Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Willkommen bei der Höheren Berufsbildung Uster

JavaScript

Raphael Ritter & Reto Rezzonico

Ein Angebot der Berufsfachschule Uster und der Höheren Fachschule Uster

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Dozent

- Raphael Ritter
  - Bsc. Computer Science
  - EMBA General Management

- Geschäftsführer 2BIT GmbH
  - Senior Consultant

- Dozent Bereich JavaScript HFU
- Dozent Bereich AngularJS und Hybride Entwicklung HSR

2BIT
Business Intelligence | Mobile Apps | Consulting

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Dozent

- Reto Rezzonico
  - Bachelor of Science ZFH in Informatik
- Senior Consultant 2BIT GmbH
- Full Stack Software Engineer
  - Aktueller Schwerpunkt Web und App Entwicklung

2BIT
Business Intelligence | Mobile Apps | Consulting

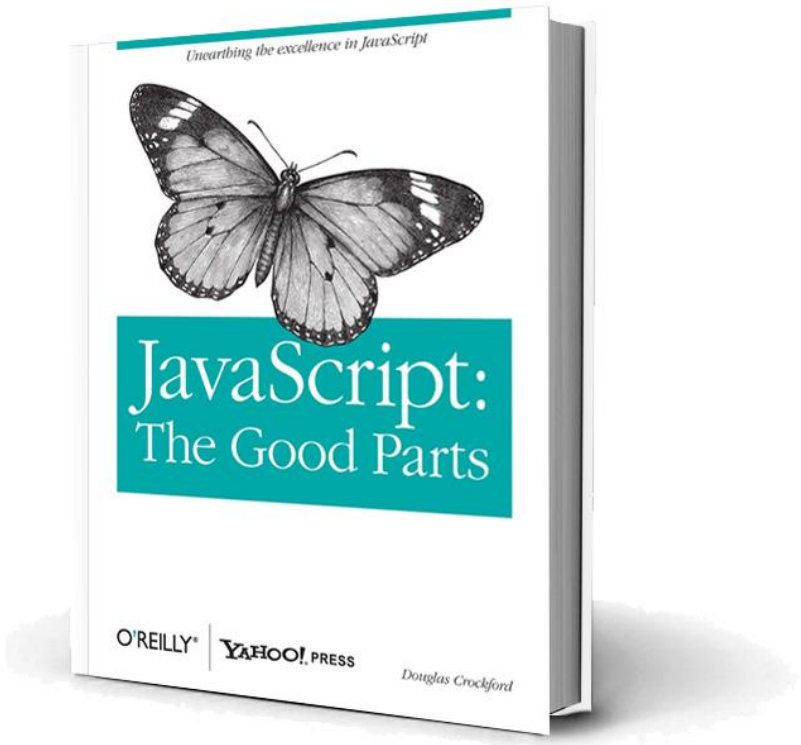
Vorstellung

Bitte stellt euch kurz vor:

- Name / Vorname
- Beruf
- Motivation
- Ziele für das Modul JavaScript

Modulübersicht

- Literatur zusätzlich zum Unterricht



Modulübersicht

- JavaScript Einführung und Grundlagen
- OOP
- DOM
- JSON
- AJAX
- JQuery
- Animationen
- ECMA Script 5
- JS Frameworks
- Tools
- TypeScript
- Frameworks

Modulübersicht

- JavaScript Einführung und Grundlagen
  - Sprach Konstrukte (Abend 1)
  - Grundlagen & Typen (Abend 2)
  - DOM (Abend 3)

- JavaScript Advanced
  - jQuery (Abend 4) [Lernzielkontrolle Grundlagen]
  - Google Maps / JSON / Handlebars (Abend 5)
  - Security & Architektur (Abend 6) [MLZ Vorstellung]
  - Patterns & Idioms (Abend 7) [Lernzielkontrolle]

