EINFÜHRUNG IN JAVASCRIPT

CHRISTINE KOPPELT

17. APRIL 2016

ABOUT ME

- Softwareentwicklerin (Java, Python, JavaScript, Scala)
- Co-Organisatorin node.js User Group München

WAS SIND EINSATZMOGLICHKEITEN VON JAVASCRIPT?


MIN
SEC
5
00

THEMEN FÜR DEN WORKSHOP

- Grundlagen der Sprache (Spracheigenschaften, Datentypen, Syntax)
- weitere Spracheigenschaften
- Modularisierung
- Testen & Codeanalyse (mocha.js, ESLint)
- Buildskripte & Paketmanagement mit npm
- IDEs und Editoren

JAVASCRIPT - DIE GRUNDLAGEN

ERSCHEINUNGSJAHR

1990

1995

2000

INTERPRETIERT ODER KOMPILIERT?

TYPISIERUNG

	dynamisch	statisch
schwach	?	?
stark	?	?

ÜBERBLICK

- Ursprünglich als Sprache für dynamische Webseiten entwickelt
- Standardisiert als ECMAScript
  - ES5 (2009)
  - ES6 (2015)
  - Transpiler
- Mehrere Implementierungen
  - Google, Mozilla, Apple, Microsoft, Oracle

5 PRIMITIVE DATENTYPEN

number, string, boolean, null, undefined

```
aNumber = 1.3,
anotherNumber = 10,
aString = 'Hello world',
theTruth = true,
nothing = null,
evenLess = undefined;
```

ALLES ANDERE SIND OBJEKTE

... inklusive Funktionen und Arrays
... optional: Attribute und Methoden

OBJEKT INITIALISIERER

```
var myLaptop = {manufacturer: "Lenovo", year: 2013, cpu: {model: "i5", freq: "2.6GHz"}};
```

VERWENDUNG EINER KONSTRUKTOR FUNKTION

```
function Laptop(manufacturer, year, cpu) {
  this.manufacturer = manufacturer;
  this.year = year;
  this.cpu = cpu;
  this.logInfo = function (){
    console.log("Manufacturer: " + manufacturer + " Year: " + year);
  }
}

var myLaptop = new Laptop("Lenovo", 2013, {model: "i5", freq: "2.6GHz"});
```

ZUGRIFF AUF ATTRIBUTE

```
myLaptop.manufacturer
myLaptop["manufacturer"]
```

ARRAY

```
var myArray = ["a text", null, 23, false];
myArray[2];
```

```
var myArray = ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango'];
var citrus = myArray.slice(1, 3);

// citrus contains ['Orange','Lemon']
```

VARIABLEN DEKLARIEREN

VAR

```
var declaringVariables = function() {
    var aLocalVariable = 1;
    aGlobalVariable = 2;
};

function varVisibility() {
    //i *is* visible out here
    for( var i = 0; i < 5; i++ ) {
        //i is visible to the whole function
    };
    //i *is* visible out here
};
```

LET

```
function letVisibility() {
    //j is *not* visible out here
    for( let j = 0; j < 5; j++ ) {
        //j is only visible in here (and in the for() parentheses)
    };
    //j is *not* visible out here
};

function myFunction() {
    let string1 = 'string1'; //function block scoped
    var string2 = 'string2'; //function block scoped
};
```

VARIABLEN DEKLARIEREN

CONST

Repräsentiert eine konstante Referenz auf einen Wert, nicht einen konstanten Wert

```
const names = [];
names.push( "Jordan" );
console.log( names );
names = [1,2];  // Error!
```

THIS

- In Konstruktoren und Methoden verhält sich `this` wie erwartet

```
class Square {
  constructor(width) {
      this.width = width;
  }
  area() {
      return this.width * this.width;
  }
}
var square = new Square(5);
square.area();
```

- Ausserhalb von Funktionen ist `this` das globale Objekt (`window` im Browser)
- In echten Funktionen, in Callbacks, in geschachelten Funktionen ist das Verhalten oft unerwartet

IMPLIZITE TYPKONVERTIERUNG

```
"12345" * 1 === 12345  // true
[] + [] === "" //true
{} + [] === 0 // true
```

Details: http://www.2ality.com/2013/04/quirk-implicit-conversion.html

== VS ===

===: STRICT EQUALITY, IDENTITY

- Prüft beide Operanden auf exakte Gleichheit
- Sowohl Typ als auch Wert müssen gleich sein

