	}

	/**
	 * Returns all methods off the Javascript object.
	 * @return JavascriptMethod[] all methods
	 */
	public JavascriptMethod[] getMethods() {
			return methods;
	}

	/**
	 * Returns the names off all object methods.
	 * @return String[] all method names
	 */
	public String[] getMethodNames() {
			final String[] methodNames = new String[methods.length];
			for (int i = 0; i < methodNames.length; i++) {
				methodNames[i] = methods[i].getName();
			}
			return methodNames;
	}

	/**
	 * Returns the names off all object attributes.
	 * @return String[] all attribute names
	 */
	public String[] getAttributeNames() {
			return attributeNames;
	}

	/**
	 * Returns the name.
	 * @return String the name
	 */
	public String getName() {
			return name;
```

```
        }

        /**
         * Returns all child object names.
         * @return String[] all child object names
         */
        public String[] getChildObjectNames() {
              return childObjectNames;
        }

}
```

D.VI.3 JsCodeScanner.java

```
package de.kt.scripteditor.javascript;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.text.TextAttribute;
import org.eclipse.jface.text.rules.EndOfLineRule;
import org.eclipse.jface.text.rules.IRule;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;
import org.eclipse.jface.text.rules.RuleBasedScanner;
import org.eclipse.jface.text.rules.SingleLineRule;
import org.eclipse.jface.text.rules.Token;
import org.eclipse.jface.text.rules.WhitespaceRule;
import org.eclipse.jface.text.rules.WordRule;

import de.kt.scripteditor.util.ColorProvider;
import de.kt.scripteditor.util.GenericWhitespaceDetector;

/**
 * A scanner for the default content of a *.js document. It defines
 * keywords, constants, strings, functions and single line comments.
 * @author Karsten Thiemann
 */
public class JsCodeScanner extends RuleBasedScanner {

      private static final String[] fgKeywords = JsSyntax.getKeywords();
      private static final String[] fgConstants = JsSyntax.getConstants();

      /**
       * Constructor.
       * @param fgColorProvider the ColorProvider
       */
      public JsCodeScanner(ColorProvider provider) {

            final IToken keyword =
                  new Token(
                        new TextAttribute(provider.getColor
                              (ColorProvider.KEYWORD)));
            final IToken constant =
                  new Token(
                        new TextAttribute(provider.getColor
                              (ColorProvider.CONSTANT)));
            final IToken string =
                  new Token(
                        new TextAttribute(provider.getColor
                              (ColorProvider.STRING)));
            final IToken comment =
                  new Token(
                        new TextAttribute(
                             provider.getColor(ColorProvider.JS_COMMENT)));
            final IToken function =
                  new Token(
                        new TextAttribute(provider.getColor
                              (ColorProvider.KEYWORD)));
            final IToken other =
                  new Token(
                        new TextAttribute(provider.getColor
                              (ColorProvider.DEFAULT)));

            final List rules = new ArrayList();

            //comment rules are added here for the use in HTML-pages
            // Add rule for single line comments.
```

```
            rules.add(new EndOfLineRule("//", comment)); //$NON-NLS-1$

            //Add rule for multi-line comments.
            rules.add(new MultiLineRule("/*", "*/", comment, (char) 0, true));
                  //$NON-NLS-1$ //$NON-NLS-2$

            // Add rule for strings and character constants.
            rules.add(new SingleLineRule("\"", "\"", string, '\\'));
                  //$NON-NLS-1$ //$NON-NLS-2$
            rules.add(new SingleLineRule("'", "'", string, '\\'));
                  //$NON-NLS-1$ //$NON-NLS-2$

            // Add generic whitespace rule.
            rules.add(new WhitespaceRule(new GenericWhitespaceDetector()));

            // Add word rules for keywords, types, and constants.
            final WordRule wordRule = new WordRule(new JsWordDetector(), other);

            wordRule.addWord("script", constant); //$NON-NLS-1$
            wordRule.addWord("SCRIPT", constant); //$NON-NLS-1$


            for (int i = 0; i < fgKeywords.length; i++)
                  wordRule.addWord(fgKeywords[i], keyword);
            for (int i = 0; i < fgConstants.length; i++)
                  wordRule.addWord(fgConstants[i], constant);
            rules.add(wordRule);

            final IRule[] result = new IRule[rules.size()];
            rules.toArray(result);
            setRules(result);
      }

}
```

D.VI.4 JsCompletionProcessor.java

```
package de.kt.scripteditor.javascript;

import java.util.Iterator;
import java.util.Vector;

import org.eclipse.jface.text.ITextViewer;
import org.eclipse.jface.text.contentassist.CompletionProposal;
import org.eclipse.jface.text.contentassist.ICompletionProposal;
import org.eclipse.jface.text.contentassist.IContentAssistProcessor;
import org.eclipse.jface.text.contentassist.IContextInformation;
import org.eclipse.jface.text.contentassist.IContextInformationValidator;

/**
 * The ContentAssistProcessor of javascript areas.
 * @author Karsten Thiemann
 */
public class JsCompletionProcessor implements IContentAssistProcessor {

      private static final int ATTRIBUTE = 1;
      private static final int OBJECT = 2;
      private static final int FUNCTION = 3;
      private static final int OTHER = 4;

      protected Vector proposalList = new Vector();

/**
* This method returns a list of completion proposals as ICompletionProposal
* objects. The proposals are based on the word at the offset in the document
* where the cursor is positioned.
* @see org.eclipse.jface.text.contentassist.IContentAssistProcessor#
* computeCompletionProposals(ITextViewer, int)
 */
      public ICompletionProposal[] computeCompletionProposals(
            ITextViewer viewer,
            int documentOffset) {
                  final JsStringAnalyser stringAnalyser =
                  new JsStringAnalyser(viewer, documentOffset);

            //letzter Teil ist Objekt oder Funktion (nicht void)
            if (stringAnalyser.isLastPartIsObject()) {
                  //add childObjectNames
                  final String[] childObjectNames =
                        JsSyntax.getChildObjectNamesForObject(
                              stringAnalyser.getLastPart());
                  for (int j = 0; j < childObjectNames.length; j++) {
                        proposalList.add(new JsProposal(childObjectNames[j],
                              OBJECT));
                  }
                  //    add attributeNames
                  final String[] attributeNames =
                        JsSyntax.getAttributeNamesForObject(
                              stringAnalyser.getLastPart());
                  for (int j = 0; j < attributeNames.length; j++) {
                        proposalList.add(new JsProposal(attributeNames[j],
                              ATTRIBUTE));
                  }
                  //add methodNames
                  final String[] methodNames =
                        JsSyntax.getMethodNamesForObject
                              (stringAnalyser.getLastPart());
                  for (int j = 0; j < methodNames.length; j++) {
                        proposalList.add(new JsProposal(methodNames[j],
                              FUNCTION));
                  }
```

```
            return getAdaptedArray(stringAnalyser);
      } else
            //Parent ist Objekt oder Funktion (nicht void)
            if (stringAnalyser.isParentIsObject()) {

                  if (stringAnalyser.isLastCharIsClosedBracket()
                        || stringAnalyser.isLastCharIsPoint()) {
                  //unbekannte oder void-Funktion oder falscher Punkt
                        return null;
                  }

                  final String wordPart = stringAnalyser.getLastPart();
                  //add childObjectNames
                  final String[] childObjectNames =
                        JsSyntax.getChildObjectNamesForObject(
                              stringAnalyser.getParentObjectString());
                  for (int j = 0; j < childObjectNames.length; j++) {
                        if (childObjectNames[j]
                              .toUpperCase()
                              .startsWith(wordPart.toUpperCase()))
                              proposalList.add(
                                    new JsProposal(childObjectNames[j],
                                          OBJECT));
                  }
                  //add attributeNames
                  final String[] attributeNames =
                        JsSyntax.getAttributeNamesForObject(
                              stringAnalyser.getParentObjectString());
                  for (int j = 0; j < attributeNames.length; j++) {
                        if (attributeNames[j]
                              .toUpperCase()
                              .startsWith(wordPart.toUpperCase()))
                              proposalList.add(
                                    new JsProposal(attributeNames[j],
                                          ATTRIBUTE));
                  }
                  //add methodNames
                  final String[] methodNames =
                        JsSyntax.getMethodNamesForObject(
                              stringAnalyser.getParentObjectString());
                  for (int j = 0; j < methodNames.length; j++) {
                        if (methodNames[j]
                              .toUpperCase()
                              .startsWith(wordPart.toUpperCase()))
                              proposalList.add(
                                    new JsProposal(methodNames[j],
                                          FUNCTION));
                  }

                  return getAdaptedArray(stringAnalyser);
            }

      //alle Standardvorschläge anzeigen
      final Object[] allWords = JsSyntax.getAllWords();
      // iterate over all the different categories
      //TODO unabhängig werden von der Reihenfolge!
      //constants, keywords, getAllGlobalMethodNames(),
      //getAllObjectNames()
      for (int i = 0; i < allWords.length; i++) {
            final String[] list = (String[]) allWords[i];
            int type = OTHER;
            switch (i) {
                  case 2 :
                        type = FUNCTION;
                        break;
                  case 3 :
                        type = OBJECT;
```

```
                        break;
                  default :
                        type = OTHER;
                        break;
            }
            // iterate over the current category
            for (int y = 0; y < list.length; y++) {
                  if (list[y]
                        .toUpperCase()
                        .startsWith(stringAnalyser.getLastPart()
                        .toUpperCase()))
                        proposalList.add(new JsProposal(list[y], type));
            }
      }
      return getAdaptedArray(stringAnalyser);

}

/*
 * Turns the vector into an Array of ICompletionProposal objects
 */
protected ICompletionProposal[] getAdaptedArray(JsStringAnalyser analyser)
{
      final ICompletionProposal[] result =
            new ICompletionProposal[proposalList.size()];

      int index = 0;

      for (Iterator i = proposalList.iterator(); i.hasNext();) {
            final JsProposal proposal = (JsProposal) i.next();
            String keyWord = proposal.name;
            String displayString = keyWord;
            int cursorOffset = keyWord.length();
            if (proposal.type == FUNCTION) {
                  displayString += " - Function"; //$NON-NLS-1$
                  keyWord += "()"; //$NON-NLS-1$
                  cursorOffset++;
            } else if (proposal.type == OBJECT) {
                  displayString += " - Object"; //$NON-NLS-1$
            } else if (proposal.type == ATTRIBUTE) {
                  displayString += " - Attribute"; //$NON-NLS-1$
            }
            if (analyser.isLastPartIsObject()
                  && !analyser.isLastCharIsPoint()) {
                  keyWord = "." + keyWord; //$NON-NLS-1$
                  cursorOffset++;
            }

            result[index] =
                  new CompletionProposal(
                        keyWord,
                        analyser.getOffset(),
                        analyser.getLength(),
                        cursorOffset,
                        null,
                        displayString,
                        null,
                        null);
            index++;

      }

      proposalList.removeAllElements();
      return result;
}

/* (non-Javadoc)
```

```
	 * Method declared on IContentAssistProcessor
	 */
	public IContextInformation[] computeContextInformation(
		ITextViewer viewer,
		int documentOffset) {
		return null;
	}

	/* (non-Javadoc)
	 * Method declared on IContentAssistProcessor
	 */
	public char[] getCompletionProposalAutoActivationCharacters() {
		return new char[] { '.' };
	}

	/* (non-Javadoc)
	 * Method declared on IContentAssistProcessor
	 */
	public char[] getContextInformationAutoActivationCharacters() {
		return null;
	}

	/* (non-Javadoc)
	 * Method declared on IContentAssistProcessor
	 */
	public IContextInformationValidator getContextInformationValidator() {
		return null;
	}

	/* (non-Javadoc)
	 * Method declared on IContentAssistProcessor
	 */
	public String getErrorMessage() {
		return null;
	}

	/**
	 * A JsProposal consists of a name and a type. Types are attribute,
	 * object, function or other, defined in JsCompletionProcessor.
	 * @author Karsten Thiemann
	 */
	protected class JsProposal {
		final String name;
		final int type;

		/**
		 * Constructor.
		 * @param name the name
		 * @param type the type
		 */
		JsProposal(String name, int type) {
			this.name = name;
			this.type = type;
		}
	}
}
```

D.VI.5 JsStringAnalyser.java

```
package de.kt.scripteditor.javascript;

import java.util.StringTokenizer;
import java.util.Vector;

import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.ITextViewer;

/**
 * Used to scan and detect for Javascript keywords, functions or objects
 * @author Karsten Thiemann
 */
public class JsStringAnalyser {
	private String wordPart = ""; //$NON-NLS-1$
	private String parentObjectString = ""; //$NON-NLS-1$
	private int documentOffset;
	private boolean lastCharIsPoint = false;
	private boolean lastCharIsClosedBracket = false;
	private boolean lastPartIsObject = false;
	private boolean parentIsObject = false;
	private final String[] objectNames = JsSyntax.getAllObjectNames();
	private final String[] globalMethodNames =
		JsSyntax.getAllGlobalMethodNames();
	private final Vector elements;

	/**
	 * Constructor.
	 * @param viewer is a text viewer
	 * @param documentOffset into the document
	 */
	public JsStringAnalyser(ITextViewer viewer, int offset) {
		elements = new Vector();
		documentOffset = offset;
		int docOffset = documentOffset - 1;
		int bracketcount = 0;
		try {
			final StringBuffer buffer = new StringBuffer();
			if (viewer.getDocument().getChar(docOffset) == '.') {
				lastCharIsPoint = true;
			} else if (viewer.getDocument().getChar(docOffset) == ')') {
				lastCharIsClosedBracket = true;
			}
			//string left of docOffset e.g. "document.getEl"
			while (((docOffset) >= viewer.getTopIndexStartOffset())
				&& (Character
					.isJavaIdentifierPart(
						viewer.getDocument().getChar(docOffset))
					|| viewer.getDocument().getChar(docOffset) == '.'
					|| viewer.getDocument().getChar(docOffset) == ')'
					|| viewer.getDocument().getChar(docOffset) == '('
					|| bracketcount > 0)) {
				if (viewer.getDocument().getChar(docOffset) == ')') {
					bracketcount++;
				}
				if (bracketcount == 0) {
					buffer.append(viewer.getDocument().getChar
						(docOffset));
				}
				if (viewer.getDocument().getChar(docOffset) == '(') {
					bracketcount--;
				}
				docOffset--;
			} //we've been one step too far : increase the offset
			docOffset++;
			wordPart = buffer.reverse().toString();
```

```
                generateStringVector(wordPart);
                analyse(0);
          } catch (BadLocationException e) { // do nothing
          }
    }
    /**
    * Computes the type of the wordPart and it´s parent.
    */
    private void analyse(int step) {
          if (step >= elements.size())
                return;
          //Fehler!
          final String element = (String) elements.elementAt(step);

          if (step == 0) {
                if (isObject(element)) {
                      lastPartIsObject = true;
                } else if (isGlobalMethod(element)) {
                      if (JsSyntax.getGlobalMethod(element).isVoidMethod()) {
                            lastPartIsObject = false;
                            parentIsObject = false;
                            return;
                            //nicht weiter analysieren, da unbekannt
                      } else {
                            lastPartIsObject = true;
                            elements.set(
                                  0,
                                  JsSyntax.getGlobalMethod(element)
                                        .getReturnTypeName());
                      }
                } else {
                      //unknown or constant
                      lastPartIsObject = false;
                      parentIsObject = false;
                      return;
                      //nicht weiter analysieren, da unbekannt
                }
          } else {
                //step>0
                parentIsObject = lastPartIsObject;
                if (parentIsObject) {
                      parentObjectString = (String) elements.elementAt(
                            step - 1);
                }
                if (isObject(element)) {
                      lastPartIsObject = true;
                } else if (
                      parentIsObject
                            && isMethodOfParent(
                                  element,
                                  (String) elements.elementAt(step - 1))) {
                      if (JsSyntax
                            .getMethodForObject(
                                  element,
                                  (String) elements.elementAt(step - 1))
                            .isVoidMethod()) {
                            lastPartIsObject = false;
                            return;
                            //nicht weiter analysieren, da unbekannt
                      } else {
                            lastPartIsObject = true;
                            elements.set(
                                  step,
                                  JsSyntax
                                        .getMethodForObject(
                                              element,
                                              (String) elements.elementAt(
```

```
                                        step - 1))
                                    .getReturnTypeName());
                        }
                  } else {
                        lastPartIsObject = false;
                        if (elements.size() > step + 1) {
                  //es gibt noch weitere elemente, aber unbekannte Syntax...
                              parentIsObject = false;
                              parentObjectString = ""; //$NON-NLS-1$
                        }
                        return;
                  }
            }

            if (++step >= elements.size()) {
                  if (step > 1) {
                        parentObjectString = (String) elements.elementAt(
                              step - 2);
                  }
                  return;
            } else {
                  analyse(step);
            }

