OBS: Det är första gången som vi gör en uppgift med olika delar och steg inom delarna. Tidigare har U1 bara gått ut på att ni skulle, på egen hand, programmera en viss app. Vi hoppas att uppdelningen ger en mer lärorik uppgift. Vi ber om ursäkt om vi har missat något, hör gärna av er om det är något som inte verkar stämma.

U1 - Inlämningsuppgift 1

U1 består av två delar: Kodning och Flow Charting.

I koddelen ska du koda en app (Averager) med hjälp av ganska detaljerade instruktioner. Lärandemålen för denna del är: att förstå existerande kod, att på nätet läsa information om befintliga JS-funktioner, att koda egna lösningar baserat på liknande exempel, att med rätt termer förklara vad en viss kod gör. Lösningen som föreslås, och som du följer under uppgiften, är strukturerad på ett sätt som vi hoppas ger dig viktiga insikter i programmering.

I koddelen ingår att spela in videos där du förklarar hur koden som du har skapat (eller tillsammans med andra) fungerar. Detta för att det är extremt viktigt att du ser till att du förstår koden som du lämnar in.

En Flow Chart används för att planera / synliggöra hur man tänker lösa ett problem. I U1 har vi skapat en enkel Flow Chart för appen Averager som du ska studera och förstå. Sedan ska du ändra i flow-chart så att den skulle fungera för en något annorlunda version av Averager-appen. Lärandemålen här är att introduceras till Flow Charts och att börja så smått utveckla strategier för att lösa programmeringsproblem.

Allt som behövs för U1 finns här:
https://mau.box.com/s/83zza7wedwnm5v8vbr96kuyaxakojh0a

Averager

Averager är en app som gör ganska onödiga grejer men som är en bra övning.

Det är viktigt att du löser uppgiften i ordning (1, 2, 3 etc). Det är bara att följa instruktionerna i JS-filerna i de olika mapparna.

All kod du skriver ska hanteras via en repository (bara en repository för alla deluppgifter) på Github.

Du ska också ladda upp på Canvas in en zip-fil med all kod som du skapar. Använd mappstrukturen och filerna som redan finns i mappen Averager. Ingen av mina videos ska dock finnas med i det du lämnar in.

Glöm inte att inkludera en länk till din Github-repository. Det ska göras i filen Averager/AveragerApp/index.js. Se instruktionerna där för mer info.

Alla måste lämna in sin individuella lösning på U1 på Canvas. Det är helt ok att lösa U1 tillsammans med andra. Ni kan till och med lämna in samma kod. Ni kan dock INTE lämna in samma videos. De måste vara individuella. Det är superviktigt att var och en av er visar att ni förstår koden ni lämnar in.

Flow Chart

I flowChart-mappen finns det ett dokument som visar:

1. En lista över alla funktioner som används i Averager
2. En enkel Flow Chart för Averager.

Listan har vi lämnat ut som ett exempel på en funktionslista. I kommande uppgifter kommer ni att behöva skapa egna funktionslistor.

Uppgiften går ut på att studera Averagers FlowChart så att du förstår hur den är uppbyggd och hur den är relaterad till koden. Vi rekommenderar att du studerar den efter det att du har kodat klart Averager. Notera att inte alla funktioner har tagits med i FlowCharter, vi har endast tagit med de som kallar på andra funktioner.

När du väl har förstått kopplingen mellan FlowChart och koden så ska du skapa din egen FlowChart för en modifierad version av Averager: AveragerGroupSame. Denna version av appen fungerar exakt likadant som vanliga Averager med en skillnad: När användaren markerar en siffra så kommer alla de andra likadana siffror att automatiskt markeras också. Detsamma gäller om användaren avmarkerar en siffra. Så om användaren markerar en 5 så kommer alla andra 5 i grinden att också markeras. Och de kommer också att inkluderas när programmet räknar ut summan, genomsnittet och antalet. Om användaren istället avmarkerar en 5 så kommer alla andra 5 att avmarkeras automatiskt. Och då ingår de inte när man räknar ut summan, genomsnittet och antalet.

Du behöver inte koda detta, bara modifiera FlowChart för att vara en lösning på detta nya problem.

Du kan utgå från att det finns en funktion som heter arrayOfSameNumbers(number). Funktionen tar alltså emot ett argument som är en siffra och returnerar en array (referens till array) som inkluderar alla numberDivs som innehåller den siffran.

Du ska även spela in en kort video som förklarar ändringarna som du har gjort i Flow Chart och hur du tror att programmet nu kommer att fungera.

