WEBT FS 2020

Victor Fernández

Februar 2020

Inhaltsverzeichnis

Teil I
SW01 - Einführung

1 Inhalt

- Klassifikation von Web-Applikationen
  - Statische Webseiten
  - Dynamische Webseiten
    - Serverseitige Erzeugung: Datenbank-orientiert
    - Serverseitige Erzeugung: mit Webservices
    - Clientseitige Erzeugung
- Formen mobiler Applikationen
- Grundlegende Adressierungskonzepte

1.1 Verstehen der verschiedenen Varianten von Web Applikationen

Statische Websites Einfache statische Webauftritte, z.B. Homepage
- Vorteile
  - Einfache Basistechnologien: HTML und CSS
  - Einfache Infrastruktur notwendig: Webserver
  - Automatische Erfassung durch Suchmaschinen
- Nachteile
  - Aktualisierungs- und Konsistenzprobleme
  - Keine Anwendungsfunktionalität

Dynamische Websites (Inhalt dynamisch generiert) Verschiedene Ansätze:
- Datenbankorientierte Web-Applikation
- Verwendung von Webservices
- Clientseitiges Erzeugen der dynamischen Inhalte

1.2 Verstehen von dynamischen Webseiten

Konzept Das Problem mit statischen Websites ist, dass man mit der Seite selber nicht interagieren kann. Das ziel von dynamischen Websites ist es, eine **Benutzer-System-Interaktion** über das Web zu ermöglichen. Gewünschte Interaktionen beinhalten (nicht abschliessend):
- Suche - Ergebnis
- Eingabe - Speicherung
- Bestellung - Lieferung
- Profil Angabe - Personalisierung

Datenbankorientiert
- Server erstellt ein **Formular** zur Verfügung, das vom **Web-Browser angezeigt** wird
- Benutzer füllt Formular aus und drückt auf Submit Button
- Browser entnimmt die vom Benutzer eingegebenen Daten aus den entsprechenden Feldern und schick sie an den Server zusammen mit dem Namen

1.3 Kennen der Arbeitsweise sowie der Vor- und Nachteile von nativen Applikationen, hybriden Applikationen und Web Applikationen

1.4 Kennen der grundlegenden Adressierungsmechanismen

Teil II
SW02 - HTML

2 HTML

- Das W3C (World-Wide-Web Consortium) ist für Standardisierung zuständig
- Aktuellster fertiger Standard ist Version 5.2 (Dez. 2017)
- Zudem wurden verschiedene Themen rund um HTML5 in eigenen Spezifikationen verabschiedet, bzw. sind in der Entwicklung (AAM, HTML Extension Specifications, ARIA,. . .)

2.1 Verstehen wie HTML Informationen strukturiert und wie der Aufbau von HTML Dokumenten ist

2.1.1 HTML tags

2.2 Wissen wie der syntaktische Aufbau von HTML ist

<head>
- Enthält Kopfdaten wie Metainformation, Titel, Stil, Scriptdefinitionen, Adress- und Zielfensterbasis
- Ist in jedem HTML Dokument zu finden
  - Metainformationen werden durch Metatags repräsentiert

```
<meta charset="utf-8">
<meta name="author" content="Hans Wurst">
```

  - Stildefinitionen (CSS - Cascaded Style Sheet) legen Darstellungen fest

```
<style>
    h1 { color: white; }
    p  { font-wight: bold; }
</style>
```

<body>
- HTML <body> kennzeichnet den Anfang und das Ende des sichbaren Inhalts der WEbseite
- Browser zeigen nur den Inhalt zwischen dem öffnenden und schlissenden body-Tag im Browserfenster
- Enthält weitere HTML Tags welsche die Information strukturieren, aber auch **Scripts**, welche an der aufgeführten Stelle ausgeführt werden
- Ein HTML-Dokument darf nur einen body-Tag haben

Text- und Informations-Strukturierung

- Absatz
- Zeilenumbruch
- Vorformatierung
- Überschriften
- Waagrechte Linien
- Container

```
<p>..</p>
<br />
<pre>..</pre>
<h1>..</h1> bis <h6>..</h6>
<hr />
<div>..</div> oder <span>..</span>
```

