Blockwoche: Web Programming Lab 🚀

Übung «Architektur Technologie-Radar»

Ihr dürft einen Technologie-Radar umsetzen. Folgende Anforderungen sind bekannt:

- Technologien können in einem geschützten Bereich erfasst, geändert, publiziert und gelöscht werden.
- Publizierte Technologien erscheinen in der Technologie-Radar Ansicht.
- Besucher des Technologie-Radars sehen die publizierten Technologien und können diese kommentieren.
- Die Technologien auf dem Radar müssen schnell geladen werden.
- Die Verwaltung der Technologien muss sicher sein. D.h. nur bestimmte Benutzer (Rolle CTO / Tech-Lead) dürfen die Technologien verwalten.
- Der Technologie-Radar soll auf allen Bildschirmgrössen lesbar sein.

Aufgaben (15') Link auf das Mural-Board

- Identifiziert mögliche nicht funktionale Anforderungen und beschreibt exemplarisch 1 – 2 Qualitätsszenarien
- Optional: Diskutiert mögliche bekannte Architekturansätze und Technologie-Einsätze des Technologie-Radars
- 2-3er Gruppen, 1 – 2 Vorstellungen der Gruppe

Lösungsvorschlag - Qualitätsziele

Merkmal Performance: „Die Technologien müssen schnell geladen werden”

- Der Leser der Technologien auf dem Technologie-Radar empfindet das Laden und Navigieren durch die Technologie-Einträge auch bei einer 3G-Verbindung als flüssig. Das Laden und Rendern eines Technologie-Steckbriefs in einem modernen Browser ist bei einer 3G Verbindung unter 1 Sekunde.

Mögliche Architekturentscheide:

- Z.B. Trennung CMS- und Auslieferungsplattform
- Z.B. Assets auf Content-Delivery-Network, Caching-Mechanismen, etc.
- Z.B. Einsatz JAMStack – Technologien in Markdown, Auslieferung von vorgeneriertem HTML

Merkmal Sicherheit :“Die Verwaltung der Technologie-Beiträge muss sicher sein.“

- Attacken von einem Angreifer, wie z.B. eine Brute-Force Attacke in der Technologie-Radar-Administration, werden geloggt und der Angreifer entsprechend blockiert.
- Werden Technologie-Steckbriefe durch einen Angreifer verändert, werden diese Veränderungen geloggt und können innerhalb von einem Tag wiederhergestellt werden.

Mögliche Architekturentscheide:

- Z.B. Einsatz einer Web Application Firewall; Einsatz eines Captchas, Einsatz von einem IAM-System

Lösungsvorschlag - Qualitätsziele

Merkmal Benutzerfreundlichkeit: „Der Technologie-Radar soll auf allen Bildschirmgrössen lesbar sein.“

Der Besucher kann die Technologien auf seinem iPhone (Mobile) und iPad (Tablet) lesen (Um ganz explizit zu sein: Grössen resp. Viewports spezifizieren). Die Technologie-Einträge passen sich der entsprechenden Bildschirmgrösse an und das Blog-System ist bedienbar.

Mögliche Architekturentscheide:

- Z.B. Einsatz Responsive UI-Framework, Gezielter Einsatz CSS Media-Queries

JavaScript Sprachkonzepte I

Eigenschaften, History und Versionen, Setup Dev Umgebung

Übungsfragen im Plenum (5’)

1. Auf welchen Plattformen wird JavaScript verwendet resp. kann verwendet werden?

   - Client – Webbrowser (Chrome)
   - Backend Applikationen (NodeJS)
   - Mobile Applikationen (Ionic)
   - Desktop Applikationen (Electron)
   - Datenbanken (CouchDB)
   - IaC (Pulumi)

2. Nenne mindestens drei Eigenschaften von JavaScript und erkläre diese.

   High-Level (nicht auf die darunterliegende Plattform kümmern), dynamisch (verschiedene sachen zur laufzeit), schwach typisiert, mehrere pardigmen, interpretiert (braucht kein kompilierungsschritt)

3. Finde heraus, welche Firmen die ECMAScript Spezifikation massgäblich mitprägen (Chairgroup).

   https://www.ecma-international.org/technical-committees/tc39/

Let’s dive into code….

