



# Webseiten-Entwicklung mit HTML, CSS und JavaScript

Autoren: Luca Rief, Daniel Appenmaier

Version: 1.3

PUBLIC

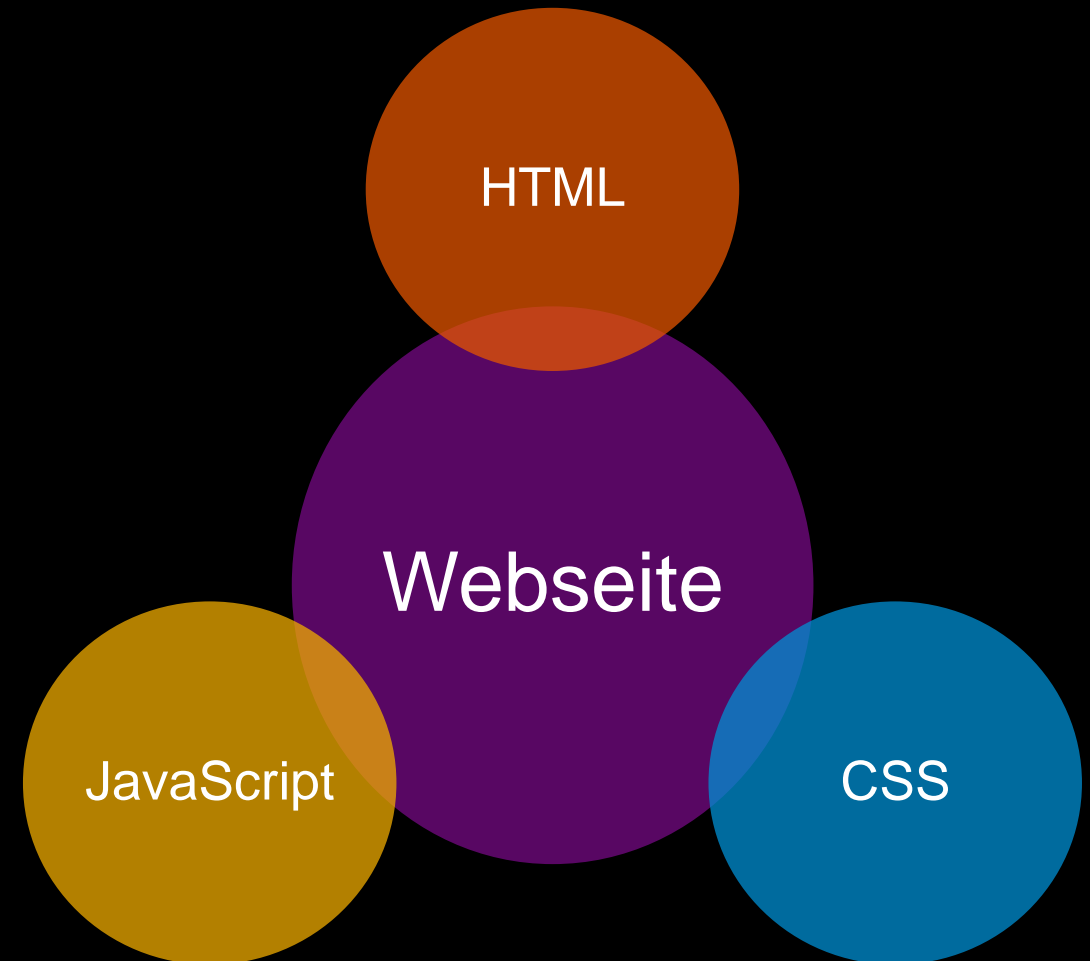
# Agenda

- Einführung
- HTML: Aufbau der Webseite
- CSS: Aussehen der Webseite
- JavaScript: Verhalten der Webseite

**Einführung**

# Webseiten

- Webseiten bestehen aus Quellcode, der von einem Browser interpretiert wird
- Der Quellcode stellt sozusagen den Bauplan für die Webseite dar
- Zur Erstellung des Quellcodes und damit der Webseite werden 3 Sprachen benötigt
  - **HTML** strukturiert den Inhalt
  - **CSS** gestaltet die HTML-Elemente
  - **JavaScript** steuert das Verhalten



# Tools zur Entwicklung von Webseiten

- Zur Entwicklung von Webseiten benötigt man generell nur einen Texteditor (z.B. Notepad, Wordpad etc.) sowie einen Browser (z.B. Google Chrome, Mozilla Firefox etc.)
- Der Nachteil einfacher Texteditoren besteht darin, dass diese dem Entwickler keinerlei Unterstützung bei der Entwicklung geben
- Professionelle Texteditoren wie *Notepad++* oder Quelltext-Editoren wie *Visual Studio Code* bieten eine Reihe an Funktionen (z.B. Syntax-Highlighting, Code-Vervollständigung etc.), die das Entwickeln erleichtern
- Die meisten Browser bieten standardmäßig die Möglichkeit, über **F12** die Entwicklertools aufzurufen, die das Analysen von Webseiten ermöglichen sowie das Aufspüren von Fehlern erleichtern