      }

      private void generateStringVector(String wordPart) {
            final StringTokenizer tokenizer = new StringTokenizer(wordPart,
                  "."); //$NON-NLS-1$
            while (tokenizer.hasMoreTokens()) {
                  elements.add(tokenizer.nextToken());
            }
      }

      private boolean isObject(String element) {
            for (int i = 0; i < objectNames.length; i++) {
                  if (objectNames[i].equalsIgnoreCase(element)) {
                        return true;
                  }
            }
            return false;
      }

      private boolean isGlobalMethod(String element) {
            for (int i = 0; i < globalMethodNames.length; i++) {
                  if (globalMethodNames[i].equalsIgnoreCase(element)) {
                        return true;
                  }
            }
            return false;
      }

      private boolean isMethodOfParent(String element, String parent) {
            final String[] methodNames = JsSyntax.getMethodNamesForObject
                  (parent);
            for (int i = 0; i < methodNames.length; i++) {
                  if (methodNames[i].equalsIgnoreCase(element)) {
                        return true;
                  }
            }
            return false;
      }

      /**
       * Returns the offset for the insert.
       * @return int the offset
       */
```

```
	public int getOffset() {
		if (lastPartIsObject) {
			return documentOffset;
		}
		return documentOffset - getLastPart().length();
	}

	/**
	 * @return boolean true if the last char is '.'
	 */
	public boolean isLastCharIsPoint() {
		return lastCharIsPoint;
	}

	/**
	 * @return boolean true if the last char is ')'
	 */
	public boolean isLastCharIsClosedBracket() {
		return lastCharIsClosedBracket;
	}

	/**
	 * @return boolean true if the last wordPart is a Javascript object.
	 */
	public boolean isLastPartIsObject() {
		return lastPartIsObject;
	}

	/**
	 * @return boolean true if the prelast wordPart is a Javascript object.
	 */
	public boolean isParentIsObject() {
		return parentIsObject;
	}

	/**
	 * Returns the name of the last wordParts parent.
	 * For document.getEl it would return "document".
	 * @return String the name of the last wordParts parent
	 */
	public String getParentObjectString() {
		return parentObjectString;
	}

	/**
	 * Returns the last wordPart.
	 * @return String the last wordPart
	 */
	public String getLastPart() {
		if (elements.size() > 0)
			return (String) elements.lastElement();
		return wordPart;
	}
	/**
	 * Returns the length of the last wordPart.
	 * @return int the length of the last wordPart
	 */
	public int getLength() {
		if (lastPartIsObject) {
			return 0;
		}
		return getLastPart().length();

	}

}
```

D.VI.6 JsSyntax.java

```
package de.kt.scripteditor.javascript;

import java.util.Iterator;
import java.util.Vector;

import de.kt.scripteditor.ScripteditorPlugin;

/**
 * Represents the syntax of Javascript. Provides information about Javascript
 * objects, global methods, keywords and constants.
 * @author Karsten Thiemann
 */
public class JsSyntax {

	private static Vector jsObjects;	//all JavaScript-Objects from the
					//syntax-xml-file
	private static Vector globalMethods;	//all global JavaScript-Functions
	private static String[] keywords;	//all JavaScript-Keywords
	private static String[] constants;	//all JavaScript-Constants
	private static String[] allWords;	//union of objects,functions,
					//keywords and constants

	/**
	 * parse the xml-syntaxFile
	 */
	static {
		JsSyntaxParser parser = new JsSyntaxParser();

		parser.parseDocument(
			ScripteditorPlugin.getDefault().getDescriptor()
				.getInstallURL()
				+ "JsSyntax.xml"); //$NON-NLS-1$

		keywords = parser.getKeywords();
		constants = parser.getConstants();
		globalMethods = parser.getGlobalMethods();
		jsObjects = parser.getJsObjects();
	}

	/**
	 * Returns all object names.
	 * @return String[] all object names
	 */
	public static String[] getAllObjectNames() {
		final String[] objectNames = new String[jsObjects.size()];
		for (int i = 0; i < objectNames.length; i++) {
			objectNames[i] =
				((JavascriptObject) jsObjects.elementAt(i)).getName();
		}
		return objectNames;
	}

	/**
	 * Returns the method names of the given object.
	 * @param objectName the name of a javascript object
	 * @return String[] the method names of the given object
	 */
	public static String[] getMethodNamesForObject(String objectName) {
		final Iterator it = jsObjects.iterator();
		while (it.hasNext()) {
			JavascriptObject jso = (JavascriptObject) it.next();
			if (jso.getName().equalsIgnoreCase(objectName)) {
				return jso.getMethodNames();
			}
```

```
		}
		return new String[0];
	}

	/**
	 * Returns the child object names of the given object.
	 * @param objectName the name of a javascript object
	 * @return String[] the child object names of the given object
	 */
	public static String[] getChildObjectNamesForObject(String objectName) {
		final Iterator it = jsObjects.iterator();
		while (it.hasNext()) {
			final JavascriptObject jso = (JavascriptObject) it.next();
			if (jso.getName().equalsIgnoreCase(objectName)) {
				return jso.getChildObjectNames();
			}
		}
		return new String[0];
	}

	/**
	 * Returns the attribute names of the given object.
	 * @param objectName the name of a javascript object
	 * @return String[] the attribute names of the given object
	 */
	public static String[] getAttributeNamesForObject(String objectName) {
		final Iterator it = jsObjects.iterator();
		while (it.hasNext()) {
			final JavascriptObject jso = (JavascriptObject) it.next();
			if (jso.getName().equalsIgnoreCase(objectName)) {
				return jso.getAttributeNames();
			}
		}
		return new String[0];
	}

	/**
	 * Returns the names of all global methods.
	 * @return String[] the names of all global methods
	 */
	public static String[] getAllGlobalMethodNames() {
		final String[] globalMethodNames = new String[globalMethods.size()];
		for (int i = 0; i < globalMethodNames.length; i++) {
			globalMethodNames[i] =
				((JavascriptMethod) globalMethods.elementAt(i))
					.getName();
		}
		return globalMethodNames;
	}

	/**
	 * Returns all words, defined in the syntax as an array of String arrays.
	 * @return Object[] all words, defined in the syntax
	 */
	public static Object[] getAllWords() {
		return new Object[] {
			constants,
			keywords,
			getAllGlobalMethodNames(),
			getAllObjectNames()};
	}

	/**
	 * Returns the (globas) JavascriptMethod with the given name
	 * @param name the name
	 * @return JavascriptMethod the (globas) JavascriptMethod
	 * with the given name or null
```

```
     */
    public static JavascriptMethod getGlobalMethod(String name){
          final Iterator it = globalMethods.iterator();
          while (it.hasNext()) {
                final JavascriptMethod element = (JavascriptMethod) it.next();
                if(element.getName().equalsIgnoreCase(name)){
                      return element;
                }
          }
          return null;
    }

    /**
     * Returns the object method with the given name for the given object.
     * @param methodName the methods name
     * @param objectName the javascript objects name
     * @return JavascriptMethod the object method with the given
     * name for the given object or null
     */
    public static JavascriptMethod getMethodForObject(String methodName,
          String objectName) {
          final Iterator it = jsObjects.iterator();
          while (it.hasNext()) {
                final JavascriptObject jso = (JavascriptObject) it.next();
                if (jso.getName().equalsIgnoreCase(objectName)) {
                      final JavascriptMethod [] methods = jso.getMethods();
                      for (int i = 0; i < methods.length; i++) {
                            if(methods[i].getName().equalsIgnoreCase
                                  (methodName)){
                                  return methods[i];
                            }
                      }
                }
          }
          return null;
    }

    /**
     * Returns all JavaScript constants.
     * @return String[] all constants
     */
    public static String[] getConstants() {
          return constants;
    }

    /**
     * Returns all JavaScript keywords.
     * @return String[] all keywords
     */
    public static String[] getKeywords() {
          return keywords;
    }

}
```

D.VI.7 JsSyntaxParser.java

```
package de.kt.scripteditor.javascript;

import java.util.Vector;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import de.kt.scripteditor.util.AbstractXMLParser;

/**
 * Parses a JsSyntax-XMLFile.
 * @author Karsten Thiemann
 */
public class JsSyntaxParser extends AbstractXMLParser {

	private final Vector constants = new Vector();
	private final Vector keywords = new Vector();
	private final Vector globalMethods = new Vector();
	private final Vector jsObjects = new Vector();

	/* (non-Javadoc)
	 * @see de.kt.jseditor.util.AbstractXMLParser#doContent
	 * (org.w3c.dom.Document)
	 */
	protected void doContent(Document d) {
		doConstants(d);
		doKeywords(d);
		doGlobalMethods(d);
		doObjects(d);
	}

	/**
	 * @param d the Document
	 */
	private void doObjects(Document d) {
		final NodeList nodeList = d.getElementsByTagName("objects");
			//$NON-NLS-1$
		for (int i = 0; i < nodeList.getLength(); i++) {
			NodeList objecttNodeList = nodeList.item(i).getChildNodes();
			for (int j = 0; j < objecttNodeList.getLength(); j++) {
				if (objecttNodeList
					.item(j)
					.getNodeName()
					.equalsIgnoreCase("object")) { //$NON-NLS-1$

					final Element e = (Element) objecttNodeList
						.item(j);
					String name = ""; //$NON-NLS-1$
					final Vector childObjectNames = new Vector();
					final Vector methods = new Vector();
					final Vector attributeNames = new Vector();
					if (e.hasAttribute("name")) { //$NON-NLS-1$
						name = e.getAttribute("name"); //$NON-NLS-1$
					}
					final NodeList childObjectNodes =
						e.getElementsByTagName("childObject");
							//$NON-NLS-1$
					final NodeList methodNodes =
						e.getElementsByTagName("method");
							//$NON-NLS-1$
					final NodeList attributeNodes =
						e.getElementsByTagName("attribute");
							//$NON-NLS-1$
```

```
//Zur besseren Lesbarkeit nach links gerückt
for (int k = 0; k < childObjectNodes.getLength(); k++) {
      final Element elem = (Element) childObjectNodes.item(k);
      if (elem.hasAttribute("name")) { //$NON-NLS-1$
            childObjectNames.add(elem.getAttribute("name")); //$NON-NLS-1$
      }
}
for (int l = 0; l < attributeNodes.getLength(); l++) {
      final Element elem = (Element) attributeNodes.item(l);
      if (elem.hasAttribute("name")) { //$NON-NLS-1$
            attributeNames.add(elem.getAttribute("name")); //$NON-NLS-1$
      }
}
for (int m = 0; m < methodNodes.getLength(); m++) {
      final Element elem = (Element) methodNodes.item(m);
      String methodName = ""; //$NON-NLS-1$
      String returnTypeName = ""; //$NON-NLS-1$
      boolean isVoid = true;

      if (elem.hasAttribute("name")) { //$NON-NLS-1$
            methodName = elem.getAttribute("name"); //$NON-NLS-1$
      }
      if (elem.hasAttribute("returnType")) { //$NON-NLS-1$
            returnTypeName = elem.getAttribute("returnType"); //$NON-NLS-1$
      }
      if (elem.hasAttribute("isVoid")) { //$NON-NLS-1$
            isVoid =(elem.getAttribute("isVoid") //$NON-NLS-1$
                        .equalsIgnoreCase("yes")) //$NON-NLS-1$
                        ? true
                        : false;
      }
      methods.add(
            new JavascriptMethod(
                  methodName,
                  returnTypeName,
                  isVoid));
}
//alle Eigenschaften des Objektes gesammelt. Objekt erstellen
final JavascriptMethod[] methodArray = new JavascriptMethod[methods.size()];
for (int k = 0; k < methodArray.length; k++) {
      methodArray[k] = (JavascriptMethod) methods.elementAt(k);
}

String[] childObjectNamesArray = new String[childObjectNames.size()];
for (int k = 0; k < childObjectNamesArray.length; k++) {
      childObjectNamesArray[k] = (String) childObjectNames.elementAt(k);
}

final String[] attributeNamesArray = new String[attributeNames.size()];
for (int k = 0; k < attributeNamesArray.length; k++) {
      attributeNamesArray[k] = (String) attributeNames.elementAt(k);
}

jsObjects.add(
      new JavascriptObject(
            name,
            methodArray,
            childObjectNamesArray,
            attributeNamesArray));
}
                  }
            }

      }

      private void doConstants(Document d) {
            final NodeList nodeList = d.getElementsByTagName("constants");
```

```
                //$NON-NLS-1$
        for (int i = 0; i < nodeList.getLength(); i++) {
                final NodeList constantNodeList = nodeList.item(i)
                        .getChildNodes();
                for (int j = 0; j < constantNodeList.getLength(); j++) {
                        if (constantNodeList
                                .item(j)
                                .getNodeName()
                                .equalsIgnoreCase("constant")) { //$NON-NLS-1$
                                final Element e = (Element) constantNodeList
                                        .item(j);
                                if (e.hasAttribute("name")) { //$NON-NLS-1$
                                        constants.add(e.getAttribute("name"));
                                                //$NON-NLS-1$
                                }
                        }
                }
        }
}

private void doKeywords(Document d) {
        final NodeList nodeList = d.getElementsByTagName("keywords");
                //$NON-NLS-1$
        for (int i = 0; i < nodeList.getLength(); i++) {
                NodeList keywordNodeList = nodeList.item(i).getChildNodes();
                for (int j = 0; j < keywordNodeList.getLength(); j++) {
                        if (keywordNodeList
                                .item(j)
                                .getNodeName()
                                .equalsIgnoreCase("keyword")) { //$NON-NLS-1$
                                Element e = (Element) keywordNodeList.item(j);
                                if (e.hasAttribute("name")) { //$NON-NLS-1$
                                        keywords.add(e.getAttribute("name"));
                                                //$NON-NLS-1$
                                }
                        }
                }
        }
}

private void doGlobalMethods(Document d) {
        final NodeList nodeList = d.getElementsByTagName("functions");
                //$NON-NLS-1$
        for (int i = 0; i < nodeList.getLength(); i++) {
                NodeList functionNodeList = nodeList.item(i).getChildNodes();
                for (int j = 0; j < functionNodeList.getLength(); j++) {
                        if (functionNodeList
                                .item(j)
                                .getNodeName()
                                .equalsIgnoreCase("method")) { //$NON-NLS-1$
                                Element e = (Element) functionNodeList.item(j);
                                String methodName = ""; //$NON-NLS-1$
                                String returnTypeName = ""; //$NON-NLS-1$
                                boolean isVoid = true;

                                if (e.hasAttribute("name")) { //$NON-NLS-1$
                                        methodName = e.getAttribute("name");
                                                //$NON-NLS-1$
                                }
                                if (e.hasAttribute("returnType")) { //$NON-NLS-1$
                                        returnTypeName = e
                                        .getAttribute("returnType"); //$NON-NLS-1$
                                }
                                if (e.hasAttribute("isVoid")) { //$NON-NLS-1$
                                        isVoid =
                                                (e.getAttribute("isVoid")
                                                        .equalsIgnoreCase("yes"))
                                                        //$NON-NLS-1$ //$NON-NLS-2$
```

```
                                                ? true
                                                : false;
                                    }
                                    globalMethods.add(
                                          new JavascriptMethod(
                                                methodName,
                                                returnTypeName,
                                                isVoid));
                              }
                        }
                  }
            }

            /**
             * Returns all keywords.
             * @return String[] all keywords
             */
            public String[] getKeywords() {
                  final String[] k = new String[keywords.size()];
                  for (int i = 0; i < k.length; i++) {
                        k[i] = (String) keywords.elementAt(i);
                  }
                  return k;
            }

            /**
             * Returns all constants.
             * @return String[] all constants
             */
            public String[] getConstants() {
                  final String[] c = new String[constants.size()];
                  for (int i = 0; i < c.length; i++) {
                        c[i] = (String) constants.elementAt(i);
                  }
                  return c;
            }

            /**
             * Returns all global methods.
             * @return Vector all global methods
             */
            public Vector getGlobalMethods() {
                  return globalMethods;
            }

            /**
             * Returns all javascript objects.
             * @return Vector all javascript objects
             */
            public Vector getJsObjects() {
                  return jsObjects;
            }
}
```

D.VI.8 JsWordDetector.java

```
package de.kt.scripteditor.javascript;

import org.eclipse.jface.text.rules.IWordDetector;

/**
 * A JavaScript word detector. JavaScript uses the same rules as Java does.
 * @author Karsten Thiemann
 */
public class JsWordDetector implements IWordDetector {