Setup (10’)

- 👉 https://github.com/web-programming-lab/javascript-sprachkonzepte

JavaScript Sprachkonzepte II

Variabeln, primitive Datentypen, Operatoren

Übung I (ca. 5’)

1. Was ist der Unterschied zwischen `let` und `var`?

   let = blockscoped, var = functionscoped

2. Für was wird `typeof()` verwendet?

   Gibt den Typ von einer Variable zurück

3. Was bedeutet schwach typisiert?

   Typ kann zu einer Laufzeit ändern resp. nicht initial an den Typ gebunden

4. Was ist der Unterschied zwischen == und ===?

   == lose gleichheit: beide wurden in den gleichen Typ konvertiert, nach der Konvertierung wird der final vergleich mit === ausgeführt
   === strikte gleichheit: keiner der werte wird zur laufzeit implizit konvertiert, falls Datentypen ungleich sind ist auch der verlgeich false

5. Was ist der Unterschied zwischen Primitives und Non-Primitives?

   Primitives = immutable; String, Number, boolean, BigInt, String, Symbol
   Non-primitives – not immutable; Object
   Non-primitives = referenzen

Übung II (5’)

```
console.log('' + 1 + 0); // ?
console.log('' - 1 + 0 ); // ?
console.log(true + false); // ?
console.log(6 / '3'); // ?
console.log('2' * '3' ); // ?
console.log(7 / 0); // ?
console.log(null + 1); // ?
console.log(undefined + 1); // ?
console.log(21 == '21'); // ?
console.log(undefined == null); // ?
console.log(undefined === null); // ?
console.log(21 === 21); // ?
```

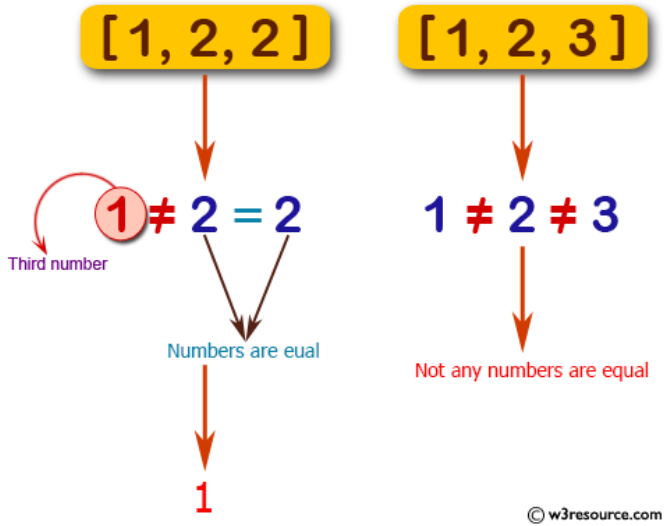
exercise-equality.js

Undefined becomes NaN after a numer conversion

`console.log("" + 1 + 0);` →10
`console.log("" - 1 + 0 );` →-1 (string konktatenation und convertierung in string funktioniert nur mit dem plus operator)
`console.log(true + false);` → 1
`console.log(6 / "3"); // ?` → 2
`console.log("2" * "3" );` → 6
`console.log(7 / 0); //` → Infinity
`console.log(null + 1);` →1 (null wird zu number konvertiert, gemäss Spez: 0)
`console.log(undefined + 1); // ?` → NaN (undefined wird zu NaN) →
Achtung Prüfung nicht mit === sondern via isNaN
console.log(21 == ‘21’) → True
Console.log(undefined == null) →True → keine saubere Erklärung, für mich: beide Werte beschreiben die Absenz von einem Wert, daher true
Console.log(undefined === null) → False
Console.log(21 === 21) → True

Übung III (10’)

1. Gegeben sind drei Nummern. Davon sind zwei gleich. Finde die Dritte.



https://gist.github.com/bitsmuggler/7fe7a2d8cbe69177a689cba18ac28c01

2. Erweitern Sie die Übung so, dass lediglich Ganzzahlen eingegeben werden können (Rückgabe ‘null’, falls keine Ganzzahlen).

exercises / exercise-number.js

JavaScript Sprachkonzepte III

Objects, Arrays, Functions, Prototype Chains, Classes

Übung I (10’)

```
// Vor dem Funktionsaufruf
let menu = { width: 200, height: 300, title: "Titel" };
multiProperties(menu);
// Nach dem Funktionsaufruf
menu = { width: 400, height: 600, title: "Titel" };
```

Schreibe eine Funktion, welche alle number Properties um 2 multipliziert (generisch!).

Übung: exercises/exercise-multiproperties.js

→ Beachte, dass die Funktion nichts zurückgibt!

Lösung:
https://gist.github.com/bitsmuggler/04ffd0035cfe85fb0dbda657aa2b8b4a

Übung II (10’)

```
const salaries = {
    patrick: 100000,
    andreas: 110000,
    gwendolin: 91000,
    nayoona: 45000
 };
```

Schreibe eine Funktion, welche alle Löhne des Objektes ‘salaries’ aufsummiert (generisch!).

