SPENGERGASSE

ausbildung mit zukunft

JavaScript

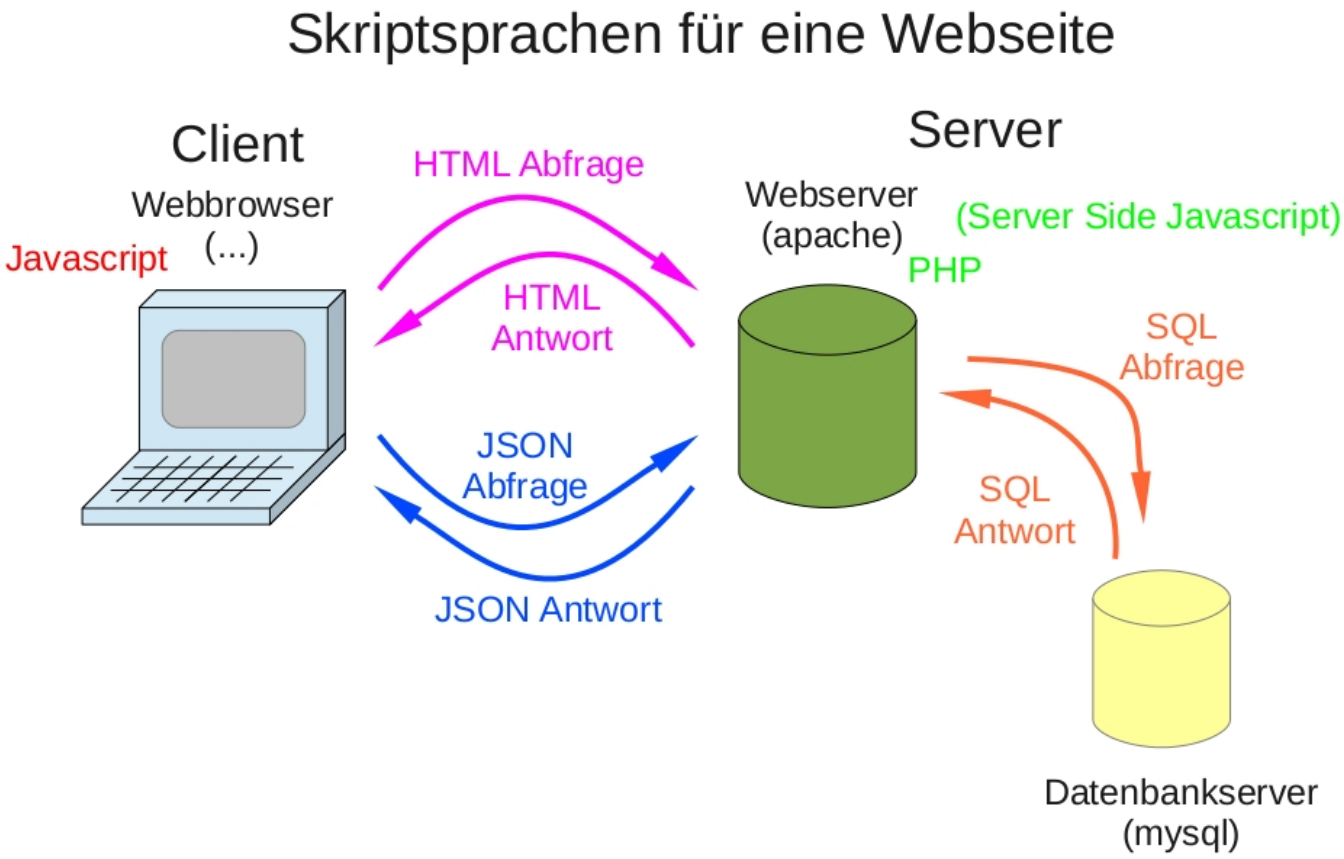
DI (FH) Levent Öztürk

Übersicht

- Wo wird Javascript ausgeführt
- Was ist eine Funktion
- Wann wird Javascript ausgeführt
- Javascript einbinden
- Javascript in HTML aufrufen
- Variablen setzen und ausgeben
- DOM (Document Object Model)
- Einfaches Editieren mit FireFox
- Quellen

