## 27.4 Komponente XmlHtml

Die Komponente gb.xml.html stellt Ihnen Klassen zur Verfügung, mit denen Sie ein neues HTML-Dokument schreiben oder ein existierendes HTML-Dokument ändern können.

Die Komponente gb.xml.html von Adrien Prokopowicz besitzt nur die beiden Klassen *HtmlDocument* und *XmlElement*. Die Klasse XmlElement ist eine Re-Implementation der Klasse XmlElement aus http://gambaswiki.org/wiki/comp/gb.xml/xmlelement und wird hier nicht weiter beschrieben → Kapitel 27.0 XML.

Die Komponente basiert auf gb.xml und die Klasse HtmlDocument (gb.xml.html) beerbt die Klasse XmlDocument in gb.xml.

Ein Objekt der Klasse HtmlDocument können Sie erzeugen. Es repräsentiert ein HTML-Dokument:

```
Dim hHtmlDocument As HtmlDocument
hHtmlDocument = New HtmlDocument ( [ FileName As String ] )
```

### 27.4.1 Eigenschaften

Die Klasse *HtmlDocument* verfügt über diese Eigenschaften:

| Eigenschaft | Datentyp | Beschreibung |
|---|---|---|
| All | XmlNode[] | Zurückgegeben wird ein Array mit allen XML-Knoten. |
| Base | String | Gibt die Basis-URL zurück, die für alle relativen URLs verwendet wird, die in einem Dokument enthalten sind. Wenn im Dokument kein <base>-Tag vorhanden ist, gibt diese Eigenschaft Null zurück und erzeugt bei der Einstellung ein neues <base>-Element. |
| Html5 | Boolean | Wird die Eigenschaft auf True gesetzt, dann wird ein HTML5-Dokument erzeugt und der Dokument-Typ im Prolog html5-konform angegeben: <!DOCTYPE html>. Wird die Eigenschaft auf False gesetzt, dann wird der Dokument-Typ so deklariert: <!DOCTYPE html PUBLIC "-/W3C**DTD XHTML 1.0 Strict**EN" "http:**www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> |
| Title | String | Setzt den Titel für das HTML-Dokument. |
| Lang | String | Setzt die Sprache im lang-Attribut im Dokument-Typ-Tag. |
| Favicon | String | Ein Favicon als Icon in der Adresszeile eines Webbrowsers wird über einen Datei-Pfad zur Icon-Bild-Datei festgelegt. |
| Head | XmlElement | Gibt das <head>-Element zurück oder setzt das <head>-Element des Dokuments. Wenn kein <head>-Element im Dokument vorhanden ist, wird es beim Lesen dieser Eigenschaft erzeugt. |
| StyleSheets | .HtmlDocumentStyleSheets | Die virtuelle Klasse besitzt drei Methoden, deren wichtigste die Add-Methode ist: Sub Add ( Source As String [ , Media As String ] ). Die Methode fügt ein CSS-StyleSheet-Tag in das HTML-Dokument ein. Der Standard-Wert für den optionalen Parameter 'Media' ist "screen". |
| Scripts | .HtmlDocumentScripts | Die virtuelle Klasse besitzt drei Methoden, deren wichtigste die Add-Methode ist: Sub Add ( Source As String ). Die Methode fügt ein Skript-Tag in das HTML-Dokument ein. |
| Body | XmlElement | Gibt das <body>-Element des Dokuments zurück oder legt dieses fest. Wenn kein <body>-Element im Dokument vorhanden ist, wird es beim Lesen dieser Eigenschaft erzeugt: → https://developer.mozilla.org/de/docs/Web/HTML/Element/body. |
| Content | String | Gibt den kompletten Inhalt des HTML-Dokuments in einem String zurück. |

Tabelle 27.4.1.1 : Eigenschaften der Klasse HtmlDocument

### 27.4.2 Methoden

Die Klasse *HtmlDocument* besitzt diese Methoden:

