Die Grundlagen der Webentwicklung

Begrifflichkeiten

Client/Server

Webseiten und Webanwendungen (Definition und Unterschied siehe Ende des Abschnitts) bestehen aus einem Teil, der auf der Clientseite ausgeführt wird (dem Frontend), und einem Teil, der auf der Serverseite ausgeführt wird (dem Backend). Auf Serverseite sorgt ein Webserver dafür, dass die Webseite bereitgestellt wird. Auf Clientseite greift man über einen Webclient (auch nur Client oder auch User Agent genannt) auf die Webanwendung zu. In der Regel handelt es sich dabei um einen Webbrowser (kurz: Browser), aber es gibt auch noch andere Arten von Clients wie beispielsweise Screen-reader, kommandozeilenbasierte bzw. programmatisch gesteuerte HTTP-Clients oder sogenannte Headless Browser, die keine grafische Oberfläche haben. Ruft man im Browser eine Webseite auf, ist der Ablauf dabei Folgender: Auf Clientseite gibt der Nutzer im Browser die Adresse ein (auch URL für Uniform Resource Locator, siehe auch nächster Abschnitt) und bestätigt mit Enter bzw. dem entsprechenden Button im Browser das »Laden« der Webseite. Der Browser generiert daraufhin im Hintergrund eine Anfrage, die über das HTTP-Protokoll an den Server gesendet wird. Diese Anfrage nennt man auch HTTP-Anfrage oder HTTP-Request. Auf Serverseite nimmt der Webserver die Anfrage entgegen und generiert eine passende Antwort (HTTP Antwort bzw. HTTP Response), die er dann an den Client zurückschickt. Der Browser wiederum nimmt die Antwort entgegen und rendert, sprich visualistert die Webseite. Eventuell benötigte Ressourcen wie Bilder etc. lädt der Browser dabei automatisch nach, damit sie dargestellt werden können.

Begriffsdefinition

- Eine *Webseite* bezeichnet ein einzelnes HTML-Dokument, das unter einer bestimmten URL abgerufen werden kann.
- Eine *Website* dagegen ist eine Zusammenfassung verschiedener solcher einzelner Webseiten, beispielsweise die Schulhomepage https://www.htl-hl.ac.at
- Eine *Webanwendung* handelt es sich um eine Website, die sich eher wie eine Desktopanwendung anfühlt. Beispiele hierfür sind die im Browser verfügbaren Anwednungen von Microsoft (Office 365 - Word, Excel, ...)

Zusammenhang von URLs, Domains und IP-Adressen

Die Adresse, die in die Adressleiste des Browsers eingegeben wird, wird als URL (Uniform Resource Locator) bezeichnet.
Eine URL besteht dabei aus verschiedenen Teilen:

https://	**www.htl-hl.ac.at**	**:80**	**/web/abt/it/index.html**	**?language=de**	**#medientechnik**
Schema/Protokoll	Host	Port	Pfad	Query String	Fragment (Verweis innerhalb eines HTML Dokuments)

- **Protokoll/Schema:** Definiert das zu verwendende Protokoll. Mögliche Protokolle sind beispielsweise: - HTTP (Hypertext Transfer Protocol): Für das Übertragen von Webseiten wird das Protokoll HTTP verwendet oder dessen sichere Variante HTTPS (Hypertext Transfer Protocol Secure). - FTP (File Transfer Protocol): Dieses Protokoll dient dem Übertragen von Dateien zu oder von einem FTP-Server. - SMTP (Simple Mail Transfer Protocol): Dieses Protokoll kommt für das Übertragen von E-Mails zum Einsatz.
- **Host (auch Hostname):** Identifiziert den Webserver eindeutig. Der Host besteht dabei aus Subdomain, Domain und Top-Level-Domain. Der Host *www.htl-hl.ac.at* beispielsweise besteht aus der Subdomain *www*, der Domain *htl-hl*, der Second-Level-Domain *ac* und der Top-Level-Domain *at*.
- **Port:** Gibt an, über welchen Kanal auf den Webserver zugegriffen werden soll. In der Regel ist dieser Teil beim normalen Browsen nicht sichtbar, da die Standardports (beispielsweise 80 für HTTP oder 443 für HTTPS, siehe auch https://de.wikipedia.org/wiki/Liste_der_standardisierten_Ports von den Browsern in der Adressleiste nicht angezeigt werden. Für die lokale Entwicklung und insbesondere für die Entwicklung von Webservices werden häufiger individuellen Ports eingesetzt. Beispielsweise könnte eine URL eines lokal am Rechner laufenden Webservices so aussehen: http://localhost:8080/myservice/api/users.
- **Pfad (auch Path):** Gibt den Pfad auf dem Webserver zu der angefragten Datei an (bzw. allgemeiner zu der angefragten Ressource, weil es sich nicht immer um eine Datei im physischen Sinne handeln muss). In der URL https://www.htl-hl.ac.at/web/fileadmin/img/HTL_Logo_fin_RGB.svg beispielsweise ist */web/fileadmin/img/HTL_Logo_fin_RGB.svg* der Pfad. Der Pfadtrenner ist dabei immer ein vorwärtsgerichteter Schrägstrich (Slash).
- **Query String:** Hierüber können in Form von Schlüssel-Wert-Paaren zusätzliche Informationen übergeben werden, anhand derer der Webserver die HTTP-Antwort generieren kann. Der Query String wird dabei über ein Fragezeichen eingeleitet, die einzelnen Query-String-Parameter sind durch ein kaufmännisches *Und &* verbunden und die Schlüssel und Werte jeweils durch ein Gleichheitszeichen getrennt, also zum Beispiel https://www.htl-hl.ac.at/web/index.php?id=217&q=Medientechnik.