# HTML: Aufbau der Webseite

# HTML

- HTML steht für **H**ypertext **M**arkup **L**anguage
- HTML ist keine Programmiersprache sondern eine sogenannte **Auszeichnungssprache** und beschreibt den Aufbau von Webseiten
- Die erste Spezifikation zu HTML wurde 1992 veröffentlicht
- HTML ist eine der Kernsprachen des *World Wide Web*



# HTML-Tags

- **Tags** sind Anweisungen in spitzen Klammern, auch HTML-Markup genannt. Sie legen eine Struktur über Text und Bilder der Webseite und versorgen die Browser mit Informationen über den Aufbau der Seite
- Tags können beliebig ineinander verschachtelt werden
- Man unterscheidet zwischen öffnenden Tags (**<Tag>**) und schließenden Tags (**</Tag>**)

```
<TagA>  
  <TagB>Inhalt 1</TagB>  
  <TagC>Inhalt 2</TagC>  
</TagA>
```

# Attribute

- **Attribute** bringen zusätzliche Informationen in ein Tag (z.B. den Pfad auf eine Bilddatei)
- Attribute und Attributs-Ausprägungen werden innerhalb des öffnenden Tags festgelegt

```
<TagA attributA="Wert 1" attributB="Wert 2">  
  <TagB attributC="Wert 3">Inhalt 1</TagB>  
  <TagC>Inhalt 2</TagC>  
</TagA>
```

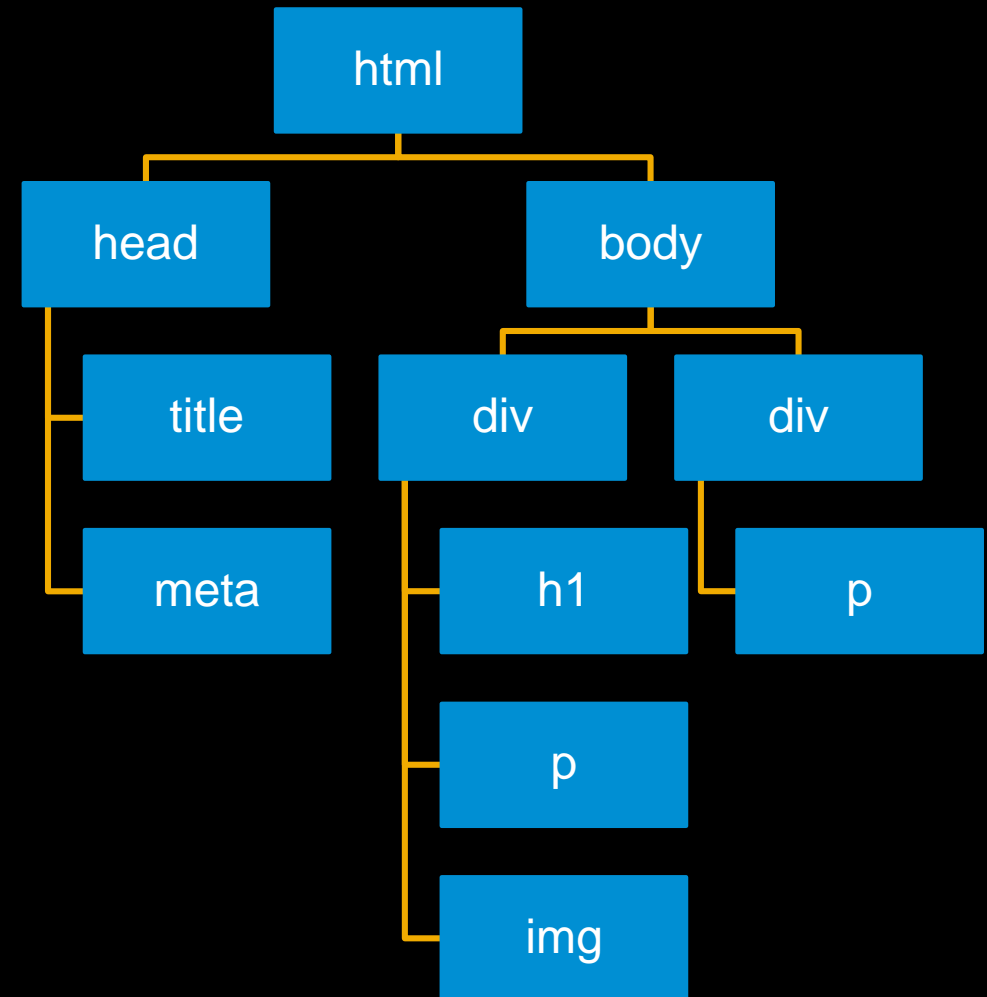
# Grundgerüst eines HTML-Dokuments

- Das Grundgerüst eines HTML-Dokuments besteht typischerweise aus einem Kopfbereich (Tag **<head>**) sowie dem Inhaltsbereich (Tag **<body>**)
- **Hinweis:** Kommentare (Tag **<!-->**) werden vom Browser ignoriert

```
<!DOCTYPE html>
<html>                                <!--Beginn des HTML-Dokuments-->
  <head>                              <!--Beginn des Kopfbereichs-->
    <title>Titel</title>             <!--Titel der Webseite-->
  </head>                             <!--Ende des Kopfbereichs-->
  <body>                              <!--Beginn des Inhaltsbereichs-->
    Inhalt                           <!--Inhalt der Webseite-->
  </body>                             <!--Ende des Inhaltsbereichs-->
</html>                              <!--Ende des HTML-Dokuments-->
```