| Methode | Rückgabetyp | Beschreibung |
|---|---|---|
| Sub Open ( FileName As String ) | - | Entfernt den gesamten Inhalt dieses Dokuments und lädt seinen neuen Inhalt aus der angegebenen Datei. *FileName* enthält den Datei-Pfad. Wenn die angegebene Datei ungültig ist oder nicht geöffnet werden kann, so wird ein Fehler ausgelöst. |
| GetElementById (Id As String [ , Depth As Integer ] ) | XmlElement | Die Funktion gibt das Element mit der angegebenen ID als XML-Element zurück. Der optionale Parameter *Depth* bestimmt die Such-Tiefe. |
| GetElementsByTagName ( TagName As String [ , Mode As Integer, Depth As Integer ] ) | XmlElement | Gibt alle Elemente des HTML-Dokuments in einem Array von XML-Elementen zurück, deren Tag-Name 'TagName' mit dem Namen im Dokument übereinstimmt. Das Modus-Argument definiert die verwendete Vergleichsmethode. Es unterstützt GB.Binary, GB.IgnoreCase und GB.Like. Weitere Informationen finden Sie unter 'Vordefinierte Konstanten'. Das Depth-Argument definiert, wo die Suche gestoppt werden soll: Bei einem negativen Wert wird nur am Ende des Baums gestoppt (Standard), 1 überprüft nur das Root-Element, 2 überprüft nur die direkten Kinder des Root-Elements und so weiter ... . |
| GetElementsByNamespace ( Namespace As String [ , Mode As Integer, Depth As Integer ] ) | XmlElement | Gibt alle Elemente des HTML-Dokuments in einem Array von XML-Elementen zurück, deren Namespace-Präfix mit dem Namen übereinstimmt. Das Modus-Argument definiert die verwendete Vergleichsmethode. Es unterstützt GB.Binary, GB.IgnoreCase und GB.Like. Weitere Informationen finden Sie unter 'Vordefinierte Konstanten'. Das Depth-Argument definiert, wo die Suche gestoppt werden soll: Bei einem negativen Wert wird nur am Ende des Baums gestoppt (Standard), 1 überprüft nur das Root-Element, 2 überprüft nur die direkten Kinder des Root-Elements und so weiter ... . |
| GetElementsByClassName ( ClassName As String [ , Depth As Integer ] ) | XmlElement | Gibt alle Elemente des HTML-Dokuments in einem Array von XML-Elementen zurück, deren Klassen-Name mit dem Namen übereinstimmt. Das Depth-Argument definiert, wo die Suche gestoppt werden soll: Bei einem negativen Wert wird nur am Ende des Baums gestoppt (Standard), 1 überprüft nur das Root-Element, 2 überprüft nur die direkten Kinder des Root-Elements und so weiter ... . |
| CreateElement ( TagName As String ) | XmlElement | Die Funktion erzeugt ein neues Element 'TagName'. Der Funktionswert ist ein XML-Element. |
| FromString ( Data As String ) | - | Entfernt den existierenden Inhalt des HTML-Dokuments vollständig und lädt den neuen Inhalt aus dem angegebenen XML-String. Wenn die angegebene Zeichenfolge ungültig ist, wird ein Fehler ausgelöst. |
| ToString ( [ Indent As Boolean ] ) | String | Die Funktion gibt den Inhalt des HTML-Dokuments in einem String zurück, der zum Beispiel später erneut eingelesen werden kann. Wenn der optionale Parameter 'Indent' auf True gesetzt ist, dann wird der Inhalt mit passenden Einrückungen zurückgegeben. |
| Save ( FileName As String [ , Indent As Boolean ] ) | - | Speichert den Inhalt des HTML-Dokuments unter dem mit *FileName* angegebenen Datei-Pfad. Wenn der optionale Parameter 'Indent' auf True gesetzt ist, dann wird der Inhalt des HTML-Dokuments mit passenden Einrückungen formatiert. |

Tabelle 27.4.2.1 : Methoden der Klasse HtmlDocument

### 27.4.3 Projekte

Welche Gründe mag es geben, warum jemand mit den Klassen der Komponente gb.xml.html ein HTML-Dokument schreiben sollte und nicht mit den Methoden der Klasse XmlWriter, was naheliegend wäre? Ein Grund ist darin zu sehen, dass die Klassen von gb.xml.html spezialisiert sind, um ein