- Sind alles **Blockelemente**, das heisst, ein neuer Absatz (Zeilenumbruch) wird eingeleitet

Verfügungen (Hypertext-Referenzen)
- Link

```
<a href="pfad/datei">Linktext</a>
```

- Sowohl lokal als auch ins Internet möglich

```
<a href="/index.html">Home</a>
<a href="http://www.hslu.ch">Gehe zu HSLU</a>
```

- Mail-Links

```
<a href="mailto:hans@muster.ch">Mail schreiben</a>
```

  - Sollte vermieden werden, da Spambots diese automatisiert auslesen
- Interne Verknüpfung mittels Anker

```
<a href="#Kapitel1">Kapitel 1</a>
```

- als Ziel dieses Links

```
<p id="Kapitel1">Kapitel 1</p>
```

- Öffnen im neuen Fenster

```
<a href="adresse" target="_blank">Adresse</a>
```

Grafiken

- Grafikformate gif, jpg, png, ...
- Einbinden mit

```
<img src="pfad/bildname" alt="Beschreibung" />
```

- alt-Attribut verwenden spezielle Browser (Barrierefreiheit) oder Suchmaschinen (z.B. Google Bildersuche), unbedingt angeben
- Anzeigegrösse veränderbar mit Attributen `width` und `height`
- Rahmen: border="1px"
- Als Hintergrund der Seite

```
<body background="bildname">
```

Klickbare Grafiken: Imagemaps

1. Definition des Bildes

```
<map name="karte">
    <area shape="circle" coords="50,50,45" href="Ziel.html" alt="Reiseziel" />
</map>
```

   - Wird häufig zu Navigationszwecken verwendet

Beispiel Imagemap-Code

```
<body>
    ...
    <img src="transmap.gif" alt="list of stations" usemap="#transmap">
    <map name="transmap">
        <area shape="rect" coords="59,390,172,441" href="#ACACIA" />
        <area shape="rect" coords="280,21,390,63" href="#ALMOND" />
        <area shape="rect" coords="135,52,243,124" href="#APPLE" />
        <area shape="rect" coords="141,235,189,284" href="#ASH" />
        <area shape="rect" coords="110,336,205,388" href="#BEECH" />
        <area shape="rect" coords="152,289,236,334" href="#BIRCH" />
        <area shape="rect" coords="330,231,402,288" href="#CHERRY" />
        ...
    </map>
    ...
    <a name="ACACIA">Acacia</a><img src="red.gif" alt="Red Line" />
    ...
    <a name="ALMOND">Almond</a><img src="yellow.gif" alt="Yellow Line" />
    ...
</body>
```

Metatags

- Werden nicht zur Gestaltung sondern **zur Beschreibung des Inhalts** verwendet (daher der Name: Meta-Information)
- Werden im `head` Bereich eingefügt
- Aufbau:

```
<meta name="Schlüsselwort" content="Inhalt">
```

- Die wichtigsten Metatags

- keywords
- description
- language
- page-topic (Thema für Suchmaschinen und Kataloge)
- audience (Zielgruppe in Suchmaschinen und Katalogen)
- robots (zur Linkverfolgung)
- refresh (und expires →Ablaufdatum)
- copyright

Sonderzeichen

- Damit Sonderzeichen korrekt dargestellt werden, muss das charset metatag korrekt gesetzt sein

```
<meta charset="utf8">
<meta charset="iso-8859-1">
```

- Eine Alternative ist die Zeichen speziell zu kodieren:
  Beispiel: aus **ü** wird **ü**
- Dies ist auch notwendig bei Zeichen, welche mit dem HTML-Markup kollidieren:
  & zu &, < zu <, > zu >,

Masseinheiten

- Verwendet in Attributen zur Bestimmung der Dimensionen verschiederer Elemente wie Bilder, Schriften, Ränder, Abstände
- Die wichtigsten Einheiten sind:
  - pt: Punkt, **absolute** Angabe, 1 Punkt entspricht 1/72 Inches
  - in: Inch, **absolute** Angabe, 1 Inch enspricht 2.54cm
  - mm: Millimeter, **absolute** Angabe
  - px: Pixel, **absolute/relative** Angabe Abhängig von der **Pixeldichte** des Ausgabegeräts
  - em: M, **relative** Angabe, auf die Schriftgrösse des Elements bezogen (mit Ausnahmen)
  - %: Prozent, **relative** Angabe, je nach CSS-Eigenschaft relativ zur elementeigenen Grösse, oder zu der des Elternelements, oder zu einem allgemeineren Kontext

