# Web-Entwicklung

## HTML, Javascript, PHP

## *Foliensammlung*

# HTML – Hypertext Markup Language

- HTML ist eine „Sprache“ zum Strukturieren und Vernetzen von Dokumenten
- HTML kennt
  - ✔Normalen Text
  - ✔Strukturanweisungen (sog. „Tags“), die auch Verweise („Links“) zu anderen Dokumenten, lokal und im Internet enthalten
  - ✔Schnittstellen zu Programmen („Plugins“) und Programmiersprachen („Skripts“)
  - ✔Erweiterungen, die Layout und interaktives Verhalten des Dokumentes festlegen (CSS, „Cascading Style Sheets“)

Die Wikipedia-Seite zu HTML enthält auch eine Kurzeinführung zu den HTML-Tags.

# Empfohlene Literatur

- Sprach-Referenz:

  ☞ http://de.selfhtml.org/html/referenz/

- Umfangreicher Kurs zum Selbststudium:

  ☞ http://de.selfhtml.org/

# Grundstruktur HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
 <HEAD>
  <TITLE>Mein erstes HTML-Dokument</TITLE>
 </HEAD>
 <BODY>

Hier ist ein unsichtbarer Kommentar:

<!-- Dies ist ein Kommntar, er erscheint nicht in der Ausgabe,
     sondern nur im HTML-Quelltext! -->

Den Quelltext der Seite sieht man bei den meisten Browsern mit Tastenkombination
Steuerung-U.

Text wird der Darstellung automatisch umgebrochen und linksbündig formatiert.

Mehrfache          Leerzeichen          werden        als       eins    dargestellt.

Mehrfache Zeilenumbrüche

ebenfalls.
 </BODY>
</HTML>
```

# HTML-Tags

| | |
|---|---|
| <head>...</head> | Abschnitt mit „unsichtbaren“ Informationen über das Dokument |
| <title>...</title> | Text für die Titelleiste, nur innerhalb des Headers |
| <html>...</html> | Rahmt das gesamte Dokument ein |
| <body>...</body> | Block, der den „sichtbaren“ Teil des Dokumentes enthält |
| <h1 align=center>...</h1> | Überschrift Ebene 1, zentriert |
| <a href=“http://webserver/dokument“>Text</a> | Verweis auf Dokument im Internet |
| <img src=“adresse-bild“ width=breite height=höhe> | Bild an dieser Stelle einblenden |
| ... | |

# HTML-Tags

## Listen

| | |
|---|---|
| <ul> … </ul> | Unsortierte Liste |
| <ol> … </ol> | Nummerierte Liste |
| <li> Text... | Listenelement (in beiden) |
| | |
| <dl> … </dl> | Beschreibungsliste |
| <dt> | Stichwort in der DL |
| <dd> | Beschreibung in der DL |

# HTML-Tags

## Abschnitte und Textauszeichnungen

| Tag | Wirkung |
|---|---|
| <BR> | Zeilenumbruch |
| <P ALIGN=LEFT\|RIGHT\|CENTER>...</P> | Absatz (Paragraph) mit der angegebenen Formatierung, lässt Abstand nach oben und unten |
| <DIV ….> … </DIV> | Abschnitt / Bereich, ohne Abstand |
| <HR WIDTH=100> | Horizontale Linie |
| <B>Text...</B> | Fettdruck |
| <I>Text...</I> | Schrägschrift (Italics) |
| <CODE>Text...</CODE> od. <TT>...</TT> | Quelltexte in Schreibmaschinenschrift |
| <EM>Text...</EM> | Hervorheben (EMphasize) |

# Hyperlinks → das „H“ in „HTML“

- Das „Anchor“-Tag kann auf weitere Dokumente im Internet oder lokal verweisen.

```
<A HREF="http://knopper.net/bw/swt/">Hier klicken</A>
<A HREF="musik.mp3">Audio abspielen</A>
```

- Das „Anchor“-Tag kann auch auf bestimmte Stellen im gleichen oder einem anderen Dokument verweisen, wenn diese einen NAMEn besitzen:

```
<A HREF=#ende>Ans Ende Springen.</A>
…
…
<A NAME=ende></A>
```

# HTML-Tabellen