HTML-Dokument zu schreiben. Außerdem stellt Ihnen die Klasse HtmlDocument mit einem Dokument vom Typ HtmlDocument einen DOM-Baum zur Verfügung, den Sie aufbauen und gezielt verändern können. Stellen Sie sich die folgende Situation vor: Sie schreiben ein CMS ähnlich zum DokuWiki in Gambas. Auf Anfrage eines Webbrowsers läuft dieses CMS als CGI-Skript auf dem Server und erzeugt entsprechend der angeforderten URL ein HTML-Dokument aus verschiedenen Ressourcen wie zum Beispiel Datenbank-Daten, Eingaben aus Formularen oder Daten aus XML-Dateien, deren Steuerung zum Beispiel XML-Konfigurationsdateien übernehmen. Je nachdem, welche Module in Ihrem CMS geladen sind, werden der HTML-Seite weitere Elemente hinzugefügt, wie zum Beispiel ein automatisch generiertes Inhaltsverzeichnis oder ein Besucher-Zähler im Fußbereich der Seite. Eine solche HTML-Seite sequentiell – also mit einer Reihe von Zeilen der Form "sHTML &= ..." – zu schreiben, ist aufwändig, weil jede Routine genau an der richtigen Stelle aufgerufen werden muss. Wie soll reagiert werden, wenn für den Besucherzähler eine eigene CSS- oder Javascript-Datei eingebunden werden soll? Das muss der Prozedur, die den HTML-Kopfbereich <head>...</head> zusammenbaut natürlich schon vorher bekannt sein! Hier ist es wesentlich besser, wenn man ein Dokument vom Typ HtmlDocument einsetzt. Jede einzubindende Ressource kann dann als neuer Knoten an die Stellen in den DOM-Baum eingehängt werden, wo sie hin gehört. Es ist so u.a. kein Problem, ein neues Stylesheet über die Methode HtmlDocument.Stylesheets.Add(...) zu einem HtmlDocument hinzuzufügen. Erst wenn der DOM-Baum komplett ist, wird er in einen (wohlgeformten!) HTML-String gerendert. Was die Verwendung von gb.xml.html generell angeht, besticht die Komponente durch den Einsatz des 'Document Object Model' (DOM). Sie können im DOM-Baum auch elegant nach interessierenden Informationen suchen. Zum Beispiel gibt die Methode

```
HtmlDocument.Root.GetChildrenByFilter("a[href^=ftp://]")
```

als Funktionswert alle Link-Tags <a>...</a> in einem Array XmlElement[] zurück – also eine Link-Liste, deren Einträge jeweils auf einen FTP-Server verweisen. Sie werden feststellen, dass das selbst mit einem regulären Ausdruck ungleich schwieriger wäre, weil Sie sich zum Beispiel nicht darauf verlassen können, dass 'href' das erste Attribut ist, das in einem Link-Tag aufgelistet ist!

Die vorgestellten Projekte zeigen Ihnen die Verwendung der Klassen der Komponente gb.xml.html.

### 27.4.3.1 Projekt 1

Ein HTML5-Dokument wird im ersten Projekt *neu* geschrieben. Dessen Inhalt zeigt sich so in einem Webbrowser:

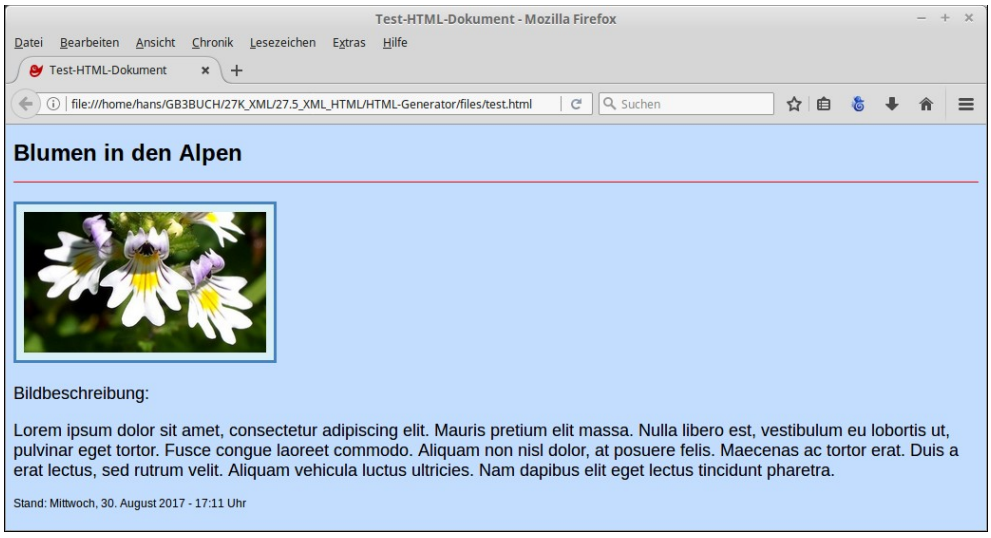


Abbildung 27.4.3.1.1: Inhalt der HTML5-Datei test.html in einem Webbrowser

Der Quelltext wird vollständig angegeben und anschließend kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Public Sub Form_Open()
[4]    FMain.Resizable = True
[5]    FMain.Caption = "HTML5-GENERATOR"
```