	/* (non-Javadoc)
	 * Method declared on IWordDetector.
	 */
	public boolean isWordPart(char character) {
		return Character.isJavaIdentifierPart(character);
	}

	/* (non-Javadoc)
	 * Method declared on IWordDetector.
	 */
	public boolean isWordStart(char character) {
		return Character.isJavaIdentifierStart(character);
	}
}
```

D.VII de.kt.scripteditor.javascripteditor

D.VII.1 JsActionContributor.java

```
package de.kt.scripteditor.javascripteditor;


import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.Separator;
import org.eclipse.ui.IActionBars;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.IWorkbenchActionConstants;
import org.eclipse.ui.editors.text.TextEditorActionContributor;
import org.eclipse.ui.texteditor.ITextEditor;
import org.eclipse.ui.texteditor.ITextEditorActionDefinitionIds;
import org.eclipse.ui.texteditor.RetargetTextEditorAction;

/**
 * Contributes interesting actions of the JavascriptEditor
 * to the desktop's Edit menu and the toolbar.
 * @author Karsten Thiemann
 */
public class JsActionContributor extends TextEditorActionContributor {

	protected final RetargetTextEditorAction contentAssistProposal;

	/**
	 * Default constructor.
	 */
	public JsActionContributor() {
		super();
		contentAssistProposal= new RetargetTextEditorAction(
			JsEditorMessages.getResourceBundle(),
			"ContentAssistProposal."); //$NON-NLS-1$
		contentAssistProposal.setActionDefinitionId
			(ITextEditorActionDefinitionIds.CONTENT_ASSIST_PROPOSALS);
	}

	/*
	 * @see IEditorActionBarContributor#init(IActionBars)
	 */
	public void init(IActionBars bars) {
		super.init(bars);
		final IMenuManager menuManager= bars.getMenuManager();
		final IMenuManager editMenu= menuManager
			.findMenuUsingPath(IWorkbenchActionConstants.M_EDIT);
		if (editMenu != null) {
			editMenu.add(new Separator());
			editMenu.add(contentAssistProposal);
		}
	}

	private void doSetActiveEditor(IEditorPart part) {
		super.setActiveEditor(part);
		ITextEditor editor= null;
		if (part instanceof ITextEditor)
			editor= (ITextEditor) part;
		contentAssistProposal.setAction(getAction(editor,
			"ContentAssistProposal")); //$NON-NLS-1$
	}
```

```
	/*
	 * @see IEditorActionBarContributor#setActiveEditor(IEditorPart)
	 */
	public void setActiveEditor(IEditorPart part) {
		super.setActiveEditor(part);
		doSetActiveEditor(part);
	}

	/*
	 * @see IEditorActionBarContributor#dispose()
	 */
	public void dispose() {
		doSetActiveEditor(null);
		super.dispose();
	}
}
```

D.VII.2 JsDocumentProvider.java

```
package de.kt.scripteditor.javascripteditor;

import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.IDocumentPartitioner;
import org.eclipse.jface.text.rules.DefaultPartitioner;
import org.eclipse.ui.editors.text.FileDocumentProvider;


/**
 * The document provider for *.js files.
 * @author Karsten Thiemann
 */
public class JsDocumentProvider extends FileDocumentProvider {
	/** legal content types */
	private final static String[] TYPES =
		new String[] {
			JsPartitionScanner.JS_COMMENT };
	private static JsPartitionScanner scanner = null;

	/**
	 * Constructor.
	 */
	public JsDocumentProvider() {
		super();
	}

	/* (non-Javadoc)
	 * Method declared on AbstractDocumentProvider
	 */
	protected IDocument createDocument(Object element) throws CoreException {
		final IDocument document = super.createDocument(element);
		if (document != null) {
			IDocumentPartitioner partitioner = createJsPartitioner();
			document.setDocumentPartitioner(partitioner);
			partitioner.connect(document);
		}
		return document;
	}

	/**
	 * Returns a partitioner for .js files.
	 */
	private IDocumentPartitioner createJsPartitioner() {
		return new DefaultPartitioner(getJsPartitionScanner(), TYPES);
	}

	/**
	 * Returns a scanner for creating js partitions.
	 */
	private JsPartitionScanner getJsPartitionScanner() {
		if (scanner == null) {
			scanner = new JsPartitionScanner();
		}
		return scanner;
	}
}
```

D.VII.3 JsEditor.java

```
package de.kt.scripteditor.javascripteditor;

import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.text.source.ISourceViewer;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.texteditor.ITextEditorActionDefinitionIds;
import org.eclipse.ui.texteditor.TextOperationAction;
import org.eclipse.ui.views.contentoutline.IContentOutlinePage;

import de.kt.scripteditor.javascripteditor.outline.JsContentOutlinePage;
import de.kt.scripteditor.util.AbstractScriptEditor;
import de.kt.scripteditor.util.SnipletLibrary;

/**
 * The editor for *.js files.
 * @author Karsten Thiemann
 */
public class JsEditor extends AbstractScriptEditor {
	/** The outline page */
	private JsContentOutlinePage outlinePage;

	public JsEditor() {
		super();
	}

	/**
	 * Sets the input of the outline page to null.
	 * @see org.eclipse.ui.IWorkbenchPart#dispose()
	 */
	public void dispose() {
		JsEditorEnvironment.disconnect(this);
		if (outlinePage != null)
			outlinePage.setInput(null);
		super.dispose();
	}

	/**
	 * Sets the input of the outline page.
	 * @see org.eclipse.ui.IWorkbenchPart#doSetInput()
	 */
	public void doSetInput(IEditorInput input) throws CoreException {
		super.doSetInput(input);
		if (outlinePage != null)
			outlinePage.setInput(input);
	}
	/** Returns the JsContentOutlinePage if request is for a an
	 * outline page.
	 */
	public Object getAdapter(Class required) {
		if (IContentOutlinePage.class.equals(required)) {
			if (outlinePage == null) {
				outlinePage =
					new JsContentOutlinePage(getDocumentProvider(),
						this);
				if (getEditorInput() != null)
					outlinePage.setInput(getEditorInput());
			}
			return outlinePage;
		}
		return super.getAdapter(required);
	}

	/* (non-Javadoc)
```

```
	 * Method declared on AbstractTextEditor
	 */
	protected void initializeEditor() {
		super.initializeEditor();
		JsEditorEnvironment.connect(this);
		setDocumentProvider(new JsDocumentProvider());
		setSourceViewerConfiguration(new JsSourceViewerConfiguration());
		setEditorContextMenuId("#JsEditorContext"); //$NON-NLS-1$
		setRulerContextMenuId("#JsRulerContext"); //$NON-NLS-1$
	}

	/* (non-Javadoc)
	 * @see org.eclipse.ui.texteditor.AbstractTextEditor#createActions()
	 */
	protected void createActions() {
		super.createActions();
		final IAction action = new TextOperationAction
(JsEditorMessages.getResourceBundle(), "ContentAssistProposal.", this,
ISourceViewer.CONTENTASSIST_PROPOSALS); //$NON-NLS-1$
		action.setActionDefinitionId(
			ITextEditorActionDefinitionIds.CONTENT_ASSIST_PROPOSALS);
		setAction("ContentAssistProposal", action); //$NON-NLS-1$

	}

	/* (non-Javadoc)
	 * @see de.kt.jseditor.util.AbstractScriptEditor#updateSniplets()
	 */
	protected void updateSniplets() {
		sniplets = SnipletLibrary.getJsSniplets();
	}
}
```

D.VII.4 JsEditorEnvironment.java

```
package de.kt.scripteditor.javascripteditor;

import org.eclipse.jface.text.rules.RuleBasedScanner;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.javascript.JsCodeScanner;
import de.kt.scripteditor.javascript.JsSyntax;
import de.kt.scripteditor.util.ColorProvider;
import de.kt.scripteditor.util.JsFunctionScanner;

/**
 * The JsEditorEnvironment maintains singletons used by the javascript editor.
 * @author Karsten Thiemann
 */
public class JsEditorEnvironment {

	private static ColorProvider colorProvider;
	private static JsCodeScanner codeScanner;
	private static JsFunctionScanner functionScanner;
	private static JsSyntax jsSyntax;

	private static int refCount = 0;

	/**
	 * Initializes the receiver if it is the first activation.
	 */
	public static void connect(Object client) {
		if (++refCount == 1) {
			colorProvider = new ColorProvider();
			codeScanner = new JsCodeScanner(colorProvider);
			functionScanner = new JsFunctionScanner();
			ScripteditorPlugin
				.getDefault()
				.getPreferenceStore()
				.addPropertyChangeListener(
				colorProvider);
		}
	}

	/**
	 * A disconnection has occured - clear the receiver if it is the last
	 * deactivation.
	 */
	public static void disconnect(Object client) {
		if (--refCount == 0) {
			codeScanner = null;
			ScripteditorPlugin
				.getDefault()
				.getPreferenceStore()
				.removePropertyChangeListener(
				colorProvider);
			colorProvider.dispose();
			colorProvider = null;
			functionScanner = null;
		}
	}

	/**
	 * Returns the singleton scanner.
	 */
	public static RuleBasedScanner getJsCodeScanner() {
		return codeScanner;
	}
```

```
	/**
	 * Returns the singleton function scanner.
	 */
	public static RuleBasedScanner getJsFunctionScanner() {
		return functionScanner;
	}

	/**
	 * Returns the singleton color provider.
	 */
	public static ColorProvider getColorProvider() {
		return colorProvider;
	}

}
```

D.VII.5 JsEditorMessages.java

```
package de.kt.scripteditor.javascripteditor;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

/**
 * The JsEditorMessages manages the access to the editors ressource bundle.
 * @author Karsten Thiemann
 */
public class JsEditorMessages {

	private static final String RESOURCE_BUNDLE =
		"de.kt.scripteditor.javascripteditor.JsEditorMessages";
			//$NON-NLS-1$

	private static final ResourceBundle resourceBundle =
		ResourceBundle.getBundle(RESOURCE_BUNDLE);
	/**
	 * private constructor to avoid instantiation.
	 */
	private JsEditorMessages() {
	}

	/**
	 * @param key the key of the stored string
	 * @return String the string with the given key
	 */
	public static String getString(String key) {
		try {
			return resourceBundle.getString(key);
		} catch (MissingResourceException e) {
			return "!" + key + "!"; //$NON-NLS-2$ //$NON-NLS-1$
		}
	}

	public static ResourceBundle getResourceBundle() {
		return resourceBundle;
	}
}
```

D.VII.6 JsPartitionScanner.java

```
package de.kt.scripteditor.javascripteditor;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.text.rules.EndOfLineRule;
import org.eclipse.jface.text.rules.IPredicateRule;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;
import org.eclipse.jface.text.rules.RuleBasedPartitionScanner;
import org.eclipse.jface.text.rules.Token;

/**
 * Partition scanner for *.js files. Specifies the different partitions
 * of the document (comments and default).
 * @author Karsten Thiemann
 */
public class JsPartitionScanner extends RuleBasedPartitionScanner {

      public final static String JS_COMMENT = "__js_comment"; //$NON-NLS-1$


  /**
   * Constructor.
   */
      public JsPartitionScanner() {
            super();
            final List rules = new ArrayList();

            final IToken jsComment = new Token(JS_COMMENT);

            // Add rule for single line comments.
            rules.add(new EndOfLineRule("//", jsComment)); //$NON-NLS-1$

            // Add rule for multi-line comments.
            rules.add(new MultiLineRule("/*", "*/", jsComment, (char) 0, true));
                  //$NON-NLS-1$ //$NON-NLS-2$

            final IPredicateRule[] result = new IPredicateRule[rules.size()];
            rules.toArray(result);
            setPredicateRules(result);
      }
}
```

D.VII.7 JsSourceViewerConfiguration.java

```
package de.kt.scripteditor.javascripteditor;

import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.ITextDoubleClickStrategy;
import org.eclipse.jface.text.TextAttribute;
import org.eclipse.jface.text.contentassist.ContentAssistant;
import org.eclipse.jface.text.contentassist.IContentAssistant;
import org.eclipse.jface.text.presentation.IPresentationReconciler;
import org.eclipse.jface.text.presentation.PresentationReconciler;
import org.eclipse.jface.text.rules.BufferedRuleBasedScanner;
import org.eclipse.jface.text.rules.DefaultDamagerRepairer;
import org.eclipse.jface.text.rules.Token;
import org.eclipse.jface.text.source.ISourceViewer;
import org.eclipse.jface.text.source.SourceViewerConfiguration;
import org.eclipse.swt.graphics.RGB;

import de.kt.scripteditor.javascript.JsCompletionProcessor;
import de.kt.scripteditor.util.ColorProvider;
import de.kt.scripteditor.util.ScriptEditorDoubleClickStrategy;

/**
 * SourceViewerConfiguration used by the JsEditor.
 * @author Karsten Thiemann
 */
public class JsSourceViewerConfiguration extends SourceViewerConfiguration {

	/**
	 * Default constructor.
	 */
	public JsSourceViewerConfiguration() {
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public String[] getConfiguredContentTypes(ISourceViewer sourceViewer) {
		return new String[] {
			IDocument.DEFAULT_CONTENT_TYPE,
			JsPartitionScanner.JS_COMMENT};
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public IContentAssistant getContentAssistant(ISourceViewer sourceViewer) {

		final ContentAssistant assistant = new ContentAssistant();
		assistant.setContentAssistProcessor(
			new JsCompletionProcessor(),
			IDocument.DEFAULT_CONTENT_TYPE);

		assistant.enableAutoActivation(true);
		assistant.setAutoActivationDelay(500);
		assistant.setProposalPopupOrientation(
			IContentAssistant.PROPOSAL_OVERLAY);
		assistant.setContextInformationPopupOrientation(
			IContentAssistant.CONTEXT_INFO_ABOVE);
		assistant.setContextInformationPopupBackground(
			JsEditorEnvironment.getColorProvider().getColor(
				new RGB(150, 150, 0)));

		return assistant;

	}
```

```
	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public ITextDoubleClickStrategy getDoubleClickStrategy(
			ISourceViewer sourceViewer,
			String contentType) {
			return new ScriptEditorDoubleClickStrategy();
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public String[] getIndentPrefixes(
			ISourceViewer sourceViewer,
			String contentType) {
			return new String[] { "\t", "    " };  //$NON-NLS-1$ //$NON-NLS-2$
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public IPresentationReconciler getPresentationReconciler(ISourceViewer
			sourceViewer) {

			final ColorProvider provider =
				JsEditorEnvironment.getColorProvider();
			final PresentationReconciler reconciler =
				new PresentationReconciler();

			DefaultDamagerRepairer dr =
				new DefaultDamagerRepairer(
					JsEditorEnvironment.getJsCodeScanner());
			reconciler.setDamager(dr, IDocument.DEFAULT_CONTENT_TYPE);
			reconciler.setRepairer(dr, IDocument.DEFAULT_CONTENT_TYPE);

			dr =
				new DefaultDamagerRepairer(
					new SingleTokenScanner(
						new TextAttribute(
							provider
							.getColor(ColorProvider.JS_COMMENT))));
			reconciler.setDamager(dr, JsPartitionScanner.JS_COMMENT);
			reconciler.setRepairer(dr, JsPartitionScanner.JS_COMMENT);

			return reconciler;
	}

	/* (non-Javadoc)
	 * Method declared on SourceViewerConfiguration
	 */
	public int getTabWidth(ISourceViewer sourceViewer) {
			return 4;
	}

	/**
	 * Single token scanner.
	 */
	static class SingleTokenScanner extends BufferedRuleBasedScanner {
			public SingleTokenScanner(TextAttribute attribute) {
				setDefaultReturnToken(new Token(attribute));
			}
	};

}
```

D.VII.8 JsEditorMessages.properties

```
# Actions

ContentAssistProposal.label=Content Assist@Ctrl+SPACE
ContentAssistProposal.tooltip=Content Assist
ContentAssistProposal.description=Content Assist
```

D.VII.9 JsEditorMessages_de_DE.properties

```
# Actions

ContentAssistProposal.label=Content Assist@Ctrl+SPACE
ContentAssistProposal.tooltip=Content Assist
ContentAssistProposal.description=Content Assist
```

D.VIII de.kt.scripteditor.javascripteditor.outline

D.VIII.1 JsContentOutlinePage.java

