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 differences between Java and JavaScript + node:

That Java is a compiled language and JavaScript a scripted language
Java needs to be compiled before run, it needs a jvm (java virtual machine) to run.

Java is both a language and a platform
Java includes an execution engine, a compiler, and a set of libraries in it.

General differences in language features
Java is type strong, whereas JavaScript is not.

Blocking vs. non-blocking
JavaScript is executed at runtime, since it is a script language.
Node is often serverside, and javascript is primarily used for webapplications. JavaScript needs an engine like Node.js to run outside a browser.

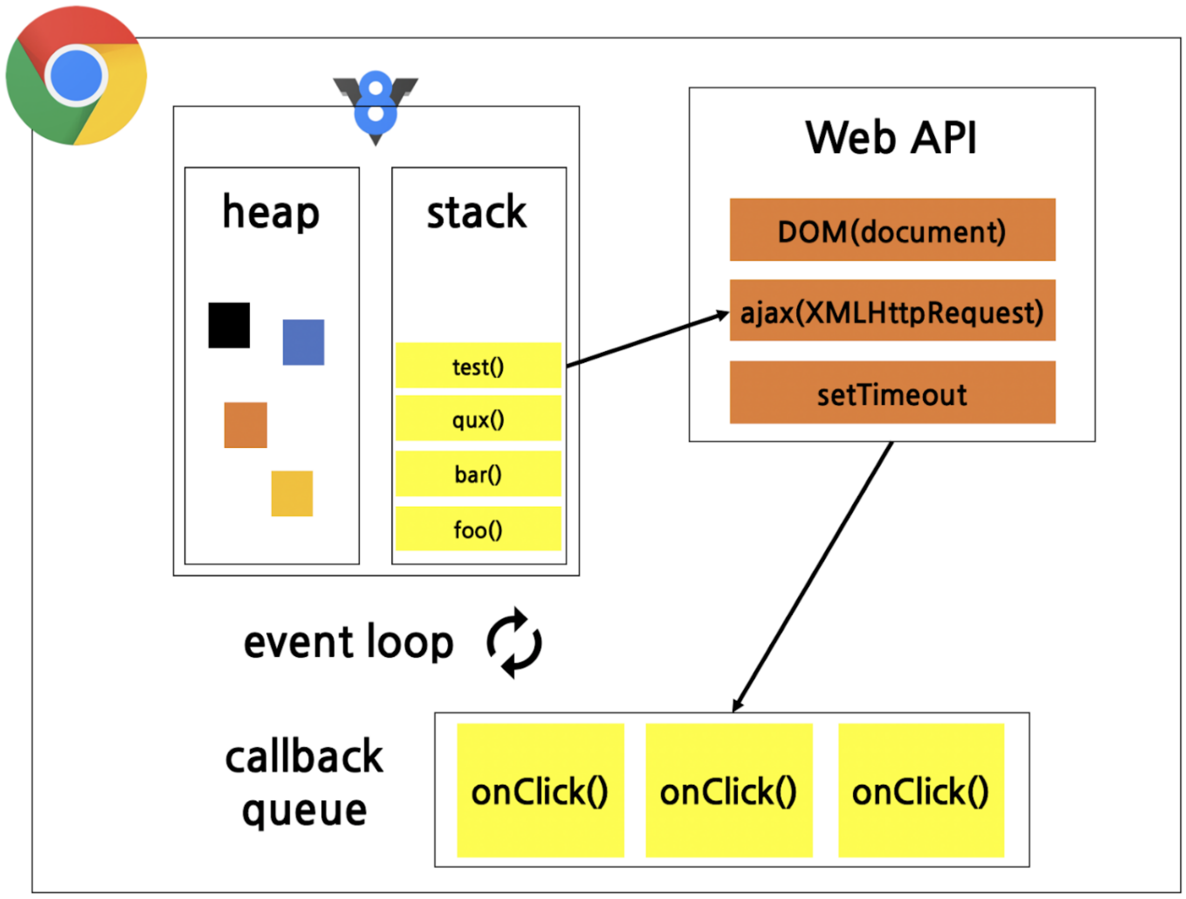
JavaScript is single threaded language, which means that a stack is blocking task-calls, unless asynchronous methods are used. Java can use multible threads.

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

Node.js makes is possible to run JavaScript outside a browser (built on Chromes V8 engine).
Npm (node package manager) is used to get dependencies 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.

