Period-1 Vanilla JavaScript, Es-next, Node.js, Babel + Webpack and TypeScript-1

Note: This description is too big for a single exam-question. It will be divided up into several smaller questions for the exam

Explain and Reflect:

- **Explain the differences between Java and JavaScript + node. Topics you could include:**
    - **that Java is a compiled language and JavaScript a scripted language**
      Java skal compiles før det det kan afvikles. Hvorimod et scripted language kan afvikles on the go. Javascript f.eks. som bruges til webapplikationer.
    - **Java is both a language and a platform**
      Java kører i Java virtuel machine. Også kaldet (JVM). I mens JavaScript kører direkte i webbrowseren.
    - **General differences in language features**.
      Java er meget stærkt typesprog. Hvorimod JavaScript er i en helt anden boldgade. Derudover er JavaScript single threaded.
    - **Blocking vs. non-blocking**
      Hvis man ”blokerer” noget, vil stakken ikke blive kørt før den blokerede kode er kørt.
- **Explain generally about node.js, when it “makes sense” and *npm*, and how it “fits” into the node echo system.**
    - **Node.js:** Bruges til at skabe backend kode i javascript. Men har altså forvandlet et ellers clientside programmeringssprog til at blive brugt på server delen.
    - **Npm:** Er en package management system. Som bruges til at hente/dele eller installere packages.
- **Explain about the Event Loop in JavaScript, including terms like; blocking, non-blocking, event loop, callback queue and "other" API's. Make sure to include why this is relevant for us as developers.**
    - **Event loop**: Javascripts eventloop tjekker konstant callstacken for at se om der er en funktion som skal køre. I mens den gør dette, tilføjer den de funktioner den finder, og eksekverer dem i en efter en i rækkefølge.
    - **Blocking & non-blocking:** Med blocking meners der at hvis en funktion afventer data eller andre ting, som gør at eventloopet bliver nød til at vente på denne funktion for at forsætte. Man stopper altså selve eventloopet grundet denne ene funktion.
      Non-blocking referer derfor til at man laver en funktion som eventloopet ikke skal vente på.
    - **Callback queue:** Er køen, hvori funktioner bliver kørt. Disse bliver kørt i først inde, først ude rækkefølge.
- **What does it mean if a method in nodes API's ends with xxxxxxSync?**
    - Asynchronus/Synchronous

- **Explain the terms JavaScript Engine (name at least one) and JavaScript Runtime Environment (name at least two)**
    - Javascript Engine: V8 (Google Chrome)
    - Javascript Runtime Environment: Spidermonkey(Firefox), Nitro (Safari)

- **Explain (some) of the purposes with the tools *Babel* and *WebPack and how they differ from each other*. Use examples from the exercises.**
    - Babel: Er en Javascript transpiler, som konventerer JavaScript til gammel ES5 Javascript. Som kan kører i alle browsere.
    - WebPack: Som navnet pakker dette tools vores imports i en fil.

Explain using sufficient code examples the following features in JavaScript (and node)

- **Variable/function-Hoisting**
  Link til opgave: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%201%20(Introduction)/Opgaver/hoisting.js

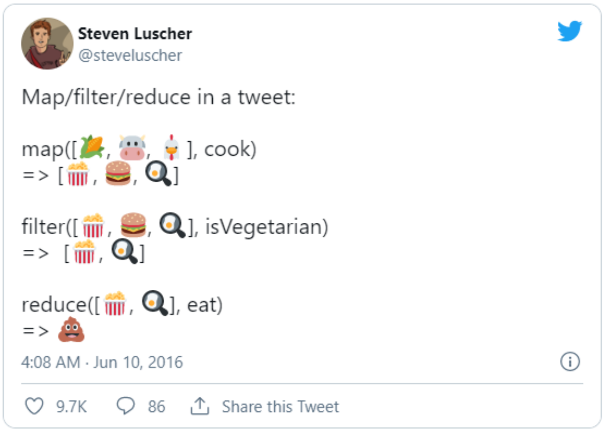
- ***`this`* in JavaScript and how it differs from what we know from Java/.net.**

- **Function Closures and the JavaScript Module Pattern**
  Link til opgave: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%201%20(Introduction)/Opgaver/closures.js

  https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%201%20(Introduction)/Opgaver/closuresInLoop.js

- **User-defined Callback Functions (writing functions that take a callback)**
  Link til opgave: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%201%20(Introduction)/Opgaver/callbacks.js

- **Explain the methods `map`, `filter` and `reduce`**
  Stødte på denne sjove tweet som demonstrerede hvad disse arrayfunktioner betyder.