Aufbau von Webapplikationen

Für die Entwicklung von Webapplikationen sind insbesondere die Entwicklung im Frontend drei Sprachen ausschlaggebend und für Webentwickler unverzichtbar.
Konkret sind dies die *Auszeichnungssprache HTML*, die *Stilsprache CSS* und die *Programmiersprache JavaScript*, die in der Regel in Kombination auf einer Webseite zum Einsatz kommen.

Hypertext Markup Language - HTML

Mit Hilfe von HTML wird über HTML-Elemente und HTML-Attribute die **Struktur** einer Webseite definiert und die Bedeutung (die Semantik) einzelner Komponenten auf einer Webseite festgelegt. Beispielsweise wird mit HTML beschrieben, welcher Bereich auf der Webseite den Hauptinhalt darstellt, welcher die Navigation und welcher den Fußbereich, und Komponenten wie Formulare, Listen, Schaltflächen, Eingabefelder oder Tabellen definiert.

Versionen

- HTML: Die erste Version (noch ohne Versionsnummer) wurde bereits 1992 veröffentlicht, basierend auf Vorschlägen, die Tim Berners-Lee (der Erfinder von HTML) bereits 1989 am CERN, der *Europäischen Organisation für Kernforschung*, erarbeitet hatte.
- HTML 4.0: Wurde 1997 veröffentlicht und ist insofern wichtig, als ab dieser Version CSS (Cascading Style Sheets) zur Verfügung standen.

- XHTML 1.0: Mit dem Erscheinen der Sprache XML (Extensible Markup Language) wurde entschieden, die Sprache HTML nach Regeln von XML neu zu formulieren. Diese Regeln sind strenger als die von HTML 4.0, beispielsweise dürfen Attribute nur in Kleinbuchstaben geschrieben werden und müssen immer einen Wert aufweisen.
- HTML5: Wurde im Jahr 2014 veröffentlicht und ersetzte die vorherigen Versionen von HTML (und XHTML). HTML5 (übrigens absichtlich mit der Versionsnummer direkt an dem *HTML* geschrieben) führte eine Reihe von neuen Funktionalitäten ein, sowie zahlreiche neue Web-APIs. Der HTML-Standard wird - wie viele andere Standards im Web - vom W3C gepflegt, dem World Wide Web Consortium. Dabei legte das W3C lange Zeit viel Wert darauf, stabile Spezifikationen für verschiedene Standards zu verabschieden, was allerdings der Weiterentwicklung des HTML-Standards nicht unbedingt zuträglich war und sie unnötig verzögerte. Als Reaktion darauf wurde von verschiedenen Browserherstellern die WHATWG, die Web Hypertext Application Technology Working Group gegründet, die fortan den HTML-Standard als *lebende Spezifikation* (Living Standard) betrachtete und HTML seitdem ohne konkrete Versionsnummer weiterentwickelt (siehe https://html.spec.whatwg.org/). Mittlerweile haben sich das W3C und die WHATWG allerdings bezüglich der Weiterentwicklung des Standards geeinigt und arbeiten nun gemeinsam an dem Living Standard.

Versionen sind im HTML-Dokument mittels *Doctype* zu definieren (z.B. `<!DOCTYPE html>` für HTML5).