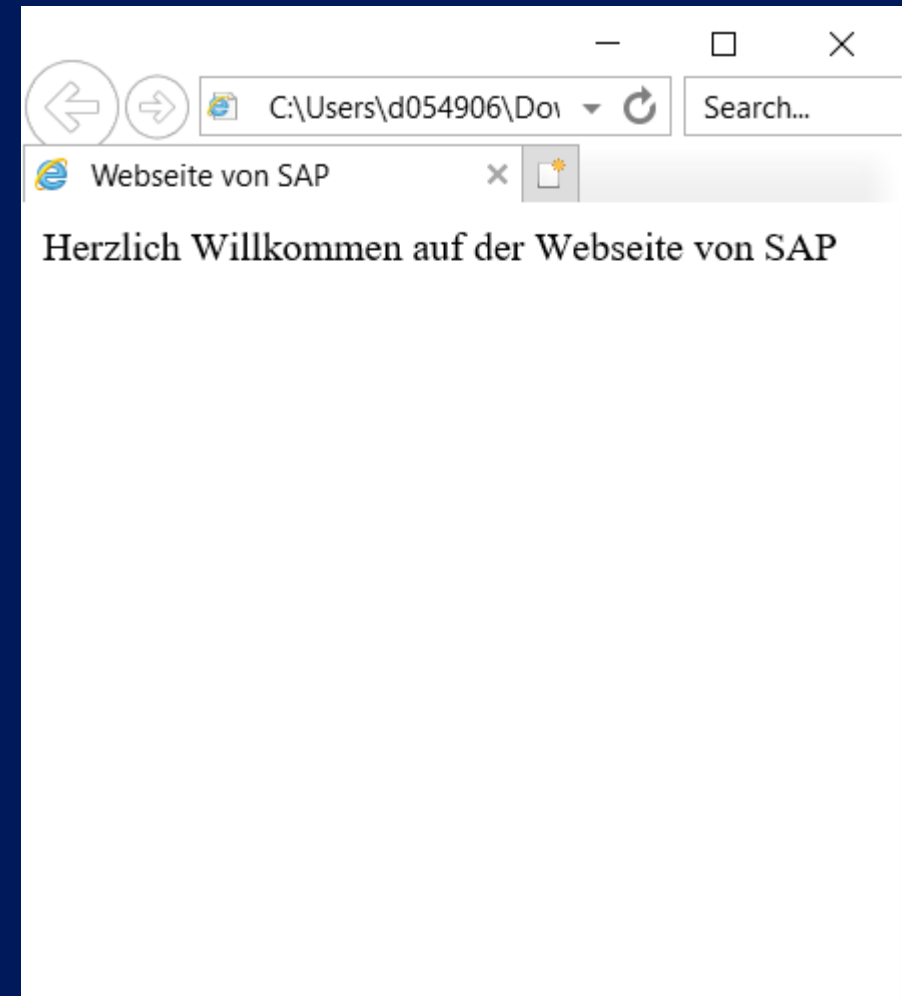
# Document Object Model

- Die Inhalte eines HTML-Dokuments werden vom Browser Schritt für Schritt in eine Objektstruktur überführt
- Diese Objektstruktur wird als DOM (**Document Object Model**) bezeichnet und ermöglicht den Zugriff auf einzelne Elemente
- **Hinweis:** die Identifizierung eines HTML-Elements erfolgt über das Attribut *id*
- **Hinweis:** mit dem Attribut *class* können verschiedene HTML-Elemente zu einer gemeinsamen Klasse zusammengefasst werden



# Aufgabe 1

- Erstelle ein HTML-Dokument (*index.html*) mit dem Titel: „Webseite von *Name*“, welches den Text „Herzlich Willkommen auf der Webseite von *Name*“ anzeigt



# Inhaltsbereich

- Im Inhaltsbereich (Tag **<body>**) werden sämtliche Inhalte der Webseite (Texte, Aufzählungen, Grafiken, Verweise) festgelegt

Attribut	Beschreibung	Beispiel
<b>text</b>	Textfarbe	"white"
<b>link</b>	Farbe eines Verweises	"red"
<b>vlink</b>	Farbe eines besuchten Verweises	"blue"
<b>topmargin</b>	Abstand nach oben in Pixel	50
<b>leftmargin</b>	Abstand nach links in Pixel	50
<b>bgcolor</b>	Hintergrundfarbe	"black"
<b>background</b>	Hintergrundbild	"sap.png"

# Textformatierungen

- Texte können in HTML über vordefinierte Tags angepasst werden
- **Hinweis:** Umlaute und Sonderzeichen müssen in HTML über spezielle Codes abgebildet werden

```
<h1>größligte &Uuml;berschrift</h1>  
<h6>kleinste &Uuml;berschrift</h6>  
<b>fetter Text</b>  
<i>kursiver Text</i>  
<u>unterstrichener Text</u>  
<s>durchgestrichener Text</s>  
<center>zentrierter Text</center>  
<sup>hochgestellter Text</sup>  
<sub>tiefgestellter Text</sub>  
<marquee>Lauftext</marquee>
```