Wo wird JavaScript ausgeführt

- Übersicht



Wo wird JavaScript ausgeführt

- Eine Webseite wird als HTML Abfrage geholt
  - Am Server wird eventuell durch ein PHP-Code ein HTML-Code erzeugt
  - wird eventuell eine SQL-Abfrage verwendet
  - HTML-Code wird an den Client (Browser) als Antwort auf die Abfrage geschickt
- Besitzt der HTML-Text auch JavaScript-Code, so wird dieser am Client (im Browser) ausgeführt
- zusätzliche Informationen können über JSON beim Server nachgefragt
  - Beispiel: Google Vorschläge zum Vervollständigen der Suchanfrage nach den ersten drei Buchstaben

Wie wird JavaScript ausgeführt

- wird eine Seite als Datei geöffnet und JavaScript ausgeführt (kein Server notwendig)
- Client und Server können auf derselben Maschine laufen
  - Beispiele: XAMPP oder Client auf dem Linux Server

Externe Java Script

- Wird eine Aufgabe zu verschiedenen Zeiten mehrfach benötigt, so wird der Code für diese Aufgabe in einer Funktion zusammengefasst.
  - Im Verzeichnis script befindet sich **funktionen.js**

```
// hier befinden sich zwei Funktionen
 function function1()
 {
     document.write(„hier ist funktion1\n");
 }
 function function2(wert1,wert2)
 {
     return wert1*wert2;
 }
```

Externe Java Script

- Funktionslibrary einbinden
  - Beispiel: Im Verzeichnis script befindet sich **funktionen.js**
  - Diese Datei wird in die HTML-Datei **index.htm** durch folgender Eintrag eingebunden

```
<head>
    <script language="javascript" type="text/javascript"
       src="script/funktionen.js">
    </script>
</head>
```

Externe Java Script

- Funktionen aufrufen
  - Beispiel: in der Datei Index.htm wird die Funktion function1() aufgerufen.

```
<body>
    <script language="JavaScript1.2">
       function1();
    </script>
</body>
```

  - Ergebnis: `hier ist funktion1`

Externe Java Script

- Der Funktion können Parameter mitgegeben werden, die verarbeitet werden sollen.
  - Beispiel: function2() bekommt zwei Parameter und das Ergebnis wird als Returnwert zurückgegeben

```
<body>
    <script language="JavaScript1.2">
       for (n=1;n<10;n++) {
           document.write(n+"\t"+function2(n,3)+"<br>");
       }
    </script>
</body>
```

Wann wird JavaScript ausgeführt

- Befehle außerhalb einer Funktion werden sofort während des Ladens der Seite ausgeführt.

```
<head>
    <script type="text/javascript">
       window.alert („Javascript direkt ausgeführt.");
    </script>
</head>
```

Wann wird JavaScript ausgeführt

- Funktionen werden für einen späteren Aufruf registriert. Im Head Bereich definiert.

```
<head>
    <script type="text/javascript">
       function Summe (Zahl1, Zahl2) {
           return Zahl1 + Zahl2;
       }
    </script>
</head>
```

Wann wird JavaScript ausgeführt

- Code kann nach dem vollständigen Laden und Darstellen der Seite im Browser gestartet werden

```
<body
onload="window.alert('onload:5*7='+function2(5,7));">
   <h3> Der Code wird nach dem vollständigen Darstellen der Seite gestartet<h3>
</body>
```

Wann wird JavaScript ausgeführt

- Code kann über Links ausgeführt werden

```
<a href="javascript:
window.alert('onload:5*7='+function2(3,7));">
Java Script </a>
```

- oder Buttons auf der Webseite aktiviert

```
<form name="Test" action="">
   <input type="text" size="10" name="wert1">*
   <input type="text" size="10" name="wert2">=
   <input type="text" size="20" name="Ergebnis"><br>
   <input type="button" value="Ausrechnen"
       onclick="Ergebnis.value =
       function2(wert1.value,wert2.value)">
</form>
```

Wann wird JavaScript ausgeführt

- oder mit der Maus überfahren werden
  - onmouseover

```
<form name="Test" action="">
   <input type="text" size="40" name="Ergebnis"><br>
   <input type="button" value="Aktuelles Datum"
       onmouseover="Ergebnis.value = Date()">
</form>
```

Ereignisverarbeitung

- Einbindung in HTML und Ereignisverarbeitung (Event-Handling)
  - Einbindung in HTML mit dem script-Element
  - Grundlagen der Ereignisverarbeitung
  - Arbeiten mit dem Event-Objekt
  - Fortgeschrittene Ereignisverarbeitung