```
<table>
 <tr>
  <th>Überschrift Spalte 1</th><th>Überschrift Spalte 2</th>
 </tr>
 <tr>
   <td> Zeile 1, Spalte 1</td><td>Zeile 1, Spalte 2</td>
 </tr>
 <tr>
   <td> Zeile 2, Spalte 1</td><td>Zeile 2, Spalte 2</td>
 </tr>
 <tr>
   <td> Zeile 3, Spalte 1</td><td>Zeile 3, Spalte 2</td>
 </tr>
</table>
```

# Kodieren von Sonderzeichen – HTML Entities (1)

- Problem 1: Der Zeichensatz, in dem der Dokumentenquelltext geschrieben ist, entspricht nicht unbedingt dem Zeichensatz, den der Browser erwartet.
→ z.B. Fehler bei der Darstellung von Umlauten

- Problem 2: Für manche Sonderzeichen gibt es keine Taste auf der Tastatur, z.B.: π ğ ø Å

- Problem 3: Das Kleiner-Zeichen < und das Kaufmanns-UND & haben in HTML eine spezielle Bedeutung und können nicht „direkt“ ausgegeben werden.

# Kodieren von Sonderzeichen – HTML Entities (2)

Lösung: Kodierung als „HTML-Entity“:

**&Name;** oder **&#Nummer;**

| HTML-Quelltext | Wirkung | Merken mit |
|---|---|---|
| < | < | „Less Than“ |
| > | > | „Greater Than“ |
| & | & | „Ampers And“ |
| ä Ä | ä Ä | „A Umlaut“ |
| ö Ö | ö Ö | „O Umlaut“ |
| ü Ü | ü Ü | „U Umlaut“ |
| ß | ß | „S Z Ligatur“ (Scharfes S) |
| € | € | „Euro“ |

Türkische Sonderzeichen: Siehe hier.
Kyrillische Zeichen (Griechisch, Russisch): Siehe hier.

# Zeichenkodierung des gesamten Dokuments

Im Header des HTML-Dokuments kann der Zeichensatz, der für die Darstellung verwendet werden soll (und in dem das Dokument geschrieben ist), angegeben werden (s.a.):

```
<HEAD>
 <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8">
</HEAD>
```

# (Pre-)Formatierte Texte

- Trick A: Häufig werden Tabellen dafür verwendet, um Formularelemente oder allgemein, Teile des Dokumentes relativ zueinander zu positionieren.

- Trick B: Mit <pre> ... </pre> „eingerahmt“ erscheinen die Texte „beinahe“ immer genau dort, wo sie im HTML-Quelltext eingegeben wurden, sind aber gleichzeitig im „Schreibmaschinen-Zeichensatz“.

- Beides gilt als „schlechter Stil“, die moderne Variante zur absoluten oder relativen Positionierung von Teilen des Dokumentes sind die ☞„Cascading Stylesheets“ (CSS), die wir aber in der Vorlesung nicht im Detail behandeln werden.

# Beispiele und Übungen

→ S. Live-Beispiele Ordner

# Formulare – Web 2.0 – Interaktion!

Das <FORM>-Tag leitet einen interaktiven Bereich einer Webseite ein. In diesem können sich Elemente wie Buttons, Textfelder, Popup-Menüs etc. befinden. Es ist ein Fehler, das FORM-Tag wegzulassen, wenn ein Formular programmiert werden soll!

```
<FORM METHOD=… ACTION=…>

 … Formular-Elemente …

</FORM>
```

METHOD: Art der Übertragung (später!)
ACTION: Das Programm oder die Webseite, die die Daten erhalten soll (später!)

# Formulare abschicken oder zurücksetzen

```
<FORM METHOD=… ACTION=…>

 … Formular-Elemente …

<input name=start type=submit value=Abschicken>
<input type=reset value="Alles Zurücksetzen">

</FORM>
```

Der SUBMIT-Button überträgt die Daten an das mit ACTION angegebene Programm (später!). Der RESET-Button setzt alle Eingabefelder auf den Ursprungswert zurück (löscht sie aber nicht notwendigerweise).

# Formular-Elemente

## Das einfache Texteingabe-Feld

```
<input name=Kennung
       type=text
       size=80
       maxlength=120
       value="Vorgabe">
```

Ab HTML5 gibt es auch einen „placeholder=...“-Parameter, der einen Tipp im Feld anzeigt, welcher nach Eingabe verschwindet.

**Vorgabe**

# Formular-Elemente

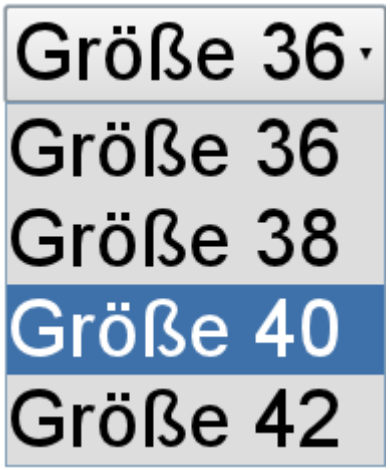
## Das mehrzeilige Texteingabe-Feld

