Grundkurs i JavaScript

Grupparbete

🚨 VIKTIGT

- **Börja inte jobba** med uppgiften förrän ni har fått examinationskraven i den agila kursen. Ni får betyg i den kursen baserat på er ARBETSPROCESS i denna uppgift.
- Det går inte att bli godkänd på någon av kurserna om man inte deltar i grupparbetet ⇒ det är svårt att göra ett grupparbete själv.
- Om ni inte fått kontakt med alla i gruppen fredag förmiddag den 13 december så måste ni höra av er till en lärare.
- Grupparbete kan endast ge G som betyg.

Sammanfattning av "att göra" mellan 12 dec och 10 jan

- Koda ett quiz i TypeScript genom att använda agila arbetsmetoder (scrum)
- Dokumentera den agila processen i gruppen i repot
- Skriva en individuell (slut)reflektion

Meny Innehåll

Gruppindelning

▶ **Grupper**

Att jobba i grupp (kod)

▶ **Hur ni ska göra**

Att jobba i grupp (psykologi)

▶ **Läs mig**

Det här ska ni göra

> Lära er sätta upp JavaScript-projekt från scratch

> Skapa ett quiz med TypeScript i grupp

> Förstå agilt arbetssätt

Börja såhär (i grupp)

Förslag: en person gör stegen 2-10 på sin dator via skärmdelning.

▶ **1. Installera Node.js**

▶ **2. Starta GitHub Classroom-repo**

▶ **3. Installera Vite & TypeScript**

▶ **4. Fyll i gruppkontrakt**

▶ **5. Fixa .gitignore**

▶ **6. Fixa config-filen**

▶ **7. Installera ESLint**

▶ **8. Installera Prettier**

▶ **9. Installera Sass eller Tailwind för CSS**

▶ **10. Aktivera publicering av sidan på GitHub Pages**

▶ **11. Testa att allt fungera**

▶ **12. Ladda ner repot till varje gruppmedlems dator**

▶ **13. Test-merge:a via branches**

▶ **14. Ta en paus!**

▶ 🤘 **Individuell övning**

Projektet ska innehålla följande dependencies:

Dependency	Användningsområde
Vite	För att köra och publicera projektet
ESLint	För konsekvent kodstil
TypeScript	För att felsäkra JavaScript
Sass eller Tailwind	För att generera komprimerad CSS
Prettier	För att skriva konsekvent kod

Tips inför er planering

1. Rita en wireframe/ett flöde för hur programmet ska fungera.
2. Bryt ner wireframen i issues i GitHub.

3. Skriv pseudokod för varje issue.
4. Fundera på vad ni kan sedan innan; vad har ni gjort i övningar och webshops-uppgiften som ni kan återanvända.

Commits

Kom överens om en commit-stil ni vill använda, dvs. hur ni skriver era commit-meddelanden.

Viktigt : när ni jobbar i grupp/gemensamt så ska ni använda funktionen "co-author commits", se här . Detta för att era "live-kodningssessioner" ska synas på varje individuell persons "bidrag" för bedömningsunderlag. (Enklast att göra i GitHub for Desktop - då behöver man bara trycka på en knapp)

▶ **Tips och rekommendationer**

Uppgiftsbeskrivning

Ni får välja quiz-tema själva.

För G

- Skapa ett quiz med minst 20 frågor
- Varje fråga ska ha 3 svarsalternativ och endast 1 svarsalternativ ska vara korrekt
- Frågorna ska presenteras i slumpmässig ordning, och du ska visa 10 frågor per spelomgång
- Om användaren väljer att spela igen, så ska inte samma 10 frågor komma upp på nytt
- Du ska få poäng för rätt svar
- Det ska bara visas en fråga åt gången på skärmen
- Det ska finnas en tidräkning (uppåt). Tidräkningen ska stanna när alla frågor har besvarats.
- Det ska visas en bekräftelseruta som visar hur många frågor spelaren svarade rätt på (av totalt antal frågor), och hur lång tid det tog.

- Varje person ska ha gjort minst två pull requests.

För extra utmaning

▶ **Ej krav**

Betyg och bedömning

Krav för IG

- Du har inte deltagit i arbetet och/eller gjort commits (det syns tydligt i Insights-fliken på ert repo)
- Du/ni har inte följt instruktionerna
- Du/ni uppvisar avsaknad av kunskaper i såväl HTML, CSS som JavaScript och du/ni har ignorerat semantik och tillgänglighet

Krav för G

JavaScript

- Koden är korrekt formaterad.
- Koden innehåller enstaka buggar, men fungerar på det stora hela.
- Ni har gjort åtminstone en mobilversion; om ni återanvänder samma layout i tablet/desktop så går det bra.
- Sidan ska vara någorlunda snygg (CSS).
- Ni har checkat av "vanliga misstag" från tidigare uppgifter, såväl HTML & CSS som JavaScript. Se här , här och här .
- Sidan är tillgänglighetsgranskad.
- Sidan (HTML-koden) är validerad.
- Ni har gjort en Lighthouse-analys på live-sidan.
- Koden är skriven i TypeScript.
- Ni har skrivit minst 1 interface.
- Ni har använt minst 1 modul-fil (t.ex. lagt frågorna i en separat fil).

