Sprachen und Technologien

HTML

- **H**yper**t**ext **M**arkup **L**anguage
- Ab 1989 durch Tim Berners-Lee
  - Später durch W3C und WHATWG
- **Auszeichnungssprache** zur Strukturierung von Webinhalten
  - Enthält nur Struktur und Inhalt
  - Inhalte sind (mittlerweile) statisch
  - Formatierung (mittlerweile) durch CSS
- Aktuell in Version 5.2
- Syntaktisch ähnlich zu XML (SGML)
  - Basiert auf Tags und Attributen
- Laufzeitumgebung: **Browser**

HTML - Syntax

- Besteht aus drei Teilen
  - Doctype
    - Zur Typdefinition
  - Head
    - Metainformation
    - Ressourcenreferenzierung
  - Body
    - Inhalte

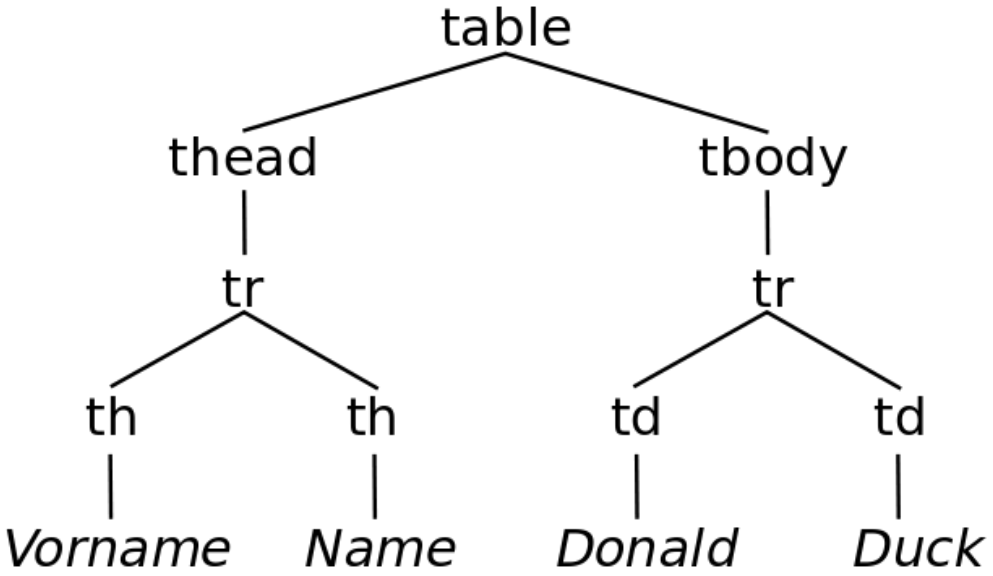
```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel der Webseite</title>
    <!-- weitere Kopfinformationen -->
    <!-- Kommentare werden im Browser
  </head>
  <body>
    <p>Inhalt der Webseite</p>
  </body>
</html>
```

HTML - DOM

- **D**ocument **O**bject **M**odel
- Programmierschnittstelle zum einheitlichen Zugriff auf HTML Elemente
  - Zu Manipulation durch Javascript
  - Auch in XML genutzt
- Stellt Struktur als abstrakten Baum bereit
- Erleichtert Adressierung von Knoten über relative Beziehungen
  - Parent
  - Child
  - Sibling

HTML - DOM

```
<table>
  <thead>
    <tr>
      <th>Vorname</th>
      <th>Name</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Donald</td>
      <td>Duck</td>
    </tr>
  </tbody>
</table>
```



CSS

- **C**ascading **S**tyle **S**heets (1995)
- Erlaubt separate Formatierung von HTML und XML
  - Trennung von Inhalt und Formatierung möglich
  - Unterschiedliche Formatierung abhängig von Anforderung
    - Über sog. **Media Queries**
- Referenziert Elemente (auch) über DOM
  - Sehr ausgefeilte Selektoren möglich
- Anweisungen legen Eigenschaften von Elementen fest
  - Werden anhand von Position im DOM überschrieben oder geerbt
    - Der mehr spezifische Selektor gewinnt
- Laufzeitumgebung: **Browser**

CSS - Syntax

- Selektor (-Liste)
- Anweisung
  - Beinhalten Eigenschaften

```
Selektor1 [, Selektor2 [, …] ] {
    Eigenschaft-1: Wert-1;
    …
    Eigenschaft-n: Wert-n[;]
}
/* Kommentar */
/* In eckigen Klammern stehen optionale Angaben */
```

CSS - Beispiel

```css
p.info {
    font-family: arial, sans-serif;
    line-height: 150%;
    margin-left: 2em;
    padding: 1em;
    border: 3px solid red;
    background-color: #f89;
    display: inline-block;
}
p.info span {
    font-weight: bold;
}
p.info span::after {
    content: ": ";
}
```

```html
<p class="info">
    <span>Hinweis</span>
    Sie haben sich erfolgreich angemeldet.
</p>
```

Hinweis: Sie haben sich erfolgreich angemeldet.

