

WEBTECHNOLOGIEN
02 – JAVASCRIPT –
EINFÜHRUNG, VARIABLEN UND
KONTROLLSTRUKTUREN

PROF. DR. MARKUS HECKNER

WAS IST JAVASCRIPT?

- Cross-platform
- Objektorientiert
- Hohe Nutzung und hohe Verbreitung
Nutzung (99% der Clients haben JavaScript aktiviert, 90% aller Websites verwenden JavaScript)
- Läuft häufig im Browser
- Zunehmend auch auf dem Server
(Fokus dieses Kurses)



Quellen: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript
<https://developer.yahoo.com/blogs/ymn/many-users-javascript-disabled-14121.html>

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Skalare Datentypen speichern einen Wert, verfügen aber über keine Methoden (d.h. sind keine Objekte) – Aber mit String, Number und Boolean existieren Wrapperobjekte (später)
- Keyword **let** vs **var**
 - **let** – Blockscope innerhalb einer Funktion, d.h. unsichtbar in geschweiften Klammern
 - **var** – Global in der Funktion
 - **let** ist moderner, wir verwenden **let** (auch wenn man oft **var** in Codebeispielen findet)

Vgl. z.B. <http://typedarray.org/JavaScript-refresh/#primitive-composite> oder
https://developer.mozilla.org/de/docs/Glossary/einfache_datenelemente

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Im Gegensatz zu C, Java oder C++, keine Unterscheidung der Datentypen (double, int, ...) bei der Definition / Initialisierung der Variablen
- Typ **number** wird für alle Ganzzahlen und Dezimalzahlen verwendet
- Da kein Konzept int und double, kein type-casting

```
let a = 4;  
let b = 6.9;  
let c = -5;
```

Operator, der Zeichenkette zurückgibt, die den Typ des ausgewerteten Operanden beschreibt...

```
// outputs: 1.3333333333333333  
console.log(a / 3);  
// outputs: number number number  
console.log ( typeof a, typeof b, typeof c );
```

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Umwandlung einer Fließkommazahl in eine Ganzzahl durch Nutzung der Methode floor (= abrunden) des Math Objekts
- Math ist ein globales Objekt, das überall in JavaScript genutzt werden kann

```
let posNum = 45.95;  
let negNum = -45.95;
```

```
// outputs: 45  
console.log(Math.floor(posNum));  
// outputs: -46  
console.log(Math.floor(negNum));
```

?

Welche Ausgaben erzeugt der Code?

!

Liste global verfügbarer Objekte:
https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Vergleich von Strings erfolgt auf **lexikalischer** Basis mit ==
- Konkatenierung erfolgt mittels +

```
let firstNameOne = "Bill";
let firstNameTwo = "Bill";
let lastName = "Gates";

let completeName = firstNameOne + " " +
    lastName;
//outputs: true
console.log(firstNameOne == firstNameTwo);
//outputs: Bill Gates
console.log(completeName);
```

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Bei Zugriff auf Eigenschaft oder Methode eines Strings mit `.-Operator` kann der String wie ein Objekt verwendet werden...
- Strings sind dann Objekte (wie in Java...) und bieten Methoden zum Zugriff auf die Daten des Objekts an...
- Iterieren eines Strings durch Abfrage der Länge und Zugriff über Indizes

```
let lastName = "Gates";  
for (let i = 0; i < lastName.length; i++) {  
  console.log(lastName.charAt(i));  
}
```

"G""a""t""e""s"

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Strings lassen sich auch mit „Number“ konkatenieren
- Erzeugen eines Strings aus Zahlen durch Voranstellen eines leeren Strings möglich

```
let firstString = 1 + 2 + 3 + "hello";  
let secondString = "hello" + 1 + 2 + 3;  
let thisAlsoWorks = "" + 1 + 2 + 3;
```

```
console.log(firstString);    // outputs: 6hello  
console.log(secondString);  // outputs: hello123  
console.log(thisAlsoWorks); // outputs: 123
```

?

Welche Ausgaben erzeugt der Code?