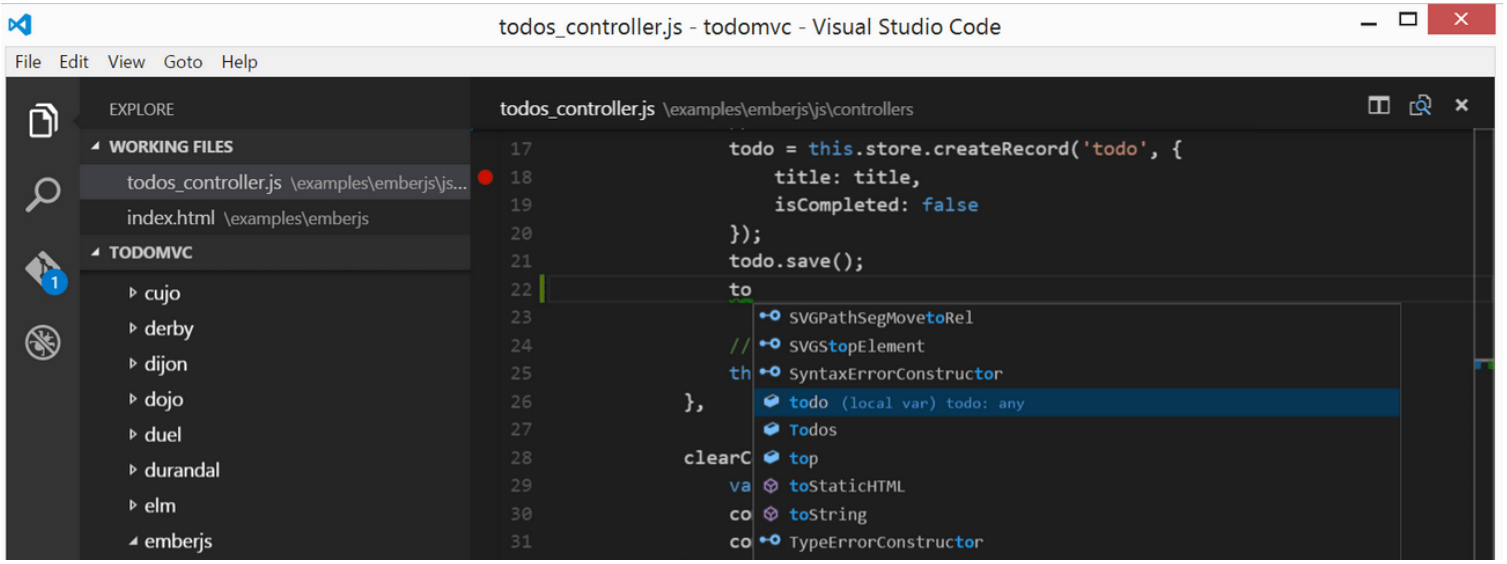
- JavaScript Beyond
  - TypeScript (Abend 8)
  - Frameworks (Abend 9) [MLZ Abgabe]

Modulübersicht

- Es wird zwei Lernzielkontrollen geben
  - Eine nach dem Grundlagen Block von drei Vorlesungen
  - Eine weitere nach der sechsten Vorlesung
- Diese beiden Modulprüfungen ergeben 50% der Modulnote

- Die restlichen 50% ergibt die Modullernzielkontrolle die am Ende des Moduls abgegeben werden muss
  - Die Lernzielkontrolle wird am sechsten Abend vorgestellt
    - Am siebten und achten Abend erhaltet ihr jeweils eine Stunde um daran zu arbeiten und Fragen zu stellen

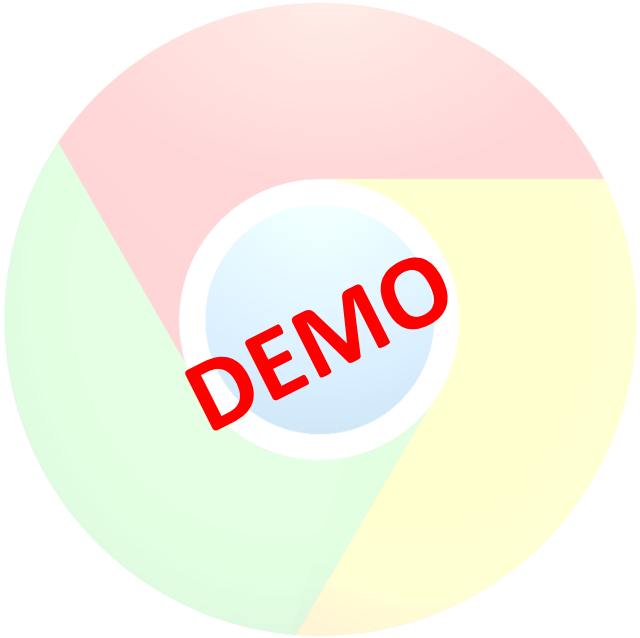
Visual Studio Code



https://code.visualstudio.com/

Chrome

- Chrome wird als Webbrowser eingesetzt im Unterricht, da er verschiedene gute Features zum debuggen von Code bietet



