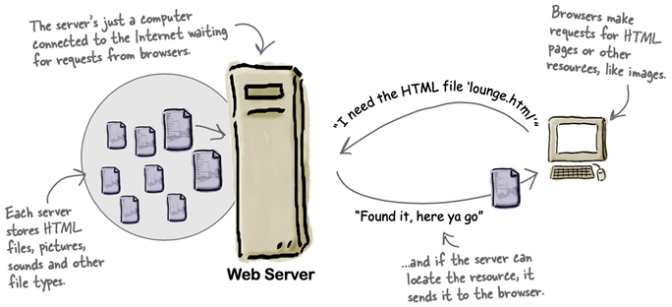
Web-Entwicklung

Jil Zerndt, Lucien Perret
December 2024

Intro

WEB-Architektur

- Client-Server (Browser-Webserver)
- Interaktion: Request/Response



Technologien

Client-Seitig

- Beschränkt auf das, was der Browser kann
- HTML + CSS + JavaScript + noch ein paar Sachen
- → Front-end Entwickler

Server-Seitig

- Praktisch unbeschränkt: Plattform, Programmiersprache, ...
- Erzeugt und gesendet wird das, was der Browser kann
- → Back-end Entwickler

JavaScript

JS-Grundlagen

Web-Konsole In JS mit dem Keyword `console`:

- `console.log(message)`: Loggt eine Nachricht
- `console.clear()`: Löscht die Konsole
- `console.trace(message)`: Stack trace ausgeben
- `console.error(message)`: stderr ausgeben
- `console.time()`: Startet einen Timer
- `console.timeEnd()`: Stoppt den Timer

Website für Konsolen-API: `https://nodejs.org/api/console.html`

Datentypen

Bekannte Datentypen wie bei Java, spezielle Datentypen:

- `undefined`: Variable wurde deklariert, aber nicht initialisiert
- `null`: Variable wurde deklariert und initialisiert, aber nicht belegt
- `Symbol`: Eindeutiger, unveränderlicher Wert
- `Number`: Ganze Zahlen, Fließkommazahlen, NaN, Infinity
  - Infinity: $1/0$, $-1/0$
  - NaN: $0/0$, $\sqrt{-1}$
- `BigInt`: Ganze Zahlen beliebiger Größe
- `Object`: Sammlung von Schlüssel-Wert-Paaren
- `Function`: Funktionen sind Objekte

mit dem Keyword `typeof` kann der Datentyp zurückgegeben werden

typeof 12	//	'number'
typeof(12)	// 'number'	
typeof 2 n	//	'bigint'
typeof Infinity	// 'number'	
typeof NaN	// 'number' !!	
typeof 'number'	// 'string'	

Variablen Definition

- `var`: Global oder lokal
- `let`: Nur lokal
- `const`: Konstante

Variablenbindung

- `var`: Funktionsbindung
- `let`: Blockbindung
- `const`: Blockbindung

```
let width = 10
console.log(width * width)          // → 100

let answer = true, next = false
let novalue
console.log(novalue)                // → undefined
```

Operatoren

- Arithmetische Operatoren: $+, -, *, /, \%, ++, --$
- Zuweisungsoperatoren: $=, += , -= , *= , /= , \%= , **= , <<= , >>=, >>>=, \&= , ^= , |=$
- Vergleichsoperatoren: $==, ===, !=, !==, >, <, >=, <=$
- Logische Operatoren: &&, ||, !
- Bitweise Operatoren: &, |, ^ , <<, >>, >>>
- Sonstige Operatoren: `typeof`, `instanceof`

Vergleich mit == und ===

- `==`: Vergleicht Werte, konvertiert Datentypen
- `===`: Vergleicht Werte und Datentypen ohne Konvertierung

ebenfalls: `!=` und `!==`

Verzweigungen, Wiederholung und Switch Case

- `if (condition) {...} else {...}`
- `switch (expression) { case x: ... break; default: ... }`
- `for (initialization; condition; increment) {...}`
- `while (condition) {...}`
- `do {...} while (condition)`
- `for (let x of iterable) {...}`

```
switch (<ausdruck>) {
    case <wert1>:
    break
    default:
        ...
        break
}

if (<ausdruck>) \{
\} else \{

}

for (let i=1; i<50; i*=2) {
    console.log(i)
}
4 // -> 1, -> 2, -> 4, > 8, > 16, -> 32
```

