DANIEL MIKESCH

JAVASCRIPT 101

IF - ELSE - ELSE IF - SWITCH

```
//if-else
if(true) {
    console.log('FOO!'); //FOO!
    console.log('BAR!'); //BAR!
} else {
    console.log('GRR!'); //-> wird nicht ausgeführt
}


//if ohne { }
if(true) console.log('FOO!'); //FOO!

//if ohne { }
if(false) console.log('GRRR!'); //-> wird nicht ausgeführt


//if ohne { } sollte man nicht verwenden da es schnell mal passiert das
man übersieht dass ein if ohne { } nur für die erste nach dem if
stehende Anweisung gilt. (BAR! wird hier jeweils ausgegeben)

if(false)
    console.log(‘FOO!');
    console.log(‘BAR!');

if(false) console.log('FOO!'); console.log('BAR!');
```

```
//else ohne { } sollte man
//auch nicht verwenden
if(false) {
   console.log('FOO!');
} else console.log('BAR!');


//sonderfall else if
if(a) {
   //...
} else if(b) {
   //...
} else if(c) {
   //...
}


switch (expression) {
  case choice1:
    //...
    break;
  case choice2:
    //...
    break;
  default:
    //..
}
```

FOR - WHILE - DO - DO WHILE

```
for (initializer; exitCondition; finalExpression) {
  // code to run
}
```

```
while (exitCondition) {
  // code to run
}
```

```
do {
  // code to run
} while(exitCondition)
```

```
for (var propertyname in object) {
  console.log( object[propertyname] );
}
```

```
for (var value of iteratable) {
  console.log( value );
}
```

NUMBER - METHODES / FUNCTIONS

- parseInt(someString, [radix]) //Wandelt einen String in eine Ganzzahl um.
  optionaler Parameter radix gibt an welches Zahlensystem
  verwendet werden soll (binary => 2, hex => 16, dec => 10...)
  per default wird 10 / Decimal System verwendet
  ACHTUNG: Manche Browser verhalten sich nicht Standardkonform und verwenden bei Strings die mit "0" beginnen das Octalsystem (Basis 8) an!
- parseFloat(someString) //Wandelt einen String in eine Kommazahl um
- someNumber.toFixed([digits]); //Formatiert eine Zahl auf n Kommastellen
  Achtung: werden Kommastellen abgeschnitten wird gerundet 1.07.toFixed(1) == "1.1"

SPECIAL “NUMBERS”

- Number.MAX_VALUE & Number.MIN_VALUE
  Die höchste bzw niedrigste in JS darstellbare Zahl
- Number.MAX_SAFE_INTEGER & Number.MIN_SAFE_INTEGER
  Die höchste bzw niedrigste in JS darstellbare GANZE Zahl
- Infinity
  + oder - Unendlich z.B.: 10 / 0 ergibt Infinity
  Infinity + 1 === Infinity
  10 / Infinity === 0

NAN – NOT A NUMBER

- Ergebnis einer nicht durchführbaren mathematischen Operation. z.B.:
  Math.sqrt(-1) ergibt NaN
- Ergebnis einer fehlgeschlagenen Typkonvertierung z.B.:
  parseInt(“fooobar”) ergibt NaN
- ACHTUNG!!! Vergleiche mit NaN werden immer als false gewertet
  NaN > 5 //false NaN == NaN //false NaN != NaN //true
  NaN < 5 //false NaN === NaN //false NaN !== NaN //true
- Zur Überprüfung ob eine Variable den Wert NaN hat verwendet man die isNaN Funktion isNaN(NaN) //true

MATH – “HELPEROBJECT”

- Math.random()
  Eine zufällige (pseudorandom) Kommazahl zwischen 0 und 1 wobei 1 nie erreicht wird. Also 0 bis 0.999999999…
- Math.abs(number)
  Der Absolutwert einer Zahl |x| (Entfernt das Vorzeichen)
- Math.floor(number)
  Abrunden auf eine Ganze Zahl => Math.floor(1.9) === 1
- Math.ceil(number)
  Aufrunden auf eine Ganze Zahl => Math.ceil(1.1) === 2
- und viele mehr… sqrt, log, cos, sin, tan, log2, log10, …
  https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Math