# Trennlinien

- Der Tag `<hr>` erzeugt eine horizontale Trennlinie (**horizontal row**)
- **Hinweis:** beim Attribut **align** sind die Werte **left** (linksbündig), **right** (rechtsbündig), **center** (zentriert) und **justify** (Blocksatz) möglich

Attribut	Beschreibung	Beispiel
<b>width</b>	Breite in Pixel	<b>500</b>
<b>size</b>	Höhe in Pixel	<b>5</b>
<b>align</b>	Ausrichtung	<b>"center"</b>
<b>color</b>	Farbe	<b>"white"</b>
<b>noshade</b>	Schattierung deaktivieren	

# Zeilenumbrüche und Absätze

- Zeilenumbrüche (**breaks**) werden in HTML mit Hilfe des Tags **<br>**, Absätze (**paragraphs**) mit Hilfe des Tags **<p>** umgesetzt

Attribut	Beschreibung	Beispiel
<b>align</b>	Ausrichtung	<b>"center"</b>

# Aufzählungen

- HTML unterscheidet zwischen geordneten (Tag `<ol>`) und ungeordneten (Tag `<ul>`) Aufzählungen (*lists*)
- Einzelne Einträge werden mit dem Tag `<li>` geöffnet und mit dem Tag `</li>` geschlossen

```
<!--ungeordnete Aufzählung-->
```

```
<ul>
```

```
  <li>Eintrag 1</li>
```

```
  <li>Eintrag 2</li>
```

```
  <li>Eintrag 3</li>
```

```
</ul>
```

```
<!--geordnete Aufzählung-->
```

```
<ol>
```

```
  <li>Eintrag 1</li>
```

```
  <li>Eintrag 2</li>
```

```
  <li>Eintrag 3</li>
```

```
</ol>
```

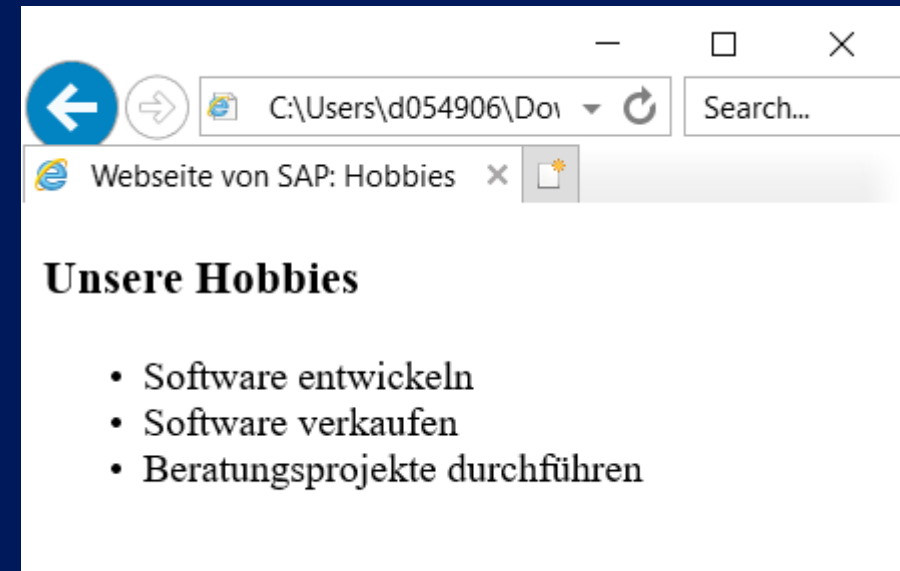
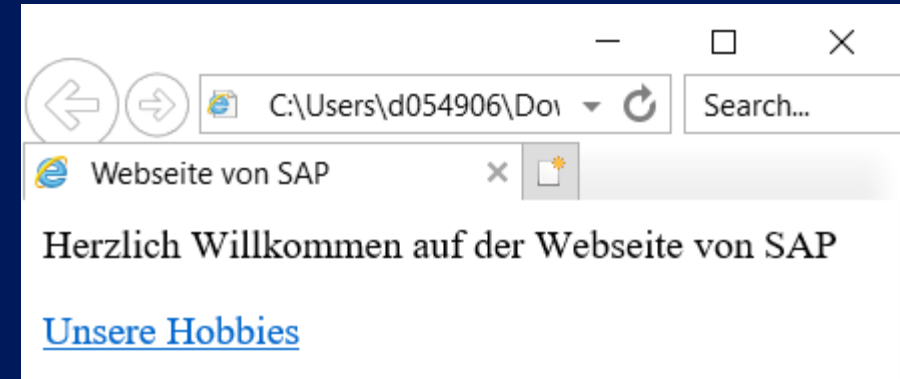
# Verweise

- Verweise (**anchors**, Tag `<a>`) ermöglichen die Navigation sowohl innerhalb der Webseite, als auch zu externen Dokumenten
- **Hinweis**: dem Attribut `href` können interne Seiten, externe Seiten sowie Verweise auf HTML-Elemente als Wert zugewiesen werden
- **Hinweis**: wird dem Attribut `target` der Wert `_blank` zugewiesen, wird der Verweis in einem neuen Tab geöffnet

Attribut	Beschreibung	Beispiel
<code>href</code>	Verweis	<code>"http://www.sap.com"</code>
<code>title</code>	Tooltip	<code>"SAP.com"</code>
<code>target</code>	Ziel	<code>"_blank"</code>