If a promise is called, it is sent to web api, and then callback



What does it mean if a method in nodes API's ends with xxxxxxSync?
It means that it is a method that is blocking the stack.

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

- JavaScript engines executes JavaScript code. Node.js is built on Chrome V8 engine.
- Mozilla Firefox has SpiderMonkey, and Apples Safari has Nitro.

Node.js is a runtime Environment, and google Chrome is a runtime enviromnet as a browser.

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 used transpile new ES to older versions.

Webpack explain link: https://www.valentinog.com/blog/webpack/#getting-started-with-webpack

"webpack is a module bundler "

Example: JavaTypeScript/webpack-startcode

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

Variable/function-Hoisting
In JavaScript, a variable acan be used before it has been declared.

Example: JavaTypeScript/Period1/day1/hoisting.js

***`this`* in JavaScript and how it differs from what we know from Java/.net.**
In JavaScript *this* refers to the owner of the function we are executing, or rather the object that a function is a method of.
In Java, *this* refers to the current instance object on which the method is executed.
Example: JavaTypeScript/Period1/day1/thisInJavaScript.js

Function Closures and the JavaScript Module Pattern
A closure gives you access to an outer functions scope from an inner function.
Example: JavaTypeScript/Period1/day1/closures.js

User-defined Callback Functions (writing functions that take a callback)
Function as parameter
Example: JavaTypeScript/Period1/day1/myFilterAndmyMap.js

Explain the methods `map`, `filter` and `reduce`
Returns new array, not a reference to old.
Example: JavaTypeScript/Period1/day1/myFilterAndmyMap.js

Provide examples of user-defined reusable modules implemented in Node.js (learnynode - 6)
Importing and exporting.
Example: JavaTypeScript/Period1/day1/learnyounode/mymodule.js (and make-it-modular.js)

Provide examples and explain the es2015 features: `let, arrow functions, this, rest parameters, destructuring objects and arrays,` 🟨 `maps/sets` etc.
Link with examples: https://babeljs.io/docs/en/learn/

🟨 **Provide an example of ES6 inheritance and reflect over the differences between Inheritance in Java and in ES6.**
Not many differences in ES6, but huge difference from ES5.
Example: JavaTypeScript/Period1/day5/typescriptexercise/src/classes.ts

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

From EventEmitter = require('events')

Example: JavaTypeScript/Period1/day2/dosDetector.js

ES6,7,8,ES-next and TypeScript

Provide examples with es-next, running in a browser, using Babel and Webpack
ES Next is a term used to refer to future versions of ECMAScript that have not been released.
Example: JavaTypeScript/webpack-startcode

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

- **Typescript** is a superset of js, which means it can do anything that javascript can do, and same functionality as babel.
- **Babel** can take newer versions of ES, and compile it down to older versions.

Example: JavaTypeScript/Period1/day5/typescriptexercise/tsconfig.json (change ES)

Provide examples to demonstrate the benefits of using TypeScript, including, types, interfaces, classes and generics
Example: JavaTypeScript/Period1/day5/typescriptexercise/

Explain how we can get typescript code completion for external imports.
With command: *npm install node-fetch and npm install @types/node-fetch --save-dev*

🟨 **Explain the ECMAScript Proposal Process for how new features are added to the language (the TC39 Process)**
process found here: https://tc39.es/process-document/

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:

JavaScript is single threaded, and async/await fixes the management of the stack. ES6 introduces a better syntaxs for asynchronous calls.

Promise API: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

~~**Example(s) that demonstrate how to avoid the callback hell (“Pyramid of Doom")**~~
Example: JavaTypeScript/Period1/day3/exercise1.js

Example(s) that demonstrate how to implement our own promise-solutions.
Example: JavaTypeScript/Period1/day3/exercise1.js

Example(s) that demonstrate error handling with promises
Example: JavaTypeScript/Period1/day3/exercise1/2.js

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

Example: JavaTypeScript/Period1/day3/exercise3.js

Explain about JavaScripts async/await, how it relates to promises and reasons to use it compared to the plain promise API.
Reason to use: way better syntax

Provide examples to demonstrate:

Why this often is the preferred way of handling promises
Example: JavaTypeScript/Period1/day3/exercise2.js

Error handling with async/await
Example: JavaTypeScript/Period1/day3/exercise1.js

Serial or parallel execution with async/await.
Example: JavaTypeScript/Period1/day3/exercise3.js

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