Agenda

- Einführung JavaScript

- JavaScript Grundlagen
  - Typen
  - Funktionen
  - Operatoren

Unterlagen der HSR

Einige Inhalte sind aus den Folien / Übungen des CAS Frontend Engineering der Hochschule für Technik Rapperswil.

www.hsr.ch
http://hsr.ch/CAS-Front-End-Engineering.12432.0.html

Einführung

- Skript Sprache welche hauptsächlich in Web Browsern eingesetzt wird
- Der Einsatz auf der Serverseite nimmt aber derzeit stark zu
- Sprachkern als ECMAScript standardisiert.
  - http://www.ecmascript.org/
  - V1: 1997
  - V5.1 (aktuell): 2011
  - http://www.ecma-international.org/ecma-262/5.1/
  - V6: Mitte 2015

Einführung

- JavaScript ist dynamisch typisiert.
- Erlaubt objektorientierte, prozedurale und funktionale Programmierung.
- JavaScript wird üblicherweise nicht kompiliert sondern interpretiert.

Einführung

- Dynamische Manipulation von Webseiten über DOM.
- Client-Seitige Validierung von Formularen.
- Senden und empfangen von Daten ohne erneutes Laden der Seite.
- Animationen
- Berechnungen
- Zum Teil auch als Server-Seitige Skript Sprache (jsnode)
- AJAX für Web 2.0

Einführung

- JavaScript ist nicht gleich Java !!!
  - Java verhält sich zu JavaScript wie eine Ameise zu einem Ameisenbären
  - JavaScript hat mit Java nur den ersten Teil des Namens gemeinsam

JavaScript

- Interpretiert
- Objekte aus anderen Objekten (prototypes)
- Weak typing
- Von Brendan Eich

Java

- Compiliert
- Objekte aus Klassen
- Strong typing
- Von Sun Microsystems

Einführung

- Vorteile
  - Benötigt kein Browser Plugin
    (im Gegensatz zu Flash, Flex, Applets, Silverlight …)
  - Logik im Client spart Rechenzeit auf dem Server und Roundtrips
  - Schnellere Antwortzeiten für den Client

Einführung

- Nachteile von JavaScript
  - Browser Inkompatibilitäten
  - nicht mehr so schlimm wie vor der Standardisierung
  - JavaScript kann deaktiviert werden
  - Sicherheitsprobleme (Ausbruch aus der Sandbox)
  - XSS Angriffe

Einführung

- Sandbox Prinzip
  - Der interpretierte Code läuft in einer Sandbox.
  - Es besteht nur Zugriff auf die aktiven Objekte im Browser.
  - Kein Zugriff auf Dateien, Speicher, etc.
  - Jede Applikation wird isoliert betrachtet
  - Erweiterte Zugriffe sind möglich, wenn diese der Benutzer explizit erlaubt
  - Keine Persistenz (ausser Cookies)
  - Netzwerkzugriff nur zum „Ausgangs“ Host

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Einführung

Hat jemand eine Idee was dieser Code machen könnte?

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>JavaScript</title>
</head>
<body onload="alert('Seite wurde geladen.');">
  <h1>JavaScript</h1>
</body>
</html>
```

Einführung

- Im Head Bereich der HTML Datei
  <script type="text/javascript">..</script>
- Als externe Datei die im Headbereich eingebunden wird
  <script type="text/javascript" src="functions.js"></script>
- Inline im normalen HTML Inhalt der Webseite
  <script type="text/javascript">..</script>
- In den Attributen eines Elements
  <p style="cursor:pointer"
  onmouseover="this.innerHTML='Hallo Maus!'"
  onmouseout="this.innerHTML='Komm zu mir, Maus!'">
  Komm zu mir, Maus!
  </p>
- Was sind die jeweiligen Vor- Nachteile?

Einführung

- In HTML

```
...
<script type="text/javascript">
  alert('Seite wurde geladen.');
</script>
```

- In externer Datei

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

Einführung

- Mit dem Element <noscript> lässt sich ein alternativer Text (Fehlermeldung) ausgeben wenn der Browser JavaScript nicht unterstützt oder es deaktiviert ist.

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script type=”text/javascript” >
    document.write("JavaScript ist aktiv !!!");
</script>
<noscript>
    Ihr Browser unterstützt JavaScript nicht!
</noscript>
</body>
</html>
```