## Aufgabe 2

- Erstelle ein HTML-Dokument (*hobbies.html*) welches eine Aufzählung Deiner Hobbies (samt passender Überschrift) anzeigt
- Erweitere das HTML-Dokument *index.html* aus Aufgabe 1 um einen Link auf das HTML-Dokument *hobbies.html*



# Bilddateien

- Bilddateien können in HTML über den Tag `<img>` eingebunden werden
- **Hinweis:** liegen das HTML-Dokument und die Bilddatei nicht im gleichen Verzeichnis muss der Pfad zur Bilddatei ausgehend vom HTML-Dokument angegeben werden

Attribut	Beschreibung	Beispiel
<b>src</b>	Pfad zur Bilddatei	<b>"image.png"</b>
<b>alt</b>	Alternativtext	<b>"Grafik"</b>
<b>width</b>	Breite in Pixel	<b>200</b>
<b>height</b>	Höhe in Pixel	<b>200</b>
<b>border</b>	Rahmenstärke in Pixel	<b>5</b>
<b>align</b>	Ausrichtung	<b>"center"</b>

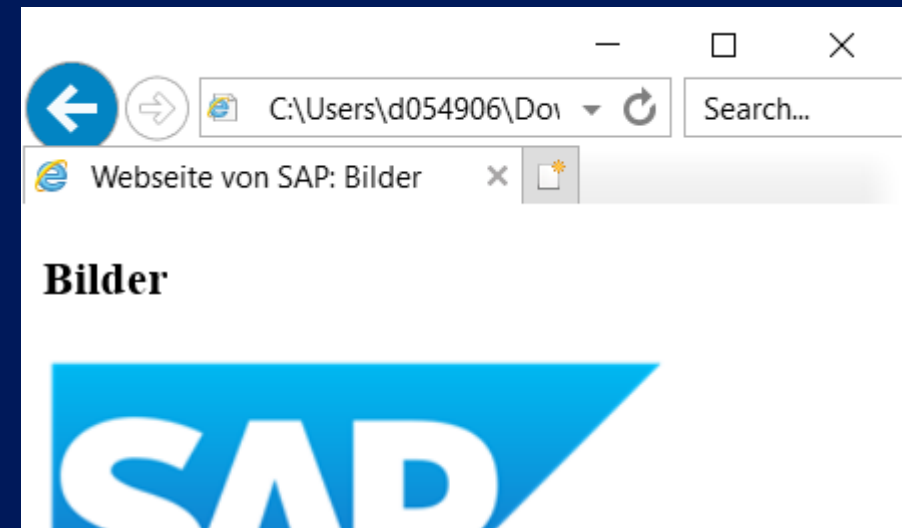
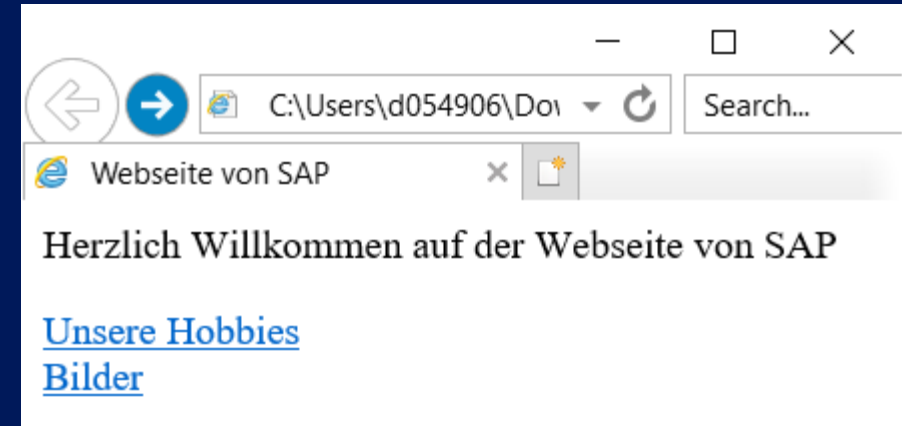
# Bereiche

- Mit Hilfe des Tags `<div>` können mehrere HTML-Elemente zu einem Bereich (**division**) zusammengefasst werden
- **Hinweis:** standardmäßig ist der Rahmen eines Bereichs nicht sichtbar

```
<div>
  <h1>Überschrift</h1>
  <p>Paragraph</p>
  <ol>
    <li>Eintrag 1</li>
    <li>Eintrag 2</li>
    <li>Eintrag 3</li>
  </ol>
</div>
```

## Aufgabe 3

- Erstelle ein HTML-Dokument (*images.html*) welches mehrere Bilder anzeigt
- Erweitere das HTML-Dokument *index.html* aus Aufgabe 2 um einen Link auf das HTML-Dokument *images.html*



# CSS: Aussehen der Webseite

# CSS

- CSS steht für **C**ascading **S**tyle **S**heets
- CSS ist keine Programmiersprache sondern eine sogenannte **Stylesheet-Sprache** und beschreibt das Aussehen von Webseiten
- CSS wurde 1994 veröffentlicht und ist, wie HTML, eine Kernsprache des *World Wide Web*