Elemente und Attribute

Aufbau von HTML-Elementen inklusive Attribute:

- Element:

```
<a href="https://www.htl-hl.ac.at">
    Link zur Schulhomepage
</a>
```

- Öffnendes Tag (Start-Tag):

```
<a href="https://www.htl-hl.ac.at">
```

- Schließendes Tag (End-Tag):

```
</a>
```

- Attribut: `href="https://www.htl-hl.ac.at"`
- Attributname: `href`
- Attributwert: `"https://www.htl-hl.ac.at"`

Blockelemente und Inline-Elemente

- *Blockelemente* beginnen bei der Darstellung im Browser mit einer neuen Zeile und beanspruchen den gesamten horizontal zur Verfügung stehenden Platz. Beispiele hierfür sind die verschiedenen Überschriften (`<h1>` bis `<h6>`), Listen (`<ul>`) und Container (`<div>`).
- *Inline-Elemente* werden in derselben Zeile dargestellt und der benötigte horizontale Platz ist auf den Inhalt beschränkt. Beispiele hierfür sind Links, die verschiedenen Textauszeichnungen (`<b>`, `<em>`, `<strong>`) sowie Bilder (`<img>`).

Gruppierungen

Über das `<div>`-Element lassen sich Texte und Elemente als *Block* gruppieren, über das `<span›`-Element lassen sich Texte und Elemente inline gruppieren. Beide Elemente kommen vor allem dann zum Einsatz, wenn man eigene, individuelle UI-Komponenten (User-Interface-Komponenten) erstellen möchte, zu denen es keine Elemente im HTML-Standard gibt, wie beispielsweise Verzeichnisbäume etc.

Maskierungszeichen

Einige Zeichen wie etwa das `<`-Zeichen oder das `›`-Zeichen werden von der Sprache HTML selbst verwendet und können daher nicht ohne weiteres im Text einer Webseite verwendet werden. (Bei einem `<`-Zeichen denkt der Browser sonst beispielsweise, dass ein öffnendes Tag beginnt.) Solche Sonderzeichen müssen stattdessen maskiert werden und über sogenannte Maskierungscodes bzw. Escape-Codes gekennzeichnet werden. Statt `<` wird beispielsweise der Maskierungscode `&lt;` verwendet, und statt `>` `&gt;`.

Besondere Attribute

`id` und `class`-Attribute sind für die Arbeit mit CSS wichtig. Dadurch können Elemente eindeutig identifiziert werden beziehungsweise einer Klasse zugeordnet werden.

Kommentare

Oft kann es sinnvoll sein, den HTML-Code mit Kommentaren zu versehen, beispielsweise um anderen Entwicklern (oder sich selbst) Hinweise auf Besonderheiten im Code zu geben. Kommentare können in HTML über `<!--` eingeleitet werden und mit `--›` abgeschlossen werden: `<!-- Hier steht ein Kommentar --›`

Multimediadateien

Neben Bildern unterstützt HTML 5 auch andere Multimediadateien wie Audiodateien `<audio>` oder Videos `<video`.

Metadaten

Innerhalb des `<head›`-Elements können mittels `<meta›`-Elemente sogenannte Metadaten zu definieren. Beispielsweise können dies Schlüsselwörter zu dem Inhalt der Webseite oder Informationen zu dem Autor der Webseite sein, was insbesondere im Hinblick auf die Suchmaschinenoptimierung (Search Engine Optimization, kurz SEO) relevant ist.

Strukturelle Elemente

Eine einzelne Webseite besteht aus verschiedenen Bereichen und kann mit Hilfe von Elementen strukturiert werden. Seit HTML5 gibt es eine Reihe von zusätzlichen Elementen, mit denen die Struktur einer Webseite noch besser beschrieben werden kann. Beispiele hierfür sind die Elemente `<header>` (Kopfbereich einer

Webseite), `<footer>` (Fußbereich), `<nav>` (Navigation), `<article>` (einzelne Artikel innerhalb einer Webseite), `<aside>` (Randinformationen) und `<section>` (Gruppierung verwandter Inhalte).

Stylesheets und Javascript-Dateien

Ebenfalls innerhalb des `<head>`-Elements können unter Verwendung des `<link>`-Elements und des `<style>`-Elements CSS-Dateien eingebunden werden. Javascript-Dateien dagegen können über das Element `<script>` eingebunden werden.

Cascading Style Sheets - CSS

Nur HTML alleine macht eine Webseite visuell noch nicht sonderlich ansprechend. Für diesen Teil ist CSS zuständig. Hierüber wird mit Hilfe von sogenannten CSS-Regeln *gestaltet*, wie die einzelnen Komponenten, die zuvor in HTML definiert wurden, visuell dargestellt werden. Sie legen also über CSS das **Design und Layout** einer Webseite fest. Beispielsweise definieren Sie Textfarbe, Textgröße, Umrandungen, Hintergrundfarben, Farbverläufe etc.