Videos

Du kommer att behöva skapa några videos där du förklarar kodsnuttar eller funktioner. Du ska alltid förklara koden rad för rad. Det är enormt viktigt att du använder rätt termer och det kan bli kompletteringar på detta. Vi förväntar oss korrekta beskrivningar och kommer att vara strikta. Du kan förklara på svenska eller engelska.

Några termer som ni förväntas använda korrekt:

Variabler	En **variabel** "hämtas" inte. En variabel måste **deklareras**. Den kan också få ett **värde** "**assignat**" (**tilldelat**) till sig (eller ett värde kan **sparas** i den), eller värdet som finns sparat i dem **används** på något sätt.
For-loopar	For-loopar (och andra loopar) **itereras** Varje "körning" (fel ord) är en **iteration** (rätt ord) Variabeln som "räknar" i loopen kallas **counter**. Counter **inkrementeras** / **dekrementeras** i varje iteration Inför varje iteration kontrolleras om **villkoret uppfylls** (är sant) En for-loop **avslutas** när villkoret inte längre fylls.

Array	En array sparas aldrig i en variabel. Det är bara **referensen till arrayen** som sparas i variabeln. I övningen "More fun with Arrays" från lektion 4 står det att man inte brukar prata så mycket om referenser till arrayer och det stämmer. Men i U1 måste ni göra det. En array består av ett gäng värden (eller **element**). Varje element har sin plats i arrayen. Platsen identifieras med en **index**-siffra. I lektioner har Erik har kallat platsen "cell". På engelska kallas den "**slot**", detta är mer korrekt.
Funktioner	En funktion "hämtas" inte. En funktion måste **deklareras**. En funktion "**tar emot**" **parametrar**. Dessa parametrar får ett värde när funktionen "**anropas**". "**Argument**" används ofta som synonym för "parameter". Vid korrekt bruk ska "parameter" användas när man pratar om funktionsdeklaration (då vet vi inte vilket värde parameter kommer att få). "Argument" ska användas när man pratar om funktionsanrop (och vi vet vilket värde det har). Det är ok om ni, i denna övning, använder bara "parameter" eller bara "argument" när ni förklarar er kod. En funktion kan ha ett **returvärde** (markeras med kommandot *return*). I så fall **returnerar** den det värdet som skrivs efter *return*. Annars returnernar funktionen värdet *undefined*.
HTML-element	HTML-element "hämtas" inte. Ett korrekt sätt att uttrycka sig är "Referensen till elementet X sparas i (eller assignas till) variablen Y. .querySelector() returnerar referensen till ett element, inte elementet i sig.

Upplägg

Tisdag 01.02	Uppgiften delas ut. Grundläggande info om Github och hur man använder den på enklaste sätt.
Torsdag 03.02	Eventuella frågor om U1. Undantag: FlowChart. Det är viktigt att ni kämpar med den på egen hand. Möjlighet att fråga även om tidigare övningar eller begrepp. Eventuella frågor om Github.
Fredag 04.02	Kl07:00: Lösningar till delarna 1-4 publiceras. Kl10:00: Genomgång av publicerade lösningar. Andra frågor.
Söndag 06.02	Kl23:59: Den föreslagna lösningen till Average App publiceras Din lösning behöver inte vara som den publicerade. Den publicerade är kopplat till FlowChart. Vi förväntar oss lösningar från er som liknar väldigt mycket den som publiceras på söndag eftersom ni har ganska noggranna instruktioner.
Tisdag 08.02	Kl23:59: Deadline. Canvas-uppgiften stänger.

Style-Guides

Den slutgiltiga app-koden i mappen AveragerApp ska följa dessa regler:

A. All kod måste vara korrekt indenterad (2 spaces per nivå). Detta gäller HTML-, CSS- och JS-kod.
   Notera att vi inte kommer att lämna feedback om var i koden ni har indenterat fel.
B. Ni får inte använda några globala variabler i JS-koden. Ni måste dock använda globala funktioner.
C. Alla namn på variabler, klasser, id:en, funktioner, etc måste vara genomtänkta och professionella.
D. Koden i JS-filen ska följa denna struktur (notera att det skiljer sig från tidigare övningar):
    A. Globala variabler (Ni inte får dock inte ha några globala variabler i AveragerApp)
    B. Funktionsdeklarationer (tidigare har det sagts att detta ska komma sist)
    C. Eventlisteners för befintliga HTML-element
    D. Direktkod

Betyg

Det går bara att få U eller G i denna uppgift.