# Einbinden von Stylesheets

- Zum Einbinden von CSS-Formatierungen existieren unterschiedliche Möglichkeiten: Inline-Style, interne Stylesheets und externen Stylesheets
- **Hinweis:** im Rahmen des Workshops wird nur auf das Einbinden externer Stylesheets eingegangen

```
/* CSS-Datei style.css */  
  
body {  
    color: white;  
    background-color: black;  
}
```

```
<!--HTML-Dokument index.html-->  
  
<html>  
    <head>  
        <link rel="stylesheet" href="style.css"/>  
    </head>  
    <body>  
    </body>  
</html>
```

# Syntax

- CSS-Formatierungen bestehen aus einem **Selektor**, mehreren **Attributen** sowie den dazugehörigen **Werten**
- Selektoren legen fest, welche CSS-Formatierungen für welche Elemente gelten
- **Hinweis**: jede Formatierung muss mit einem Semikolon abgeschlossen werden
- **Hinweis**: Kommentare beginnen in CSS mit `/*` und enden mit `*/`
- **Hinweis**: Neben selbst festgelegten Klassen gibt es auch eine Reihe vordefinierter Pseudoklassen

```
* { color: white; }      /* Selektor: alle Elemente */
h3 { color: red; }       /* Selektor: HTML-Element */
#id1 { border-color: blue; } /* Selektor: Element-Id */
.classA { color: green; } /* Selektor: Klasse */
a:visited { color: yellow; } /* Selektor: Pseudoklasse */
h3:hover { color: blue; } /* Selektor: Pseudoklasse */
```

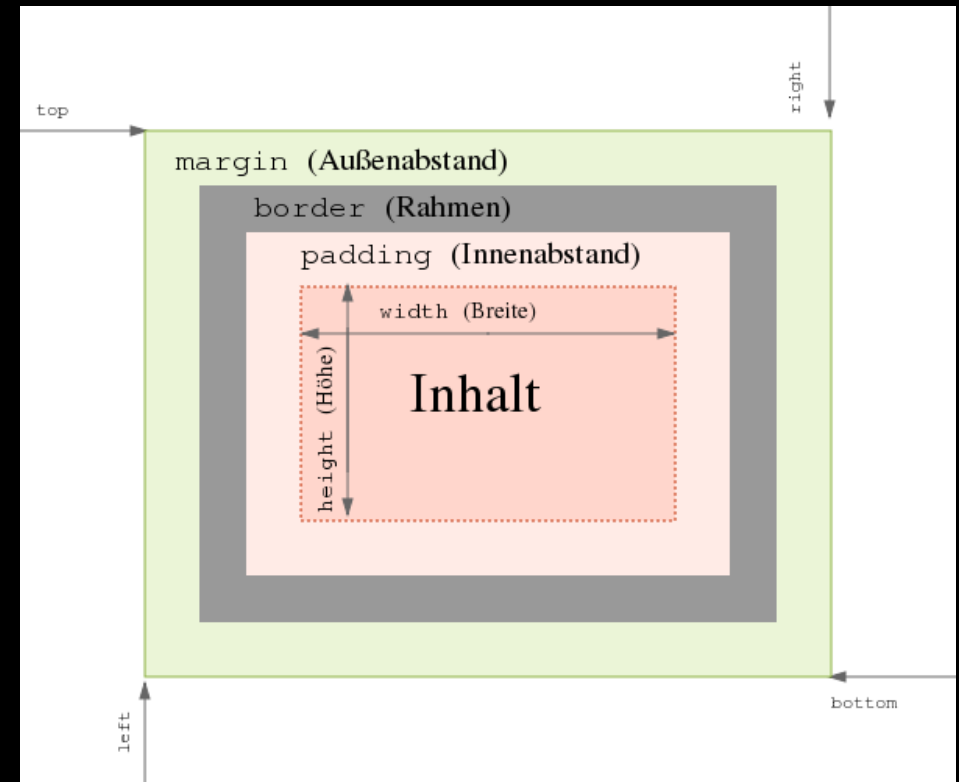
# Farbwerte

- Farbwerte können in CSS auf unterschiedliche Art und Weise angegeben werden: in Textform (vordefinierte Farben), als Hexadezimalzahl, oder als RGB- bzw. RGBA-Wert
- **Hinweis:** bei RGB werden Farben durch Angabe der Rot-, Grün- und Blauwerte festgelegt, bei RGBA wird zusätzlich noch der Alphakanal (Transparenz) angegeben

```
color: white;           /* Text */
color: #ff0000;         /* Hexadezimal */
border-color: rgb(0, 0, 255); /* RGB */
color: rgba(0, 255, 0, 0.5); /* RGBA */
```

# Box-Modell

- Die Darstellung von HTML-Elementen erfolgt über das Zeichnen von Rechtecken
- Das Box-Modell legt dabei die Größe der gezeichneten Rechtecke fest
  - Breite und Höhe des Inhalts (*width*, *height*)
  - Innenabstand (*padding*)
  - Breite des Rahmens (*border*)
  - Außenabstand (*margin*)



