**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 is both a language and a platform
  + General differences in language features.
  + Blocking vs. non-blocking

Min besvarelse:

Java bliver compiled til bytekode og kørt på en virtuel maskine (JVM). JavaScript kan blive læst direkte af browseren, og det eneste der ændres i koden er ofte at det bliver ’minified’.

Java er også en platform, hvilket gør at Java er platform uafhængig, da det ikke skal bruge noget særlig fra den platform som platformen bliver kørt på.

* Explain generally about node.js, when it “makes sense” and *npm*, and how it “fits” into the node echo system.

Min besvarelse:

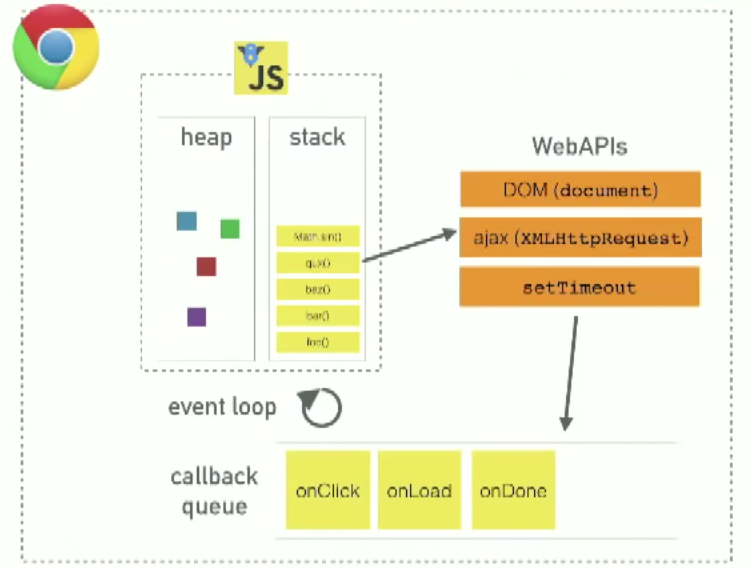
Node.js bruges når man gerne vil have JavaScript på sin backend/server side. NPM er Node.js’s ’package manager’ og bruger til at hente alle de mange gode biblioteker der er tilgængelige for Node.js.

* 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.

Min besvarelse:

JavaScript bruger en såkaldt ’stack’ med et Event Loop, som gør at langsomme (blokerende) processer bliver håndteret uden for Event Loopet i en thread pool, således at processerne ikke blokerer i Event Loopet som i sig selv kun har én enkelt tråd. Dette design har Java ikke og derfor kan processer blokere, og derfor gøre fx en webside meget uresponsiv.

Det er vigtigt at kende til Event Loop, da det eksempelvis påvirker den rækkefølge som elementer bliver vist på en hjemmeside.



* What does it mean if a method in nodes API's ends with xxxxxx**Sync**?

Min besvarelse:

Det betyder at metoden er synkron og derfor vil metoden være blokerende. Det kan være en fordel hvis man vil sikre sig at næste metode først bliver udført når den blokerende metode er færdig.

* Explain the terms JavaScript Engine (name at least one) and JavaScript Runtime Environment (name at least two)
* Explain (some) of the purposes with the tools *Babel* and *WebPack and how they differ from each other*.       Use examples from the exercises.

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

* Variable/function-Hoisting

See hoisting.js

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

See this.js

* Function Closures and the JavaScript Module Pattern

See closures.js

* User-defined Callback Functions (writing functions that take a callback)

See callback.js

* Explain the methods map, filter and reduce

See mapfilterreduce.js

* Provide examples of user-defined reusable modules implemented in Node.js (learnynode - 6)

See exportmodule.js & importmodule.js

* Provide examples and explain the es2015 features: let, arrow functions, this, rest parameters, destructuring objects and arrays,   maps/sets etc.

See es5.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

See emit.js

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)

**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.
* Example(s) that demonstrate error handling with promises
* Example(s) that demonstrate how to execute asynchronous (promise-based) code in **serial** or **parallel**

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.

Se the exercises for Period-1 to get inspiration for relevant code examples