```
[6]    btnHTMLShow.Enabled = False
[7]  End
[8]
[9]  Public Sub btnGenerateHTMLDokument_Click()
[10]   TextArea1.Clear()
[11]   TextArea1.Insert(WriteHTMLDocumentDOM().ToString(True))
[12]   SaveHTMLFile()
[13]   btnHTMLShow.Enabled = True
[14] End
[15]
[16] Public Sub btnHTMLShow_Click()
[17]   If Exist(Application.Path &/ "files/test.html") Then
[18]      Shell "firefox " & Application.Path &/ "files/test.html"
[19]   Endif
[20]   btnHTMLShow.Enabled = False
[21] End
[22]
[23] Private Sub SaveHTMLFile()
[24]   WriteHTMLDocumentDOM().Save(Application.Path &/ "files/test.html", True)
[25] End
[26]
[27] Private Function WriteHTMLDocumentDOM() As HtmlDocument
[28]
[29]   Dim hHtmlDocument As New HtmlDocument
[30]   Dim hXMLElement As XmlElement
[31]   Dim sLongText As String
[32]
[33]   sLongText = File.Load("texts/image.description.txt")
[34]
[35]   hHtmlDocument.Html5 = True
[36]   hHtmlDocument.Lang = "de"
[37]   hHtmlDocument.Title = "Test-HTML-Dokument"
[38]   hHtmlDocument.StyleSheets.Add("../css/main.css")
[39]   hHtmlDocument.Favicon = "../images/favicon.png"
[40]   hHtmlDocument.Scripts.Add("../scripts/datetime.js")
[41]
[42]   hHtmlDocument.Body.NewElement("h1")
[43]   hXMLElement = hHtmlDocument.GetElementsByTagName("h1", gb.IgnoreCase)[0]
[44]   hXMLElement.AppendText("Blumen in den Alpen")
[45]
[46]   hHtmlDocument.Body.NewElement("hr")
[47]   hXMLElement = hHtmlDocument.GetElementsByTagName("hr", gb.IgnoreCase)[0]
[48]   hXMLElement.SetAttribute("class", "line")
[49]
[50]   hHtmlDocument.Body.NewElement("br")
[51]
[52]   hHtmlDocument.Body.NewElement("img")
[53]   hXMLElement = hHtmlDocument.GetElementsByTagName("img", gb.IgnoreCase)[0]
[54]   hXMLElement.SetAttribute("src", Application.Path &/ "images/augentrost.jpg")
[55]   hXMLElement.SetAttribute("width", "255")
[56]   hXMLElement.SetAttribute("height", "148")
[57]   hXMLElement.SetAttribute("alt", "euphrasia rostkoviana")
[58]
[59]   hHtmlDocument.Body.NewElement("p")
[60]   hXMLElement = hHtmlDocument.GetElementsByTagName("p", gb.IgnoreCase)[0]
[61]   hXMLElement.AppendText("Bildbeschreibung:")
[62]
[63]   hHtmlDocument.Body.NewElement("p")
[64]   hXMLElement = hHtmlDocument.GetElementsByTagName("p", gb.IgnoreCase)[1]
[65]   hXMLElement.AppendText(sLongText)
[66]
[67]   hHtmlDocument.Body.AppendText("Stand:")
[68]   hHtmlDocument.Body.NewElement("span")
[69]   hXMLElement = hHtmlDocument.GetElementsByTagName("span", gb.IgnoreCase)[0]
[70]   hXMLElement.SetAttribute("id", "datetime")
[71]   hXMLElement.SetAttribute("class", "id")
[72]
[73]   Return hHtmlDocument
[74]
[75] End
```

Kommentar

- Die Funktion *WriteHTMLDocumentDom()* in den Zeilen 27 bis 74 hat als Funktionswert ein HTML-Dokument vom Typ HtmlDocument.
- Zuerst wird in der Zeile 29 ein neues, leeres HtmlDokument erzeugt.
- In der Zeile 35 wird ein HTML5-Prolog erzeugt.
- In den Zeilen 36 bis 40 werden die Tags für die Sprache, den Titel, ein CSS-StylSheet, das Favicon und ein Skript (JavaScript) in den Head-Bereich eingefügt.
- In der Zeile 62 wird zum Beispiel ein zweites Absatz-Tag im Body-Bereich erzeugt und in den

DOM-Baum eingehängt. Dann wird dieses zweite Absatz-Tag im DOM-Baum gesucht und einem Element zugewiesen, dem dann als Inhalt der etwas längere Bildbeschreibungstext zugeordnet wird. Das Element wird dann in den Baum im zweiten Absatz-Tag eingehängt. Dieses Vorgehen finden Sie prinzipiell bei allen eingefügten (Tag-)Elementen.

- Eine Besonderheit stellen die Zeilen 66 bis 70 dar. Hier wird der Funktionswert einer Funktion (aktuelles Datum und aktuelle Zeit) in einem Java-Skript in einem speziellen Bereich-Tag (span) ausgegeben und anschließend css-formatiert. Achtung: Der (Zeit-)Wert ist statisch und ändert sich nur bei einem erneuten Aufruf der HTML-Seite! Die Zeile 69 wird nur dann verständlich, wenn Sie sich die Datei 'datetime.js' und den Abschnitt *.id* der CSS-Datei 'main.css' durchlesen.
- Der Inhalt des HTML-Dokuments wird in einer TextArea angezeigt (Zeile 11).
- Die Zeilen 23 bis 25 übernehmen das Speichern des fertigen HTML-Dokuments im Projekt-Ordner. Ein Speichern-Dialog wird in der Prozedur SaveHTMLFile() nicht angeboten.
- Die Anzeige der HTML-Datei erfolgt im Webbrowser *FireFox* in den Zeilen 16 bis 21. Alternativ können Sie auch '*Desktop.Open("file://" & Application.Path &/ "files/test.html")*' als Ersatz für die Zeile 18 verwenden, wenn stets der aktuelle System-Webbrowser genutzt werden soll.
- Die Zeilen 54 bis 57 könnten Sie durch die Zeile 54* ersetzen, um die drei Attribute festzulegen. Die erste, mehrzeilige Variante ist m.E. besser lesbar, weil man bestimmte Zeichen nicht maskieren muss:

```
[54]  hXMLElement.SetAttribute("src", Application.Path &/ "images/augentrost.jpg")
[55]  hXMLElement.SetAttribute("width", "255")
[56]  hXMLElement.SetAttribute("height", "148")
[57]  hXMLElement.SetAttribute("alt", "euphrasia rostkoviana")
```

```
[54*]  hXMLElement.SetAttribute("src", Application.Path &/ "images/augentrost.jpg\" width=\"255\"
height=\"148\" alt=\"euphrasia rostkoviana")
```

Das Programm erzeugt den folgenden Inhalt der HTML-Datei:

```
<!DOCTYPE html>
<html lang="de">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>
   Test-HTML-Dokument
  </title>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="/home/hans/.../css/main.css" type="text/css" media="screen" />
  <link rel="icon" href="/home/hans/.../images/favicon.png" />
  <script src="/home/hans/.../scripts/datetime.js" type="text/javascript">
  </script>
 </head>
 <body>
  <h1>
   Blumen in den Alpen
  </h1>
  <hr class="line" />
  <br />
  <img src="/home/hans/.../images/augentrost.jpg" width="255" height="148" alt="Euphrasia rostkoviana" />
  <p>
   Bildbeschreibung:
  </p>
  <p>
   Augentrost oder Euphrasia rostkoviana: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris
pretium elit massa. Nulla libero est, vestibulum eu lobortis ut, pulvinar eget tortor. Fusce congue laoreet
commodo. Aliquam non nisl dolor, at posuere felis. Maecenas ac tortor erat. Duis a erat lectus, sed rutrum
velit. Aliquam vehicula luctus ultricies. Nam dapibus elit eget lectus tincidunt pharetra.
  </p>
  Stand:
  <span id="datetime" class="id">
  </span>
 </body>
</html>
```

Kommentar

- Die angegebenen Pfade werden nur gekürzt angezeigt.
- Die Zeile 35 im Programm-Quelltext erzeugt einen Fehler, da zwei <meta>-Tags generiert werden, die beide rot markiert sind. Vom Entwickler Adrien Prokopowicz wurde der Fehler bereits in der Entwickler-Version unter: https://gitlab.com/gambas/gambas/commit/6fd076a9ef55926dee-e27e60a0f7f24bb4839f94 beseitigt.
- Ohne die erste, fehlerhafte rote Zeile bestätigt der HTML-Validator auf der Webseite https://vali-

dator.w3.org/nu/: *Document checking completed. No errors or warnings to show*. Prima!

- Die Anzeige des Inhalts der o.a. HTML-Datei sehen Sie in der Abbildung 27.4.3.1.1 (oben).

Nutzen Sie die stabile Gambas-Version 3.10.0 oder darunter, dann kann Ihnen die folgende Funktion helfen, ein wohlgeformtes HTML-Dokument zu erzeugen:

```
Private Function WorkAround(HtmlDocument As HtmlDocument) As HtmlDocument

  Dim hHtmlDocument As New HtmlDocument
  Dim hXMLElement As XmlElement

  hHtmlDocument = HtmlDocument
  hXMLElement = hHtmlDocument.GetElementsByTagName("meta")[0] ' Holt das erste <meta>-Tag
' Sie müssen auf das übergeordnete Element zugreifen, um das angegebene <meta>-Tag zu entfernen
  hXMLElement.Parent.RemoveChild(hXMLElement)

  Return hHtmlDocument

End
```

Es wird das erste <meta>-Tag aus dem DOM-Baum entfernt. Jetzt müssen Sie nur noch im Quell-Text die Zeile 11 gegen die darunter aufgeführte tauschen:

```
TextArea1.Insert(WriteHTMLDocumentDOM().ToString(True))
TextArea1.Insert(WorkAround(WriteHTMLDocumentDOM()).ToString(True))
```

Gleiches für die Zeile 24:

```
WriteHTMLDocumentDOM().Save(Application.Path &/ "files/test.html", True)
WorkAround(WriteHTMLDocumentDOM()).Save(Application.Path &/ "files/test.html", True)
```

Im Projekt1, das Ihnen als Archiv im Download-Bereich zum Testen zur Verfügung gestellt wird, werden die erforderlichen Korrekturen in Abhängigkeit von der verwendeten Gambas-Version (→ System.FullVersion) bereits *automatisch* vorgenommen.

### 27.4.3.2 Projekt 2

Im zweiten Projekt wird der Inhalt einer existierenden HTML-Datei *stark geändert*. Der Schwerpunkt liegt in der Suche nach ausgewählten, zu ändernden Datei-Inhalten, dem Editieren und Ändern dieser Inhalte sowie dem Einhängen der geänderten Inhalte in den DOM-Baum. Zum Ändern zählt auch das Löschen einzelner HTML-Tags und dessen Inhalt → Kapitel 27.2.3 Änderungen an einem XML-Dokument. Auch dort werden Änderungen über den DOM-Baum realisiert.

Für das zweite Projekt wird nur der Quelltext der Funktion *UpdateHTMLDocument(sPath As String)* angegeben, der intern umfangreich kommentiert ist. Beachten Sie, dass der folgende Quelltext nur eine *individuelle* Änderungsfunktion repräsentiert:

```
Private Function UpdateHTMLDocument(sPath As String) As HtmlDocument

  Dim hHtmlDocument As New HtmlDocument
  Dim hXMLElement, hExistChild, hNewChild As XmlElement
  Dim aH1, aStyle As XmlElement[]
  Dim sAttribute, sLongText, sText, sOldText, sNewText As String
  Dim iWidth, iHeight As Integer
  Dim iWH As New Integer[]

  sLongText = File.Load(Application.Path &/ "texts/description.txt")
' Der Inhalt der Original-Datei wird in ein HTML-Dokument geschrieben
' Bestehender Inhalt wird überschrieben.
  hHtmlDocument.Open(sPath)

' Spezialfall: Dem existierenden <html>-Tag wird ein Attribut über die Eigenschaft .Lang hinzugefügt
  hHtmlDocument.Lang = "DE_de"


' Das erste <meta>-Tag im Head-Bereich über seinen Namen finden und sichern
  hExistChild = hHtmlDocument.Head.GetChildrenByTagName("meta")[0]
' Ein zweites <meta>-Tag erzeugen
  hNewChild = New XmlElement("meta")
' Das zweite <meta>-Tag NACH dem ersten <meta>-Tag in den Dokument-Baum einfügen
  hHtmlDocument.Head.InsertAfter(hExistChild, hNewChild)
' Dem zweiten <meta>-Tag 2 Attribute hinzufügen
  hXMLElement = hHtmlDocument.Head.GetChildrenByTagName("meta")[1]
```

```
  hXMLElement.NewAttribute("name", "author")
  hXMLElement.NewAttribute("content", "Hans Lehmann - Osterburg - 2017")
' Zurück zum Head-Bereich im Dokument-Baum
  hXMLElement = hXMLElement.Parent

' Das zweite <meta>-Tag im Head-Bereich über seinen Namen finden und sichern
  hExistChild = hHtmlDocument.Head.GetChildrenByTagName("meta")[1]
' Ein drittes <meta>-Tag erzeugen
  hNewChild = New XmlElement("meta")
' Das dritte <meta>-Tag NACH dem zweiten <meta>-Tag in den Dokument-Baum einfügen
  hHtmlDocument.Head.InsertAfter(hExistChild, hNewChild)
' Dem dritten <meta>-Tag 2 Attribute hinzufügen
  hXMLElement = hHtmlDocument.Head.GetChildrenByTagName("meta")[2]
  hXMLElement.NewAttribute("name", "description")
  hXMLElement.NewAttribute("content", "Änderung und Erweiterung einer HTML-Datei (DOM).")
' Zurück zum Head-Bereich im Dokument-Baum
  hXMLElement = hXMLElement.Parent