Steven Luscher
@steveluscher

Map/filter/reduce in a tweet:

map([🌽, 🐮, 🐔], cook)
=> [🍿, 🍔, 🍳]

filter([🍿, 🍔, 🍳], isVegetarian)
=> [🍿, 🍳]

reduce([🍿, 🍳], eat)
=> 💩

4:08 AM · Jun 10, 2016

  - **Map:** Vi laver et nyt array fra et eksisterende og ”applyer” en funktion til hvert element.
  - **Filter:** Vi filtrer vores array, ved at bruge et conditional statement mod hvert element. Skulle denne returnere ”true” bliver dette element skubbet til vores ”output array”.
  - **Reduce:** Vi forvandler vores array til 1 value.

  Link til opgave: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%201%20(Introduction)/Opgaver/callbacks.js

- **Provide examples of user-defined reusable modules implemented in Node.js (learnynode - 6)**
  Link til opgaver: https://github.com/Erayous/Fullstack-Javascript/tree/main/P1-Handin%20(period%201)/Dag%202%20(NodeJs%20%26%20Modules-npm)/Opgaver

- **Provide examples and explain the es2015 features: let, arrow functions, this, rest parameters, destructuring objects and arrays, maps/sets etc.**
  Link til opgave: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%202/Opgaver/es2015Features.js

- Provide an example of ES6 inheritance and reflect over the differences between Inheritance in Java and in ES6.

- **Explain and demonstrate, how to implement event-based code, how to emit events and how to listen for such events**
  Link til opgaver: https://github.com/Erayous/Fullstack-Javascript/tree/main/P1-Handin%20(period%201)/Dag%202%20(NodeJs%20%26%20Modules-npm)/Opgaver

ES6,7,8,ES-next and TypeScript

- **Provide examples with es-next, running in a browser, using Babel and Webpack**
- **Explain the two strategies for improving JavaScript: Babel and ES6 + ES-Next, versus Typescript. What does it require to use these technologies: In our backend with Node and in (many different) Browsers**
- **Provide examples to demonstrate the benefits of using TypeScript, including, types, interfaces, classes and generics**
- **Explain how we can get typescript code completion for external imports.**
- **Explain the ECMAScript Proposal Process for how new features are added to the language (the TC39 Process)**

https://github.com/Erayous/Fullstack-Javascript/tree/main/P1-Handin%20(period%201)/Dag%205%20(Introduction%20to%20TypeScript)/Opgaver/src

Callbacks, Promises and async/await

Explain about (ES-6) promises in JavaScript including, the problems they solve, a quick explanation of the Promise API and:

- ~~Example(s) that demonstrate how to avoid the callback hell ("Pyramid of Doom")~~
- Example(s) that demonstrate how to implement **our own** promise-solutions.
  https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%203%20(Es2015%20promises%20%26%20async-await)/Opgaver/crypto-module.js
- Example(s) that demonstrate error handling with promises
  https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%203%20(Es2015%20promises%20%26%20async-await)/Opgaver/crypto-module.js

- **Example(s) that demonstrate how to execute asynchronous (promise-based) code in serial or parallel**
  https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%203%20(Es2015%20promises%20%26%20async-await)/Opgaver/ex1-b.js

Explain about JavaScripts **async/await**, how it relates to promises and reasons to use it compared to the plain promise API.

Provide examples to demonstrate

- **Why this often is the preferred way of handling promises**
- **Error handling with async/await**
- **Serial or parallel execution with async/await.**

Opgave med serial: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%203%20(Es2015%20promises%20%26%20async-await)/Opgaver/serial.js

Opgave med Parallel: https://github.com/Erayous/Fullstack-Javascript/blob/main/P1-Handin%20(period%201)/Dag%203%20(Es2015%20promises%20%26%20async-await)/Opgaver/parallel.js