Einführung

- // Für einzeilige Kommentare
- /* für ganze Kommentarblöcke
- die auch über mehrere Zeilen gehen können */

Einführung

- JavaScript ausführen
  - Am Ende vom Laden der Seite

```
...
<body onload="alert('Seite wurde geladen.');">
        <h1>JavaScript</h1>
</body>
```

  - Beim Klick auf einen HTML Button

```
<input type="button" value="Check" onclick="raten();" />
```

  - Formular Validierung

```
<form onsubmit="return validiereFormular();">
```

Einführung

Referenzen

http://de.selfhtml.org/javascript/
http://www.w3schools.com/js/
http://www.ecmascript.org/docs.php

Übungen

- Übung 1.1 selbständig lösen

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Grundlagen

Grundlagen von JavaScript

JavaScript Funktion

- Funktionen sind ein grundlegendes Konzept von Programmiersprachen
- In JavaScript werden sie mit dem Schlüsselwort function definiert
- Argumente und Rückgabewert sind optional

```
function functionName(arg0, arg1, ... ,argN)
{
        // Anweisungen
        return value;
}
```

JavaScript Funktion

- Funktionsdefinition

```
function sayHello(name, message) {
        alert(“Hello “ + name + “,” + message)
}
```

- Aufruf

```
sayHello(“Brendan”, “ how are you today?”)
```

JavaScript Funktion

- Funktionsdefinition

```
function add(a, b) {
        return a + b;
}
```

- Aufruf

```
var c = add( 10, 11 );
```

Lokale Variablen

- Variablen innerhalb von Funktionen sollten immer mit var deklariert werden.
- So sind sie nur lokal gültig

- Beispiel

```
function test() {
        var message = “hi”; //local variable
}
test();
alert(message); //undefined
```

Lokale Variablen

- Eine Deklaration ohne var ist zwar auch gültig, führt aber zu einer globalen Variable

- Beispiel

```
function test() {
        message = “hi”; //global variable
}
test();
alert(message); //”hi”
```

Geltungsbereiche von Variabeln (Scope)

- JavaScript unterscheidet zwischen lokalen und globalen Variablen.
- Lokale Variablen sind nur innerhalb des Blockes (bspw. eine Funktion) gültig in der sie definiert wurden
- Globale Variablen sind in der ganzen JavaScript-Anwendung gültig
  - Damit eine Variable als lokale Variable angesehen wird, muss diese mit var definiert werden.
- Wenn man das var vergisst, erstellt man automatisch eine globale Variable was zu unschönen Nebeneffekten führen kann
- Best Practices:
  - Den Variablen immer ein var voranstellen
  - Gewollte Ausnahmen im Code kommentieren

Übungen

- Übung 1.2 selbständig lösen
- Übung 1.3 selbständig lösen

Boolsche Variablen

- Boolesche Variablen können als Wert nur true (wahr) oder false (falsch) annehmen.

```
var ist2000einSchaltjahr = true;
var ist2004einSchaltjahr = true;
var ist3000einSchaltjahr = false;
```

Vergleichsoperatoren

- Ab der JavaScript-Version 1.2 können Gleichheit/Ungleichheit nicht nur mit == bzw. !=, sondern auch mit === bzw. !== abprüft werden. In diesem Fall werden die Werte zusätzlich auf ihren Variablentyp hin überprüft.

Operator	Beschreibung
==	Gleichheitsoperator
!=	Ungleichheitsoperator
<	Kleiner-als-Operator
>	Größer-als-Operator
<=	Kleiner-als-oder-gleich-Operator
>=	Größer-als-oder-gleich-Operator

Vergleichsoperatoren

The Abstract Equality Comparison Algorithm (==)

- Definition:
  - http://www.ecma-international.org/ecma-262/5.1/#sec-11.9.3

- Beschreibung:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Comparison_Operators#Using_the_Equality_Operators

Vergleichsoperatoren

- Überlegen Sie sich, welche Resultate die folgenden Eingaben bewirken. Versuchen Sie dies mit Hilfe vom «The Abstract Equality Comparison Algorithm" zu begründen. (ca 20min.)

```
undefined == null;
undefined == false;
+undefined == +false;
!undefined == !false;
new String("A") == new String("A");
var myObject = { name : "A", valueOf : function() { return this.name} };
myObject == "A";
```