' Das erste <link>-Tag im Head-Bereich über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("link")[0]
' Den alten Attribut-Wert sichern
  sAttribute = hXMLElement.GetAttribute("href")
' Den neuen Attribut-Wert festlegen
  sAttribute = "../images/favicon.ico"
' Den neuen Attribut-Wert zuweisen
  hXMLElement.SetAttribute("href", sAttribute)

' Änderungen von CSS-Festlegungen im <style>-Tag (Head) vornehmen
' Das <style>-Tag im Head-Bereich über seinen Namen finden und sichern
  aStyle = hHtmlDocument.Head.GetChildrenByTagName("style")
  If aStyle.Count > 0 Then
   ' CSS-Festlegungen für das <body>-Tag sichern
     sText = aStyle[0].TextContent
     sOldText = "font-size: 10px"
     sNewText = "font-size: 14px"
   ' Schriftgröße ersetzen
     sText = Replace$(sText, sOldText, sNewText)
     sOldText = "Verdana"
     sNewText = "\"DejaVu Sans Mono\""
   ' Schrift ersetzen
     sText = Replace$(sText, sOldText, sNewText)
   ' Neue CSS-Festlegungen für das <body>-Tag zuweisen
     aStyle[0].TextContent = sText
  Endif

' Dem <body>-Tag ein (CSS-Farbwert-)Attribut hinzufügen
  hHtmlDocument.Body.SetAttribute("style", "color:darkblue;")

' Das erste <h1>-Tag im Body-Bereich über seinen Namen finden und sichern
  aH1 = hHtmlDocument.Body.GetChildrenByTagName("h1")
' Text im <h1>-Tag ersetzen
  If aH1.Count > 0 Then
     aH1[0].TextContent = Replace$(aH1[0].TextContent, "/var/www", "~/public_html")
  Endif

' Das erste <img-Tag über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("img")[0]
' Style-Attribut im <img>Tag über seinen Namen finden und sichern
  sAttribute = hXMLElement.GetAttribute("style")
' Breite und Höhe des Bildes ermitteln und neu festlegen (normieren)
  For Each sText In Scan(sAttribute, "*:*px;*:*px")
    If IsInteger(sText) Then iWH.Add(CInteger(sText))
  Next
  If iWH[0] < 32 Or iWH[0] > 64 Then
     If (iWH[0] >= iWH[1] And iWH[0] <> 0 And iWH[1] <> 0) Then
        iWidth = 64
        iHeight = 64 * (iWH[1] / iWH[0])
     Endif
  Endif
  If iWH[1] < 32 Or iWH[1] > 64 Then
     If (iWH[1] >= iWH[0] And iWH[0] <> 0 And iWH[1] <> 0) Then
        iHeight = 64
        iWidth = 64 * (iWH[0] / iWH[1])
     Endif
  Endif
' Neuen Attribut-Wert festlegen
  sAttribute = "width:" & Str(iWidth) & "px;height:" & Str(iHeight) & "px"
' Den neuen Attribut-Wert zuweisen
  hXMLElement.SetAttribute("style", sAttribute)

' Zurück zum Body-Bereich im Dokument-Baum
  hXMLElement = hXMLElement.Parent
```

```
' Ein neues <p>-Tag erzeugen
  hXMLElement.NewElement("p")
' Das erste <p>-Tag über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("p")[0]
' Text in das <p>-Tag einfügen
  hXMLElement.AppendText(sLongText)

' Ein neues <p>-Tag erzeugen - Variante
  hHtmlDocument.Body.NewElement("p")
' Das zweite <p>-Tag über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("p")[1]
' Text in das <p>-Tag einfügen
  hXMLElement.AppendText("Eine gewöhnliche HTML-Datei besteht grundsätzlich aus den folgenden
Abschnitten:")

' Ein neues <ol>-Tag erzeugen
  hHtmlDocument.Body.NewElement("ol")
' Das erste <ol>-Tag über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("ol")[0]
' Dem 1. <ol>-Tag ein Attribut hinzufügen
  hXMLElement.SetAttribute("style", "text-indent:2em; list-style:upper-roman")