Ereignisverarbeitung

- Folgende Ereignisse können eingebunden werden
  - onabort (bei Abbruch)
  - onblur (beim Verlassen)
  - onchange (bei erfolgter Änderung)
  - onclick (beim Anklicken)
  - ondblclick (bei doppeltem Anklicken)
  - onerror (im Fehlerfall)
  - onfocus (beim Aktivieren)
  - onkeydown (bei gedrückter Taste)
  - onkeypress (bei gedrückt gehaltener Taste)
  - onkeyup (bei losgelassener Taste)
  - onload (beim Laden einer Datei)

Ereignisverarbeitung

- onmousedown (bei gedrückter Maustaste)
  onmousemove (bei weiterbewegter Maus)
  onmouseout (beim Verlassen des Elements mit der Maus)
  onmouseover (beim Überfahren des Elements mit der Maus)
  onmouseup (bei losgelassener Maustaste)
  onreset (beim Zurücksetzen des Formulars)
  onselect (beim Selektieren von Text)
  onsubmit (beim Absenden des Formulars)
  onunload (beim Verlassen der Datei)
  javascript: (bei Verweisen)

Variablen und Arrays

- Variablentypen
- Variablennamen
- Deklaration und Gültigkeitsbereiche
- Typ anpassen
- Was ist ein Array
- Arrays deklarieren
- Zugriff auf ein Array
- Quellen

Variablen und Arrays

- Variablentypen: In Javascript werden nur drei Typen von Variablen unterschieden *(Beispiel02-02.html)*
  - numerische Variable (ganzzahlig oder mit Komma)
    - var Zahl1 = 42; // ganze Zahl
    - var Zahl2 = 14.3; // Zahl mit Nachkommastelle
  - boolean (true = wahr oder false = falsch)
    - var Bool = true, var Bool = false;
  - Zeichenketten (alles andere)
    - var Text = „Zeichenkette“;
  - Typ einer Variablen feststellen
    - typeof ()

Variablen und Arrays

- Variablennamen
  - Namen von Variablen müssen mit einem Zeichen oder einem Unterstrich (_) beginnen
  - Danach dürfen Variablen auch Ziffern enthalten
  - Es wird zwischen Groß- und Kleinschreibung unterschieden!
  - Beispiele für gültige Namen von Variablen
  - _abc, _123, A12B, fF, k_h6_B_3_d

Variablen und Arrays

- Deklaration und Gültigkeitsbereiche
  - Definition local
    - Innerhalb von {} gültig
    - var name1;
    - var name1=„Hugo“;
  - Definition global
    - Im gesamten Script gültig
    - name2= „Oberlix;
    - wert=77;

Variablen und Arrays

- Übersicht der Funktionen zur expliziten Umwandlung
  - String (Wert)
  - Number (Wert)
  - parseInt (Wert)
  - parseFloat (Wert)
  - Fehler bei Typfehler: „NaN“ = „Not a Number“

Variablen und Arrays

- Mathematische Operationen mit Umwandlung

```
<script language="JavaScript1.2">
   a=3, b=4, c=13.3, d='14.4';
   e=a*d;
   f=c+d;
   g=c+parseFloat(d);
   document.write("a=3, b=4, c=13.3, d='14.4'<br>");
   document.write("a*d="+e+"; c+d="+f+"; c+d="+g);
</script>
```

  - Ergebnis:
    - a=3, b=4, c=13.3, d='14.4'
      a*d=43.2; **c+d=13.314.4**; c+d=27.700000000000003

Variablen und Arrays

- Was ist ein Array?
  - Ein Array ist ein Zusammenschluss von mehreren Variablen
  - Sie müssen nicht vom selben Typ sein!
  - Zugriff auf Arrayelemente kann über
    - Indizierung oder
    - auch über eindeutige Namen erfolgen (assoziativ)

Variablen und Arrays

- Arrays deklarieren
  - Als Liste von Konstanten
    - var Schueler1 = [„Max“, „Muster“, 17, true];
  - Als Array-Konstruktor
    - var Schueler2 = new Array („Max“, „Muster“, 17, true);
  - Als assoziatives Array
    - var Schueler3 = new Array ();
  - Schueler3[„Vorname“] = „Max“;