STRINGS IN NUMBER KONVERTIEREN

```
let number = "3";  
//outputs: "string"  
console.log(typeof number);  
  
number = Number(number);  
//outputs: "number"  
console.log(typeof number);
```



Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Achtung: Vergleich mit `==` führt zu automatischer Typkonvertierung
- `===` unterbindet Typkonvertierung

```
let a = true;  
let b = "true";  
let c = 1;  
let d = false;
```

```
console.log ( a == true ); // outputs: true  
console.log ( b == true ); // outputs: false  
console.log ( c == true ); // outputs: true  
console.log ( d == true ); // outputs: false  
console.log ( c === true ); // outputs: false
```

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- Variablen, denen noch kein Wert zugewiesen haben, haben den Wert **undefined**
- Was passiert in C? Was passiert in Java?

```
let firstNameOne = "Bill";  
let lastName;
```

```
console.log(firstNameOne); //outputs: Bill  
console.log(lastName);     //outputs: undefined
```

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Primitive
Datentypen

number

string

boolean

undefined

null

- null repräsentiert das absichtliche Fehlen eines Werts
- null ist Schlüsselwort in JavaScript und wird ohne Anführungszeichen geschrieben

```
let userInput = null;

if (userInput === null) {
  console.log("ARGHL ...");    //outputs: ARGHL...
}
```

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Komplexe
Datentypen

Array

- Nicht gesetzte Indizes in einem Array werden als **undefined** zurückgegeben

```
let shoppingItems = [];  
shoppingItems[0] = "Orangensaft";  
shoppingItems[1] = "Tomaten";  
shoppingItems[2] = "Pizza";  
shoppingItems[4] = "Nudel";
```

```
for(let i = 0; i < shoppingItems.length; i++){  
  console.log(shoppingItems[i]);  
}
```

//Outputs: "Orangensaft", "Tomaten", "Pizza",
undefined, "Nudel"

?

Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Komplexe
Datentypen

Array

- Arrays lassen sich dynamisch mit der Methode **push** verlängern

```
let shoppingItems = [];  
shoppingItems.push("Orangensaft");  
shoppingItems.push("Cola");
```

```
for(let i = 0; i < shoppingItems.length; i++){  
  console.log(shoppingItems[i]);  
}
```

//Outputs: "Orangensaft", "Cola"

?

Welche Ausgaben erzeugt der Code?

Kontroll-
strukturen

for

while

if - else

switch
case



Buggefahr!!

In JavaScript nur **let**, kein **int**!

```
for (let step = 0; step < 5; step++) {  
  // Runs 5 times, with values of step 0 through 4.  
  console.log('Walking east one step');  
}
```

Kontroll-
strukturen

for

while

if - else

switch
case

Alternatives Iterieren über eine Liste:

```
let shoppingItems = [];  
shoppingItems[0] = "Orangensaft";  
shoppingItems[1] = "Tomaten";  
shoppingItems[2] = "Pizza";
```

Iteriert über Namen
der Elemente
(Ausgabe: 0, 1, 2)

```
for (let shoppingItem in shoppingItems) {  
  console.log(shoppingItem);  
}
```

Iteriert über Werte der Elemente
(Ausgabe: "Orangensaft", ...)

```
for (let shoppingItem of shoppingItems) {  
  console.log(shoppingItem);  
}
```


Kontroll-
strukturen

for

while

if - else

switch
case

```
while (x < 10) {  
    x++;  
}
```

Quelle: Mozilla Developer Network (2015)

Kontroll-
strukturen

for

while

if - else

switch
case

```
if (condition) {  
    //statement 1;  
} else {  
    //statement 2;  
}
```

Quelle: Mozilla Developer Network (2015)

PROMPT UND MODULO

Öffnet Popup für
Nutzereingaben

```
let number = prompt("Enter a whole number: ");

if (number % 2 === 0) {
  console.log("Number is even!");
} else if (number % 2 === 1) {
  console.log("Number is odd!");
} else {
  console.log("Hey, I asked for a whole number!");
}
```



number ist eigentlich ein String, aber implizite
Typkonvertierung, d.h. JS versucht den String
automatisch als **Number** zu interpretieren

Kontroll-
strukturen

for

while

if - else

switch
case

- Welche Ausgaben erzeugt der Code für die folgenden Werte von **fruittype**?
 - Apples
 - Mangoes
 - Kiwi

```
let fruittypes = ["Apples", "Mangoes", "Kiwi"];
let fruittype = fruittypes[0];

switch (fruittype) {
  case "Apples":
    console.log("Apples are $0.32 a pound.");
    break;
  case "Cherries":
    console.log("Cherries are $3.00 a pound.");
    break;
  case "Mangoes":
    console.log("Mangoes are $0.56 a pound.");
    break;
  default:
    console.log("Sorry, we are out of " + fruittype + ".");
}
console.log("Is there anything else you'd like?");
```

FUNKTIONEN OHNE RÜCKGABEWERTE

Kein Modifikator für
Sichtbarkeit

Kein Rückgabetyt!!!