Practice

Vergleichsoperatoren

- Aus diesem Grund **NIE** das doppelte == verwenden
- Für vergleiche von Werten **IMMER** das dreifache === verwenden
  - Nur so kann die automatische Typenconvertierung ausgehebelt werden

Logische Operatoren

Operator	Beschreibung
&&	logischer AND-Operator
||	logischer OR-Operator
!	logischer NOT-Operator

Bitoperatoren

Operator	Beschreibung
&	bitweiser AND-Operator
|	bitweiser OR-Operator
^	bitweiser XOR-Operator
~	bitweiser Komplement-Operator
<<	bitweise Verschiebung nach links
>>	bitweise Verschiebung nach rechts
>>>	bitweise Verschiebung nach rechts mit Füllnullen

Numerische Variablen

- Können Ganzzahl oder Floating Point sein

```
var pi = 3.14159265;
var millenium = 2000;
var minusEins = –1;
```

Arithmetische Operatoren

Operator	Beschreibung
+	Additionsoperator
-	Subtraktionsoperator
*	Multiplikationsoperator
/	Divisionsoperator
%	Modulo-Operator

Operator	Beschreibung
+=	Additions-Zuweisungsoperator
-=	Subtraktions-Zuweisungsoperator
*=	Multiplikations-Zuweisungsoperator
/=	Divisions-Zuweisungsoperator
%=	Modulo-Zuweisungsoperator

Inkrement / Dekrement

- Werden zum Auf- und Abwerten eines einzelnen Wertes verwendet.
- Inkrement- und Dekrement-Operatoren sind einstellige Operatoren und werden nur in Verbindung mit einem ganzzahligen oder einem Fliesskomma-Operanden benutzt.

- Der Inkrement-Operator ++ erhöht den Wert des Operanden um 1
- Der Dekrement-Operator - - erhöht den Wert des Operanden um 1

Inkrement / Dekrement

- ++zaehler; entspricht zaehler = zaehler + 1;
  (oder auch zaehler+=1;)
- Es gibt aber auch die Syntax, wo der Operator der Variablen nachgestellt wird: zaehler++;
- Die Reihenfolge von Operand und Operator ist relevant:
- Wenn der Operator vor dem Operanden steht, erfolgt die Erhöhung des Wertes bevor der Wert dem Operanden zugewiesen wird.
- Wenn er hinter dem Operanden steht, erfolgt die Erhöhung nachdem der Wert bereits zugewiesen wurde

Strings

- Strings werden in " oder ' angegeben.
- Es besteht kein Unterschied (Im Gegensatz zu PHP) zwischen einem String mit "..." oder '...‘
- Wenn man in einem String ein " oder ' Zeichen einfügen möchte, kann das mittels dem Escapezeichen \ realisiert werden.

- Beispiele

```
var s= 'Mein Text ist: "Hello world"   ';
var s= "Mein Text ist: 'Hello world' ";
var s= "Mein Text ist: \"Hello world\"";
var s= 'C:\\TEMP';
```

Strings

- Strings können mit dem Plus (+) Operator zusammengefügt werden (Concatenation)

- Beispiele:

```
var vorne = "Tages";
var hinten = "Anzeiger";
var Verlag = vorne + " " + hinten;  //liefert "Tages Anzeiger"
```

Strings

Methode	Beschreibung
charAt()	Gibt den Buchstaben an der angegebenen Stelle zurück.
charCodeAt()	Gibt den Unicode an der angegebenen Stelle zurück.
concat()	Verbindet zwei oder mehr Strings und gibt diese als neuen String zurück.
fromCharCode()	Konvertiert Unicode Codes in Buchstaben.
indexOf()	Gibt die Position vom ersten Vorkommen des angegebenen Strings zurück.
lastIndexOf()	Gibt die Position vom letzten Vorkommen des angegebenen Strings zurück.
match()	Gibt die Matches von einem Regular Expression zurück.
replace()	Ersetzt die Matches von einem Regular Expression mit dem Substring.
search()	Gibt die Position vom Match eines Regular Expressions zurück.
slice()	Gibt einen Substring zurück. Negative Werte für die Selektion von der rechten Seite.
split()	Splittet ein Array an der angegebenen Stelle und gibt ein Array zurück.
substr()	Gibt einen Substring zurück.
substring()	Gibt einen Substring zurück.
toLowerCase()	Konvertiert den String in Kleinbuchstaben.
toUpperCase()	Konvertiert den String in Grossbuchstaben.
valueOf()	Gibt den primitive Type vom String zurück.