Funktionsdefinition

- `function name(parameters) {...}`
- `const name = (parameters) => {...}`
- `const name = parameters => {...}`
- `const name = parameters => expression`

```
1 const square = function (x) {
    return x * x
}
4
5 console.log(square(12)) // -> 144
```

```
1 const square1 $=(x) \Rightarrow$ \{return $x * x$ \}
2 const square2 $=x=>x^{*} x$
```

```
1 // Beispiel einer Funktion
2 function add(a, b) {
3 return a + b;
4 }
5 // Beispiel einer Arrow-Funktion
6 const add = (a, b) => a + b;
```

Objekte und Arrays

Objekt vs Array

Was	Objekt	Array
Art	Attribut-Wert-Paare	Sequenz vo
Literalnotation	werte = { a: 1, b: 2}	liste = [1, 2
Ohne Inhalt	werte = {}	liste = []
Elementzugriff	werte[ä"] oder werte.a	liste[0]

Objekte

- Objekt Attribute sind dynamisch und können einfach erweitert werden:
- Objekt Attribute können auch einfach mit dem delete keyword entfernt werden.
- Mit in kann überprüft werden, ob ein Attribut existiert

```
let person = {
    name: "John Baker",
    age: 23,
    "exam results": [5.5, 5.0, 5.0, 6.0, 4.5]
}

    let obj = { message: "not yet implemented" }
    obj.ready = false
> obj
{ message: 'not yet implemented', ready: false }
> obj.attr
undefined

> let obj = { message: "ready", ready: true, tasks: 3 }
> delete obj.message
> obj.tasks = undefined
> obj
{ ready: true, tasks: undefined }
> "message" in obj
false
> "tasks" in obj
true
```

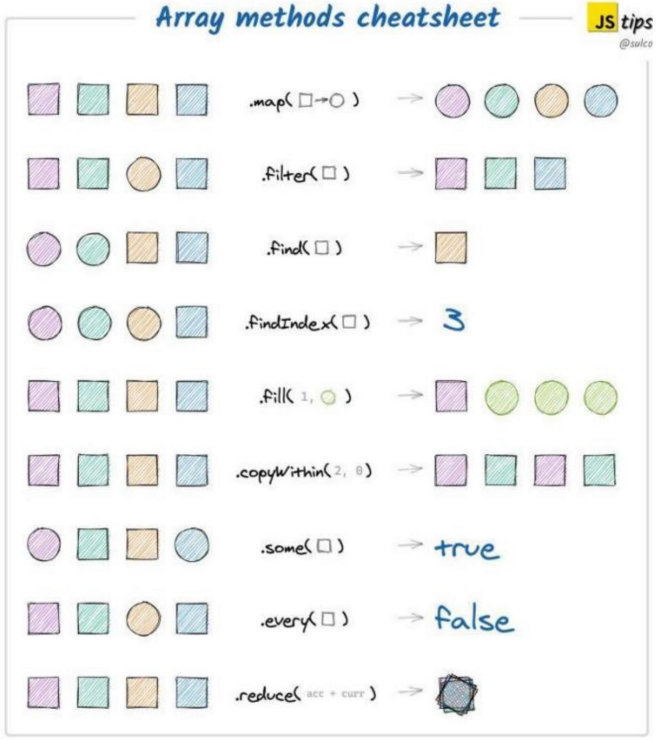
Methoden Ein Objekt kann auch Methoden enthalten:

```
> let cat = { type: "cat", sayHello: () => "Meow" }
> cat.sayHello
[Function: sayHello]
    > cat.sayHello()
'Meow'
```

Arrays

Verschiedene Hilfsfunktionen:

- Array.isArray()
- .push()
- .pop()
- Indexof, lastIndexOf
- Concat
- slice
- Shift, unshift
- .forEach(item => ....)
- .filter(item => .....)
- .map(item => ...)




```
// einfachere Variante für Arrays
for (let entry of myArray) {
  doSomethingWith(entry)
}
```