```
package de.kt.scripteditor.javascripteditor.outline;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.BadPositionCategoryException;
import org.eclipse.jface.text.DefaultPositionUpdater;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.IDocumentListener;
import org.eclipse.jface.text.IPositionUpdater;
import org.eclipse.jface.text.Position;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.RuleBasedScanner;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.viewers.ITreeContentProvider;
import org.eclipse.jface.viewers.SelectionChangedEvent;
import org.eclipse.jface.viewers.TreeViewer;
import org.eclipse.jface.viewers.Viewer;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.ui.texteditor.IDocumentProvider;
import org.eclipse.ui.texteditor.ITextEditor;
import org.eclipse.ui.views.contentoutline.ContentOutlinePage;

import de.kt.scripteditor.javascripteditor.JsEditorEnvironment;
import de.kt.scripteditor.util.AbstractScriptContentProvider;
import de.kt.scripteditor.util.IScriptContentProvider;
import de.kt.scripteditor.util.IScriptEditorOutlinePage;
import de.kt.scripteditor.util.JsFunctionScanner;
import de.kt.scripteditor.util.OutlineDocumentListener;

/**
 * The outline page of the JsEditor.
 * @author Karsten Thiemann
 */
public class JsContentOutlinePage
      extends ContentOutlinePage
      implements IScriptEditorOutlinePage {

      /**
       * Parses the editors input.
       */
      protected class ContentProvider
            extends AbstractScriptContentProvider
            implements ITreeContentProvider, IScriptContentProvider {

            protected final static String JS_OUTLINE_ELEMENTS =
                  "__javascript_outline_elements"; //$NON-NLS-1$
            protected final IPositionUpdater positionUpdater =
                  new DefaultPositionUpdater(JS_OUTLINE_ELEMENTS);
            protected List content = new ArrayList(10);

            /* (non-Javadoc)
             * @see de.kt.jseditor.util.IScriptContentProvider#parse
             * (org.eclipse.jface.text.IDocument)
             */
            public void parse(IDocument document) {
                  final RuleBasedScanner scanner =
                        JsEditorEnvironment.getJsFunctionScanner();
```

```
                scanner.setRange(document, 0, document.getLength());
                IToken token = scanner.nextToken();

//Zur besseren Lesbarkeit nach links gerückt
try {
      while (!token.isEOF()) {
            int lastFunctionOffset = 0;
            if (token.equals(JsFunctionScanner.JS_FUNCTION_TOKEN)) {
                  final int offset = scanner.getTokenOffset();
                  final int length = scanner.getTokenLength();
                  final String funcName = getFunctionName(
                        document.get(offset, length));
                  content.add(
                        new JsOutlineElement(
                              funcName,
                              new Position(offset, length),
                              JsOutlineElement.FUNCTION));
                  lastFunctionOffset = offset;
            } else if (
                  token.equals(JsFunctionScanner.JS_FIELD_TOKEN)) {
                  final int offset = scanner.getTokenOffset();
                  if (isGlobalField(document.get(
                        lastFunctionOffset,
                        offset - lastFunctionOffset))) {
                        final int length = scanner.getTokenLength();
                        final String fieldName = getVarName(document.get(offset,
                              length));

                        content.add(
                        new JsOutlineElement(
                              fieldName,
                              new Position(offset, length),
                              JsOutlineElement.FIELD));
                  }
            }
            token = scanner.nextToken();
      }
} catch (BadLocationException e) {
      // do nothing
}
            }

            /*
             * @see IContentProvider#inputChanged(Viewer, Object, Object)
             */
            public void inputChanged(
                  Viewer viewer,
                  Object oldInput,
                  Object newInput) {
                  if (oldInput != null) {
                        final IDocument document =
                              documentProvider.getDocument(oldInput);
                        if (document != null) {
                              try {
                                    document.removePositionCategory
                                          (JS_OUTLINE_ELEMENTS);
                              } catch (BadPositionCategoryException x) {
                                    // do nothing
                              }
                              document.removePositionUpdater(positionUpdater);
                              document.removeDocumentListener(listener);
                        }
                  }
                  content.clear();
                  if (newInput != null) {
                        final IDocument document =
                              documentProvider.getDocument(newInput);
```

```
                        if (document != null) {
                              document.addPositionCategory(JS_OUTLINE_ELEMENTS);
                              document.addPositionUpdater(positionUpdater);
                              document.addDocumentListener(listener);
                              parse(document);
                        }
                  }

            }

            /*
             * @see IContentProvider#dispose
             */
            public void dispose() {
                  if (content != null) {
                        content.clear();
                        content = null;
                  }
            }

            /*
             * @see IContentProvider#isDeleted(Object)
             */
            public boolean isDeleted(Object element) {
                  return false;
            }

            /*
             * @see IStructuredContentProvider#getElements(Object)
             */
            public Object[] getElements(Object element) {
                  return content.toArray();
            }

            /*
             * @see ITreeContentProvider#hasChildren(Object)
             */
            public boolean hasChildren(Object element) {
                  return element == input;
            }

            /*
             * @see ITreeContentProvider#getParent(Object)
             */
            public Object getParent(Object element) {
                  if (element instanceof JsOutlineElement)
                        return input;
                  return null;
            }

            /*
             * @see ITreeContentProvider#getChildren(Object)
             */
            public Object[] getChildren(Object element) {
                  if (element == input)
                        return content.toArray();
                  return new Object[0];
            }

      };

      protected Object input;
      protected IDocumentProvider documentProvider;
      protected ITextEditor textEditor;
      protected ContentProvider contentProvider;
      protected IDocumentListener listener;
```

```
/**
 * Creates a content outline page using the given provider
 * and the given editor.
 * @param provider the document provider
 * @param editor the editor
 */
public JsContentOutlinePage(
      IDocumentProvider provider,
      ITextEditor editor) {
      super();
      documentProvider = provider;
      textEditor = editor;
      contentProvider = new ContentProvider();
      listener =
            new OutlineDocumentListener(contentProvider,
                  textEditor, this);
}

/* (non-Javadoc)
 * Method declared on ContentOutlinePage
 */
public void createControl(Composite parent) {

      super.createControl(parent);

      final TreeViewer viewer = getTreeViewer();
      viewer.setContentProvider(contentProvider);
      viewer.setLabelProvider(new JsOutlineContentLabelProvider());
      viewer.addSelectionChangedListener(this);

      if (input != null) {
            viewer.setInput(input);
            documentProvider.getDocument(input)
                  .addDocumentListener(listener);
      }
}

/* (non-Javadoc)
 * Method declared on ContentOutlinePage
 */
public void selectionChanged(SelectionChangedEvent event) {

      super.selectionChanged(event);

      final ISelection selection = event.getSelection();
      if (selection.isEmpty())
            textEditor.resetHighlightRange();
      else {
            JsOutlineElement jsFunction =
                  (JsOutlineElement) ((IStructuredSelection) selection)
                        .getFirstElement();
            int start = jsFunction.position.getOffset();
            int length = jsFunction.position.getLength();
            try {
                  textEditor.setHighlightRange(start, length, true);
            } catch (IllegalArgumentException x) {
                  textEditor.resetHighlightRange();
            }
      }
}

/**
 * Sets the input of the outline page
 */
public void setInput(Object newInput) {
      input = newInput;
      update();
```

```
	}

	/**
	 * The dispose of the JsContentOutlinePage removes the
	 * document listener from the
	 * actual document.
	 */
	public void dispiose() {
		if (input != null) {
			documentProvider.getDocument(input).removeDocumentListener(
				listener);
		}
		super.dispose();
	}

	/**
	 * Updates the outline page.
	 */
	public void update() {
		TreeViewer viewer = getTreeViewer();

		if (viewer != null) {
			final Control control = viewer.getControl();
			if (control != null && !control.isDisposed()) {
				control.setRedraw(false);
				viewer.setInput(input);
				viewer.expandAll();
				control.setRedraw(true);
			}
		}
	}
}
```

D.VIII.2 JsOutlineContentLabelProvider.java

```
package de.kt.scripteditor.javascripteditor.outline;

import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Image;

import de.kt.scripteditor.util.EditorImages;

/**
 * Manages the images for the JavasciptEditors outline page.
 * @author Karsten Thiemann
 * @since 02.11.2003
 */
public class JsOutlineContentLabelProvider extends LabelProvider{

	public JsOutlineContentLabelProvider() {
	}

	public Image getImage(Object element) {
		if (element instanceof JsOutlineElement) {
			int t = ((JsOutlineElement) element).getType();
			if (t == JsOutlineElement.FIELD){
				return EditorImages.getImage(EditorImages.ICON_FIELD);
			}
			if (t == JsOutlineElement.FUNCTION){
				return EditorImages.getImage
					(EditorImages.ICON_FUNCTION);
			}
		}
		return null;
	}

}
```

D.VIII.3 JsOutlineElement.java

```
package de.kt.scripteditor.javascripteditor.outline;

import org.eclipse.jface.text.Position;


/**
 * JsOutlineElements are used in the outlines function list.
 * @author Karsten Thiemann
 */
class JsOutlineElement {
	public static final int FUNCTION = 1;
	public static final int FIELD = 2;

	public final String name;
	public final Position position;
	public final int type;

	/**
	 * Constructor.
	 * @param name the name
	 * @param position the position
	 */
	public JsOutlineElement(String name, Position position, int type) {
		this.name = name;
		this.position = position;
		this.type = type;

	}

	public String toString() {
		return name;
	}
	/**
	 * Returns the type of the JsOutlineElement
	 * @return int the elements type
	 */
	public int getType() {
		return type;
	}

}
```

D.IX de.kt.scripteditor.preferences

D.IX.1 ColorPreferencePage.java

```
package de.kt.scripteditor.preferences;

import org.eclipse.jface.preference.BooleanFieldEditor;
import org.eclipse.jface.preference.ColorFieldEditor;
import org.eclipse.jface.preference.FieldEditorPreferencePage;
import org.eclipse.jface.preference.IPreferenceStore;
import org.eclipse.jface.preference.PreferenceConverter;
import org.eclipse.swt.graphics.RGB;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchPreferencePage;

import de.kt.scripteditor.ScripteditorPlugin;


/**
 * PreferencePage for all colors used by the plugin.
 * @author Karsten Thiemann
 */
public class ColorPreferencePage
      extends FieldEditorPreferencePage
      implements IWorkbenchPreferencePage, IPreferenceIDs {

      public ColorPreferencePage() {
            super(GRID);
            setPreferenceStore(ScripteditorPlugin.getDefault()
                  .getPreferenceStore());
            setDescription(PreferenceMessages
                  .getString("ColorPreferencePage.Description")); //$NON-NLS-1$
            initializeDefaults();
      }
      /**
       * Sets the default values of the preferences.
       */
      public static void initializeDefaults() {
            final IPreferenceStore store = ScripteditorPlugin.getDefault()
                  .getPreferenceStore();
            PreferenceConverter.setDefault(store, JS_COMMENT_COLOR,
                  new RGB(128,128,0));
            PreferenceConverter.setDefault(store, HTML_COMMENT_COLOR,
                  new RGB(220,220,0));
            PreferenceConverter.setDefault(store, KEYWORD_COLOR,
                  new RGB(0,0,220));
            PreferenceConverter.setDefault(store, CONSTANT_COLOR,
                  new RGB(0,128,128));
            PreferenceConverter.setDefault(store, STRING_COLOR,
                  new RGB(0,200,0));
            PreferenceConverter.setDefault(store, DEFAULT_COLOR,
                  new RGB(0,0,0));
            PreferenceConverter.setDefault(store, HTML_TAG_COLOR,
                  new RGB(220,0,220));
            store.setDefault(CLOSE_TAG, true);
            store.setDefault(TAGS_TO_UPPERCASE, false);
      }

      /**
       * Creates the field editors.
       */

      public void createFieldEditors() {
            addField(
                  new BooleanFieldEditor(
```

```
                        CLOSE_TAG,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.Close_Tags"),
                        getFieldEditorParent()));
            addField(
                  new BooleanFieldEditor(
                        TAGS_TO_UPPERCASE,
                        PreferenceMessages.getString
                        ("ColorPreferencePage.Tags_to_upper_case"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        JS_COMMENT_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.Script_Comment"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        HTML_COMMENT_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.HTML_Comment"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        KEYWORD_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.Keyword"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        CONSTANT_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.Constant"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        STRING_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.String"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        DEFAULT_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.Text"),
                        getFieldEditorParent()));
            addField(
                  new ColorFieldEditor(
                        HTML_TAG_COLOR,
                        PreferenceMessages.getString
                              ("ColorPreferencePage.HTML_Tag"),
                        getFieldEditorParent()));
      }

      /* (non-Javadoc)
       * @see org.eclipse.ui.IWorkbenchPreferencePage#init
       * (org.eclipse.ui.IWorkbench)
       */
      public void init(IWorkbench workbench) {
            //do nothing
      }
}
```

D.IX.2 FilterPreferencePage.java

```
package de.kt.scripteditor.preferences;

import org.eclipse.jface.preference.BooleanFieldEditor;
import org.eclipse.jface.preference.FieldEditorPreferencePage;
import org.eclipse.jface.preference.IPreferenceStore;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchPreferencePage;

import de.kt.scripteditor.ScripteditorPlugin;

/**
 * PreferencePage for the HTML-OutlineView filter.
 * @author Karsten Thiemann
 */
public class FilterPreferencePage
      extends FieldEditorPreferencePage
      implements IWorkbenchPreferencePage, IPreferenceIDs {
      /**
       * Constructor.
       */
      public FilterPreferencePage() {
            super(GRID);
            setPreferenceStore(ScripteditorPlugin.getDefault()
                  .getPreferenceStore());
            setDescription(PreferenceMessages.getString
                  ("FilterPreferencePage.Description")); //$NON-NLS-1$
            initializeDefaults();
      }

      /**
       * Sets the default values of the preferences.
       */
      public static void initializeDefaults() {
            final IPreferenceStore store = ScripteditorPlugin.getDefault()
                  .getPreferenceStore();
            store.setDefault(A, false);
            store.setDefault(BODY, true);
            store.setDefault(BUTTON, false);
            store.setDefault(DIV, true);
            store.setDefault(FORM, true);
            store.setDefault(FRAME, false);
            store.setDefault(FRAMESET, false);
            store.setDefault(H1, false);
            store.setDefault(IFRAME, false);
            store.setDefault(IMG, false);
            store.setDefault(INPUT, false);
            store.setDefault(LI, false);
            store.setDefault(LINK, false);
            store.setDefault(MAP, false);
            store.setDefault(OBJECT, false);
            store.setDefault(OL, false);
            store.setDefault(OPTION, false);
            store.setDefault(P, false);
            store.setDefault(SELECT, false);
            store.setDefault(SPAN, false);
            store.setDefault(STYLE, false);
            store.setDefault(TABLE, true);
            store.setDefault(TD, false);
            store.setDefault(TEXTAREA, false);
            store.setDefault(TR, false);
            store.setDefault(UL, false);
      }
```

```
/**
 * Creates the field editors.
 */
public void createFieldEditors() {
      addField(new BooleanFieldEditor(A, "A", getFieldEditorParent()));
      addField(new BooleanFieldEditor(BODY, "BODY",
            getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(BUTTON, "BUTTON",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(DIV, "DIV",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(FORM, "FORM",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(FRAME, "FRAME",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(
                  FRAMESET,
                  "FRAMESET", //$NON-NLS-1$
                  getFieldEditorParent()));
      addField(new BooleanFieldEditor(H1, "H1",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(IFRAME, "IFRAME",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(IMG, "IMG",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(INPUT, "INPUT",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(LI, "LI",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(LINK, "LINK",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(MAP, "MAP",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(OBJECT, "OBJECT",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(OL, "OL",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(OPTION, "OPTION",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(P, "P",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(SELECT, "SELECT",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(SPAN, "SPAN",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(STYLE, "STYLE",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(TABLE, "TABLE",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(new BooleanFieldEditor(TD, "TD",
                  getFieldEditorParent())); //$NON-NLS-1$
      addField(
            new BooleanFieldEditor(
                  TEXTAREA,
                  "TEXTAREA", //$NON-NLS-1$
                  getFieldEditorParent()));
```

```
            addField(new BooleanFieldEditor(TR, "TR",
                  getFieldEditorParent())); //$NON-NLS-1$
            addField(new BooleanFieldEditor(UL, "UL", g
                  etFieldEditorParent())); //$NON-NLS-1$
            final GridLayout layout = new GridLayout();
            layout.numColumns = 2;
            layout.marginHeight = 0;
            layout.marginWidth = 0;
            getFieldEditorParent().setLayout(layout);
      }

      /* (non-Javadoc)
       * @see org.eclipse.ui.IWorkbenchPreferencePage#init
       * (org.eclipse.ui.IWorkbench)
       */
      public void init(IWorkbench workbench) {
      }
}
```

D.IX.3 IPreferenceIDs .java

```
package de.kt.scripteditor.preferences;

