

## 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

Javascript is used “out of the box”, and does not include executor, compiler etc.

It does however need other tools like node if it is to be used outside of a web browser. Javascript is a script language and is compiled in runtime.

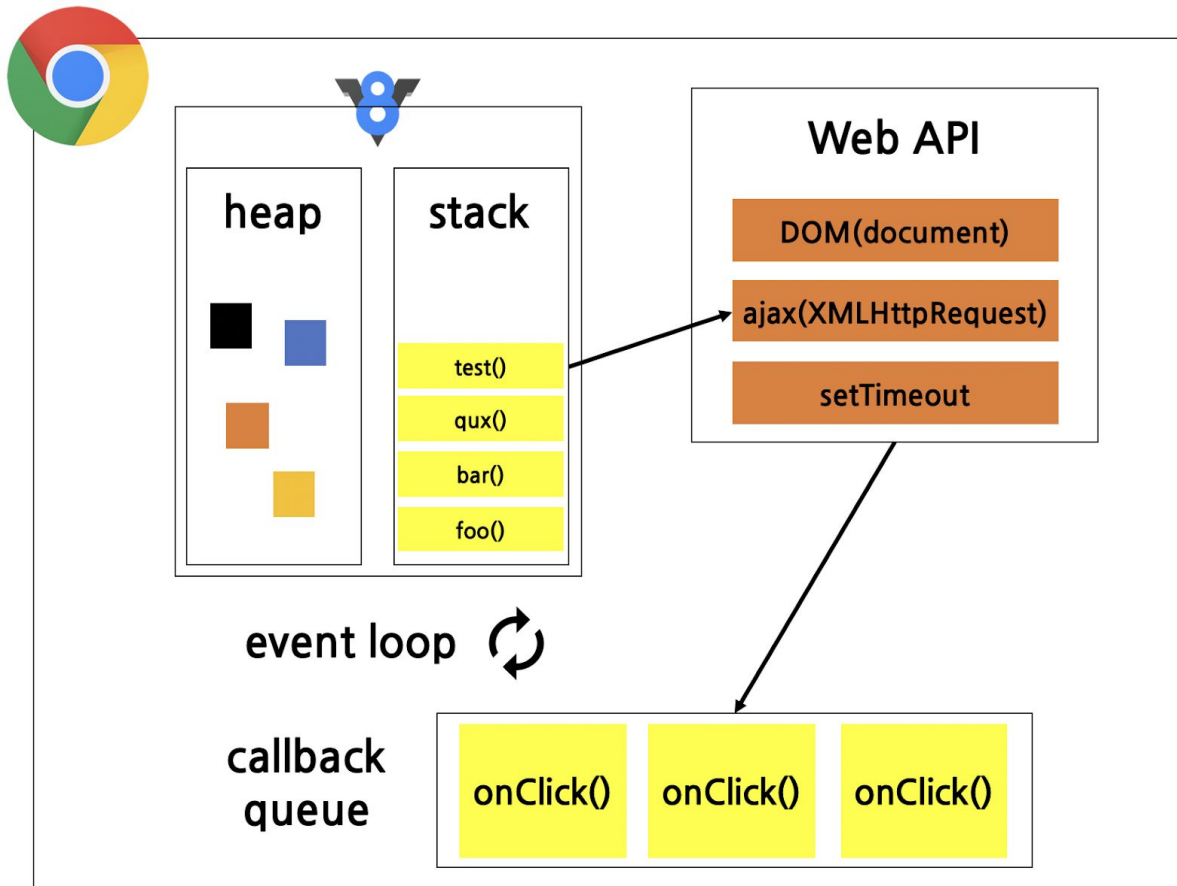
Java is a type strong language, whereas JavaScript is not. JavaScript is a single-threaded language, which means that if there is something on the stack being executed, by default JavaScript will block the “task”.

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

Node.js makes it possible to run JavaScript outside of our web browser. Npm(node package manager) is used to get dependencies which will be used in our code.

Node.js is open-source, which means that everyone can add packages to npm.

- 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.



Async tasks move to the Web API and then to the callback queue, where it will be handled when the stack is empty.

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

Sync methods will block the stack.

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

{ V8 = Google, SpiderMonkey = Mozilla, JavaScriptCore = Apple, Chakra = IE }

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

Babel is a transpiler for converting ECMAScript 2015+ into backwards compatible js that can run on older js engines

webpack

<https://github.com/Groenbek/Period1/tree/main/Day4/webpack-tutorial>

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

- Variable/function-Hoisting

<https://github.com/Groenbek/Period1/tree/main/Day1>

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

<https://github.com/Groenbek/Period1/tree/main/Day1>

- Function Closures and the JavaScript Module Pattern

<https://github.com/Groenbek/Period1/tree/main/Day1>

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

<https://github.com/Groenbek/Period1/tree/main/Day1>

- Explain the methods `map`, `filter` and `reduce`


<https://github.com/Groenbek/Period1/tree/main/Day1>

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

<https://github.com/Groenbek/Period1/tree/main/Day2>

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

<https://github.com/Groenbek/Period1/tree/main/Day4/webpack-tutorial>

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

<https://github.com/Groenbek/Period1/tree/main/Day5>

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

<https://github.com/Groenbek/Period1/tree/main/Day2>

ES6,7,8,ES-next and TypeScript

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

<https://github.com/Groenbek/Period1/tree/main/Day4/webpack-tutorial>

- 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

<https://github.com/Groenbek/Period1/tree/main/Day4/webpack-tutorial>


<https://github.com/Groenbek/Period1/tree/main/Day5>

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

<https://github.com/Groenbek/Period1/tree/main/Day5>

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

<https://github.com/Groenbek/Period1/tree/main/Day5>

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

Link: [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**

<https://github.com/Groenbek/Period1/tree/main/Day3>

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.

<https://github.com/Groenbek/Period1/tree/main/Day3>