Kein Datentyp für
Parameter!

```
var errorMsg = "DB connection failed";  
logError(errorMsg);  
  
function logError(errorMsg) {  
    console.log("Error: " + errorMsg);  
}
```

Ausgabe: "Error: DB connection failed"

?

Welche Ausgabe erzeugt der Code?

FUNKTIONEN MIT RÜCKGABEWERTEN

Kein Modifikator für
Sichtbarkeit

Kein Rückgabebetyp!!!

Kein Datentyp für
Parameter!

```
var numToSquare = 3;  
var squareResult = square(numToSquare);  
console.log(squareResult);
```

```
function square(numToSquare) {  
    return numToSquare * numToSquare;  
}
```

Ausgabe: 9

?

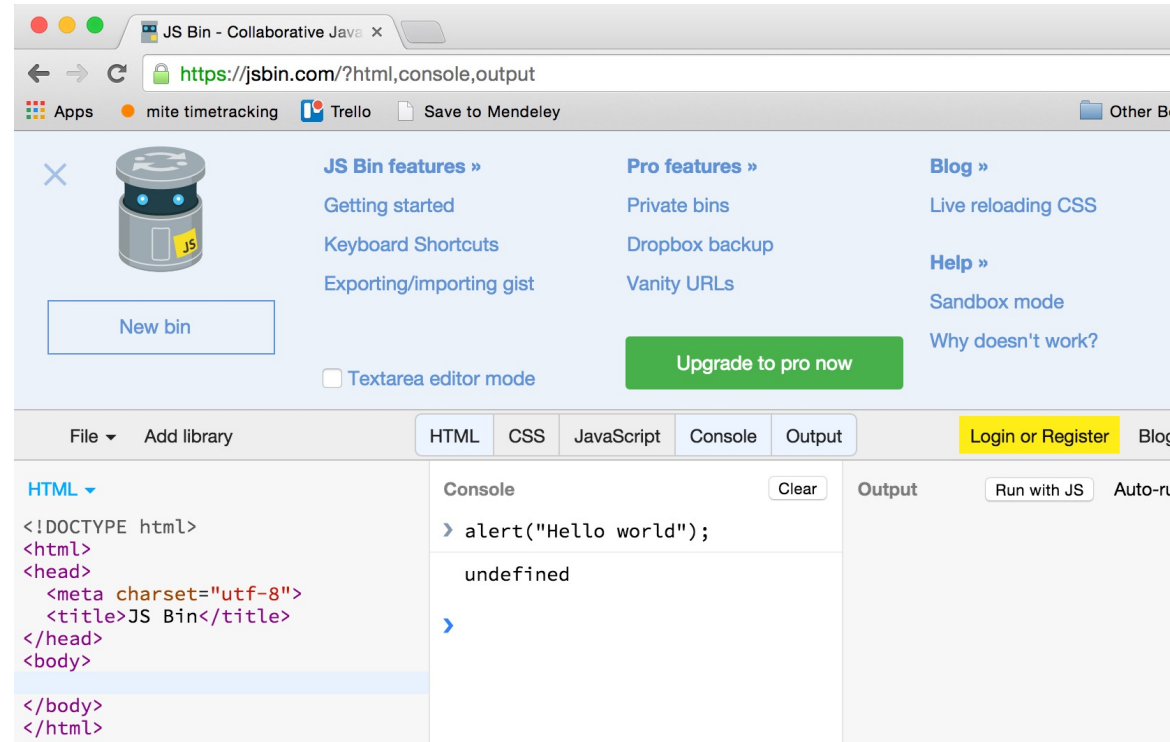
Welche Ausgabe erzeugt der Code?

KOMMENTARE

```
// I am a single line comment
```

```
/* I am a  
    multiline comment  
*/
```

LAB: ERSTE VERSUCHE MIT JSBIN.COM



Webseite mit der sich JavaScript Code für schnelle Tests ausführen lässt

QUELLEN

- Imbert, T. (2013). A JavaScript Refresh. Online verfügbar:
<http://typedarray.org/JavaScript-refresh/>. Letzter Zugriff: 11.08.2015.
- Mozilla Developer Network. (2015b). A re-introduction to JavaScript (JS tutorial).
Online verfügbar: https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript
- Mozilla Developer Network. (2015b). Introduction to Object Oriented JavaScript.
Online verfügbar: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript#JavaScript_object_oriented_programming. Letzter Zugriff: 13.08.2015