

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
- Explain generally about node.js, when it “makes sense” and *npm*, and how it “fits” into the node ecosystem.
- 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.
- What does it mean if a method in nodes API's ends with **xxxxxxSync**?
- 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
- *this* in JavaScript and how it differs from what we know from Java/.net.
- Function Closures and the JavaScript Module Pattern
- User-defined Callback Functions (writing functions that take a callback)
- Explain the methods `map`, `filter` and `reduce`
- Provide examples of user-defined reusable modules implemented in Node.js (learnynode - 6)
- Provide examples and explain the es2015 features: `let`, arrow functions, `this`, rest parameters, destructuring objects and arrays, maps/sets etc.
- 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

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

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](#)