/**
 * Interface with all IDs used in the preferences.
 * @author Karsten Thiemann
 */
public interface IPreferenceIDs {

	public static String JS_COMMENT_COLOR = "JS_COMMENT_COLOR"; //$NON-NLS-1$
	public static String HTML_COMMENT_COLOR = "HTML_COMMENT_COLOR";
		//$NON-NLS-1$
	public static String KEYWORD_COLOR = "KEYWORD_COLOR"; //$NON-NLS-1$
	public static String CONSTANT_COLOR = "CONSTANT_COLOR"; //$NON-NLS-1$
	public static String STRING_COLOR = "STRING_COLOR"; //$NON-NLS-1$
	public static String DEFAULT_COLOR = "DEFAULT_COLOR"; //$NON-NLS-1$
	public static String HTML_TAG_COLOR = "HTML_TAG_COLOR"; //$NON-NLS-1$

	public static final String CLOSE_TAG = "CLOSE_TAG"; //$NON-NLS-1$
	public static final String TAGS_TO_UPPERCASE = "TAGS_TO_UPPERCASE";
		//$NON-NLS-1$

	public static final String A = "A"; //$NON-NLS-1$
	public static final String BODY = "BODY"; //$NON-NLS-1$
	public static final String BUTTON = "BUTTON"; //$NON-NLS-1$
	public static final String DIV = "DIV"; //$NON-NLS-1$
	public static final String FORM = "FORM"; //$NON-NLS-1$
	public static final String FRAME = "FRAME"; //$NON-NLS-1$
	public static final String FRAMESET = "FRAMESET"; //$NON-NLS-1$
	public static final String H1 = "H1"; //$NON-NLS-1$
	public static final String IFRAME = "IFRAME"; //$NON-NLS-1$
	public static final String IMG = "IMG"; //$NON-NLS-1$
	public static final String INPUT = "INPUT"; //$NON-NLS-1$
	public static final String LI = "LI"; //$NON-NLS-1$
	public static final String LINK = "LINK"; //$NON-NLS-1$
	public static final String MAP = "MAP"; //$NON-NLS-1$
	public static final String OBJECT = "OBJECT"; //$NON-NLS-1$
	public static final String OL = "OL"; //$NON-NLS-1$
	public static final String OPTION = "OPTION"; //$NON-NLS-1$
	public static final String P = "P"; //$NON-NLS-1$
	public static final String SELECT = "SELECT"; //$NON-NLS-1$
	public static final String SPAN = "SPAN"; //$NON-NLS-1$
	public static final String STYLE = "STYLE"; //$NON-NLS-1$
	public static final String TABLE = "TABLE"; //$NON-NLS-1$
	public static final String TD = "TD"; //$NON-NLS-1$
	public static final String TEXTAREA = "TEXTAREA"; //$NON-NLS-1$
	public static final String TR = "TR"; //$NON-NLS-1$
	public static final String UL = "UL"; //$NON-NLS-1$
}
```

D.IX.4 PreferenceMessages.java

```
package de.kt.scripteditor.preferences;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

/**
 * The PreferenceMessages manages the access to the prefereces ressource bundle.
 * @author Karsten Thiemann
 * @since 23.10.2003
 */
public class PreferenceMessages {

	private static final String BUNDLE_NAME =
		"de.kt.scripteditor.preferences.PreferenceMessages"; //$NON-NLS-1$

	private static final ResourceBundle RESOURCE_BUNDLE =
		ResourceBundle.getBundle(BUNDLE_NAME);

	/**
	 * private constructor to avoid instantiation.
	 */
	private PreferenceMessages() {
	}

	/**
	 * @param key the key of the stored string
	 * @return String the string with the given key
	 */
	public static String getString(String key) {
		try {
			return RESOURCE_BUNDLE.getString(key);
		} catch (MissingResourceException e) {
			return '!' + key + '!';
		}
	}
}
```

D.IX.5 PreferenceMessages.properties

```
ColorPreferencePage.Description=A preference page to set the highlight colors
ColorPreferencePage.Close_Tags=Close Tags
ColorPreferencePage.Tags_to_upper_case=Tags to upper case
ColorPreferencePage.Script_Comment=Script Comment
ColorPreferencePage.HTML_Comment=HTML Comment
ColorPreferencePage.Keyword=Keyword
ColorPreferencePage.Constant=Constant
ColorPreferencePage.String=String
ColorPreferencePage.Text=Text
ColorPreferencePage.HTML_Tag=HTML Tag

FilterPreferencePage.Description=A preference page to set the outline filter
```

D.IX.6 PreferenceMessages_de_DE.properties

```
ColorPreferencePage.Description=Setzen der Syntax-Farben
ColorPreferencePage.Close_Tags=Schliessendes Tag hinzufügen
ColorPreferencePage.Tags_to_upper_case=Tags in Grossbuchstaben
ColorPreferencePage.Script_Comment=Script-Kommentar
ColorPreferencePage.HTML_Comment=HTML-Kommentar
ColorPreferencePage.Keyword=Schlüsselwort
ColorPreferencePage.Constant=Konstante
ColorPreferencePage.String=String
ColorPreferencePage.Text=Text
ColorPreferencePage.HTML_Tag=HTML Tag

FilterPreferencePage.Description=Auswahl der zu filternden Tags
```

D.X de.kt.scripteditor.util

D.X.1 AbstractScriptContentProvider.java

```
package de.kt.scripteditor.util;

/**
 * Includes Methods to extract function and field names from the editorcontent.
 * @author Karsten Thiemann
 */
public abstract class AbstractScriptContentProvider {

	/**
	 * Returns the state of the given field.
	 * @param string the name of the field
	 * @return boolean true if the given field is a global one
	 */
	protected boolean isGlobalField(String string) {
		final char[] characters = string.toCharArray();
		int bracketCount = 0;
		for (int i = characters.length - 1; i >= 0; i--) {
			if (characters[i] == '}') {
				bracketCount--;
			} else if (characters[i] == '{') {
				bracketCount++;
			}
		}
		return (bracketCount <= 0);
	}

	/**
	 * Returns the name of the field in the given line of code.
	 * @param string the declaration line
	 * @return String the name of the field
	 */
	protected String getVarName(String string) {
		final char[] characters =
			string
				.substring(string.indexOf("var ") + 4, string.length())
				.toCharArray();
		final StringBuffer name = new StringBuffer(string.length());
		boolean start = false;
		for (int i = 0; i < characters.length; i++) {
			if (Character.isJavaIdentifierPart(characters[i])) {
				name.append(characters[i]);
				start = true;
			}
			if (start && !Character.isJavaIdentifierPart(characters[i])) {
				return name.toString();
			}
		}
		return string;
	}

	/**
	 * Returns the name of the function in the given line of code.
	 * @param string the matched string
	 * @return String the name of the function
	 */
	protected String getFunctionName(String string) {
		return string
			.substring(string.indexOf("function") + 8, string.length())
			.trim();
	}
}
```

D.X.2 AbstractScriptEditor.java

```
package de.kt.scripteditor.util;

import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.MenuManager;
import org.eclipse.jface.action.Separator;
import org.eclipse.ui.editors.text.TextEditor;

/**
 * Abstract Superclass for ScriptEditors. Provides sniplet
 * actions in the context menu.
 * @author Karsten Thiemann
 */
public abstract class AbstractScriptEditor
      extends TextEditor
      implements IScriptEditor {

      protected CodeSniplet[] sniplets;

      /** (non-Javadoc)
      * Method declared on AbstractTextEditor
      */
      protected void editorContextMenuAboutToShow(IMenuManager parentMenu) {
            super.editorContextMenuAboutToShow(parentMenu);
            updateSniplets();
            final IMenuManager subMenuInsert = new MenuManager(
                  UtilMessages.getString
                        ("AbstractScriptEditor.Insert_Sniplets")); //$NON-NLS-1$
            final IMenuManager subMenuDelete = new MenuManager(
                  UtilMessages.getString
                        ("AbstractScriptEditor.Delete_Sniplets")); //$NON-NLS-1$

            parentMenu.add(new Separator());
            //TODO ins gleiche menu wie saveSniplet...
            if (subMenuInsert != null) {
                  parentMenu.add(subMenuInsert);
                  for (int i = 0; i < sniplets.length; i++) {
                        final InsertSnipletAction insertSnipletAction =
                              new InsertSnipletAction(sniplets[i]);
                        insertSnipletAction.setActiveEditor(this);
                        subMenuInsert.add(insertSnipletAction);
                  }
                  if (sniplets.length == 0) {
                        subMenuInsert.setVisible(false);
                  }
            }
            if (subMenuDelete != null) {
                  parentMenu.add(subMenuDelete);
                  for (int i = 0; i < sniplets.length; i++) {
                        final DeleteSnipletAction deleteSnipletAction =
                              new DeleteSnipletAction(sniplets[i]);
                        deleteSnipletAction.setActiveEditor(this);
                        subMenuDelete.add(deleteSnipletAction);
                  }
                  if (sniplets.length == 0) {
                        subMenuDelete.setVisible(false);
                  }
            }
      }
```

```
	/**
	 * Updates the content of the sniplet Vector.
	 */
	protected abstract void updateSniplets();

	/* (non-Javadoc)
	 * @see de.kt.jseditor.util.IScriptEditor#getCursorOffset()
	 */
	public int getCursorOffset() {
		return getSourceViewer().getSelectedRange().x;
	}
}
```

D.X.3 AbstractXMLParser.java

```
package de.kt.scripteditor.util;

import java.io.IOException;

import org.apache.xerces.parsers.DOMParser;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.widgets.Shell;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXException;

/**
 * Abstract superclass for all syntax parsers.
 * @author Karsten Thiemann
 */
public abstract class AbstractXMLParser {

	/**
	 * Instantiates a DOMParser and parses the given xml-document.
	 * @param uri the URI of the xml-document to parse
	 */
	public void parseDocument(String uri) {
		//Der Parser wird erstellt
		final DOMParser parser = new DOMParser();
		try {
			//Whitespaces werden ignoriert
			parser.setFeature(
				"http://apache.org/xml/features/dom/include-ignorable-
					whitespace", //$NON-NLS-1$
				false);
			//Das Dokument wird geparst
			parser.parse(uri);
			//Ein Document-Objekt wird erzeugt
			final Document d = parser.getDocument();
			//Root-Objekt wird erstellt
			final Element root = d.getDocumentElement();

			doContent(d);
		}
		catch (SAXException e) {
			showException(e);
		} catch (IOException e) {
			showException(e);
		}
	}

	/**
	 * In the doContent method, the parsing has to be done.
	 * @param d the Document
	 */
	protected void doContent(Document d){};

	/**
	 * Shows an exception in a MessageDialog.
	 * @param e the exception to show
	 */
	protected void showException(Exception e) {
		MessageDialog.openError(new Shell(),
			e.getMessage(),
			e.getMessage());
	}
}
```

D.X.4 CodeSniplet.java

```
package de.kt.scripteditor.util;

/**
 * A CodeSniplet is a named html or javascript fragment. It can be inserted.
 * @author Karsten Thiemann
 */
public class CodeSniplet {
	private final String name;
	private final String content;
	/**
	 * Constructor for CodeSniplet.
	 * @param name
	 * @param content
	 */
	public CodeSniplet(String name, String content) {
		this.name = name;
		this.content = content;
	}

	/**
	 * Gets the content (the code sniplet).
	 * @return String the content
	 */
	public String getContent() {
		return content;
	}

	/**
	 * Gets the name.
	 * @return String the name
	 */
	public String getName() {
		return name;
	}

}
```

D.X.5 ColorProvider.java

```
package de.kt.scripteditor.util;

import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

import org.eclipse.jface.preference.IPreferenceStore;
import org.eclipse.jface.preference.PreferenceConverter;
import org.eclipse.jface.util.IPropertyChangeListener;
import org.eclipse.jface.util.PropertyChangeEvent;
import org.eclipse.swt.graphics.Color;
import org.eclipse.swt.graphics.RGB;
import org.eclipse.swt.widgets.Display;

import de.kt.scripteditor.ScripteditorPlugin;
import de.kt.scripteditor.preferences.IPreferenceIDs;

/**
 * Manages the Colors, used by the plugin.
 * @author Karsten Thiemann
 */
public class ColorProvider implements IPropertyChangeListener, IPreferenceIDs {

	public static RGB JS_COMMENT = new RGB(128, 128, 0);
	public static RGB HTML_COMMENT = new RGB(220, 220, 0);
	public static RGB KEYWORD = new RGB(0, 0, 220);
	public static RGB CONSTANT = new RGB(0, 128, 128);
	public static RGB STRING = new RGB(0, 128, 0);
	public static RGB DEFAULT = new RGB(0, 0, 0);
	public static RGB HTML_TAG = new RGB(220, 0, 220);

	protected Map colorTable = new HashMap(10);
	private IPreferenceStore store;

	/**
	 * Constructor.
	 */
	public ColorProvider() {

		store = ScripteditorPlugin.getDefault().getPreferenceStore();
		try {
			JS_COMMENT = getStoredRGB(JS_COMMENT_COLOR);
			HTML_COMMENT = getStoredRGB(HTML_COMMENT_COLOR);
			KEYWORD = getStoredRGB(KEYWORD_COLOR);
			CONSTANT = getStoredRGB(CONSTANT_COLOR);
			STRING = getStoredRGB(STRING_COLOR);
			DEFAULT = getStoredRGB(DEFAULT_COLOR);
			HTML_TAG = getStoredRGB(HTML_TAG_COLOR);

		} catch (RuntimeException e) {
			// no value stored, defaults are used
			e.printStackTrace();
		}
	}

	/**
	 * @param s the colors name
	 * @return
	 */
	private RGB getStoredRGB(String s) {
		return PreferenceConverter.getColor(store, s);
	}

	/**
```

```
	 * Release all of the color resources held onto by the receiver.
	 */
	public void dispose() {
		final Iterator e = colorTable.values().iterator();
		while (e.hasNext())
			((Color) e.next()).dispose();
	}

	/**
	 * Return the Color that is stored in the Color table as rgb.
	 */
	public Color getColor(RGB rgb) {
		Color color = (Color) colorTable.get(rgb);
		if (color == null) {
			color = new Color(Display.getCurrent(), rgb);
			colorTable.put(rgb, color);
		}
		return color;
	}

	public void propertyChange(PropertyChangeEvent event) {
		//TODO Änderungen dynamisch übernehmen
		if (event.getProperty().equalsIgnoreCase(STRING_COLOR)) {
			if (event.getNewValue() instanceof RGB) {
				STRING = (RGB) event.getNewValue();
			} else
				STRING = PreferenceConverter.getColor(store,
					STRING_COLOR);
		}
		if (event.getProperty().equalsIgnoreCase(HTML_COMMENT_COLOR)) {
			if (event.getNewValue() instanceof RGB) {
				HTML_COMMENT = (RGB) event.getNewValue();
			} else
				HTML_COMMENT = PreferenceConverter.getColor(store,
					HTML_COMMENT_COLOR);
		}
		if (event.getProperty().equalsIgnoreCase(CONSTANT_COLOR)) {
			if (event.getNewValue() instanceof RGB) {
				CONSTANT = (RGB) event.getNewValue();
			} else
				CONSTANT = PreferenceConverter.getColor(store,
					CONSTANT_COLOR);
		}
		if (event.getProperty().equalsIgnoreCase(JS_COMMENT_COLOR)) {
			if (event.getNewValue() instanceof RGB) {
				JS_COMMENT = (RGB) event.getNewValue();
			} else
				JS_COMMENT = PreferenceConverter.getColor(store,
					JS_COMMENT_COLOR);
		}
		if (event.getProperty().equalsIgnoreCase(DEFAULT_COLOR)) {
			if (event.getNewValue() instanceof RGB) {
				DEFAULT = (RGB) event.getNewValue();
			} else
				DEFAULT = PreferenceConverter.getColor(store,
					DEFAULT_COLOR);
		}
		if (event.getProperty().equalsIgnoreCase(HTML_TAG_COLOR)) {
			if (event.getNewValue() instanceof RGB) {
				HTML_TAG = (RGB) event.getNewValue();
			} else
				HTML_TAG = PreferenceConverter.getColor(store,
					HTML_TAG_COLOR);
		}
	}
}
```

D.X.6 DeleteSnipletAction.java

```
package de.kt.scripteditor.util;

import org.eclipse.jface.action.Action;
import org.eclipse.ui.editors.text.TextEditor;

import de.kt.scripteditor.htmleditor.HTMLEditor;
import de.kt.scripteditor.javascripteditor.JsEditor;

/**
 * The DeleteSnipletAction deletes a CodeSniplet from the library.
 * @author Karsten Thiemann
 */
public class DeleteSnipletAction extends Action {
	TextEditor editor;
	private final CodeSniplet sniplet;