==: ABSTRACT EQUALITY

- Bei ungleichen Typen findet ein Konvertierung statt

```
1 == 1;    // true
"1" == 1;  // true
1 === 1;   // true
"1" === 1; // false
```

UNTERSCHIED ZWISCHEN UNDEFINED UND NULL

`undefined`: Wenn eine Variable nicht initialisiert wurde

```
var x; // undefined
x === undefined // true
```

`null`: Bei expliziter Zuweisung von "nichts"

```
var y = null; // undefined
y === undefined // false
```

FUNKTIONEN

FUNKTIONSDEKLARATION

```
function myOtherFunction (x) { console.log(x); }
myOtherFunction("Hallo Welt");
```

FUNKTIONSAUSDRUCK

```
var myFunction = function (x){ console.log(x); }
myFunction("Hallo Welt");
```

ARROW FUNCTIONS

```
var filtered = myArray.filter(function(x) {return x%2 == 0;});
```

vs.

```
var filtered = myArray.filter(x => x%2 == 0);
```

DEFAULT PARAMETERS

```
function multiply(a, b = 1) {
  return a*b;
}

multiply(5); // 5
```

NAMENSKONVENTIONEN

- Dateinamen: kleingeschrieben und enden mit `.js`
- Namen sollen aus [a-zA-Z0-9_]+ bestehen
  - Camelcase is gängig
- Variablen und Funktionen starten mit Kleinbuchstaben
- Klassennamen starten mit Großbuchstaben
- Globale Variablen in GROSSBUCHSTABEN
- Berühmte Ausnahmen: jQuery ($), lodash (_)

KOMMENTARE

```
// Einzeiliger Kommentar
var x = 5; // Beschreibung des Codes
/*
Mehrzeiliger
Kommentar
*/
```

SCHLEIFEN

```
var n = 0;
while (n < 3) {
  n++;
}
```

```
for (var i = 0; i < 9; i++) {
   console.log(i);
}
```

```
let iterable = "netzwerk";
for (let value of iterable) {
  console.log(value);
}
```

ARRAYS

```
[2, 5, 9, 123, 42, 47].forEach(x => console.log(x));
```

```
["Peter", "Bob", "Mary"].map(name => "Hello " + name);
```

CONSOLE

- Zugriff auf die Debugging-Konsole des Browsers
- Funktioniert im IE8/9 nur bei geöffneten Entwicklertools
- Logging

```
console.log
console.warn
```

JSON (JAVASCRIPT OBJECT NOTATION)

- Datenformat zum Datenaustausch zwischen Anwendungen
- Unterstützte Datentypen: Nullwert, Boolean, Zahl, Zeichenkette, Array, Objekt
- Datentypen können beliebig tief verschachtelt werden
- JavaScript Objekte können ohne aufwändige Konfiguration von/nach JSON konvertiert werden

```
{
   "firstName": "Sandy",
   "lastName": "Smith",
   "isCustomer": true,
   "age": 25,
   "phoneNumbers": [{
      "type": "home",
      "number": "212 5551234"
   }, {
      "type": "office",
       "number": "646 5554567"
   }]
}

JSON.stringify(myObject)
JSON.parse (jsonstring);
```

ÜBUNGEN - SETUP

- Installiere node.js: https://nodejs.org
- Git Repository auschecken oder als zip herunterladen: https://github.com/cko/ost2016-javascript
- `npm install` aufrufen
- Tests aufrufen: `npm test`

ÜBUNGEN - MATERIAL

- MDN JavaScript Reference
- DevDocs

ÜBUNG 1

- Implementiere die Funktion `fizzbuzz`. Diese bekommt ein Array von Zahlen übergeben. Zahlen die durch 3 teilbar sind sind durch 'fizz', Zahlen die durch 5 teilbar sind durch 'buzz' und Zahlen die durch 3 und 5 teilbar sind durch 'fizzbuzz' zu ersetzen.

ÜBUNG2

- Implementiere die Funkktion `getMyFavoriteBooks` die eine Liste von Objekten mit den folgenden Attributen zurückgibt:
  - author
  - title
  - year

ÜBUNG3

- Implementiere die Funktion `getPriceOfProduct` die ein Objekt als JSON-String übergeben bekommt und von diesem das Attribut `price` zurückliefert.

FAZIT

Was habt ihr in diesem Kapitel gelernt?