```
<textarea name=Kennung
          cols=20
          rows=5> Vorgabetext... </textarea>
```

# Formular-Elemente

- „Popup-Menüs“ sind eine platzsparende Art, aus einer Liste genau *ein* Element auszuwählen.

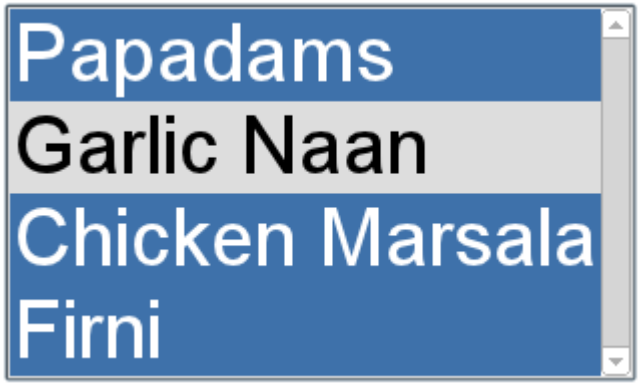
```
<select name=Kennung>
     <option value=36>Größe 36</option>
     <option value=38>Größe 38</option>
     <option value=40>Größe 40</option>
     <option value=42>Größe 42</option>
</select>
```

# Formular-Elemente

- Die gleiche Konstruktion gibt es auch mit Mehrfach-Auswahl.

```
<select name=Kennung multiple>
 <option value=101>Papadams</option>
 <option value=102>Garlic Naan</option>
 <option value=103>Chicken Marsala</option>
 <option value=104>Firni</option>
</select>
```



# Formular-Elemente

- Checkboxen sind „Schalter“, die an- und abgewählt werden können.

```
<input type=checkbox name=Kennung1 value=1>
  Bitte keine Werbung!<br>

<input type=checkbox name=Kennung2 value=2>
  Ich möchte am Gewinnspiel teilnehmen.<br>
```

☑ Bitte keine Werbung!
☐ Ich möchte am Gewinnspiel teilnehmen.

# Formular-Elemente

- Radioboxen sind „Wechselschalter“, von denen immer nur *einer* ausgewählt sein kann.

```
<input type=radio name=Kennung value=a>
  Antwort A ist richtig!<br>
<input type=radio name=Kennung value=b>
  Antwort B ist richtig!<br>
<input type=radio name=Kennung value=c>
  Antwort C ist richtig!<br>
```

- (●) Antwort A ist richtig!
- ( ) Antwort B ist richtig!
- ( ) Antwort C ist richtig!

# ACTION!

- Noch können wir mit den HTML-Formularen wenig anfangen, denn hierzu müsste ein *Programm* die Auswertung der eingegebenen Daten übernehmen, und eine Ergebnisseite „zurückschicken“, oder zumindest irgend etwas *tun*.

- Das Attribut **`action=programm-adresse`** innerhalb des `<form>`-Tag teilt dem Browser mit, wohin er die Eingabedaten zu schicken hat, und woher er dann ggf. auch wieder ein Ergebnisdokument zurückgeschickt bekommt.

- Wir werden auf die Programmiersprache ☞ PHP zurückgreifen, um Formulare auszuwerten und Ergebnisse darzustellen. Hierzu wird es notwendig sein, einen WWW-Server mit PHP-Unterstützung verfügbar zu haben!

# ☞ Javascript

- ...hat (so gut wie) nichts mit der Programmiersprache ☞ JAVA zu tun.
- ...ist in den meisten Browsern standardmäßig eingebaut, und unterstützt interaktive Funktionen, die vom Browser selbst durchgeführt werden.
- ...ist leider je nach Hersteller unterschiedlich implementiert (teilweise sehr gravierende Unterschiede in Syntax und Semantik von Dokumentstrukturen und Befehlen).
- ...wird oft eingesetzt, um „dynamische Webseiten“, „Rollover“-Funktionen und Browser-Bedienungserleichterungen, sogar einfache Spiele oder komplexe Grafikfunktionen, zu programmieren.*)

*) Wir beschränken uns in der Vorlesung auf die wichtigsten Funktionen für den „Hausgebrauch“.

# Javascript skripten – das <script>-Tag

Eine Möglichkeit, Javascript innerhalb einer HTML-Seite auszuführen, ist die <script>...</script>-Umgebung.