Variablen und Arrays

- Array Beispiele
  - Zugriff auf ein Array

```
<script language="JavaScript1.2">
   var Schueler1 = ["Max", "Muster", 17, true];
   var Schueler2 = new Array("Hugo","Schuster",18, false);
   var Schueler3 = new Array ();
   Schueler3["Vorname"] = "Hans";
   document.write(Schueler1[0]+" "+Schueler1[1]+"
       "+Schueler1[2]+" "+Schueler1[3]+"<br>");
   document.write(Schueler2[0]+" "+Schueler2[1]+"
       "+Schueler2[2]+" "+Schueler2[3]+"<br>");
   document.write(Schueler3["Vorname"]+"<br>");
</script>
```

Variablen und Arrays

- Anzahl der Arrayelemente
  - Schueler1.length()

```
<script language="JavaScript1.2">
   var Schueler1 = ["Max", "Muster", 17, true];
   for (i=0;i< Schueler1.length;i++){
       document.write(Schueler1[i]+" ");
   }
</script>
```

  - Ergebnis?

Variablen und Arrays

- Arrayelemente anhängen, löschen, ersetzten
  - pusch()

```
<script language="JavaScript1.2">
   var monate = new Array("Januar", "Februar", "März");
   monate.push("April", "Mai", "Juni", "Juli", "August");
   monate.push("Settember", "Optober");
</script>
```

  - splice()

```
monate.splice(8, 2, "September","Oktober", "November",
"Dezember");
// 9. und 10. Elemente werden gelöscht und 4 Elemente
angehängt
```

Variablen und Arrays

- Arrayelemente sortieren
  - sort()

```
monate.sort();
for (i=0;i< monate.length;i++) {
       document.write(monate[i]+", ");
}
```

  - Ausgabe

```
April, August, Dezember, Februar, Januar, Juli, Juni, Mai,
März, November, Oktober, September,
```

Document Object Model (DOM)

- Was ist eine Objekt
- Attribute vorhandener Objekte
- Methoden vorhandener Objekte
- Darstellung einer Webseite im DOM
- Beispiele für Objekte
- Quellen

Document Object Model (DOM)

- Was ist ein Objekt?
  - Dient der Abbildung realer Objekte in Software
  - Objekte enthalten
    - Attribute (zugeordnete Variablen)
    - Methoden (zugeordnete Funktionen)
  - Attribute werden nur durch Methoden verändert (Kapselung).
  - Attribute und Methoden werden vom Objekt durch einen Punkt getrennt.

Document Object Model (DOM)

- Attribute vorhandener Objekte
  - window
    - *innerHeight*, *innerWidth*: innere Größe der Seite
    - *outerHeight*, *outerWidth*: äußere Größe der Seite
    - *screenX*, *screenY*: linkes, oberes Eck der Seite
  - screen
    - height, width: Größe des Bildschirms
    - availHeight, availWidth: verwendbarer Bildschirm

Document Object Model (DOM)

- Methoden vorhandener Objekte
  - window
    - alert (): Text in einem eigenen Fenster ausgeben
    - open (): neues Fenster aufmachen
    - prompt (): Eingabe über ein eigenes Fenster
  - Location
    - assign (): neues Dokument laden
    - reload (): gleiches Dokument noch einmal laden
    - replace (): aktuelles Dokument ersetzen

Document Object Model (DOM)

