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

* Explain the differences between Java and JavaScript + node. Topics you could include:
  + that Java is a compiled language and JavaScript a scripted language

**Java er et ”compiled language” dvs det skal kompileres før det kan afvikles.**

**Javascript primært til webapplikationer**

**Node er js til server siden.**

* + Java is both a language and a platform

**Java er en et sprog og platform som er platform dependent, alle javaprogrammer kører i java virtuel machine(jvm) , i mens javascript kører out of the box i browseren.**

* + General differences in language features.

**Java er et type stærkt sprog, Javascript er ikke.**

**Javascript bliver integreret med HTML for at give funktion til hjemmesider/apps.**

**Javascript er platform independent kan kører på alt (windows, maco sv)**

**Javascript er single threaded.**

* + Blocking vs. non-blocking

Blocking vil sige at stakken bliver ”blokeret” indtil det ”blockede” kode er kørt.

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

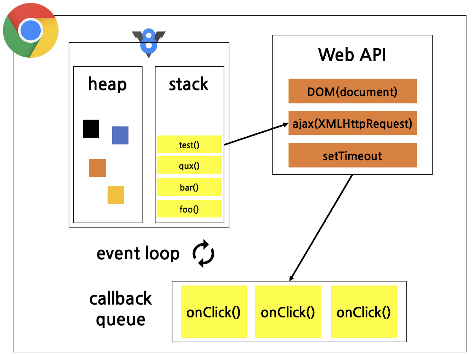
**Node.js kan bruges til at skrive server site kode i javascript / Køre Javasript udenfor en browser.**

**Det er open-source – alle kan bruge node packet manageren og dele og hente kode.**

* 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.

**dvs. når man f.eks laver et request et node serveren om at hente en fil, så bliver den request ”smidt” over til en API ”worker” som ligger det tilbage i vores callback kø, og når stacken er tom bliver den kørt.**



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

**Så betyder det at det ikke en en Async metode, dvs den ”blocker”**

**Eksempel i p1/Day3/KodeMedFraVideo/block-NonBlock.js**

* Explain the terms JavaScript Engine (name at least one) and JavaScript Runtime Environment (name at least two)

**V8(Google Chrome) , Spidermonkey(Mozilla Firefox), Nitro (Apple 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: Bruges til at transpile nyt EcmaScript kode til tidligere versioner. Så det kan køres i ældre engines.**

**Webpack: Samler/pakker alle imports osv i 1 fil eller flere.**

**Eksempel: p1/Day4/opg2babel**

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

* Variable/function-Hoisting

**Eksempel: P1/Day1/hoisting.js**

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

**I javascript referere this til “owner objektet”, I en funktion referer det til det globale Object**

**I java refere det til objeket da det blive instansiret**

**Eksempel: P1/Day1/this.js**

* Function Closures and the JavaScript Module Pattern

**Eksempel: P1/Day1/Opgaverne/closures.js**

**Eksempel: P1/Day1/Opgaverne/closuresExported.js**

**Eksempel: P1/Day1/Opgaverne/closuresOpgNameAge.js**

**Eksempel: P1/Day1/Opgaverne/closuresInLoop.js**

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

**Eksempel: P1/Day1/Opgaverne/MagicOfCallBacks.js**

* Explain the methods map, filter and reduce

**Eksempel: P1/Day1/Opgaverne/MagicOfCallBacks.js**

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

**Eksempel: P1/Day2/Opgaverne/osData.js**

**Eksempel: P1/Day2/Opgaverne/nodeServer.js**

**Eksempel: P1/Day2/Opgaverne/dosDetector.js**

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

**Eksempel: P1/Day2/Opgaverne/es2015Features.js**

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

**Eksempel: P1/Day5/Exercises/src/ClassesInheritance.ts**

* Explain and demonstrate, how to implement event-based code, how to emit events and how to listen for such events

**Eksempel: P1/Day2/Opgaverne/osData.js**

**Eksempel: P1/Day2/Opgaverne/nodeServer.js**

**Eksempel: P1/Day2/Opgaverne/dosDetector.js**

ES6,7,8,ES-next and TypeScript

* Provide examples with es-next, running in a browser, using Babel and Webpack

**Alt I stage-4 er i ES-next.**

* 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

**P1/Day5/KodeMedITimen/src/logger.ts**

**P1/Day5/Exercises/src**

* Explain how we can get typescript code completion for external imports.

**Npm install @types/lodash f.eks.**

* Explain the ECMAScript Proposal Process for how new features are added to the language (the TC39 Process)
* Stage 0: Strawperson
* [Stage 1: Proposal](https://nitayneeman.com/posts/introducing-all-stages-of-the-tc39-process-in-ecmascript/#stage-1-proposal)
* [Stage 2: Draft](https://nitayneeman.com/posts/introducing-all-stages-of-the-tc39-process-in-ecmascript/#stage-2-draft)
* [Stage 3: Candidate](https://nitayneeman.com/posts/introducing-all-stages-of-the-tc39-process-in-ecmascript/#stage-3-candidate)
* [Stage 4 Finished](https://nitayneeman.com/posts/introducing-all-stages-of-the-tc39-process-in-ecmascript/#stage-4-finished)

**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.

**Eksempel: P1/Day3/Exercises/ex1-crypto-module.js**

* Example(s) that demonstrate error handling with promises

**Eksempel: P1/Day3/Exercises/ex1-crypto-module.js**

* Example(s) that demonstrate how to execute asynchronous (promise-based) code in **serial** or **parallel**

**Eksempel: P1/Day3/Exercises/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.

**Eksempel: P1/Day3/Exercises/ex3-serial.js**

**Eksempel: P1/Day3/Exercises/ex3-parallel.js**

Første del af spørgsmålene er diskuteret i studie-gruppen bestående af:

Michael Ipsen, Sebastian Bentley, Sebastian Hansen og Rasmus Grønbæk.