HTML, CSS und JavaScript *

Jens Lechtenbörger

VM Neuland im Internet 2021

Inhaltsverzeichnis

1 Einführung

Dieses Dokument beinhaltet Verweise auf Quellen rund um HTML, CSS und JavaScript und soll ohne Anspruch von Vollständigkeit einen Einstieg in ausgewählte Aspekte dieser Themen ermöglichen.

1.1 Lernziele

- Separation of Concerns am Beispiel HTML, CSS, JavaScript erläutern
- HTML, CSS und JavaScript am Beispiel erläutern und anpassen

1.2 Verschiedene Einstiege in die Thematik

Für einen ersten Einstieg könnte das im folgenden Unterabschnitt dargestellte Demo-HTML-Fragment dienen. Alternativ überspringen Sie den Unterabschnitt zunächst und kommen für Tests später zurück.

*Dieses PDF-Dokument ist eine minderwertige Version einer OER-HTML-Seite; freies Repository mit Org-Mode-Quelltexten.

1.3 Demo-HTML-Fragment

Das folgende HTML-Fragment ist (dank des Plugins Klipse) im oberen Bereich *editierbar* und zeigt im unteren Bereich die vom Browser dargestellte Ansicht.

```
<style>
h1.demo-headline { color: blue; }
i { color: darkblue; }
.green { color: green; }
#demo-id { color: red; }
</style>
<h1 class="demo-headline">Hello World!</h1>
<p class="green">This is a <i>paragraph</i> of text with class ‘‘green’’.</p>
<p id="demo-id">This is another paragraph with id ‘‘demo-id’’.</p>
```

Ganz grob bestehen HTML-Dokumente aus **Elementen**, die durch öffnende und schließende **Tags** definiert werden. Das Beispiel-Fragment beginnt mit dem Element `style` (zwischen öffnendem Tag `<style>` und schließendem Tag `</style>`), das zur Definition von **CSS-Regeln** dient, die das Layout bestimmen (Schriftart, -größe, -farbe, Abstände usw.). Auf die Stildefinition folgen eine Überschrift der ersten Gliederungsebene (`h1`) und zwei Paragraphen (`p`).

CSS-Regeln werden in ihrer eigenen Sprache definiert. Hier sehen Sie verschiedene Farben, die durch unterschiedliche **Selektoren** an Teile des HTML-Dokuments gebunden werden. Eine Regel legt fest, dass Element `i` in Dunkelblau dargestellt wird. Zudem können Regeln für **Klassen** definiert werden, deren Namen ein Punkt vorangestellt wird (hier `green`), für eindeutig **identifizierte** Elemente (mit einem Hashtag wie bei `#demo-id`), für Kombinationen (Überschriften der Ebene 1 *und* der Klasse `demo-headline` durch `h1.demo-headline`) und vieles mehr.

Ändern Sie obiges Dokument, fügen Sie z. B. testweise eine Überschrift der Ebene 2 hinzu, und ändern Sie den Stil des Elements `i`. Beachten Sie, dass CSS-Vorgaben aus dem umgebenden Dokument in Kombination mit Ihren CSS-Angaben wirken (Änderungen an `i` beeinflussen beispielsweise alle kursiven Elemente im Text). Was Sie alles mit CSS ändern können, erfordert Ihre eigene Recherche. Verändern Sie etwa die Schriftgröße bestimmter Elemente mit einer eigenen CSS-Klasse.

2 HTML

Die Hypertext Markup Language (HTML) ist *die* Sprache (genauer: Sprachfamilie) des Web, die typischerweise in Kombination mit CSS und JavaScript zum Einsatz kommt. Das selfHTML-Wiki ist *die* deutschsprachige OER rund um HTML.

Lesen Sie im selfHTML-Wiki, wie mit diesen drei Sprachen Inhalt, Präsentation und Verhalten getrennt werden, und **erklären** Sie danach was unter „Separation of Concerns“ verstanden wird, welche Vorteile die Einhaltung dieses Prinzips verspricht und wie es mit der Trennung von Inhalt, Präsentation und Verhalten im Kontext von HTML, CSS und JavaScript zusammenhängt.