Agila

- Ni ska ha gjort minst två sprint-planeringar inkl. estimat (story points) i GitHub eller Trello och dokumenterat dessa i ert repo.
- Ni har valt en scrum master (dokumentera vem).

- Ni ska ha daily standups varje arbetsdag (ej helg) och dokumenterat dessa i ert repo. Ta gärna en skärmdump på er board varje dag och klistra in i dokumentationen så att jag kan se hur den utvecklas. Använd co-authoring-funktionen på GitHub om ni gör en gemensam commit.
- Ni ska efter avslutad sprint hålla retros och dokumentera dessa i ert repo.
- Ni ska ha deltagit i att presentera för produktägare
- Ni ska ha deltagit i avslutande presentation

För plus i kanten

▶ **Ej krav**

Vanliga frågor

Får vi ändra i reglerna för ESLint/Prettier?

Ja, det går bra.

Får vi lägga till paket?

Ja, det går bra. Däremot får ni inte använda ett ramverk såsom Vue, Svelte eller React.

Får vi ta bort TypeScript?

Nej.

Får vi använda Tailwind?

Ja, det går bra.

Resurser

- Klona repo till egen profil - ⚠ Om du har ett publicerings-script behöver du ta bort det innan push (och lägga till manuellt via webben), eller installera SSH-nycklar

.

Vanliga misstag 🚨

Se listan från individuell uppgift

Du hittar den här.

DOMContentLoaded (2024-12-27)

Då JavaScript-filen (eller TypeScript-filen snarare) länkas in sist i HTML-dokumentet så behöver ni inte använda er av `DOMContentLoaded` -funktionen; JavaScriptet körs först när HTML-strukturen är inladdad.

Detta är snarare ett old-school "hack" för att garantera att HTML-strukturen är inladdad.

Spara som variabel

Det finns någonting som kallas för "Big O " vars syfte är att räkna ut hur snabbt ett program är. Det är ingenting ni behöver lära er, men en liten föreklad minnesregel man kan ha i bakhuvudet är denna:

Varje gång ni skriver en punkt så blir programmet 1 snäpp långsammare.

I exemplet nedan har ni 2 punkter = programmet blir "2 långsamt".

typescript
```
document.getElementById("nextQuestionBtn")!.addEventListener("click", han
```

Om ni inte sparar referensen till knappen `nextQuestionBtn` i en variabel, utan använder `document.getElementById` varje gång, t.ex. såhär:

typescript
```
document.getElementById("nextQuestionBtn")!.addEventListener("click", han

// nån annan kod här emellan…

document.getElementById("nextQuestionBtn")!.setAttribute("disabled", "tru
```

```
6
7   // nån annan kod här emellan…
8
9   document.getElementById("nextQuestionBtn")!.removeAttribute("disabled");
```

I exemplet ovan blir programmet då "6 långsamt".

Skulle ni istället ha det som en variabel blir programmet "4 långsamt":

typescript

```
1   const nextQuestionBtn = document.getElementById("nextQuestionBtn");
2
3   nextQuestionBtn.addEventListener("click", handleNextQuestion);
4
5   // nån annan kod här emellan…
6
7   nextQuestionBtn.setAttribute("disabled", "true");
8
9   // nån annan kod här emellan…
10
11  nextQuestionBtn.removeAttribute("disabled");
```

Besparingen här kan tyckas minimal (och det är den), men bra att "sätta denna princip i ryggmärgen" att om det används mer än 1 gång ⇒ spara i en variabel.

Ta ovan exempel gånger 10, 20 eller 200 för ett stort program, och hastighetsbesparingen på 2X börjar bli signifikant.

As

Det finns två sätt att skriva på i TypeScript när man ska definiera en typ, ni avgör själva vilken ni tycker är mest läsbar/vill använda:

typescript

```
1   const landingPage = document.querySelector('#landingPage') as HTMLElement
2   const landingPage: HTMLElement = document.querySelector('#landingPage');
```

Gruppera kod

- Exporterade funktioner/metoder/variablet brukar läggas högst upp eller längst ner i dokumentet så att de är lätta att hitta.
- Gruppera alla exporterade metoder så att de "sitter ihop", alt. använd export som "nyckelord":

typescript

```typescript
function a() { /* nån kod */ }
function b() { /* nån kod */ }
function c() { /* nån kod */ }

export { a, b, c }
```

Senast uppdaterad: 2024-12-27 08:12

Föregående sida
Inför grupparbetet

Nästa sida
Det funkar inte...