CSS - Vererbung im DOM

```html
<table>
  <thead>
    <tr>
      <th>Vorname</th>
      <th>Name</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Donald</td>
      <td>Duck</td>
    </tr>
  </tbody>
</table>
```

```css
table  {
    color: red;
}
th  {
    color:blue ;
}
```

CSS - Preprocessor

- CSS kann unübersichtlich werden
  - Oft redundanter Code
  - Unnatürlich flacher Aufbau
- Preprocessor erlauben “höhere” Sprachkonstrukte wie Variablen
- Haben eigene Sprache und Syntax
- Wird vor dem Deployment meist zu CSS kompiliert
- Bekannte Vertreter
  - Sass (**S**yntactically **A**wesome **S**tyle **S**heets) 2006
    - SCSS (**S**assy **CSS**)
  - Less (**Le**aner **S**tyle **S**heets) 2006

CSS - Preprocessor

- Syntaktische Erweiterungs von CSS
- Bieten
  - Variablen (CSS3 mittlerweile auch)
  - Funktionen (inklusive Argumente)
  - Vererbung
  - Mixins
  - Verbessertes Nesting

```
@mixin table-base {
  th {
    text-align: center;
    font-weight: bold;
  }
  td, th {
    padding: 2px;
  }
}

#data {
  @include table-base;
}
```

PHP

- **P**HP: **H**ypertext **P**reprocessor (1995)
  - Ehemals **P**ersonal **H**ome **P**age
- Stark auf Webentwicklung optimierte Skriptsprache
- In C geschrieben
  - Teilweise
- Durch Community weiterentwickelt
  - Immer frei und kostenlos
  - Langsame Entwicklung bis 2014
- Immer noch schlechter Ruf in akademischen Kreisen
  - Zunehmende Professionalisierung seit 2014
  - “Arschtritt” von Javascript und Facebook

PHP - Für das Web optimiert

- Integrierte Funktionen zur HTTP Header Manipulation
- Erlaubt Kontrolle von Sockets
- Breite CGI Unterstützung
  - Ausgabe 1:1 als HTTP Response an Client
- Lässt sich auch in HTML integrieren

```
<!DOCTYPE html>
<html>
 <head>
  <title>PHP "Hello, World!" program</title>
 </head>
 <body>
  <?php echo '<p>Hello, World!</p>'; ?>
 </body>
</html>
```

PHP - Für das Web optimiert

- Integriertes Session Handling über $_SESSION und $_COOKIE
  - Serverseitige Möglichkeit zu stateful statt stateless
- Integration der Webserver SAPI über $_SERVER
  - Zugriff auf alle Umgebungs- und Laufzeitvariablen
- Integration von Requesthandling des Servers über $_REQUEST
- Integrierter eigener Webserver (für Entwicklungszwecke)
- (Pseudo-) Standards rund um Request-/Response-Handling
  - PSR-7, PSR-13, PSR-15, PSR-17, PSR-18 (5 von 13)

PHP - Syntax

- Vergangenheit als Skriptsprache kaum noch aktiv genutzt
  - Fast ausschließlich objektorientiert nutzbar

```php
class Student extends User
{
    public string $course;

    public function __construct(string $name, string $course)
    {
        $this->course = $course;
        parent::__construct($name);
    }

    public function job(): string
    {
        return "I learn " . $this->course;
    }
}
```

PHP - Besonderheiten

- Durch Möglichkeit zur HTML Einbettung **nötig: Start Tag**
  - IMMER nötig
  - Beendender Tag nur als Abgrenzung zu folgendem Inhalt
  - Webserver sendet **alles** an Client