Wenn Sie noch keine Erfahrung mit HTML haben, empfehle ich diesen HTML-Einstieg im selfHTML-Wiki, der auch Grundlagen von CSS behandelt. Weitergehende Details zu CSS finden sich dann im Einstieg in CSS. Um die Grundlagen von JavaScript zu erlernen, können Sie das Tutorial JavaScript Hero absolvieren. (Im Kontext dieses Vertiefungsmoduls ist das allerdings nicht nötig. Die weiter unten angegebenen Texte des selfHTML-Wiki zum DOM sollten Ihnen helfen, die folgenden Aufgaben zu lösen.)

Im Zuge der Web-Entwicklung können Sie beispielsweise per Docker einen Web-Server auf dem eigenen Rechner betreiben, der während der Entwicklung Ihre Web-Ressourcen ausliefert.

Zudem existieren Online-Editoren, die gleichzeitig sowohl den Quelltext einer Web-Seite als auch ihre Browser-Darstellung live anzeigen. In sehr einfacher Form funktioniert das z. B. in HTML-Dokumenten, in denen das Plugin Klipse eingebettet ist. Umfangreichere Funktionalität bieten etwa JS Bin (v4 als freie Software) oder proprietäre Alternativen wie JSFiddle (Hello-World-Beispiel von selfHTML) und CodePen.

Schließlich sei auf die **Web Developer Tools** moderner Browser hingewiesen, die Sie über das Menü oder Shortcuts aufrufen können. Probieren Sie im Firefox oder Chromium `Strg-Umschalt-I`, und schauen Sie sich den HTML-Text dieser Seite an (Reiter Inspector im Firefox, Elements im Chromium; in weiteren Reitern können Sie sich beispielsweise Netzwerkaufrufe einschließlich der HTTP-Header und Cookies ansehen). Beachten Sie, wie im Kopf des HTML-Textes (`<head>`) CSS und JavaScript eingebunden werden; im Körper (`<body>`) finden Sie diesen Text gefolgt von weiterem JavaScript.

3 JavaScript, DOM

JavaScript ist *die* Programmiersprache des Web. JavaScript-Programme können in HTML-Dokumente eingebunden und dann vom Browser ausgeführt werden. Sie können dann z. B. das HTML-Dokument dynamisch anpassen oder im Hintergrund Daten mit Servern im Internet austauschen. Die Möglichkeiten sind nahezu unbegrenzt und können sowohl für gute Zwecke geals auch Kriminelles missbraucht werden.

Als Schnittstelle zwischen JavaScript und HTML dient das Document Object Model (DOM), zu dem das selfHTML-Wiki eine Kurzerläuterung und ein etwas längeres Tutorial liefert.

Im Projekt Solid werden Anwendungen mit JavaScript geschrieben. Ne-

ben den im Getting-Started-Tutorial genannten JavaScript Client libraries existieren Ansätze zur Unterstützung der JavaScript-Frameworks React (Solid React SDK, React-Komponenten) und Angular (Solid Angular Yeoman Generator).

Die nachfolgend wiedergegebene Datei index.jsx der Demo der React-Komponenten zeigt Grundzüge der Manipulation des DOM mit JavaScript:

```
1  import App from './app';
2  import React from 'react';
3  import ReactDOM from 'react-dom';
4
5  const container = document.createElement('div');
6  document.body.appendChild(container);
7  ReactDOM.render(<App/>, container);
```

In Zeile 5 wird ein HTML-div-Element erzeugt und der Variablen `container` zugewiesen. Der `container` wird in Zeile 6 dynamisch dem aktuellen HTML-Dokument hinzugefügt. (Beides geschieht mit Standard-JavaScript-Methoden; eine Web-Suche nach den Methodennamen führt zu Erläuterungen unter `developer.mozilla.org`.) In Zeile 7 sorgt eine React-Methode dann dafür, dass dem `container` Inhalt hinzugefügt wird, der in der Datei app.jsx definiert wird.

4 Fazit

Web-Entwicklung profitiert von zahlreichen Kenntnissen, unter denen dieses Dokument nur eine subjektive Auswahl angerissen hat. Gar nicht angesprochen wurden etwa Barrierefreiheit, SEO, Optimierung für mobile Geräte, die auch im selfHTML-Wiki zur Sprache kommen; zum Thema Sicherheit liefert das Open Web Application Security Project (OWASP) eine ausgiebige Quelle.

Lizenzangaben