	/**
	 * Constructor for InsertSnipletAction.
	 * @param snipletContent
	 */
	public DeleteSnipletAction(CodeSniplet sniplet) {
		super();
		this.setText(UtilMessages.getString("DeleteSnipletAction.Delete")
			+ sniplet.getName()); //$NON-NLS-1$
		this.setId("#DeleteSniplet_" + sniplet.getName()); //$NON-NLS-1$
		this.sniplet = sniplet;
	}

	/**
	 * @see org.eclipse.jface.action.IAction#run()
	 */
	public void run() {
		if (editor instanceof HTMLEditor) {
			SnipletLibrary.deleteHTMLSniplet(sniplet);
		} else if (editor instanceof JsEditor) {
			SnipletLibrary.deleteJsSniplet(sniplet);
		}
	}

	/**
	 * Sets the active editor.
	 * @param editor the active editor to set
	 */
	public void setActiveEditor(TextEditor editor) {
		this.editor = editor;
	}

}
```

D.X.7 EditorImages.java

```
package de.kt.scripteditor.util;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;

import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.swt.graphics.Image;

import de.kt.scripteditor.ScripteditorPlugin;

/**
 * Manages the icons, used by the plugin.
 * @author Karsten Thiemann
 * @since 02.11.2003
 */
public class EditorImages {

	public static String ICON_TAG = "tag.gif"; //$NON-NLS-1$;
	public static String ICON_FUNCTION = "function.gif"; //$NON-NLS-1$
	public static String ICON_FIELD = "field.gif"; //$NON-NLS-1$

	private static String[] allImages = { ICON_TAG, ICON_FUNCTION, ICON_FIELD
};

	private static URL iconBaseURL = null;
	private static final Map cache = new HashMap();

	static {
		try {
			iconBaseURL =
				new URL(
					ScripteditorPlugin.getDefault().getDescriptor()
						.getInstallURL(),
					"icons/"); //$NON-NLS-1$
		} catch (MalformedURLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		init();
	}

	/**
	 * @param String imageName
	 * @return Image the Image with the given name
	 */
	public static Image getImage(String imageName) {
		return (Image) cache.get(imageName);
	}

	private static void init() {
		for (int i = 0; i < allImages.length; i++) {
			final ImageDescriptor descriptor = getImageDescriptor
				(allImages[i]);
			final Image img = descriptor.createImage();
			cache.put(allImages[i], img);
		}

	}

	public static ImageDescriptor getImageDescriptor(String name) {
		if (iconBaseURL != null) {
			try {
```

```
                        return ImageDescriptor.createFromURL(
                                new URL(iconBaseURL, name));
                    } catch (MalformedURLException e) {
                        // do nothing
                    }
                }
                return ImageDescriptor.getMissingImageDescriptor();
            }

}
```

D.X.8 GenericWhitespaceDetector.java

```
package de.kt.scripteditor.util;


import org.eclipse.jface.text.rules.IWhitespaceDetector;

/**
 * A java(script) aware white space detector.
 */
public class GenericWhitespaceDetector implements IWhitespaceDetector {

	/* (non-Javadoc)
	 * Method declared on IWhitespaceDetector
	 */
	public boolean isWhitespace(char character) {
		return Character.isWhitespace(character);
	}
}
```

D.X.9 InsertSnipletAction.java

```
package de.kt.scripteditor.util;

import org.eclipse.jface.action.Action;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.IDocument;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.IFileEditorInput;
import org.eclipse.ui.editors.text.TextEditor;


/**
 * This action inserts the content of a CodeSniplet at
 * the actual cursor position.
 * @author Karsten Thiemann
 */
public class InsertSnipletAction extends Action {
	private TextEditor editor;
	private final CodeSniplet sniplet;


	/**
	 * Constructor for InsertSnipletAction.
	 * @param sniplet the CodeSniplet
	 */
	public InsertSnipletAction(CodeSniplet sniplet) {
		super();
		setText(UtilMessages.getString("InsertSnipletAction.Insert")
			+ sniplet.getName()); //$NON-NLS-1$
		setId("#InsertSniplet_"+sniplet.getName()); //$NON-NLS-1$
		this.sniplet = sniplet;
	}

	/**
	 * @see org.eclipse.jface.action.IAction#run()
	 */
	public void run() {
		IEditorInput input = editor.getEditorInput();
		if (input instanceof IFileEditorInput) {
			final IFileEditorInput fileEditorInput =
				(IFileEditorInput) input;
			final IDocument document =
				editor.getDocumentProvider()
					.getDocument(fileEditorInput);
			try {
				int offset = 0;
				if(editor instanceof IScriptEditor){
					offset = ((IScriptEditor)editor)
						.getCursorOffset();
				}
				final StringBuffer buffer = new StringBuffer();
				buffer.append(document.get(0,offset));
				buffer.append(sniplet.getContent());
				buffer.append(document.get(offset,
					document.getLength()-offset));
				document.replace(0, document.getLength(),
					buffer.toString());

				//refresh des Editors! WORKAROUND!
				document.setDocumentPartitioner
					(document.getDocumentPartitioner());
				editor.setHighlightRange(offset,0,true);

			} catch (BadLocationException e) {
```

```
                        showException(e);
                    }
                }
            }

            /**
             * Sets the active editor.
             * @param editor the active editor to set
             */
            public void setActiveEditor(TextEditor editor) {
                this.editor = editor;
            }

            private void showException(Exception e) {
                MessageDialog.openError(new Shell(), e.getMessage(),
                        e.getMessage());
            }
}
```

D.X.10 IScriptContentProvider.java

```
package de.kt.scripteditor.util;

import org.eclipse.jface.text.IDocument;

/**
 * Interface for ContentProviders used by ScriptEditors.
 * @author Karsten Thiemann
 */
public interface IScriptContentProvider {
	/**
	 * Parses the document and generates the content of the OutlineView.
	 * @param document the document to parse
	 */
	public void parse(IDocument document);
}
```

D.X.11 IScriptEditor.java

```
package de.kt.scripteditor.util;

/**
 * Interface for ScriptEditors.
 * @author Karsten Thiemann
 */
public interface IScriptEditor {

	/**
	 * Returns the Cursor Positions offset
	 * @return
	 */
	public int getCursorOffset();

}
```

D.X.12 IScriptEditorOutlinePage.java

```
package de.kt.scripteditor.util;

/**
 * Interface for OutlinePages used by ScriptEditors.
 * @author Karsten Thiemann
 */
public interface IScriptEditorOutlinePage {
	/**
	 * Updates the OutlinePage.
	 */
	public void update();
}
```

D.X.13 JSFunctionScanner.java

```
package de.kt.scripteditor.util;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.jface.text.rules.IPredicateRule;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.MultiLineRule;
import org.eclipse.jface.text.rules.RuleBasedPartitionScanner;
import org.eclipse.jface.text.rules.SingleLineRule;
import org.eclipse.jface.text.rules.Token;


/**
 * This scanner scans for javascript functions.
 * It is used by the OutlineViews.
 * @author Karsten Thiemann
 */
public class JsFunctionScanner extends RuleBasedPartitionScanner {

	public final static String JS_FUNCTION = "__js_function"; //$NON-NLS-1$
	public final static IToken JS_FUNCTION_TOKEN = new Token(JS_FUNCTION);
	public final static String JS_FIELD = "__js_field"; //$NON-NLS-1$
	public final static IToken JS_FIELD_TOKEN = new Token(JS_FIELD);

	public JsFunctionScanner() {
		super();
		final List rules = new ArrayList();

		// Add rule for functions.
		rules.add(new MultiLineRule("function ", ")", JS_FUNCTION_TOKEN));
			//$NON-NLS-1$ //$NON-NLS-2$
		rules.add(new SingleLineRule("var ", ";", JS_FIELD_TOKEN));
			//$NON-NLS-1$ //$NON-NLS-2$

		final IPredicateRule[] result = new IPredicateRule[rules.size()];
		rules.toArray(result);
		setPredicateRules(result);
	}
}
```

D.X.14 OutlineDocumentListener.java

```
package de.kt.scripteditor.util;

import org.eclipse.jface.text.DocumentEvent;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.IDocumentListener;
import org.eclipse.swt.widgets.Display;
import org.eclipse.ui.texteditor.ITextEditor;


/**
 * The OutlineDocumentListener updates the OutlineViews after
 * the document changes, but only once every 800 ms.
 * @author Karsten Thiemann
 */
public class OutlineDocumentListener implements IDocumentListener {

	private final IScriptContentProvider contentProvider;
	private final IScriptEditorOutlinePage outlinePage;
	private boolean reconciling = false;
	private IDocument document;
	private final ITextEditor textEditor;

	/**
	 * @param fContentProvider
	 */
	public OutlineDocumentListener(
		IScriptContentProvider contentProvider,
		ITextEditor textEditor,
		IScriptEditorOutlinePage outlinePage) {
			this.contentProvider = contentProvider;
			this.outlinePage = outlinePage;
			this.textEditor = textEditor;
	}

	/* (non-Javadoc)
	 * @see org.eclipse.jface.text.IDocumentListener#documentAboutToBeChanged
	 * (org.eclipse.jface.text.DocumentEvent)
	 */
	public void documentAboutToBeChanged(DocumentEvent event) {
	}

	/* (non-Javadoc)
	 * @see org.eclipse.jface.text.IDocumentListener#documentChanged
	 * (org.eclipse.jface.text.DocumentEvent)
	 */
	public void documentChanged(DocumentEvent event) {
		if (reconciling) {
			return;
		} else {
			reconciling = true;
			document = event.getDocument();
			new ReconcileThread().start();
		}
	}

	/**
	 * Parse the document and update the outline if editor,
	 * document and outline available.
	 */
	public synchronized void reconcile() {
		if (outlinePage == null || textEditor == null || document == null) {
			return;
		}
		contentProvider.parse(document);
```

```
			outlinePage.update();
			reconciling = false;
		}

		protected class ReconcileThread extends Thread {
			/* (non-Javadoc)
			 * @see java.lang.Runnable#run()
			 */
			public synchronized void run() {
				try {
					Thread.sleep(800);
				} catch (InterruptedException e) {

				} finally {
					Display.getDefault().asyncExec(new Runnable() {
						public void run() {
							reconcile();
						}
					});
				}
			}
		}

}
```

D.X.15 SaveSnipletAction.java

```
package de.kt.scripteditor.util;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.InputDialog;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.ITextSelection;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IEditorActionDelegate;
import org.eclipse.ui.IEditorPart;
import org.eclipse.ui.editors.text.TextEditor;

import de.kt.scripteditor.htmleditor.HTMLEditor;
import de.kt.scripteditor.javascripteditor.JsEditor;

/**
 * This action creates a CodeSniplet in the SnipletLibrary
 * with the selected text.
 * @author Karsten Thiemann
 */
public class SaveSnipletAction implements IEditorActionDelegate {
	TextEditor textEditor;

	/**
	 * Saves a reference to the current active editor
	 * @see org.eclipse.ui.IEditorActionDelegate#setActiveEditor(IAction,
	 * IEditorPart)
	 */
	public void setActiveEditor(IAction action, IEditorPart targetEditor) {
		textEditor = (TextEditor) targetEditor;
	}

	/* (non-Javadoc)
	 * @see org.eclipse.ui.IActionDelegate#run
	 * (org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		final IDocument document =
			textEditor.getDocumentProvider().getDocument(
				textEditor.getEditorInput());
		final ITextSelection ts =
			(ITextSelection) textEditor.getSelectionProvider()
				.getSelection();
		final InputDialog dialog =
			new InputDialog(
				null,
				UtilMessages.getString
					("SaveSnipletAction.Insert_Sniplet_Name"),
						//$NON-NLS-1$
				UtilMessages.getString
					("SaveSnipletAction.Insert_Sniplet_Name_here"),
					//$NON-NLS-1$
				"", //$NON-NLS-1$
				null);
		dialog.open();
		final String snipletName = dialog.getValue();
		if (snipletName != null && !snipletName.equalsIgnoreCase("")) {
			//$NON-NLS-1$
			if (textEditor instanceof HTMLEditor) {
				SnipletLibrary.addHTMLSniplet(
					new CodeSniplet(snipletName, ts.getText()));
			} else if (textEditor instanceof JsEditor) {
				SnipletLibrary.addJsSniplet(
					new CodeSniplet(snipletName, ts.getText()));
			}
		}
```

```
	}

	/**
	 * Enables the action if text has been selected, otherwise, the action
	 * is disabled.
	 * @see org.eclipse.ui.IActionDelegate#selectionChanged(IAction,
	 * ISelection)
	 */
	public void selectionChanged(IAction action, ISelection selection) {
		if (selection != null && selection instanceof ITextSelection) {
			final ITextSelection ts = (ITextSelection) selection;
			if (ts.getLength() == 0) {
				action.setEnabled(false);
			} else {
				action.setEnabled(true);
			}
		} else {
			action.setEnabled(false);
		}
	}
}
```

D.X.16 ScriptEditorDoubleClickStrategy.java

```
//Implementierung aus: package org.eclipse.ui.examples.javaeditor.java;
//ergänzt um Tag-Findung
/*******************************************************************************
 * Copyright (c) 2000, 2003 IBM Corporation and others.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Common Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/cpl-v10.html
 *
 * Contributors:
 *     IBM Corporation - initial API and implementation
 *******************************************************************************
/

package de.kt.scripteditor.util;

import java.util.Vector;

import org.eclipse.jface.text.BadLocationException;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.ITextDoubleClickStrategy;
import org.eclipse.jface.text.ITextViewer;
import org.eclipse.jface.text.rules.IToken;
import org.eclipse.jface.text.rules.RuleBasedScanner;

import de.kt.scripteditor.htmleditor.HTMLEditorEnvironment;
import de.kt.scripteditor.htmleditor.outline.HTMLOutlineTagScanner;

/**
 * This DoubleClickStrategy marks words or the region between
 * matching brackets or tags.
 */
public class ScriptEditorDoubleClickStrategy implements ITextDoubleClickStrategy
{

      protected ITextViewer fText;
      protected int fPos;
      protected int fStartPos;
      protected int fEndPos;

      protected static char[] fgBrackets =
            { '{', '}', '(', ')', '[', ']', '"', '"' };

      /*
       * Create a ScriptEditorDoubleClickStrategy.
       */
      public ScriptEditorDoubleClickStrategy() {
            super();
      }

      /* (non-Javadoc)
       * Method declared on ITextDoubleClickStrategy
       */
      public void doubleClicked(ITextViewer text) {

            fPos = text.getSelectedRange().x;

            if (fPos < 0)
                  return;

            fText = text;

            if (!selectBracketBlock())
                  selectWord();
      }
```

```
/**
 * Represents a HTML-Tag.
 * @author Karsten Thiemann
 */
protected class Tag {
	String name;
	int offset;
	int length;
	boolean isOpenTag;
	/**
	 * Constructor for Tags.
	 * @param name the name
	 * @param offset the offset
	 * @param length the length
	 * @param isOpenTag true if the tag is an open-tag
	 */
	public Tag(String name, int offset, int length, boolean isOpenTag) {
		this.name = name;
		this.offset = offset;
		this.length = length;
		this.isOpenTag = isOpenTag;
	}

};

/**
 * Indicates the existance of a matching closed or opened tag.
 * Updates fStartPos and
 * fEndPos.
 * @return boolean true if a matching tag can be found, false otherwise
 */
protected boolean matchTags() {

	char prevChar, nextChar;
	boolean findOpenTag = false;
	boolean findCloseTag = false;
	Tag tagToFind = null;
	Vector tags = new Vector();

	fStartPos = -1;
	fEndPos = -1;

	// get the chars preceding and following the start position
	try {

		final IDocument doc = fText.getDocument();
		prevChar = doc.getChar(fPos - 1);
		nextChar = doc.getChar(fPos);

		if (prevChar == '>') {
			findCloseTag = true;
		}
		if (nextChar == '<') {
			if (doc.getChar(fPos + 1) == '/') {
				findOpenTag = true;
			}
		}
		if (!findOpenTag && !findCloseTag) {
			return false;
		}

		final RuleBasedScanner scanner =
			HTMLEditorEnvironment.getHTMLOutlineTagScanner();
		scanner.setRange(doc, 0, doc.getLength());
		IToken token = scanner.nextToken();
		//fill tag-vector
		while (!token.isEOF()) {
```