- Woher kennt der Interpreter die Definitionen unserer Klassen?
  - Import ist eine Möglichkeit
  - Autoloading eine andere
- Über PSR-4 Semantik in Klassennamen und Speicherort
  - Spezieller PHP Code löst Dateien anhand Klassennamen auf
  - Kann selbst implementiert werden!
  - **Composer**

```php
<?php

require '../path/to/needed/class.php';

$neededInstance = new NeededClass();
```

PHP - Besonderheiten

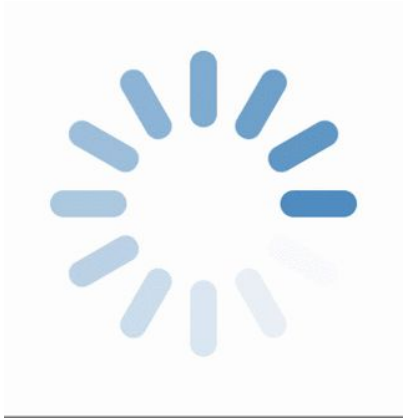
- Magische Methoden/Konstanten erlauben besondere Dynamik
  - Z.B. Aushebeln von OOP Sichtbarkeit (__get(), __set() und __call())
- Kann über C-Extensions beliebig erweitert werden
  - Sogar in C geschriebenes CMS möglich (Phalcon)
- Shared Nothing Prinzip
  - **Nichts** wird Request-übergreifend aufbewahrt
  - Keine Angst vor Inter-Request Problemen

Javascript

- Seit 1995
- Bekannteste ECMAScript Implementierung
- Dynamisch typisierte Skriptsprache
  - Oft Event-Getrieben
  - Empfehlung: FWPM “Javascript” von Sebastian Springer
- Multi-Purpose, aber ursprünglich Browser als Laufzeitumgebung
  - Dynamisches Verhalten über DOM Manipulation
- Mittlerweile omnipotent
  - Serverseitig z.B. mit Node.js
  - Embedded Scripting Sprache in CI Systemen
  - Als Basis für Standalone Anwendungen z.B. mit Electron

Javascript - Klassisch im Web

- Dynamische Interaktion und Verhalten von Webseiten als Hauptzweck
- Selektives Laden von Inhalten über **AJAX**
  - Weltbewegende Änderung für Webanwendungen
- Führen eines Client-States z.B. lokale Datenspeicherung
  - Stateful und Datenhoheit beim Client
- Verarbeitung einfacher Aufgaben, z.B. Form Validation
  - Als Usability Feature
- Animationen und dynamische Darstellung
  - Beeinflussen von CSS Klassen
- Vielzahl von Hilfsbibliotheken, z.B. jQuery
- Grundlage für **JSON**



Javascript - Syntax

- Sowohl in Scriptform als auch objektorientiert gebräuchlich

```
function LCMCalculator(x, y) { // constructor function
    let checkInt = function(x) { // inner function
        if (x % 1 !== 0)
            throw new TypeError(x + "is not an integer"); // var a =  mouseX

        return x;
    };

    this.a = checkInt(x)
    //   semicolons   ^^^^  are optional, a newline is enough
    this.b = checkInt(y);
}
```

Javascript - SPA und PWA

- **S**ingle **P**age **A**pplication bzw. **P**rogressive **W**eb**a**pp
- Moderne JS Ansätze als Framework für komplette Client Anwendung
  - Kein serverseitiges Rendering mehr
  - Inhalte in Form von Daten z.B. per Webservice Call
- Lösen Request/Response Modell scheinbar auf
  - Wirken Stateful, Nötige Requests im Hintergrund über AJAX
- Erlauben Offline Einsatz und Integration in Host APIs
- Große Auswahl an aktuellen Frameworks
  - React, Angular, Vue.js, Ember, ...

Bildquellen:

- DOM Beispielbaum Von Molily, SVG: Marlus_Gancher - drawn based on Datei:Dom-beispielbaum.png (bitmap version), Gemeinfrei, https://commons.wikimedia.org/w/index.php?curid=15766540
- CSS 3 Logo Von Rudloff - File:CSS3_and_HTML5_badges.svg, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=49121103
- Sass Logo By http://sass-lang.com/ - http://sass-lang.com/styleguide/brand/, Public Domain, https://commons.wikimedia.org/w/index.php?curid=38949377
- Composer Logo By WizardCat - getcomposer.org via Wikimedia tech blog, MIT, https://commons.wikimedia.org/w/index.php?curid=38131432
- Spinner gif https://gifimage.net/spinner-gif-13/