STRING – ESCAPING

- Escaping von " und '
  '<img src="foo.jpg">' === "<img src=\"foo.jpg\">"
  ' '
- Escaping von \
  "\\" ergibt \ "\\\\" ergibt \\ "\'\\\'" ergibt '\'
- Singe Character Escape Sequences
  \" \' \\ \n \r \t \v \b \f \0
- Multiple Character Escape Sequences
  \udddd \xdd \ddd (d steht hier für Digit also 0-9)
  z.B.: "\u00A9" ergibt © "\uD83D\uDE0E" ergibt 😎
  (4byte Unicodezeichen werden in JS als 2 Zeichen behandelt 🙄)
- Will man einen String auf mehrere Zeilen aufteilen muss der Zeilenumbruch escaped werden
  "Hallo Welt - Test \
  123" /*Hallo Welt - Test 123*/
  "Hallo Welt - Test \n\
  123" /*Hallo Welt - Test 123*/

STRING - METHODEN

“mystring”.toUpperCase()	String in nur Großbuchstaben umwandeln
“mystring”.indexOf(“suchbegriff”, [fromIndex])	Begriff suchen. Returnt den index oder -1
“mystring”.substring(0, 4)	Ein Stück herausschneiden start, ende
“mystring”.trim()	Leerzeichen hinten und vorne wegschneiden
“mystring”.split(“,”)	String in ein Array umwandeln anhand von Trennzeichen
“mystring”.replace(“search”, “replace”)	Textpassage ersetzen (nur den ersten Treffer)
“mystring”.charAt(5)	gibt den 6ten Buchstaben zurück

ES6 – TEMPLATE STRINGS

- In ES6 gibt es zusätzlich zu ‘ und “ ein weiteres Anführungszeichen den Backtick `. Strings mit Backticks werden Template-Strings genannt und verfügen über mehr Features als gewöhnliche ES5 Strings:
- Zeilenumbrüche müssen nicht mehr escaped werden.
- Der String kann Platzhalter ${...} beinhalten

```
var name = “Tom”;
var s = “Hallo ” + name +  “!\n\
Test (” + (21 * 2) + “)”;
console.log(s);
Hallo Tom!
Test(42)
```

```
const name = “Tom”;
const s = `Hallo ${name}!
Test (${(21 * 2)})`;
console.log(s);
Hallo Tom!
Test(42)
```

- “Verschachteln” ist erlaubt: `abc ${`def`}`
- Mit “Tagged Strings” kann einem Template String Logik hinzugefügt werden
  => https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/template_strings

SELECTING DOM ELEMENTS

- document.getElementById(“id”) — returnt ein Element oder null
- document.getElementsByClassName(“classXy”) — returnt ein “Array” aus Elementen
- document.getElementsByTagName(“p”) — returnt ein “Array” aus Elementen

- document.querySelector(“.classXy a.classB”) — returnt das erste Element oder null
- document.querySelectorAll(“.classXy a.classB”) — returnt ein “Array” aus Elementen

Mit den neueren querySelector und querySelectorAll Methoden lässt sich alles bewerkstelligen. Funktionieren aber erst ab IE 9. Wenn man ALTE Browser unterstützten will muss man die “alten” getElementById und co verwenden.

CREATE AND MODIFY DOM ELEMENTS

- `var myElement = document.createElement("div");` — Erzeugt ein div (ist aber noch nicht im DOM eingehängt)
- `document.createTextNode("Fuu bar 123")` — Erzeugt ein Text - "Element"

- `myElement.setAttribute("class","classname")` — Setzt Attribute
- `myElement.appendChild(otherElement)` — Hängt ein Element als Kindelement unten an
- `myElement.prepend(otherElement)` — Hängt ein Element als erstes Kindelement ein
- `parentElement.removeChild(elementToBeRemoved)` — Entfernt ein Element
- `myElement.textContent = "Fuu bar 123";` — setzt den Textinhalt
- `myElement.addEventListener("click", function(_e){…});` — Eventhandler setzen
  (man sagt auch Callbackfunktion registrieren)

HINZUFÜGEN UND ENTFERNEN VON KLASSEN

- `myElement.classList.add("classname");` — Klasse zum class="" -Attribut des Elements hinzufügen
- `myElement.classList.remove("classname");` — Klasse entfernen
- `myElement.classList.contains("classname");` — Hat das Dom-Element die Klasse "classname" ? true / false
- `myElement.classList.toggle("classname");` — Klasse hinzufügen wenn sie noch nicht gesetzt ist oder entfernen wenn sie gesetzt ist.
  Gibt true zurück wenn die Klasse hinzugefügt wurde und false wenn die Klasse entfernt wurde

STYLE EIGENSCHAFTEN EINES DOM ELEMENTS MIT JS DIREKT SETZEN ("INLINESTYLE")

- `myElement.style.display = "flex";`