```
<script type="text/javascript">
  document.writeln("Hallo, Welt!");
</script>
```

Hierbei wird oft ein Trick angewendet, um mit dem „Kommentar-Tag“ für Browser, die kein Javascript können, das Programm „unsichtbar“ zu machen.

```
<script type="text/javascript">
   <!-- // Javascript-Kommentar
   alert("Hallo, Welt!");
   // Ende. -->
</script>
```

# Javascript als Attribut für HTML-Tags

- Einzelne Funktionen oder kürzere Javascript-Programme können auch direkt in andere Tags als Attribut hineingeschrieben werden. Die entsprechenden Attribute beginnen meist mit „**on**...“.

- In diesem Beispiel wird der „Wert des Textes“ in einem Abschnitt (Paragraph <p>) beim Klick darauf verändert. Man muss hier sehr darauf achten, mit den beiden „Zitatzeichen“ " und ' nicht durcheinander zu kommen. (" über der 2 und ' neben der Eingabetaste)

```
<p onClick="this.firstChild.nodeValue='Danke.'">
   Klickst Du hier!
</p>
```

# Javascript als Attribut für HTML-Tags

- Übersicht Event-Attribute (die meisten davon funktionieren mit „on...“ als Präfix):

→ SelfHTML

# Javascript-Funktionen schreiben

Eine Funktion ist eine Folge von Befehlen, die unter einem „Namen“ zusammengefasst wird. Sie kann, wie eine mathematische Funktion, ein Ergebnis produzieren, das einer Variablen zugewiesen werden kann.

```
function funktion_name(Übergabeparameter)
 {
   Befehle, durch ; getrennt.
   return Ergebnis;
 }
```

Aufruf der Funktion (Beispiel):

```
 var Resultat = funktion_name(10);
```

# Was läuft wo?

| | Client | Server |
|---|---|---|
| **HTML** | Darstellung im Browser, lokale HTML-Dateien oder entfernte Dateien.. | Liefert auf Abruf Dateien via http-Protokoll. |
| **Javascript** | Im Browser integrierte „aktive“ Skriptsprache, oft als Abschnitt in HTML-Dateien. | |
| **PHP** | | Wird vom http-Server ausgeführt (z.B. Apache mit PHP-Modul). Nur das Ergebnis der Ausführung wird an den Client übertragen. |

# Apache-Webserver für PHP vorbereiten

Windows:

- XAMPP installieren, Apache+PHP aktivieren

Linux:

- Installation `libapache2-mod-php5`
- `sudo a2enmod php5`
- `sudo /etc/init.d/apache2 restart`
- **`sudo chmod -R a+w /var/www`**
- Anlegen einer Datei `index.php` unter `/var/www` und Abruf per Browser unter http://localhost/index.php
- Unter Knoppix ist dies bereits Voreinstellung.

# PHP lernen

- Referenz: http://php.net/
- Tutorials (in deutsch): http://www.php-einfach.de/

# PHP – Skripte in Webseiten

```
<body>
 <H1 align=center>Ein einfaches PHP-Skript</H1>
 <?php echo "Hallo, Welt!"; ?>
 <?php phpinfo(); ?>
</body>
```

# PHP-Sprachelemente

- Sehen (leider) sehr ähnlich aus wie JavaScript, aber sind durch Tags **<?php** `PHP-Skript...` **?>** eindeutig gekennzeichnet, die den Browser nie erreichen.
- Anweisungen werden durch **;** voneinander getrennt.
- Variablen beginnen mit **$**-Zeichen.
- Viele Funktionen vordefiniert.
www.php.net
- U.a. auch Datenbank-Anbindung möglich.

# Auswerten von Formularen

Bei „**METHOD=POST**“ werden Formularfelder direkt an den Server verschickt, bei „**METHOD=GET**“ über die Adresse (URL), und sind in der Browser-Adressleiste sichtbar. Dies hat auch Auswirkungen auf die Art der Abfrage in PHP.

Formular:

```
<FORM METHOD="POST" ACTION="auswert.php">
 <input name=antwort type=text>
</FORM>
```

PHP-Skript „auswert.php“:

```
<?php echo "Sie haben ";
   echo $_POST['antwort'];
   echo " eingegeben."; ?>
```

# Arrays (Datenfelder) in PHP

Bereits bekannt: `$_POST['formularfeldname']`
1. Es sind neben Zahlen in `[..]` auch Zeichenketten, z.B. 'eingabe' ... erlaubt.
2. Es lässt sich zwar die ANZAHL der Elemente auslesen, aber grundsätzlich können beliebige „Stellen“ im Array über die Indizes angesprochen werden.
3. Eine spezielle Art der `for()`-Schleife erlaubt das Auslesen von Inhalten eines Arrays nach Index:

```
foreach($_POST as $key => $value) {
 echo "Der Wert von _POST[...] an der Stelle $key
ist $value.<br>\n";
}
```

→ Bildet `$_POST['key']` auf den enthaltenen `value` ab.

# Bedingungen

- In PHP können Entscheidungsabfragen und Alternativen stattfinden. Alles, was „ungleich null“, true oder „nicht leer“ ist, ist logisch WAHR (kein expliziter Typ „boolean“).

```
if($_POST['antwort']) {
 echo "Antwort ist gesetzt."; }
else {
 echo "Antwort ist nicht gesetzt, oder 0"; }

if(stristr($_POST['antwort'], "suchtext") {
 echo "Antwort enthält suchtext.";
} else {
 echo "Antwort enthält suchtext nicht.";
}
```

# Bessere Variante von „ist das Formularfeld x belegt?“

```
if( isset($_POST['formularfeldname']) ) …
```

→ Überprüft, ob das Feld überhaupt existiert, nicht nur ob es ungleich 0 ist oder Inhalt besitzt.

```
if( $_POST['formularfeldname'] != "" ) …
```

→ Überprüft, ob das Feld existiert UND nicht leer ist.

„Notice“-Korrekte Variante:

```
if(isset($_POST['name']) && $_POST['name'] != "")
…
```

# if … else … endif über größere Bereiche

Mit der Konstruktion

```
<?php if(Bedingung1): ?>
HTML-Block 1...
<?php elseif(Bedingung2): ?>
HTML-Block 2...
<?php else: ?>
HTML-Block 3...
<?php endif; ?>
```

können Teile der HTML-Seite ein- und ausgeblendet werden, je nachdem, ob die Bedingung erfüllt ist oder nicht. Achtung: Schreibweise ist hier anders (ohne {…})!

# Variablen in PHP

- Variablen in PHP sind, wie auch in JavaScript **nicht streng typisiert**, können also Zahlen oder Texte enthalten.

```
<?php
 $var1 = "1";
 $var2 = 2;
 echo "$var1 + $var2 = ";
 echo $var1 + $var2;
 echo "<br>";
 echo $var3; // Was wird hier ausgegeben?
?>
```

# Daten speichern

- Um Formulardaten auf dem Server zu speichern, können einfache, für den Webserver schreibbare Dateien verwendet werden.

```
<?php
 $datei = fopen("datei.txt", "w");
 if($datei) {
  fwrite($datei, $text);
  fclose($datei);
 } else {
  echo "Kann Datei leider nicht öffnen!");
 }
?>
```

Das Speichern in Dateien hat gewisse Nachteile...

# Dateirechte unter Linux und Windows

- Aus Sicherheitsgründen hat der Apache-Webserver normalerweise KEINE Schreibrechte auf Dateien in den fürs WWW freigegebenen Verzeichnissen.
- Unter Linux können die Dateirechte entsprechend geändert werden:
**`chmod a+w gaestebuch.txt`**
unter Windows auch, aber es ist evtl. komplizierter...
Lösungsansatz: Schreibbare Dateien generell in ein Verzeichnis legen, das für alle schreibbar ist (z.B. **`C:\temp\gaestebuch.txt`**), aber: Gefahr versehentlichen Löschens.
- Bei gleichzeitigem Schreibzugriff mehrerer Programme auf eine Datei kann Chaos entstehen.
- Professioneller Lösungsansatz: Datenorganisation mit SQL-Datenbank

# Übungen

Wir lernen die Programmiersprache PHP in Beispielen weiter kennen.

- Arrays (lineare und assoziative)
- Schleifen
- Funktionen
- Kommunikation (Mail, Session-Verwaltung)
- Formatierungshilfen, formatierte Zahlenausgaben
- Hilfsfunktionen zum dynamischen Aufbau von HTML

# Daten per PHP im Browser speichern: Session-Cookies

- Cookies sollen die zeitbegrenzte Speicherung von Session-Daten im Browser ermöglichen.

- Über den Namen des Cookies kann ein PHP-Skript zu einem späteren Zeitpunkt wieder auf die zuvor gespeicherten Informationen zugreifen, z.B. für Formular- und Authentifikationsdaten.

- Cookies können zum „Tracking“ der Browser-Aktivitäten verwendet werden, wenn Skripte verschiedener Webseiten über gemeinsame Browser-Cookies Informationen (eine einfache „Kennung“ genügt schon) übertragen, was das Erstellen von Persönlichkeitsprofilen erlaubt, daher gelten sie mitunter auch als Sicherheitsrisiko und Eingriff in die Privatsphäre.

# Session-Cookies verwenden