Farben

- Farben werden aus den RGB-Wertangaben gebildet
- Beispiel: #FF0000 ist rot, #00FF00 ist grün, #0000FF ist blau
- Werte werden in hexadezimaler Form angegeben
- Werte von #00 bis #FF (255) sind möglich
- Einige Farben sind per Namen in der DTD (Document-Type-Definition) definiert:

Black	= #00000	Green	= #008000
Silver	= #C0C0C0	Lime	= #00FF00
Gray	= #808080	Olive	= #808000
White	= #FFFFFF	Yellow	= #FFFF00
Maroon	= #800000	Navy	= #000080
Red	= #FF0000	Blue	= #0000FF
Purple	= #800080	Teal	= #008080
Fuchsia	= #FF00FF	Aqua	= #00FFFF

2.2.1 Struktur von Webseiten - Elemente

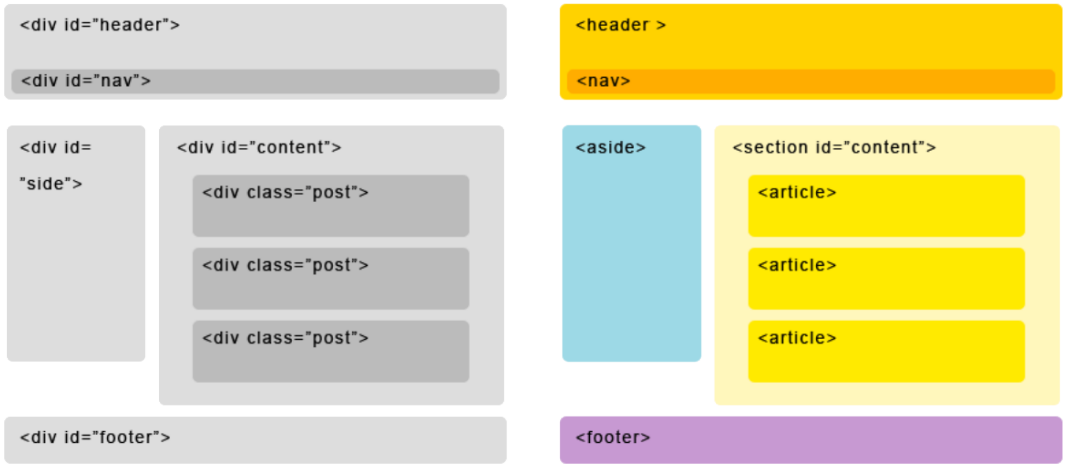

Abbildung 1: Strukturbeispiel von Webseiten

<header>
- enthält sichtbaren Kopfbereich einer Webseite
- Gruppierung einleitender Inhalte (Firmenlogos, Motto, Navigationslinks)

<footer>
- enthält Informationen, die in Webseiten am Ende stehen: Autor, Hinweise zum Urheberrecht, ein Link zum Impressum
- Position ist nicht notwendigerweise am unteren Rand
  →bei Blogeinträgen steht der footer oft neben dem Text

<article>
- stellt in sich geschlossene Abschnitte eines Dokuments dar
  →vergleichbar mit einem Zeitungsartikel
  →innterhalb von article-Elementen weitere strukturierende Elemente wie header, section oder footer

<section>
- enthält eine thematische Gruppierung von Inhalten typischerweise mit einer Überschrift
- dient dazu, den Inhalt oder auch einen article in semantische Abschnitte zu gliedern

<nav>
- umschliesst insbesondere Navigationsleisten
- kann neben einer ungeordneten Liste mit den Verweisen auch eine Überschrift oder ähnliches enthalten

