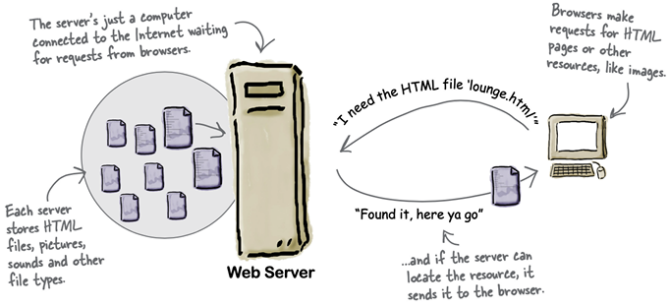
Web-Entwicklung

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

Variablen Definition

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

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) {...}`

Funktionsdefinition

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

```
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-Paar	Sequenz von Werten
Literalnotation	Werte =	Werte =

Funktionen und funktionale Programmierung

Prototypen von Objekten

Asynchrone Programmierung

Webserver

Browser-Technologien

JavaScript und DOM

Client-Server-Interaktion

UI Bibliothek

UI Komponenten

UI Implementierung

UI Einsatz

Wrap-up