Übung: exercises/exercise-salaries.js

Lösung:
https://gist.github.com/bitsmuggler/02ccf7fb5f5995c1c5b80b4aab03eede

Zusatz: Übungsfrage

```
const name = 'Patrick';
console.log( `hello ${1}` ); // ?
console.log( `hello ${"name"}` ); // ?
console.log( `hello ${name}` ); // ?

// Funktioniert das? Warum?
const user = { name: "John" };
user.name = "Pete";
```

- Hello 1
- Hello name
- Hello Patrick

- Ja, die Referenz darf aber nicht geändert werden.

JavaScript Sprachkonzepte IV

Prototype Chains und “this” Keyword

Übungsaufgabe I

exercise-prototype.js

```
let user = {
    doSth() {
        if (!this.isPausing) {
            console.log('I am doing sth.');
        }
    },
    pause() {
          this.isPausing = true;
    }
};

let admin = {

     name: 'Admin User',
     __proto__: user

};


 admin.pause();
 console.log(admin.isPausing); //? Wieso?
 console.log(user.isPausing); //? Wieso?
```

```
console.log(admin.isPausing); //true
console.log(user.isPausing); //undefined (no such property in the prototype)
```

Übungsaufgabe II - Erstelle einen Calculator (20')

- Erstelle ein Calculator mit folgenden drei Methoden:
  > read() → einlesen von 2 Werten und speichere diese als Objekt Properties
  > sum() → Summierung 2 gespeicherten Werte
  > mul() → multipliziere 2 gespeicherten Werte
- Erweitere den Rechner so, dass nicht nur 2 Werte eingelesen werden können sowie dass dieser auch mit ungültigen Werten umgehen kann.
- Übung: exercises/exercise-calculator.js

```
const calculator = {
    // ... your code ...
};
calculator.read(1, 2);
console.log( calculator.sum() );
console.log( calculator.mul() );
```

Lösung 1:
https://gist.github.com/bitsmuggler/9e5b411bca2ec301a3ef4d2c6a95c2e3
Lösung 2:
https://gist.github.com/bitsmuggler/700b4364aae228e3049db23b999336b7

Übungsaufgabe II - «Chaining» (10’)

- Implementiere die Leiter und modifiziere den Leiter Code, dass er ”chainable” ist.
- Übung: ./exercises/exercise-ladder.js

```
ladder.up().up().up().down().currentStep(); // 2
```

Lösung:
https://gist.github.com/bitsmuggler/adcf7e5cd6e2d4ab9ac5b867b3c8425d

JavaScript Sprachkonzepte V

Classes

Übungsaufgabe – Classes (10’)

- Schreibe die Person Klasse einem mit puren functions und einmal mit dem class Konstrukt.

```
const person = new Person('Patrick');
person.sayHello(); // Hello Patrick
```

Lösung:
https://gist.github.com/bitsmuggler/23cd22fbab70f73bc82b49089ae7b6d2

Zusatz: Übungsaufgabe (20’)

- Schreibe eine Uhr die tickt (nach OOP-Prinzipien).

```
const clock = new Clock('h:m:s');
clock.start();
```

- Schreibe eine erweiterte Clock (extends), bei welcher die Präzision (Anzahl ms) mitgegeben werden kann.

```
const extendedClock = new ExtendedClock('h:m:s', 2000);
extendedClock.start();
```

- Tipp: setInterval anschauen

Lösung:
https://gist.github.com/bitsmuggler/4455d7b2bbb15a974d8d59f66c0a90de
Lösung:
https://gist.github.com/bitsmuggler/8f9c0fa5f7a1e12d587382487634ef52

JavaScript Sprachkonzepte IV

Error Handling & Promises, async & await

Übungsfrage (5’)

- Gibt es hier einen Unterschied? Wenn ja, welchen?

```
promise.then(fn1).catch(fn2);

// vs.

promise.then(fn1, fn2);
```

Variante 1: Catch auf zweiter Promise
Variante 2: Catch auf erster Promise

Übungsaufgabe – Promises (15’)

- Erstelle eine Promise, welche nach einer angegeben Zeit in ms den übergebenen String wieder zurück gibt. Der Code soll wie folgt anwendbar sein:

```
delay(2000, 'hallo').then(console.log);
```

Übung: ./exercises/exercise-promise.js

https://gist.github.com/bitsmuggler/d9f3a7920cb95858cc42bfbc119a6d1e