CSS-Regeln

Die Art und Weise, wie der Inhalt bestimmter HTML-Elemente dargestellt werden soll, werden mit Hilfe von CSS-Regeln definiert. Diese Regeln bestehen prinzipiell aus zwei Teilen:

- Über den Selektor (CSS-Selektor) wird definiert, für welche HTML-Elemente die jeweilige CSS-Regel verwendet werden soll.
- Über die in geschweiften Klammern geschriebene Deklaration (CSS-Deklaration) wird festgelegt, wie genau diese HTML-Elemente dargestellt werden sollen. Deklarationen bestehen dabei ihrerseits wiederum aus einer Eigenschaft (CSS-Eigenschaft) und einem Wert (CSS-Wert), die beide durch einen Doppelpunkt voneinander getrennt sind und zusammen mit einem Semikolon enden. Eigenschaften können beispielsweise die Farbe, die Schriftart, die Abmessungen oder die Rahmenfarbe eines Elements betreffen. Über den Wert der jeweiligen Eigenschaft wird beispielsweise konkret festgelegt, welche Farbe, welche Schriftart, welche Breite oder Höhe oder welche Rahmenfarbe dargestellt werden soll. Jede Eigenschaft hat dabei bestimmte vordefinierte Werte, die gültig sind.
- CSS-Regel:

```
body {
    font-family: Arial;
}
```

- Selektor: `body`
- Deklaration: `font-family: Arial;`
- Eigenschaft: `font-family`
- Wert: `Arial`

CSS in HTML einbinden

Insgesamt gibt es drei verschiedene Möglichkeiten, CSS-Regeln zu definieren und in ein HTML-Dokument einzubinden.

- *Externes Stylesheet (external CSS):* Hierbei werden die CSS-Anweisungen als separate CSS-Datei (mit der Dateiendung css) gespeichert und mittels `<link rel="stylesheet" href="relative_path_to_file.css">` eingebunden. Diese Variante ist die sinnvollste von allen, da CSS-Code sauber von HTML-Code getrennt wird und er in mehreren HTML-Dokumenten eingebunden werden kann. Oft wird eine zentrale CSS-Datei verwendet, was bewirkt, dass sich bei einer Änderung das Aussehen aller Webseiten gleichzeitig ändert.
- *Internes Stylesheet (internal CSS):* In diesem Fall werden die CSS-Anweisungen im Kopfbereich des HTML-Dokuments innerhalb eines `<style>`-Elements definiert.
- *Inline Styles (inline CSS):* Hierbei werden die CSS-Deklarationen direkt an einem HTML-Element innerhalb des `style`-Attributs angegeben.

CSS-Selektoren

Um zu definieren, welche HTML-Elemente von einer CSS-Regel betroffen sind, stehen verschiedene Arten von *Selektoren* zur Verfügung.

Selektor	Beschreibung	Codebeispiel	Beschreibung des Codebeispiels
Typselektor	Wählt alle Elemente aus, die dem angegebenen Typ bzw. Elementnamen entsprechen.	`input: { color: green; }` `h1, h2 { color: green; }`	Selektor wählt alle `input`-Elemente aus. Selektor wählt alle `h1` und `h2` Elemente aus.

Klassenselektor	Wählt die Elemente anhand von CSS-Klassen aus, also Elemete, deren Wert des `class`-Attributs dem Wert hinter dem Punkt im Selektor entspricht.	`.valid { color: green; }`
		
		`a.valid { color: green; }`
			Der Selektor `.valid` wählt alle Elemente aus, deren `class`-Attribut den Wert `valid` hat.
			
			Der Selector `a.valid` wählt alle `<a>`-Elemente aus, deren `class`-Attribut den Wert `valid` hat.

JavaScript

HTML definiert die Struktur und CSS das Design und Layout. Beides sind - anders als mitunter fälschlicherweise zu lesen - keine Programmiersprachen und ermöglichen es daher auch nicht, Anwendungslogik innerhalb einer Webseite zu implementieren. Das ist genau der Punkt, an dem JavaScript ins Spiel kommt. Anders als die anderen beiden Sprachen ist JavaScript eine Programmiersprache und dient dazu, der Webseite (bzw. den Komponenten auf einer Webseite) dynamisches Verhalten hinzuzufügen und so

die **Interaktivität** auf der Webseite zu erhöhen.Beispiele hierfür sind die Sortierung und Filterung von Tabellendaten, dynamisches (clientseitiges) Generieren von Inhalten, clientseitige Validierung von Formulareingaben (wobei dies teilweise auch mit HTML möglich ist) und vieles mehr.