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

Udarbejdet af Jacob Simonsen, Christian Madsen, Frederik Hurup

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:

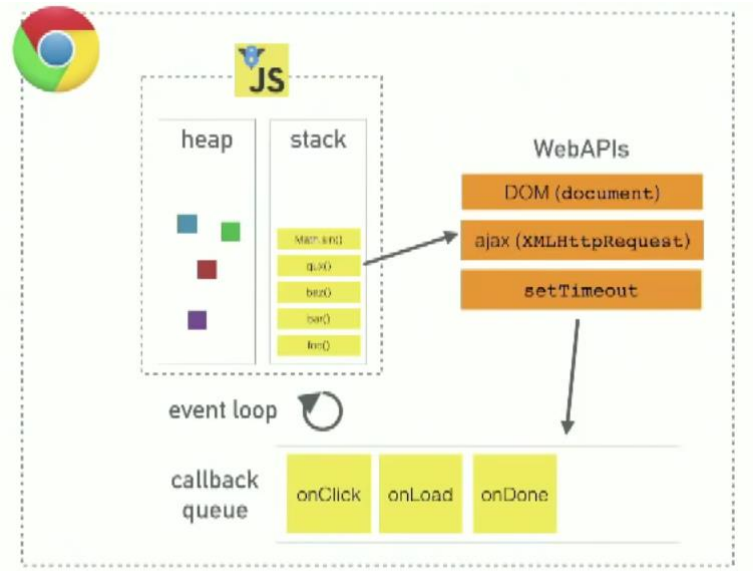
- **Explain the differences between Java and JavaScript + node. Topics you could include:**
  - **that Java is a compiled language and JavaScript a scripted language**
  - **Java is both a language and a platform**
  - **General differences in language features.**
  - **Blocking vs. non-blocking**

På grund af at javascript har firstclass funktioner er JavaScript et funktionelt programmeringssprog. JavaScript kode køre i en browser hvor Java køre virtuelt. Java er type stærkt hvor JavaScript er typesvagt. Java bliver typisk brugt til backend samt server, hvor JavaScript bliver brugt til frontend samt klient side.

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

Node.js er et backend miljø, som køre på v8 (chrome engine), som eksekvere JavaScript kode ude for en browser. Det giver mening fordi man kun skal fokusere på koden og ikke package management, som npm håndterer for én, hvilket gør at det komplimenterer node.js godt.

- **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.**



JavaScript funktioner bliver kørt en efter en, ved at blive lagt i stacken. Hvis én af funktionerne kræver et eksternt WebAPI bliver den ført ud af stacken og processeret indtil der kommer et svar, hvorefter den bliver lagt i callback-queuen indtil stacken er ledig igen, og den kan blive kørt. Så længe en funktion ligger i stacken, er den blocking og sørger for at intet andet kan køre samtidigt (medmindre det køres asynkront) og derfor er det vigtigt for udviklere at være klar over hvad der foregår i event-loopet, så man ikke ender med at ødelægge sin program med en funktion, som aldrig ender med at få et respons.

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

Det betyder at den bliver kørt synkront og dermed blokerer programmet indtil den har kørt færdig (Den slags metoder skal man helst undgå at bruge, medmindre det er nødvendigt for de efterfølgende metoder).

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

En JavaScript Engine er et computerprogram der udfører JavaScript kode, såsom Chromes V8 Engine. JS Runtime Environment er det miljø hvori programmet bliver udført, såsom node.js og alle de forskellige browsere.

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

Babel er et tool til at konvertere ES6+ kode til at være bagudkompatibelt.
WebPack er et tool til at bundle sin kode, hvilket gør den hurtigere at eksekvere i produktion.

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

- **Variable/function-Hoisting**

Hoisting er JavaScript's standard adfærd for at flytte declarations til toppen.

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

This, i Javascript refererer til det objekt, som det hører til. This, i Java referer til metoderne i klasserne for at undgå forvirring i parametrene.

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

Variable kan defineres i et lokalt eller globalt scope som afgøre hvor variablen kan kaldes. Module Pattern er der hvor man kan importere eller eksportere.

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

Skriv en funktion der tager imod en anden funktion, som fx. kan mutere data.

Eksempel: learnyounode/dayOne.js

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

Det er tre metoder som kan manipulere et array, via en Callback, derved returnerer det et nyt array. Map ændrer hvert element i arrayet. Filter filtrerer et array. Reduce reducerer et array til et enkelt element.

Eksempel: learnyounode/dayOne.js

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

Eksempler: learnyounode/Osreader.js OG learnyounode/dosDetector.js

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

Let er en block-scope variable, så den kan ikke tilgås udenfor {}.

Arrow-functions er en kortere måde at skrive funktioner på (kan ikke bruges i alle situationer).
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

This, i Javascript refererer til det objekt, som det hører til.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this

Rest parameters tillader en funktion at acceptere en ukendt mængde parametre som et array.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/rest_parameters

Destructuring arrays/objects gør det muligt at deklarere elementer til forskellige variable.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Sets er en liste af unikke elementer.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

Maps er en liste af key/value pairs (Telefonbog - unik key)
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

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

Forskellen er at man kan ændre på klasserne under runtime i JS, men det kan man ikke i Java. I JS kan man også arve fra flere klasser.

Eksempel: day5/src/ex5.ts

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

Event-based code lytter efter et event og hvis det event sker så udføres koden.

Eksempel: learnyounode/dosDetector.js

ES6,7,8,ES-next and TypeScript

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

Eksempel: babel

- **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**

Det kræver at installere de rigtige dependencies, samt generere dependencies filer.

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

Eksempler:
day5/src/chuckNorrisFetcher.ts - types
day5/src/ex1.ts – interfaces og classes
day5/src/ex4.ts - generics

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

Man skal installere @types/… for det import man har hentet.

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

0	**Strawperson**	[redacted]

1	Proposal	Identify champion
2	Draft	Initial spec. text
3	Candidate	Complete spec. text
4	Finished	Get ready for implementation

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:

Promises gør det muligt at køre sin kode parallelt via at kunne sende et request ud og køre andet koden mens man venter på respons.

- ~~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: Day3/exercises/ex1.js

- Example(s) that demonstrate error handling with promises

Eksempel: Day3/exercises/ex1.js

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

Eksempel: Day3/exercises/ex3.js

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

Async/await gør at man kan køre flere requests asynkront, men at man får koden til at vente(await) på alle svarene for at undgå fejl bagefter. Det giver også en mere clean måde at skrive promise-baseret kode, så man undgår The Pyramid of Doom.

Provide examples to demonstrate

- Why this often is the preferred way of handling promises

Eksempel: Day3/exercises/ex3.js

- Error handling with async/await

Eksempel: Day3/exercises/ex2.js

- Serial or parallel execution with async/await.

Eksempel: Day3/exercises/ex3.js

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