<aside>
- umschliesst Abschnitte einer Seite, deren Inhalt nur in einem indirekten Zusammenhang mit dem umgebenden Inhalt stehen
- Beispiele: Randbemerkungen, Fussnoten oder Links zu weitergehenden Webseiten

2.3 Kennen von geeigneten Werkzeugen für das Erstellen, Bearbeiten, Darstellen, Validieren etc. von HTML Dokumenten

2.4 Wissen um geeignete Quellen und Referenzen im Internet

Teil III
SW04 - CSS

3 Inhalt

3.1 Wissen, wie die Kaskadenkette zur schlussendlichen Darstellungsform führt

Kaskadierung Falls mehrere, sich eventuell wiedersprechnde Styles definiert sind, muss ein Regelwerk zur Anwendung kommen um zu einem Ergebnis zu gelangen. Grundprinzip dieses Regelwerkes: →Prioritätsreihenfolge:

- Gewichtung (Schlüsselwort `!important`)
- Herkunft (Web-Author, Benutzer, Browser)
- Besonderheit (je spezifischer desto mehr Gewicht)
- Ergebnisabfolge (Reihenfolge in der die Definitionen festgelegt wurden)

Prioritätenreihenfolge nach Herkunft

1. Falls Deklarationnen des Browser
2. Deklarationen des Benutzers[1]
3. Interne/Externe CSS-Anweisungen des Web-Authors
4. Inline CSS-Anweisungen des Web-Authors
5. Deklarationen des Web-Authors die `!important` enthalten
6. Deklarationen des Benutzers[1] die `!important` enthalten

- Der Vorrang der **!important-Benutzer**- gegenüber den **!important-Autoren**- Styles wurde mit dem CSS2-Standard neu festgelegt.

Unterschiedliche Ausgabemedien

- Es ist möglich für verschiedene Ausgabemedien Stylesheets zu definieren
- Dadurch können die verschiedenen Charakteristiken der Ausgabemedien (Drucker, Screen, Sprachausgabe, etc.) besser berücksichtigt werden
- Das Attribut `media` definiert, für welches Ausgabemedium der entsprechende Style definiert ist

Unterschiedliche Ausgabemedien

- Beispiel, Ausgabe für Bildschrim oder Drucker (hier im HTML-Code)

```
<head>
    <title>Seitentitel</title>
    <link href="standard.css" rel="stylesheet" media="screen" title="Standard-Layout">
    <link href="druck.css" rel="stylesheet" media="print" title="Druckoptimiertes Layout">
    <link href="aural.css" rel="stylesheet" media="speech" title="Sprachausgabe">
</head>
```

Unterschiedliche Ausgabemedien

- standard.css
  ist das Standard-Stylesheet für die Bildschrimanzeige
- print.css
  ist das Standard-Stylesheet für den Ausdruck
- aural.css
  ist für zukünftige Definitionen bezüglich der Sprachausgabe vorgesehen. Z.B. Sprechgeschwindigkeit, männliche oder weibliche Stimme, etc.

Schriftformatierung: Grundsätzliches

- definiert Schriftart bzw. -typ
  `font-family`
- Beispiel:
  `font-family:"Times New Roman";`

[1] Benutzer⇒Webseite Besucher→also seine Browsereinstellungen

- Es kann sein, dass diese Schriftart auf dem Rechner des Benutzers nicht vorhanden ist, deshalb lassen sich Alternativen angeben:
  `font-family:"Times New Roman", Garamond, serif;`
- Möchte man nur den Schrifttyp angeben, sucht das System des Benutzers die passende Schriftart dazu aus:
  `font-family:sans-serif;`

Schriftformatierung: Schriftgrösse und -neigung
- Fast jede Eigenschaft der Schriftart lassen sich steuern, sei es Grösse, Neigung, Dicke, etc.
- `font-size` legt die Grösse fest, z.B. absolut: `font-size:20px;` oder relativ: `font-size:small;`
- Die relativen Angaben sind abhängig vom OS, Browser oder anderen Einstellungen
- `font-style` steuert die Neigung, `font-variant` die Varianten der Schrift: `font-style:italic;` ergibt eine kursive Darstellung, `font-variant:small-caps;` zeigt Kapitälchen[2]