```
<?php

session_name("mein_shop");
session_set_cookie_params(10800);
session_start();

// Nur beim Anmelden → Username in Cookie setzen
if(isset($_POST['user']))
 $_SESSION['user'] = $_POST['user'];

echo "Willkommen " . $_SESSION['user'] . "!\n";

?>
```

# Session-Cookies löschen

```
<?php

session_name("mein_shop");
session_start();

$_SESSION = array(); // Cookie-Inhalte überschreiben

// Cookie ist „veraltet“
setcookie("mein_shop","/",time() - 3600,"/");

// Auch aktuelle Cookie-Session löschen
session_destroy();

?>
```

# Responsive und barrierefreie Webseiten (1)

Responsives Webdesign bedeutet:

- Webseite *reagiert* auf unterschiedliche Anforderungen
- Darstellung und Bedienung der Webseite kann sich verschiedenen Endgeräten automatisch anpassen
- Unterschiedliche Benutzerführung (Touch vs. Klick) je nach Eingabemöglichkeit
- Thematisch verwandt aber mit, abgrenzbar zu „Mobiler Webseite“ (für kleine Viewports konzipiert), „Adaptiver Webseite“ (stufenweise skalierbar) und „Liquider Webseite“ (fließendes Layout).
- auch unterschiedliche Darstellung der Bedienelemente: Dropdown- Menü oder Seitenleiste vs. „Hamburger Menü-Icon“.

# Responsive und barrierefreie Webseiten (2)

Responsives Webdesign kann realisiert werden durch:

- Oft Kombination aus Javascript und CSS3 als HTML5-Erweiterung, oder Umschalten des Inhalts per PHP

- Frameworks / Libraries für Responsive Webdesign:
http://www.w3schools.com/css/css_rwd_frameworks.asp

- Aktive Redirects durch den Webserver auf andere URL, je nach Client („mobile Versionen“ mit automatischer Wahl je nach Client/OS)

# Responsive und barrierefreie Webseiten (3)

Barrierefreies (oder „barrierearmes“) Webdesign bedeutet auch:

- Webseite berücksichtigt mögliche Einschränkungen der Interaktionsmöglichkeiten des Nutzers möglichst ohne Funktionsverlust.
- Zusätzliche Informationen werden als Alternative zur rein graphischen Darstellung zur Verfügung gestellt.
- Keine Einschränkung auf bestimmte Browser oder Ein-/Ausgabegeräte.
- Webseite soll stets leicht bedienbar und lesbar bleiben, unabhängig vom Browser

# Responsive und barrierefreie Webseiten (4)

Barrierefreies (oder „barrierearmes“) Webdesign

## Blinde Computernutzer:

- Alternative Beschreibungen („description“) und Bilddarstellungen (auch bezeichnet als „ALT-Tag“, eigentlich eher ALT-Parameter des IMG-Tag)
- Verzicht auf rein graphische Bedienelemente
- Funktion auch bei abgeschaltetem Javascript / CSS gewährleisten
- Klare und auch als reiner Text lesbare Struktur der Webseite (für Screenreader)

## Sehbehinderte Computernutzer:

- Große / skalierbare Bedienelemente und Texte
- Farb- und Kontrasteinstellungen
- Tastatur-Shortcuts

# Responsive und barrierefreie Webseiten (5)

Barrierefreies (oder „barrierearmes“) Webdesign

Motorisch eingeschränkte Computernutzer:
- Große Bedienelemente
- Ausreichend Abstand zwischen Bedienelementen für Eingabehilfen
- Sprung-Verweise auf Abschnitte in größeren Dokumenten

Andere (auch für Kinder):
- „Einfache Sprache“: Vermeidung komplizierter oder seltener Wörter und Wortkombinationen und zu langer Sätze
- Aufsplitten in mehrere Seiten und Verweise bei umfangreichen Texten
- Eingängige Symbole und konsistente, intuitive Benutzerführung