```
		if (token
				.equals
				(HTMLOutlineTagScanner.OUTLINE_OPEN_TAG_TOKEN)) {
				String tagname =
					doc.get(
						scanner.getTokenOffset(),
						scanner.getTokenLength());
				int endOfName =
					(tagname.indexOf(" ") == -1) //$NON-NLS-1$
						? tagname.length() - 1
						: tagname.indexOf(" "); //$NON-NLS-1$
				tagname = tagname.substring(1, endOfName);

				tags.add(
					new Tag(
						tagname,
						scanner.getTokenOffset(),
						scanner.getTokenLength(),
						true));
		} else if (
				token.equals(
				HTMLOutlineTagScanner.OUTLINE_CLOSE_TAG_TOKEN)) {
				String tagname =
					doc.get(
						scanner.getTokenOffset(),
						scanner.getTokenLength());
				int endOfName =
					(tagname.indexOf(" ") == -1) //$NON-NLS-1$
						? tagname.length() - 1
						: tagname.indexOf(" "); //$NON-NLS-1$
				tagname = tagname.substring(2, endOfName);

				tags.add(
					new Tag(
						tagname,
						scanner.getTokenOffset(),
						scanner.getTokenLength(),
						false));
		}
		token = scanner.nextToken();
}
//find matching Tag
String tagNameToFind = ""; //$NON-NLS-1$
int tagNumber = 0;
if (findCloseTag && !findOpenTag) {
		for (int i = 0; i < tags.size(); i++) {
				final Tag tag = (Tag) tags.elementAt(i);
				if (tag.offset + tag.length == fPos) {
					tagNameToFind = tag.name;
					fStartPos = fPos-1;
					continue;
				} else if (tag.offset + tag.length < fPos) {
					continue;
				}
				if (tag.name.equalsIgnoreCase(tagNameToFind)) {
					if (!tag.isOpenTag) {
						if (tagNumber == 0) {
							fEndPos = tag.offset;
							return true;
						} else {
							tagNumber--;
						}
					} else {
						tagNumber++;
					}
				}
		}
```

```
            } else if (findOpenTag) {
                  for (int i = tags.size() - 1; i >= 0; i--) {
                        final Tag tag = (Tag) tags.elementAt(i);
                        if (tag.offset == fPos) {
                              tagNameToFind = tag.name;
                              fEndPos = fPos;
                              continue;
                        } else if (tag.offset > fPos) {
                              continue;
                        }
                        if (tag.name.equalsIgnoreCase(tagNameToFind)) {
                              if (tag.isOpenTag) {
                                    if (tagNumber == 0) {
                                          fStartPos = tag.offset +
                                                tag.length - 1;
                                          return true;
                                    } else {
                                          tagNumber--;
                                    }
                              } else {
                                    tagNumber++;
                              }
                        }
                  }
            }

      } catch (BadLocationException x) {
            return false;
      }
      return false;
}

/**
 * Match the brackets at the current selection. Return true if successful,
 * false otherwise.
 */
protected boolean matchBracketsAt() {

      char prevChar, nextChar;

      int i;
      int bracketIndex1 = fgBrackets.length;
      int bracketIndex2 = fgBrackets.length;

      fStartPos = -1;
      fEndPos = -1;

      // get the chars preceding and following the start position
      try {

            final IDocument doc = fText.getDocument();

            prevChar = doc.getChar(fPos - 1);
            nextChar = doc.getChar(fPos);

            // is the char either an open or close bracket?
            for (i = 0; i < fgBrackets.length; i = i + 2) {
                  if (prevChar == fgBrackets[i]) {
                        fStartPos = fPos - 1;
                        bracketIndex1 = i;
                  }
            }
            for (i = 1; i < fgBrackets.length; i = i + 2) {
                  if (nextChar == fgBrackets[i]) {
                        fEndPos = fPos;
                        bracketIndex2 = i;
                  }
```

```
                }

                if (fStartPos > -1 && bracketIndex1 < bracketIndex2) {
                        fEndPos =
                                searchForClosingBracket(
                                        fStartPos,
                                        prevChar,
                                        fgBrackets[bracketIndex1 + 1],
                                        doc);
                        if (fEndPos > -1)
                                return true;
                        else
                                fStartPos = -1;
                } else if (fEndPos > -1) {
                        fStartPos =
                                searchForOpenBracket(
                                        fEndPos,
                                        fgBrackets[bracketIndex2 - 1],
                                        nextChar,
                                        doc);
                        if (fStartPos > -1)
                                return true;
                        else
                                fEndPos = -1;
                }

        } catch (BadLocationException x) {
        }

        return false;
}

/**
 * Select the word at the current selection. Return true if successful,
 * false otherwise.
 */
protected boolean matchWord() {

        final IDocument doc = fText.getDocument();

        try {

                int pos = fPos;
                char c;

                while (pos >= 0) {
                        c = doc.getChar(pos);
                        if (!Character.isJavaIdentifierPart(c))
                                break;
                        --pos;
                }

                fStartPos = pos;

                pos = fPos;
                int length = doc.getLength();

                while (pos < length) {
                        c = doc.getChar(pos);
                        if (!Character.isJavaIdentifierPart(c))
                                break;
                        ++pos;
                }

                fEndPos = pos;

                return true;
```

```
		} catch (BadLocationException x) {
		}

		return false;
	}

	/**
	 * Returns the position of the closing bracket after startPosition.
	 * @returns the location of the closing bracket.
	 * @param startPosition - the beginning position
	 * @param openBracket - the character that represents the open bracket
	 * @param closeBracket - the character that represents the close bracket
	 * @param document - the document being searched
	 */
	protected int searchForClosingBracket(
			int startPosition,
			char openBracket,
			char closeBracket,
			IDocument document)
			throws BadLocationException {
			int stack = 1;
			int closePosition = startPosition + 1;
			int length = document.getLength();
			char nextChar;

			while (closePosition < length && stack > 0) {
				nextChar = document.getChar(closePosition);
				if (nextChar == openBracket && nextChar != closeBracket)
					stack++;
				else if (nextChar == closeBracket)
					stack--;
				closePosition++;
			}

			if (stack == 0)
				return closePosition - 1;
			else
				return -1;

	}

	/**
	 * Returns the position of the open bracket before startPosition.
	 * @returns the location of the starting bracket.
	 * @param startPosition - the beginning position
	 * @param openBracket - the character that represents the open bracket
	 * @param closeBracket - the character that represents the close bracket
	 * @param document - the document being searched
	 */
	protected int searchForOpenBracket(
			int startPosition,
			char openBracket,
			char closeBracket,
			IDocument document)
			throws BadLocationException {
			int stack = 1;
			int openPos = startPosition - 1;
			char nextChar;

			while (openPos >= 0 && stack > 0) {
				nextChar = document.getChar(openPos);
				if (nextChar == closeBracket && nextChar != openBracket)
					stack++;
				else if (nextChar == openBracket)
					stack--;
				openPos--;
```

```
            }

            if (stack == 0)
                  return openPos + 1;
            else
                  return -1;
      }

      /**
       * Select the area between the selected bracket and the closing bracket.
       * Return true if successful.
       */
      protected boolean selectBracketBlock() {
            if (matchBracketsAt() || matchTags()) {

                  if (fStartPos == fEndPos)
                        fText.setSelectedRange(fStartPos, 0);
                  else
                        fText.setSelectedRange(fStartPos + 1, fEndPos -
                              fStartPos - 1);

                  return true;
            }
            return false;
      }

      /**
       * Select the word at the current selection.
       */
      protected void selectWord() {
            if (matchWord()) {

                  if (fStartPos == fEndPos)
                        fText.setSelectedRange(fStartPos, 0);
                  else
                        fText.setSelectedRange(fStartPos + 1, fEndPos -
                              fStartPos - 1);
            }
      }
}
```

D.X.17 SnipletLibrary.java

```
package de.kt.scripteditor.util;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Vector;

import org.apache.xerces.parsers.DOMParser;
import org.apache.xml.serialize.XMLSerializer;
import org.eclipse.core.runtime.Platform;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.widgets.Shell;
import org.w3c.dom.Attr;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;
import org.xml.sax.SAXException;

import de.kt.scripteditor.ScripteditorPlugin;

/**
 * The SnipletLibrary provides acess to CodeSniplets,
 * stored in XML-files.
 * @author Karsten Thiemann
 */
public class SnipletLibrary {

	private static CodeSniplet[] htmlSniplets;
	private static CodeSniplet[] jsSniplets;
	private static String HTML_FILE_NAME;
	private static String JS_FILE_NAME;

	static {
		SnipletParser parser = new SnipletParser();
		try {
			HTML_FILE_NAME =
				Platform
					.resolve(
						ScripteditorPlugin
							.getDefault()
							.getDescriptor()
							.getInstallURL())
					.toString()
					+ "HTMLSniplets.xml"; //$NON-NLS-1$

			JS_FILE_NAME =
				Platform
					.resolve(
						ScripteditorPlugin
							.getDefault()
							.getDescriptor()
							.getInstallURL())
					.toString()
					+ "JsSniplets.xml"; //$NON-NLS-1$
		} catch (IOException e) {
			showException(e);
		}
		parser.parseDocument(HTML_FILE_NAME);
		htmlSniplets = parser.getSniplets();
		parser.parseDocument(JS_FILE_NAME);
		jsSniplets = parser.getSniplets();
	}

	/**
```

```
	 * Returns all sniplets of the HTMLEditor.
	 * @return CodeSniplet[] all sniplets of the HTMLEditor
	 */
	public static CodeSniplet[] getHTMLSniplets() {
		return htmlSniplets;
	}

	/**
	 * Returns all sniplets of the JsEditor.
	 * @return CodeSniplet[] all sniplets of the JsEditor
	 */
	public static CodeSniplet[] getJsSniplets() {
		return jsSniplets;
	}

	/**
	 * Deletes the given sniplet.
	 * @param sniplet the sniplet to delete
	 */
	public static void deleteHTMLSniplet(CodeSniplet sniplet) {
		final Vector sniplets = new Vector();
		for (int i = 0; i < htmlSniplets.length; i++) {
			sniplets.add(htmlSniplets[i]);
		}
		sniplets.remove(sniplet);
		htmlSniplets = new CodeSniplet[sniplets.size()];
		for (int i = 0; i < htmlSniplets.length; i++) {
			htmlSniplets[i] = (CodeSniplet) sniplets.elementAt(i);
		}
		deleteFromXMLFile(HTML_FILE_NAME, sniplet);
	}

	/**
	 * Deletes the given sniplet.
	 * @param sniplet the sniplet to delete
	 */
	public static void deleteJsSniplet(CodeSniplet sniplet) {
		final Vector sniplets = new Vector();
		for (int i = 0; i < jsSniplets.length; i++) {
			sniplets.add(jsSniplets[i]);
		}
		sniplets.remove(sniplet);
		jsSniplets = new CodeSniplet[sniplets.size()];
		for (int i = 0; i < jsSniplets.length; i++) {
			jsSniplets[i] = (CodeSniplet) sniplets.elementAt(i);
		}
		deleteFromXMLFile(JS_FILE_NAME, sniplet);
	}

	/**
	 * Adds the given sniplet.
	 * @param sniplet the sniplet to add
	 */
	public static void addJsSniplet(CodeSniplet sniplet) {
		final Vector sniplets = new Vector();
		for (int i = 0; i < jsSniplets.length; i++) {
			sniplets.add(jsSniplets[i]);
		}
		sniplets.add(sniplet);
		jsSniplets = new CodeSniplet[sniplets.size()];
		for (int i = 0; i < jsSniplets.length; i++) {
			jsSniplets[i] = (CodeSniplet) sniplets.elementAt(i);
		}
		addToXMLFile(JS_FILE_NAME, sniplet);
	}

	/**
```

```
     * Adds the given sniplet.
     * @param sniplet the sniplet to add
     */
    public static void addHTMLSniplet(CodeSniplet sniplet) {
          final Vector sniplets = new Vector();
          for (int i = 0; i < htmlSniplets.length; i++) {
                sniplets.add(jsSniplets[i]);
          }
          sniplets.add(sniplet);
          htmlSniplets = new CodeSniplet[sniplets.size()];
          for (int i = 0; i < htmlSniplets.length; i++) {
                htmlSniplets[i] = (CodeSniplet) sniplets.elementAt(i);
          }
          addToXMLFile(HTML_FILE_NAME, sniplet);
    }

    /**
     * Stores the given sniplet in the given file.
     * @param filename the name of the xml-file
     * @param sniplet the sniplet to add
     */
    private static void addToXMLFile(String filename, CodeSniplet sniplet) {
          try {
                final DOMParser parser = new DOMParser();
          //TODO Workarround warum funktioniert der filename nicht direkt???
                filename = filename.replaceFirst("file:", "");
                      //$NON-NLS-1$ //$NON-NLS-2$
                parser.parse(filename);
                Document document = parser.getDocument();

                final Element e = document.getDocumentElement();
                final Element snipletNode = document.createElement("sniplet");
                      //$NON-NLS-1$
                final Attr nameAttribute = document.createAttribute("name");
                      //$NON-NLS-1$
                nameAttribute.setValue(sniplet.getName());
                snipletNode.setAttributeNode(nameAttribute);
                final Text textNode = document.createTextNode
                      (sniplet.getContent());
                snipletNode.appendChild(textNode);
                e.appendChild(snipletNode);
                final FileWriter fw = new FileWriter(filename, false);
                final XMLSerializer tree = new XMLSerializer(fw, null);
                tree.serialize(document);
                return;

          } catch (IOException e) {
                showException(e);
          } catch (SAXException e) {
                showException(e);
          }
    }

    /**
     * Deletes the given sniplet from the given file.
     * @param filename the name of the xml-file
     * @param sniplet the sniplet to delete
     */
    private static void deleteFromXMLFile(
          String filename,
          CodeSniplet sniplet) {
          try {
                final DOMParser parser = new DOMParser();
          //TODO Workarround warum funktioniert der filename nicht direkt???
                filename = filename.replaceFirst("file:", "");
                      //$NON-NLS-1$ //$NON-NLS-2$
                parser.parse(filename);
```

```
                final Document document = parser.getDocument();
                final NodeList nodeList = document.getElementsByTagName
                      ("sniplet"); //$NON-NLS-1$
                for (int i = 0; i < nodeList.getLength(); i++) {
                      final Element e = (Element) nodeList.item(i);
                      if (e.hasAttribute("name") //$NON-NLS-1$
                            && e.getAttribute("name").equalsIgnoreCase(
                                  //$NON-NLS-1$
                                  sniplet.getName())) {
                            e.getParentNode().removeChild(e);
                            final FileWriter fw = new FileWriter(filename,
                                  false);
                            final XMLSerializer tree = new XMLSerializer(fw,
                                  null);
                            tree.serialize(document);
                            return;
                      }
                }
          } catch (IOException e) {
                showException(e);
          } catch (SAXException e) {
                showException(e);
          }
    }

    private static void showException(Exception e) {
          MessageDialog.openError(new Shell(), e.getMessage(),
                e.getMessage());
    }
}
```

D.X.18 SnipletParser.java

```
package de.kt.scripteditor.util;

import java.util.Vector;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

/**
 * Parses the XML-File that includes the code sniplets.
 * @author Karsten Thiemann
 */
public class SnipletParser extends AbstractXMLParser {

	private final Vector sniplets = new Vector();

	/* (non-Javadoc)
	 * @see de.kt.jseditor.util.AbstractXMLParser#doContent
	 * (org.w3c.dom.Document)
	 */
	protected void doContent(Document d) {
		final NodeList nodeList = d.getElementsByTagName("sniplet");
			//$NON-NLS-1$
		for (int i = 0; i < nodeList.getLength(); i++) {
			final Element e = (Element) nodeList.item(i);
			String snipletName = ""; //$NON-NLS-1$
			String content = ""; //$NON-NLS-1$
			if (e.hasAttribute("name")) { //$NON-NLS-1$
				snipletName = e.getAttribute("name"); //$NON-NLS-1$
			}
			if (e.hasChildNodes()
				&& e.getFirstChild().getNodeType()
				== Element.TEXT_NODE) {
				content = e.getFirstChild().getNodeValue();
			}
			sniplets.add(new CodeSniplet(snipletName, content));
		}
	}

	/**
	 * Returns all CodeSniplets.
	 * @return CodeSniplet[] all CodeSniplets
	 */
	public CodeSniplet[] getSniplets() {
		final CodeSniplet[] snipletArray = new CodeSniplet[sniplets.size()];
		for (int i = 0; i < snipletArray.length; i++) {
			snipletArray[i] = (CodeSniplet) sniplets.elementAt(i);
		}
		return snipletArray;
	}
}
```

D.X.19 UtilMessages.java

```
package de.kt.scripteditor.util;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

/**
 * The UtilMessages manages the access to the
 * utils ressource bundle.
 * @author Karsten
 */
public class UtilMessages {

	private static final String BUNDLE_NAME =
"de.kt.scripteditor.util.UtilMessages"; //$NON-NLS-1$

	private static final ResourceBundle RESOURCE_BUNDLE =
		ResourceBundle.getBundle(BUNDLE_NAME);