Schriftformatierung: Schriftdicke und -farbe
- `font-weight` legt die Schriftdicke fest. Es sind sowohl absolute als auch relative Angaben möglich: `font-weight:bold;` oder `font-weight:600`
- **Achtung:** nicht jede Schriftart unterstützt diese Angaben!
- Die Eigenschaft `color` verändert die Farbe. es können Farbworte oder Tripelwerte verwendet werden:
  - `color:black;`
  - `color:#88AAFF;`
  - `color:rgb(23%,50%,95%);`

Schriftformatierung: Textdekoration
- CSS versteht u.A. folgende Arten der Textdekoration: Über-, Durch- oder Unterstrichen
- Die Eigenschaft `text-decoration` steuert die Darstellung:
  - `text-decoration:underline;`
  - `text-decoration:overline;`
  - `text-decoration:line-through;`
- Weitere Möglichkeiten:
  - `text-decoration:blink;` (zu vermeiden!)
  - `text-decoration:none;`

Schriftformatierung: Texttransformation
- Die Eigenschaft `text-transform` verändert die Gross- und Kleinschreibung eines Textes:
  - `text-transform:uppercase;`
  - `text-transform:lowercase;`
  - `text-transform:normal;`
  - `text-transform:capitalize`
- `capitalize` stellt alle Wortanfänge mit Grossbuchstaben dar

Schriftformatierung: Kurzform der Schriftformatierungen
- Um effizienter die Schriftformatierung festzulegen, können die Eigenschaften `font-family, font-size, font-variant` und `font-weight` zusammengefasst in der `font` Eigenschaft angegeben werden
- Beispiele:
  - `font:italic 14px Arial;`
  - `font:lighter 12pt monospace;`

Attribut-bedingte Formatierung
- CSS-Formatierungen lassen sich auf Elemente mit bestimmten Attributen begrenzen
- Beispiele:
  - legt Farbe für alle h1 Überschriften fest, welche ein Alignment haben
    `{h1[align] {color:blue}`
  - stellt alle Absätze, die ein Namenattribut mit Inhalt „Text“ haben, mit Kapitälchen dar
    `p[name*="Text"] {font-variant:small-caps;}`
  - weist allen zentriert ausgerichteten HTML-Elementen eine Farbe zu
    `*[align=center] {color:red;}`

[2]Kapitälchen: die Kleinbuchstaben werden als grosse Buchstaben dargestellt, doch in der höhe von Kleinbuchstaben

Individuelle CSS-Formate

- Um Elementen individuelle Formate zu geben, verwendet man die individuelle Formatierung
- Jedes Element kann ein **eindutiges** Attribut **id** besitzen (Element-ID)
  - Element-IDs kann man auch für JavaScript brauchen!
- Beispiel:
  `#eins {color:blue;}`
  `<p id="eins">Absatz</p>`

Hintergrundbilder

- Hintergrundbilder einer Seite lassen sich folgendermassen festlegen:
  `body {background-image:url(bild.jpg);}`
- Man kann aber auch nur für einen Abschnitt ein Hintergrundbild festlegen:
  `p {background-image:url(bild.jpg);}`
- Je nach Grösse kann das Bild mehrfach wiederholt werden, sowohl horizontal als auch vertikal:

```
table {
    background-image:url(bild.jpg);
    background-repeat:repeat-x;
    background-repeat:repeat-y;
}
```

Zentrieren von Containern

- Weil das `align` Attribut nur für Textabschnitte gilt, kann es **nicht** zum zentrieren von Containern bzw. neuen HTML5 Block-Elementen verwendet werden
- Man verwendet dazu die CSS Eigenschaft `margin`
- Beispiel: `<div style="margin:auto">Containerinhalt</div>`

Fliesstext - float

- Mittels der `float` Eigenschaft kann man Text um Bereiche herum fliessen lassen (oder verbieten)
- Beispiel:

```
<p>
    <span style="color:red; float:left; font-size:2.5em; padding-right:2px;">D</span>as
        ist eine umfliessende Initiale, in der nur der erste Buchstabe eines Textes gross
        geschrieben wird
</p>
```

Mehrspaltiges Layout: Definition

`<p class="blub">hallo</p>`