* *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*

My own text…

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

Node is a runtime environment used by javascript, in order to interact with the local system, for instance when we need additional files on the computer to be used, or when we want the program to write to another file. The npm is the package manager, responsible setting up and keeping track of the node environment as it is install into the system.

* ***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.***

The Event loop consist of, as a minimum example, the stack, a webapi, and the callback queue. When a function is called with async or setTimer for instance, it is given to the webapi to keep track of, and when it’s done/the time runs out, it is placed upon the callback queue, where it will go in the back of the line. Whenever the stack is empty, it will take the first object in the callback queue, and put onto the stack. As such, when code runs on the stack, it is blocking the stack, when it is handled outside the stack, and put into the event loop, it becomes non-blocking. This allows for faster code, since we don’t have to wait for potential slow blocking code, that depends on outside assistance, like fetching from an external api. This can also be used to make sure that code runs after the stack has been emptied from what else was on the stack at the time.

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

It would mean that it is a function that do NOT use a async method, and would as such be a blocking method, blocking the JS-event-queue (the stack)

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

A Javascript engine (v8 for chrome, spiderMonkey for firefox) is the program parsing the javascript code, and the part that runs the code from the top and down, and look for syntax errors, turning the js-code into machine code.

A javascript runtime enviorment (node, chrome, firefox) is what provides some objects to js to interact with, so it can interact with objects outside it’s own imidiet scope. This also includes the outside elemtents like the webapi and callback queue.

* Explain (some) of the purposes with the tools *Babel* and *WebPack and how they differ from each other*.       Use examples from the exercises.

My own text…

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

* ***Variable/function-Hoisting***

Code Examples under Week01/magicOfCallbacks/hoisting.js

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

Code Examples under Week01/magicOfCallbacks/thisInJavaScript.js

* ***Function Closures and the JavaScript Module Pattern***

Code Examples under Week01/magicOfCallbacks/closures.js

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

Code Examples under Week01/magicOfCallbacks/myPrototype.js

* ***Explain the methods map, filter and reduce***

Code Examples under Week01/magicOfCallbacks/myFilter.js, myMap.js & reduceMethod.js

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

Code Examples under Week01/learnyounode/make-it-module.js & mymodule.js

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

Code Examples under …

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

Code Examples under …

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

Code Examples under Week02/assignment2.js & dosDetector.js

ES6,7,8,ES-next and TypeScript

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

My own text…

* Explain the two strategies for improving JavaScript: Babel and ES6 (es2015) + ES-Next, versus Typescript. What does it require to use these technologies: In our backend with Node and in (many different) Browsers

My own text…

* Provide **examples** to demonstrate the benefits of using TypeScript, including, types, interfaces, classes and generics

My own text…

* Explain the ECMAScript Proposal Process for how new features are added to the language (the TC39 Process)

My own text…

**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")

My own text…

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

My own text…

* Example(s) that demonstrate how to implement **our own** promise-solutions.

My own text…

* Example(s) that demonstrate error handling with promises

My own text…

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

My own text…

* Error handling with async/await

My own text…

* Serial or parallel execution with async/await.

My own text…

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