	/**
	 * private constructor to avoid instantiation.
	 */
	private UtilMessages() {
	}

	/**
	 * @param key the key of the stored string
	 * @return String the string with the given key
	 */
	public static String getString(String key) {
		try {
			return RESOURCE_BUNDLE.getString(key);
		} catch (MissingResourceException e) {
			return '!' + key + '!';
		}
	}
}
```

D.X.20 UtilMessages.properties

```
AbstractScriptEditor.Insert_Sniplets=Insert Sniplets
AbstractScriptEditor.Delete_Sniplets=Delete Sniplets

DeleteSnipletAction.Delete=Delete:

InsertSnipletAction.Insert=Insert:

SaveSnipletAction.Insert_Sniplet_Name=Insert Sniplet Name
SaveSnipletAction.Insert_Sniplet_Name_here=Insert Sniplet Name here.
```

D.X.20 UtilMessages_de_DE.properties

```
AbstractScriptEditor.Insert_Sniplets=Sniplets einfügen
AbstractScriptEditor.Delete_Sniplets=Sniplets löschen

DeleteSnipletAction.Delete=Löschen:

InsertSnipletAction.Insert=Einfügen:

SaveSnipletAction.Insert_Sniplet_Name=Snipletname eingeben
SaveSnipletAction.Insert_Sniplet_Name_here=Snipletname hier eingeben.
```

D.XI XML-Dateien

D.XI.1 contexts.xml (kontextsensitive Hilfe)

```
<?xml version="1.0" encoding="UTF-8"?>

<contexts>
	<context id="outline_context">
		<description>Help for Scripteditor Outline View</description>
		<topic href="html/outline.html"
			label="Outline View" />
	</context>
	<context id="editor_context">
		<description>Help for Scripteditor</description>
		<topic label="Editor" href="html/editor.html"/>
	</context>
</contexts>
```

D.XI.2 plugin.xml (Manifest)

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
   id="de.kt.scripteditor"
   name="Scripteditor Plug-in"
   version="1.0.0"
   provider-name="Karsten Thiemann"
   class="de.kt.scripteditor.ScripteditorPlugin">

   <runtime>
      <library name="scripteditor.jar">
         <export name="*"/>
      </library>
      <library name="jtidy.jar">
         <export name="*"/>
      </library>
   </runtime>
   <requires>
      <import plugin="org.eclipse.core.resources"/>
      <import plugin="org.eclipse.ui"/>
      <import plugin="org.apache.xerces"/>
   </requires>


   <extension-point id="preview" name="Preview" schema="schema/preview.exsd"/>

   <extension
         point="org.eclipse.ui.editors">
      <editor
            name="Javascript Editor"
            icon="icons/js.gif"
            extensions="js"
            contributorClass=
                  "de.kt.scripteditor.javascripteditor.JsActionContributor"
            class="de.kt.scripteditor.javascripteditor.JsEditor"
            id="de.kt.scripteditor.javascripteditor.JsEditor">
      </editor>
      <editor
            name="HTML Editor"
            icon="icons/html.gif"
            extensions="htm,html"
            contributorClass=
                  "de.kt.scripteditor.htmleditor.MultiPageEditorContributor"
            class="de.kt.scripteditor.htmleditor.MultiPageEditor"
            id="de.kt.scripteditor.htmleditor.MultiPageEditor">
      </editor>
   </extension>
   <extension
         point="org.eclipse.ui.preferencePages">
      <page
            name="%html_preferences"
            class="de.kt.scripteditor.preferences.ColorPreferencePage"
            id="de.kt.scripteditor.preferences.MainPreferencePage">
      </page>
      <page
            name="%outline_preferences"
            category="de.kt.scripteditor.preferences.MainPreferencePage"
            class="de.kt.scripteditor.preferences.FilterPreferencePage"
            id="de.kt.scripteditor.preferences.FilterPreferencePage">
      </page>
   </extension>
   <extension
         point="org.eclipse.ui.newWizards">
      <category
            name="HTML"
            id="de.kt.scripteditor.htmleditor.new">
      </category>
      <wizard
```

```
         name="%new_html_file"
         icon="icons\html.gif"
         category="de.kt.scripteditor.htmleditor.new"
         class="de.kt.scripteditor.htmleditor.HTMLCreationWizard"
         id="de.kt.scripteditor.htmleditor.wizards.new.file">
      <description>
         %new_html_file_discription
      </description>
   </wizard>
</extension>
<extension
      point="org.eclipse.ui.popupMenus">
   <viewerContribution
         targetID="#HTMLEditorContext"
         id="de.kt.scripteditor.util.Sniplet">
      <action
            label="%save_sniplet"
            class="de.kt.scripteditor.util.SaveSnipletAction"
            menubarPath="additions"
            enablesFor="+"
            id="de.kt.scripteditor.util.SaveSniplet">
         <selection
               class="org.eclipse.jface.text.ITextSelection">
         </selection>
      </action>
   </viewerContribution>
   <viewerContribution
         targetID="#HTMLEditorContext"
         id="de.kt.scripteditor.htmleditor.actions.Externalize">
      <action
            label="%externalize_scripts"
            class=
         "de.kt.scripteditor.htmleditor.actions.ExternalizeScriptAction"
            menubarPath="additions"
            enablesFor="*"
            id="de.kt.scripteditor.htmleditor.actions.ExternalizeScript">
         <selection
               class="org.eclipse.jface.text.ITextSelection">
         </selection>
      </action>
   </viewerContribution>
   <viewerContribution
         targetID="#HTMLEditorContext"
         id="de.kt.scripteditor.htmleditor.actions.Insert">
      <action
            label="%insert_image"
            class="de.kt.scripteditor.htmleditor.actions.InsertImageAction"
            menubarPath="additions"
            enablesFor="*"
            id="de.kt.scripteditor.htmleditor.actions.InsertImage">
         <selection
               class="org.eclipse.jface.text.ITextSelection">
         </selection>
      </action>
   </viewerContribution>
   <viewerContribution
         targetID="#JsEditorContext"
         id="de.kt.scripteditor.util.Sniplet">
      <action
            label="%save_sniplet"
            class="de.kt.scripteditor.util.SaveSnipletAction"
            menubarPath="additions"
            enablesFor="+"
            id="de.kt.scripteditor.util.SaveSniplet">
         <selection
               class="org.eclipse.jface.text.ITextSelection">
         </selection>
      </action>
   </viewerContribution>
```

```
      <objectContribution
            objectClass="org.eclipse.core.resources.IFile"
            nameFilter="*.htm*"
            id="de.kt.scripteditor.htmleditor.CSS">
         <action
               label="%append_CSS_file"
               class=
            "de.kt.scripteditor.htmleditor.actions.AppendCSSReferenceAction"
               menubarPath="additions"
               enablesFor="+"
               id="de.kt.scripteditor.htmleditor.CSSAction">
         </action>
      </objectContribution>
   </extension>
   <extension
         point="org.eclipse.ui.editorActions">
      <editorContribution
            targetID="de.kt.scripteditor.htmleditor.MultiPageEditor"
            id=
      "de.kt.scripteditor.htmleditor.actions.InsertImageActionContribution">
         <action
               label="%insert_image"
               icon="icons/insertImage.gif"
               tooltip="Insert Image"
               class="de.kt.scripteditor.htmleditor.actions.InsertImageAction"
               toolbarPath="MultiPageEditor"
               id="de.kt.scripteditor.htmleditor.actions.InsertImageAction">
         </action>
      </editorContribution>
   </extension>
   <extension
         point="org.eclipse.help.toc">
      <toc
            file="toc.xml"
            primary="true">
      </toc>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
            file="contexts.xml">
      </contexts>
   </extension>

</plugin>
```

D.XI.3 toc.xml (Hilfe-Inhaltsverzeichnis)

```
<?xml version="1.0" encoding="UTF-8"?>
<?NLS TYPE="org.eclipse.help.toc"?>

<toc label="HTML/Javascript Editor">
      <topic label="Allgemeines"  href="html/overview.html">
            <topic label="Features" href="html/features.html"/>
            <topic label="Editor" href="html/editor.html"/>
            <topic label="Outline View" href="html/outline.html"/>
            <topic label="Assistenten" href="html/assistant.html"/>
            <topic label="CSS" href="html/css.html"/>
            <topic label="Code-Bibliothek" href="html/library.html"/>
      </topic>
      <topic label="Danksagungen"  href="html/thanks.html" />
</toc>
```

D.XII Vorschau-Erweiterungspunkt

D.XII.1 preview.exsd (Schnittstellendefinition)

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Schema file written by PDE -->
<schema targetNamespace="de.kt.scripteditor">
<annotation>
      <appInfo>
         <meta.schema plugin="de.kt.scripteditor" id="preview" name="Preview"/>
      </appInfo>
      <documentation>
         A preview plugin provides one or more preview pages that render HTML.
      </documentation>
   </annotation>

   <element name="extension">
      <complexType>
         <sequence>
            <element ref="preview" minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="point" type="string" use="required">
            <annotation>
               <documentation>
               </documentation>
            </annotation>
         </attribute>
         <attribute name="id" type="string">
            <annotation>
               <documentation>
               </documentation>
            </annotation>
         </attribute>
         <attribute name="name" type="string">
            <annotation>
               <documentation>
               </documentation>
            </annotation>
         </attribute>
      </complexType>
   </element>

  <element name="preview">
      <annotation>
         <documentation>
            Provides a preview that renders HTML.
         </documentation>
      </annotation>
      <complexType>
         <attribute name="name" type="string" use="required">
            <annotation>
               <documentation>
               </documentation>
            </annotation>
         </attribute>
         <attribute name="id" type="string" use="required">
            <annotation>
               <documentation>
               </documentation>
            </annotation>
         </attribute>
         <attribute name="class" type="string" use="required">
            <annotation>
               <documentation>
                  Must implement de.kt.scripteditor.htmleditor.IHTMLPreview.
               </documentation>
               <appInfo>
                  <meta.attribute kind="java"
```

```
                        basedOn="de.kt.scripteditor.htmleditor.IHTMLPreview"/>
                </appInfo>
             </annotation>
          </attribute>
       </complexType>
    </element>

    <annotation>
       <appInfo>
          <meta.section type="since"/>
       </appInfo>
       <documentation>
          [Enter the first release in which this extension point appears.]
       </documentation>
    </annotation>

    <annotation>
       <appInfo>
          <meta.section type="examples"/>
       </appInfo>
       <documentation>
          The following is an example of the extension point usage:
    &lt;p&gt;
    &lt;pre&gt;
       &lt;extension point=&quot;de.kt.scripteditor.preview&quot;&gt;
          &lt;preview
             id=&quot;de.kt.scripteditor.preview1&quot;
             name=&quot;Sample Preview 1&quot;
             class=&quot;de.kt.scripteditor.Preview1&quot;&gt;
          &lt;/preview&gt;
       &lt;/extension&gt;
    &lt;/pre&gt;
    &lt;/p&gt;
       </documentation>
    </annotation>

    <annotation>
       <appInfo>
          <meta.section type="apiInfo"/>
       </appInfo>
       <documentation>
          Plug-ins that want to extend this extension point must implement
&lt;samp&gt;de.kt.scripteditor.htmleditor.IHTMLPreview&lt;/samp&gt; interface.
       </documentation>
    </annotation>

    <annotation>
       <appInfo>
          <meta.section type="implementation"/>
       </appInfo>
       <documentation>
          [Enter information about supplied implementation of this extension
             point.]
       </documentation>
    </annotation>

    <annotation>
       <appInfo>
          <meta.section type="copyright"/>
       </appInfo>
       <documentation>
       </documentation>
    </annotation>

</schema>
```

E Quelltexte Vorschau-Plug-In

E.I de.kt.scripteditor.preview (Vorschau-Plug-In)

E.I.1 HTMLPreview.java

```
package de.kt.scripteditor.preview;
import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.FillLayout;
import org.eclipse.swt.ole.win32.OLE;
import org.eclipse.swt.ole.win32.OleAutomation;
import org.eclipse.swt.ole.win32.OleControlSite;
import org.eclipse.swt.ole.win32.OleFrame;
import org.eclipse.swt.ole.win32.Variant;
import org.eclipse.swt.widgets.Composite;
import de.kt.scripteditor.htmleditor.IHTMLPreview;

/**
 * This class provides a preview, using the Microsoft Internet Explorer.
 * @see IHTMLPreview
 */
public class HTMLPreview implements IHTMLPreview {

	private OleAutomation automation;
		private String name = "IE Preview";
		/* (non-Javadoc)
		 * @see de.kt.jseditor.htmleditor.IHTMLPreview#getPreviewComposite()
		 */
		public Composite getPreviewComposite(Composite parent) {
			final Composite composite = new Composite(parent, SWT.NONE);
			final FillLayout layout = new FillLayout();
			composite.setLayout(layout);
			final OleFrame frame = new OleFrame(composite, SWT.NONE);
			final OleControlSite controlSite =
				new OleControlSite(frame, SWT.NONE, "Shell.Explorer");
					//$NON-NLS-1$
			controlSite.doVerb(OLE.OLEIVERB_INPLACEACTIVATE);
			automation = new OleAutomation(controlSite);
			return composite;
		}

		/* (non-Javadoc)
		 * @see de.kt.jseditor.htmleditor.IHTMLPreview#updatePreview
		 * (java.lang.String)
		 */
		public void updatePreview(String url) {
			int[] rgdispid = automation.getIDsOfNames(new String[] {
				"Navigate" }); //$NON-NLS-1$
			int dispIdMember = rgdispid[0];
			final Variant[] rgvarg = new Variant[1];
				// this is the URL parameter
			rgvarg[0] = new Variant("file:///" + url); //$NON-NLS-1$
			final Variant pVarResult = automation.invoke(dispIdMember,
				rgvarg);
		}

		/* (non-Javadoc)
		 * @see de.kt.jseditor.htmleditor.IHTMLPreview#getName()
		 */
		public String getName() {
			return name;
		}
}
```

E.I.2 PreviewPlugin.java

```
package de.kt.scripteditor.preview;

import org.eclipse.ui.plugin.*;
import org.eclipse.core.runtime.*;
import org.eclipse.core.resources.*;
import java.util.*;

/**
 * The main plugin class to be used in the desktop.
 * @author Karsten Thiemann
 */
public class PreviewPlugin extends AbstractUIPlugin {
	//The shared instance.
	private static PreviewPlugin plugin;
	//Resource bundle.
	private ResourceBundle resourceBundle;

	/** The constructor. */
	public PreviewPlugin(IPluginDescriptor descriptor) {
		super(descriptor);
		plugin = this;
		try {
			resourceBundle= ResourceBundle.getBundle
				("de.kt.scripteditor.preview.PreviewPluginResources");
		} catch (MissingResourceException x) {
			resourceBundle = null;
		}
	}

	/**
	 * Returns the shared instance.
	 */
	public static PreviewPlugin getDefault() {
		return plugin;
	}

	/**
	 * Returns the workspace instance.
	 */
	public static IWorkspace getWorkspace() {
		return ResourcesPlugin.getWorkspace();
	}

	/**
	 * Returns the string from the plugin's resource bundle,
	 * or 'key' if not found.
	 */
	public static String getResourceString(String key) {
		ResourceBundle bundle= PreviewPlugin.getDefault()
			.getResourceBundle();
		try {
			return bundle.getString(key);
		} catch (MissingResourceException e) {
			return key;
		}
	}

	/**
	 * Returns the plugin's resource bundle,
	 */
	public ResourceBundle getResourceBundle() {
		return resourceBundle;
	}
}
```

E.II XML-Dateien

E.II.1 plugin.xml (Manifest)

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
   id="de.kt.scripteditor.preview"
   name="Preview Plug-in"
   version="1.0.0"
   provider-name="Karsten Thiemann"
   class="de.kt.scripteditor.preview.PreviewPlugin">

   <runtime>
      <library name="preview.jar"/>
   </runtime>
   <requires>
      <import plugin="org.eclipse.core.resources"/>
      <import plugin="org.eclipse.ui"/>
      <import plugin="de.kt.scripteditor"/>
   </requires>


   <extension
         id="IEPreview"
         name="IE Preview"
         point="de.kt.scripteditor.preview">
      <preview
            name="de.kt.scripteditor.preview.HTMLPreview"
            class="de.kt.scripteditor.preview.HTMLPreview"
            id="de.kt.scripteditor.preview.HTMLPreview">
      </preview>
   </extension>

</plugin>
```

F Programmierrichtlinien

Aus Platzgründen befinden sich die Programmierrichtlinien der Condat Informationssysteme AG auf der beiliegenden CD.