# Text-, Rahmen- und Aufzählungsformatierungen

- **Hinweis:** beim Attribut *style* sind unter Anderem die Werte *solid* (ausgefüllt), *dotted* (gepunktet) sowie *groove* (geritzt)
- **Hinweis:** beim Attribut *list-style-type* sind für ungeordnete Aufzählungen unter Anderem die Werte *disc* (Punkt), *circle* (kreis) und *square* (Quadrat), bei geordneten Aufzählungen die Werte *decimal* (Ziffer), *upper-roman* (römische Ziffer) sowie *lower-alpha* (Kleinbuchstabe) möglich

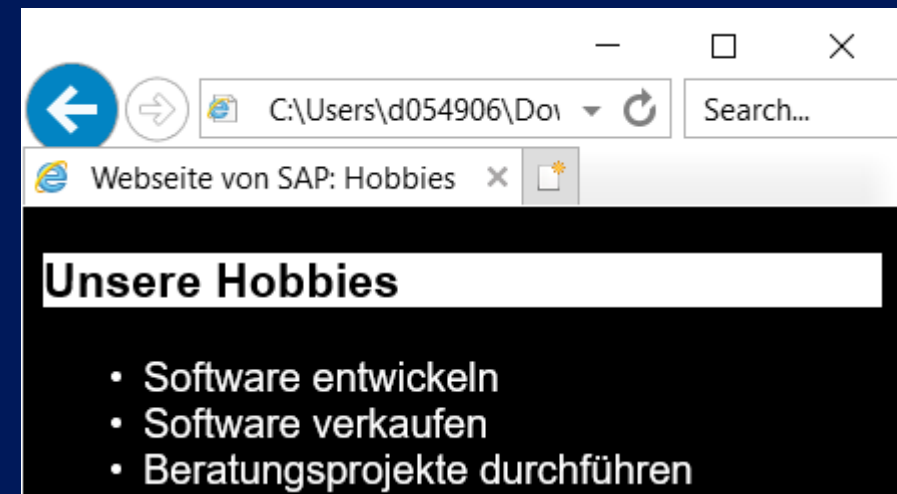
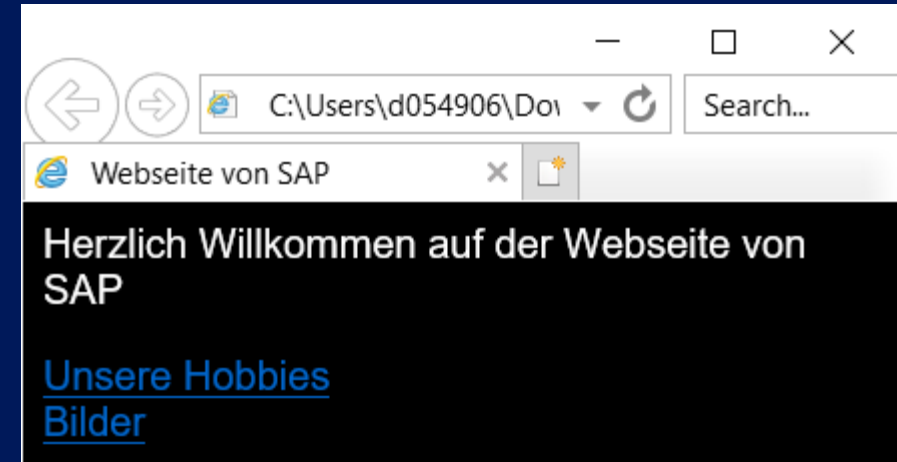
```
/* Textformatierungen */
color: white;                               /* Schriftfarbe */
font-size: 15px;                            /* Schriftgröße */
font-weight: bold;                          /* Schriftdicke */
font-style: italic;                         /* Schriftstil */
font-family: "Arial", sans-serif;          /* Schriftart */
text-decoration: underline;                 /* Dekoration */

/* Rahmenformatierungen */
border-color: rgb(0, 0, 255);                /* Linienfarbe */
border-style: solid;                         /* Linienstil */
border-radius: 10px;                         /* Radius der Ecken */
border-width: 2px;                           /* Linienstärke */

/* Aufzählungsformatierungen */
list-style-type: circle;                     /* Aufzählungszeichen */
```

# Aufgabe 4

- Erstelle eine CSS-Datei (*style.css*) und lege dort entsprechende Formatierungsregeln für Deine Webseite fest
- Binde die CSS-Datei *style.css* in die HTML-Dokumente aus Aufgabe 3 ein



# JavaScript: Verhalten der Webseite

# JavaScript

- JavaScript ist eine **Skriptsprache**, die 1995 von der Firma Netscape entwickelt wurde, um dynamische Webseiten erstellen zu können
- JavaScript hieß ursprünglich LiveScript und ist nicht mit der Programmiersprache Java „verwandt“
- JavaScript stellt eine der beliebtesten Programmiersprachen dar und gilt als Quasi-Standard für die Entwicklung webbasierter Oberflächen



# Einbinden von JavaScript-Dateien

- JavaScript kann entweder direkt oder in Form von externen JavaScript-Dateien in HTML eingebunden werden
- **Hinweis:** im Rahmen des Workshops wird nur auf das Einbinden externer JavaScript-Dateien eingegangen
- **Hinweis:** Kommentare können in JavaScript mit `//`, Kommentarblöcke mit `/*` und `*/` eingefügt werden

```
/* JavaScript-Datei index.js */  
console.log('Hello World');
```

```
<!--HTML-Dokument index.html-->  
<html>  
  <head>  
    <script src="index.js"></script>  
  </head>  
  <body>  
  </body>  
</html>
```

