HTML, CSS und JavaScript *

Jens Lechtenbörger

VM OER 2020/2021

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
- Exemplarische Web-Seiten über Web-Server in Docker testen
- Zielsetzungen von Same-Origin Policy und Cross-Origin Resource Sharing erläutern
- XSS als Beispiel für Injection-Angriffe erläutern
  - (Mindestens) ein Beispiel für unbeabsichtigte Ausführung von Cross-Origin JavaScript angeben und beheben

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

- (Über gesamtes VM hinweg: Eignung von HTML für OER diskutieren)

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.

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.

3 JavaScript, Same- und Cross-Origin

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.

In der Regel befolgen Browser die sogenannte Same-Origin-Policy, nach der beispielsweise JavaScript von einer Web-Seite nur dann auf das DOM einer zweiten Web-Seite zugreifen darf, wenn beide Web-Seiten denselben Ursprung (engl. *origin*, deutsch manchmal auch irreführend Domäne) aufweisen, ihre URLs also (1) dasselbe Schema (z. B. beide `http` oder beide `https`), (2) denselben Servernamen (IP-Adresse oder Domänenname, z. B. `www.uni-muenster.de`) und (3) denselben Port (z. B. Port 8080) aufweisen. Durch diese Einschränkung der Same-Origin-Policy soll verhindert werden, dass schützenswerte Inhalte einer Web-Seite (z. B. Cookies) von einer anderen (bösartigen) Web-Seite ausgelesen oder verändert werden.

Um Ressourcen aus verschiedenen Domänen zu kombinieren und die Zusammenarbeit zwischen verschiedenen Ursprüngen gezielt zuzulassen, gibt es einerseits den Ansatz Cross-Origin Resource Sharing (CORS). Andererseits stellt die JavaScript-Methode Window.postMessage() ein Beispiel für die sichere Cross-Origin-Kommunikation zwischen Window-Objekten (z. B. Seite und Pop-up oder iframe) dar, das in diesem Blog-Beitrag aufgegriffen wird. Unter anderem verweist der Blog-Beitrag auf eine Live-Demo auf CodePen. Bearbeiten Sie zu dieser Demo bitte folgende Aufgaben!

1. Die Live-Demo funktioniert bereits länger nicht mehr. Welches offensichtliche Problem besteht? Wenn Sie den Quelltext ansehen, erkennen Sie vielleicht auch ein subtileres Problem, das die Demo bereits im letzten Jahr hat scheitern lassen.

2. In Learnweb finden Sie ein ZIP-Archiv mit auf der Demo basierendem Code. Vergewissern Sie sich, dass Sie den Code verstehen und testen ihn dann, indem Sie mit Docker zwei Web-Server (auf unterschiedlichen Ports, die Sie dem Quelltext entnehmen) nutzen. Geben Sie an, wie Sie die Server starten und was Sie zum Test im Browser gemacht haben.

4 Sicherheit

Programmierung im Web-Kontext ist anfällig für diverse Sicherheitslücken, wozu das Open Web Application Security Project (OWASP) zahlreiche OER bereitstellt (vgl. „List of Attacks“).

Lesen Sie, was unter Injection Theory verstanden wird, in deren Rahmen Cross Site Scripting (XSS) (Erläuterungen bei OWASP, Wikipedia-Artikel) erklärt werden kann. Zum Schutz vor XSS liefert OWASP ein Cheat Sheet. Beachten Sie, dass beispielsweise die Dokumentation zur Java-Script-Methode `postMessage()` folgenden Hinweis enthält: „always verify the syntax of the received message“

Beantworten Sie folgende Fragen.

1. Skizzieren Sie *kurz* das Ziel von XSS-Angriffen sowie grundlegende Ideen ihrer Abwehr. Sie sollten dabei nicht die Punkte des Cheat Sheet von OWASP wiedergeben, sondern versuchen, so etwas wie eine allgemeine Philosophie zu formulieren.

2. Versuchen Sie, Cross-Origin-JavaScript im Beispiel zu `postMessage()` zur Ausführung zu bringen (mit dem modifizierten Code in Learnweb). Geben Sie im Erfolgsfall an, wie der Angriff aussieht und welche Änderungen nötig sind, um den Angriff abzuwehren. Erklären Sie im Misserfolgsfall, was Sie ausprobiert haben bzw. an welchen Stellen Sie gesucht haben.

3. Wie beurteilen Sie diese Lerneinheit zu HTML, CSS und JavaScript vor dem Hintergrund Ihres persönlichen Vorwissens bezüglich Schwierigkeit und Aufwand?

5 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 wurde bereits OWASP als ausgiebige Quelle empfohlen.

Lizenzangaben