Strings

```
var s = "Hello World";

var l = s.length;
// l == 11;

var substring = s.substring(6, 11);
//substring == "World"

var i = s.indexOf("lo");
// i == 3

var u = s.toUpperCase();
// u == "HELLO WORLD"
```

Typeof Operator

- Der typeof Operator liefert den Datentypen einer Variable als String
- Mögliche Rückgabewerte
  - number
  - boolean
  - string
  - undefined
  - function
  - object

Typumwandlung

- JavaScript ist in Sachen Variablentypen nicht so strikt wie andere Programmiersprachen
- Eine Variable kann auch ihren Typ während des Programmablaufs ändern
- Formulareingaben liegen beispielsweise stets als Zeichenketten vor
- Aus diesem Grund kann es vorkommen, dass der Variablentyp explizit umgewandelt werden muss

```
var zahl1 = parseInt("47");  //liefert 47 als Zahl
var zahl12 = parseFloat("47.11");  //liefert 47.11 als Zahl
```

Arrays

- Definieren von Arrays

```
var myCars=new Array();
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";

var myCars=new Array("Saab","Volvo","BMW");

var myCars=["Saab","Volvo","BMW"];
```

- Zugriff auf Arrayelemente

```
var car = myCars[1];
```

Arrays

- Mit new Array() und anschliessender Zuweisung

```
var a = new Array();
a[0] = "Januar";
a[1] = "Februar";
a[2] = "März";
```

- Anlegen eines Arrays inkl. enthaltenen Werte

```
var a = new Array("Januar", "Februar", "März");
```

- Kurzform für Arrays

```
var a = ["Januar", "Februar", "März"];
```

Tipps zu Variablen

- Gross- und Kleinschreibung wird unterschieden
  - (Wie auch überall sonst in JavaScript)

- Bezeichner in camelCase (z.B. clickButtonEventListener)

- Geltungsbereich (Scope) von Variablen möglichst klein halten

- Variablen können den Typ ändern

```
var meinZaehler= 100;
meinZaehler= "Jetzt bin ich ein String";
```

If Anweisung

```
if (BEDINGUNG) {
  ...
}

if (BEDINGUNG) {
  ...
} else {
  ...
}

if (BEDINGUNG) {
  ...
} else if (BEDINGUNG) {
  ...
} else {
  ...
}
```

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Switch

```
var resultat = "";
switch (n) {

case 1:
  resultat = "eins";
  break;

case 2:
  resultat = "zwei";
  break;

default:
  resultat = "unbekannt";
}
```

While

```
while (BEDINGUNG) {
  ...
}
```

```
while (BEDINGUNG) {
  break; // verlässt die Schlaufe
}
```

```
while (BEDINGUNG) {
  continue; // überspringt die aktuelle Iteration
}
```

Do-while

- **continue** und **break** werden auch hier unterstützt

```
do {
  ...
} while (BEDINGUNG)
```

For - Schleife

```
for(i = 0; i < 10; i++){
  alert(i);
}
```

continue und **break** werden auch hier unterstützt

Math

```
var res1 = Math.floor(1.22);
// res1 == 1

var res2 = Math.ceil(1.22);
// res2 == 2

var res3 = Math.floor(Math.random()*10);
// e.g. res3 == 4

var res4 = Math.max(1,2,3);
// res4 == 3

var res5 = Math.sqrt(256);
// res5 == 16
```

Übungen

- Übung 1.4 selbständig lösen
- Übung 1.5 selbständig lösen
- Übung 1.6 selbständig lösen

Practice

Zugriff auf HTML Elemente

- Zugriff anhand der ID

```
var eingabeElement = document.getElementById("eingabe");
var ausgabeElement = document.getElementById("ausgabe");
```

- Lesen von Werten eines HTML Input Elementes

```
var value = eingabeElement.value;
```

- Schreiben von Werten eines HTML Input Elementes

```
eingabeElement.value = „Hello World“;
```

Übungen

- Übung 1.7 selbständig lösen
- Übung 1.8 selbständig lösen

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Rückblick