' Ein neues <li>-Tag erzeugen
  hXMLElement.NewElement("li")
' Das erste <li>-Tag über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("li")[0]
' Listen-Text in das <li>-Tag einfügen
  hXMLElement.AppendText("Dokumenttyp – Deklaration/Prolog (Angabe zur verwendeten HTML - Version).")
' Zurück auf die 'ol'-Ebene im Dokument-Baum
  hXMLElement = hXMLElement.Parent

  hXMLElement.NewElement("li")
  hXMLElement = hHtmlDocument.GetElementsByTagName("li")[1]
  hXMLElement.AppendText("Kopf (head) - Kopfdaten wie zum Beispiel 'title' oder 'meta'-Angaben.")
  hXMLElement = hXMLElement.Parent

  hXMLElement.NewElement("li")
  hXMLElement = hHtmlDocument.GetElementsByTagName("li")[2]
  hXMLElement.AppendText("Körper (body) – Inhalt als Text mit Überschriften, Verweisen, Grafik-
Verweisen ... .")
  hXMLElement = hXMLElement.Parent

' Ein neues <p>-Tag erzeugen
  hHtmlDocument.Body.NewElement("p")
' Das dritte <p>-Tag über seinen Namen finden und sichern
  hXMLElement = hHtmlDocument.GetElementsByTagName("p")[2]
' Text in das <p>-Tag einfügen
  hXMLElement.AppendText("SCHLUSS -> LETZTER ABSATZ")

  Return hHtmlDocument

End
```

Mit Hilfe der o.a. Funktion kann der Inhalt der Original-HTML-Datei mit Inline-CSS-Anweisungen:

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   INDEX.HTML
  </title>
  <meta charset="utf-8" />
  <link rel="icon" href="../images/favicon.png" />
  <style>
   body {background-color: #C3DDFF;font-family: Verdana;font-size: 10px}
   h1 {font-family: Arial;font-size: 32px;color: blue;}
  .line {border:none; border-top:1px solid #0000FF; color:#FFFFFF; background-color: #FFFFFF; height: 1px;}
  </style>
 </head>
 <body>
   <br />
   <img src="../images/html5_logo.png" alt="HTML5" style="width:256px;height:256px">
   <br />
   <h1>
     HTML-Datei im Webordner /var/www
    </h1>
    <hr class="line"></hr>
 </body>
</html>
```

in eine HTML5-Datei mit geändertem Inhalt konvertiert werden:

```
<!DOCTYPE html>
<html lang="DE_de">
 <head>
  <title>
   INDEX.HTML
  </title>
  <meta charset="utf-8" />
  <meta name="author" content="Hans Lehmann - Osterburg - 2017" />
  <meta name="description" content="Änderung und Erweiterung einer HTML-Datei (DOM)." />
  <link rel="icon" href="../images/favicon.ico" />
  <style>
   body {background-color: #C3DDFF;font-family: &quot;DejaVu Sans Mono&quot;;font-size: 14px}
   h1 {font-family: Arial;font-size: 32px;color: blue;}
  .line {border: none; border-top: 1px solid #0000FF; color: #FFFFFF; … ; height: 1px;}
  </style>
 </head>
 <body style="color:darkblue;">
  <br />
  <img src="../images/html5_logo.png" alt="HTML5" style="width:64px;height:64px" />
  <br />
  <h1>
     HTML-Datei im Webordner ~/public_html
  </h1>
  <hr class="line" />
  <p>
   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris pretium elit massa. Nulla libero est,
vestibulum eu lobortis ut, pulvinar eget tortor. Fusce congue laoreet commodo. Aliquam non nisl dolor, at
posuere felis. Maecenas ac tortor erat. Duis a erat lectus, sed rutrum velit. Aliquam vehicula luctus ul-
tricies. Nam dapibus elit eget lectus tincidunt pharetra.
  </p>
  <p>
   Eine gewöhnliche HTML-Datei besteht grundsätzlich aus den folgenden Abschnitten:
  </p>
  <ol style="text-indent:2em; list-style:upper-roman">
   <li>
    Dokumenttyp – Deklaration/Prolog (Angabe zur verwendeten HTML - Version).
   </li>
   <li>
    Kopf (head) - Kopfdaten wie zum Beispiel 'title' oder 'meta'-Angaben.
   </li>
   <li>
    Körper (body) – Inhalt als Text mit Überschriften, Verweisen, Grafik-Verweisen ... .
   </li>
  </ol>
  <p>
   SCHLUSS -&gt; LETZTER ABSATZ
  </p>
 </body>
</html>
```


Abbildung 27.4.3.2.1: Inhalt der geänderten HTML-Datei in einem Webbrowser