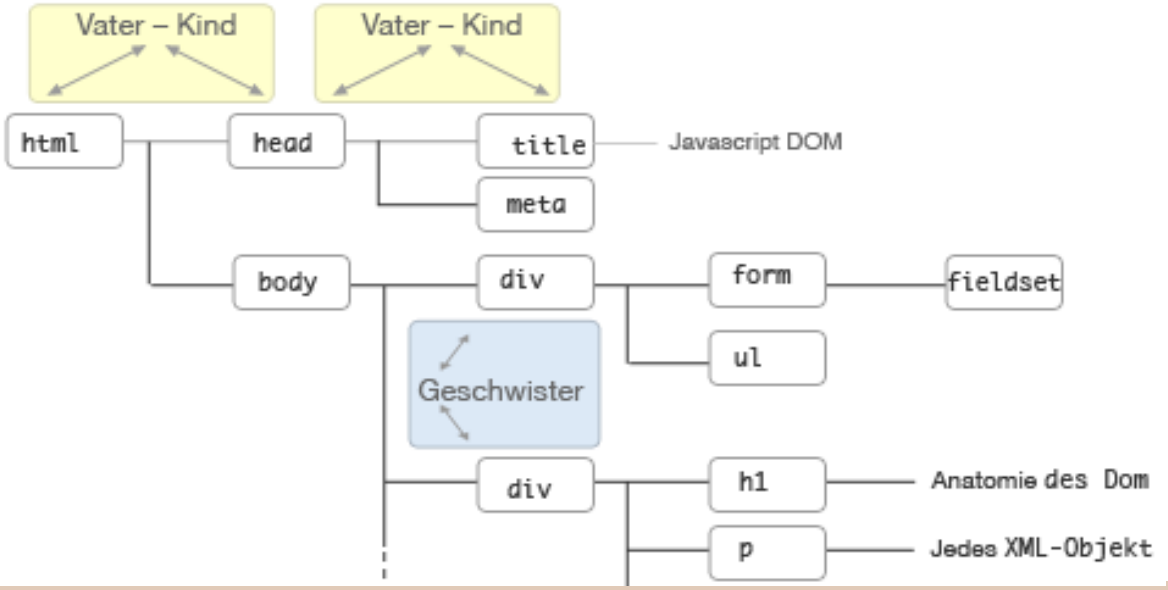
- Darstellung einer Webseite im DOM
  - Die Wurzel stellt das Objekt „document“ dar.
  - Jeweils durch das Attribut „childNodes“ kann der gesamte Baum durchgegangen werden, in dem das Dokument gespeichert ist.
  - Gibt es mehrere Frames, können diese über das Objekt „frames“ erreicht werden.
  - Im Objekt „document“ gibt es Listen
    - anchors, forms, images, links, styleSheets, ...

Javascript DOM

- Das DOM ist der Standard für den Zugriff auf alle Tags und Attribute der gesamten Webseite.
- Eine Baumstruktur setzt die Elemente eines HTML-Dokuments in Beziehung zueinander, so dass Javascript von einem Element aus durch die Seite navigieren und HTML-Tags einfügen oder entfernen kann.

Javascript DOM

- Das HTML-Element
  - ist der Vaterknoten des HEAD- und des BODY-Elements. Das HEAD-Element wiederum hat zwei Kinder: das TITLE- und das META-Element.



Javascript DOM

- Das HTML-Element
  - TITLE- und META-Element sind Nachfahren desselben Eltern-Elements – *siblings* in der Sprache des DOM..
  - Die Methoden und Eigenschaften des DOM
    - erzeugen neue Tags
    - manipulieren die Attribute
    - verschieben Elemente mitsamt allen untergeordneten Elementen
    - steuern den Fluss der Ereignisse ohne Neuladen des Dokuments.

Javascript DOM

- Suchen eines bestimmten HTML-Elements
  - getElementById ()
  - getElementsByClassName ()
  - getElementsByName ()
  - getElementsByTagName ()
- Beispiele:

Javascript Objekt

- Objekt orientierte Ansätze
  - Klasse Object()
  - Attribute definieren
  - Methoden definieren
  - Prototype
  - Vererbung

Javascript DOM

- Objekt orientierte Ansätze
  - Beispiel

```
function Person(v,n,geb,g,t,e){
p= new Object();
p.VorName=v;
p.NachName=n;
p.Gebdat=geb;
p.Geschlecht=g;
p.Tel=t;
p.Email=e;
return p;
}
```

Javascript DOM

- Objekt orientierte Ansätze
  - Beispiel: Person wird erzeugt und ins Array gespeichert

```
var leute= new Array();
var p=Person (”Max", ”Muser", "12.5.1984", "m",
"069912345678", "ahmet@bbb.cc");
leute.push(p);
```

```
for(i=0;i<leute.length;i++){
document.write(i+" "+leute[i].VorName+" "+
leute[i].NachName +" "+leute[i].Gebdat+" "+
leute[i].Geschlecht +" "+leute[i].Tel+" "+
leute[i].Email+"<br>");}
```

Quellenangaben

- http://molily.de/js/
- http://www.rolandgeyer.at/kurse/german/javasrc/js002.html
- http://de.selfhtml.org/javascript/sprache/eventhandler.htm
- http://www.mediaevent.de/javascript/Javascript-Basis-Variablen.html
- http://www.peterkropff.de/site/javascript/variablen.htm

Quellenangaben

- http://www.mediaevent.de/javascript/DOM.html
- http://www.w3schools.com/js/default.asp