# Grundlegende Sprachelemente in JavaScript

- Variablen ermöglichen das Speichern von Informationen und werden in JavaScript mit dem Schlüsselwort **let** deklariert
- Für mathematische Operationen stehen die Operatoren **+** (Addition), **-** (Subtraktion), **\*** (Multiplikation) und **/** (Division) zur Verfügung
- **Hinweis:** mit Hilfe von **console.log()** können Informationen auf der Konsole ausgegeben werden

```
let result;           // Deklaration einer Variablen
result = 4 * 3;       // Verwenden math. Operationen
console.log(result);  // Ausgabe auf der Konsole
```

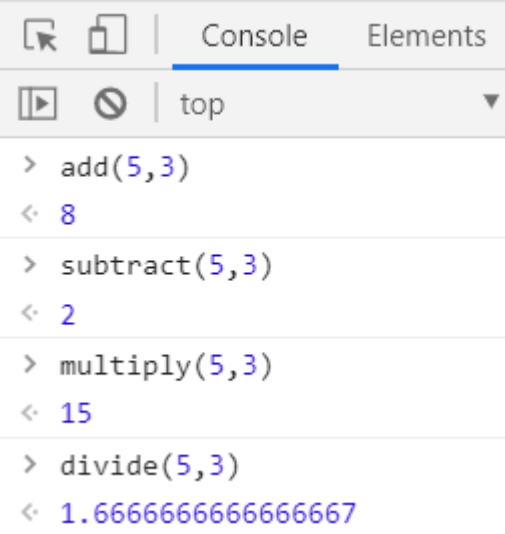
# Funktionen

- Funktionen ermöglichen die Kapselung von Quellcode
- **Parameter** ermöglichen es, der Funktion beim Aufrufen Werte mitzugeben
- Das Schlüsselwort **return** ermöglicht die Rückgabe eines Wertes an den Aufrufer der Funktion

```
function multiply(a, b) { // Definition einer Funktion
  let result;
  result = a * b;
  return result;          // Rückgabe eines Wertes
}
```

# Aufgabe 5

- Erstellen eine JavaScript-Datei (*calculator.js*) und definiere dort für jede Rechengrundart eine Funktion
- Erstelle ein HTML-Dokument (*calculator.html*) und binde dort die JavaScript-Datei *calculator.js* sowie die CSS-Datei *style.css* ein
- Erweitere das HTML-Dokument *index.html* aus Aufgabe 4 um einen Link auf das HTML-Dokument *calculator.html*



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays four function calls and their results:

```
> add(5,3)
< 8
> subtract(5,3)
< 2
> multiply(5,3)
< 15
> divide(5,3)
< 1.6666666666666667
```

# HTML-Formularelemente

- Zur Gestaltung interaktiver Webseiten stehen in HTML verschiedene HTML-Formularelemente wie Bezeichner (Tag `<label>`), Eingabefelder (Tag `<input>`) und Drucktasten (Tag `<button>`) zur Verfügung
- **Hinweis:** beim Attribut `type` (Tag `<input>`) sind unter anderem die Werte `text` (Textfeld), `password` (Passwortfeld), `color` (Farbwähler), `radio` (Auswahlschalter) und `checkbox` (Kontrollkästchen) möglich

```
<label for="input1">Textfeld</label>
<input type="text" id="input1" maxlength=30/>
<button type="button">Drucktaste</button>
```

# Zugriff auf HTML-Elemente

- Der Zugriff auf HTML-Elemente erfolgt über die Methode ***document.getElementById()***
- Das Attribut ***value*** ermöglicht den Zugriff auf den Wert eines HTML-Formularelements, das Attribut ***innerHTML*** auf „normale“ HTML-Elemente

```
let input1Value = document.getElementById('input1').value;  
let p1Value = document.getElementById('p1').innerHTML;
```

# Ereignisbehandlung

- Bei verschiedenen Aktionen (z.B. dem Betätigen einer Drucktasten) werden Ereignisse ausgelöst, auf die mit Hilfe von JavaScript-Funktionen reagiert werden kann
- Zu jedem Ereignis existiert eine dazugehörige Eigenschaft (z.B. die Eigenschaft **onClick**)

```
/* JavaScript-Datei index.js */  
function print() {  
    let input1Value = document.getElementById('input1').value;  
    document.getElementById('p1').innerHTML = input1Value;  
}
```

```
<!--Auszug HTML-Dokument index.html-->  
...  
<label for="input1">Eingabe</label>  
<input type="text" id="input1" maxlength=30/>  
<button type="button" onclick="print()">Ausgeben</button>  
<p id="p1"/>  
...
```

# Aufgabe 6

- Erweitere das HTML-Dokument *calculator.html* aus Aufgabe 5 um zwei Eingabefelder samt Bezeichner, vier Drucktasten sowie ein Ausgabefeld
- Passe die JavaScript *calculator.js* so an, dass die Werte zur Berechnung des Ergebnisses aus den Eingabefelder gelesen werden und dass das Ergebnis im Ausgabefeld angezeigt wird
- **Hinweis:** die Funktion *Number()* ermöglicht die Umwandlung einer Zeichenkette in eine Zahl

