

Laboration 3

Objekt i JavaScript

1M322 Webbteknik 2, 7,5hp
Medieteknik



1. Öppna mappen L3

Öppna mappen *L3* i Visual Studio Code (VSC) och öppna filen *index.html* i Live Server.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure with a folder named "L3" containing "css" (with files "layout.css" and "style.css"), "img", "js" (with file "script.js"), and "index.html".
- Editor View:** The file "index.html" is open, showing the following code:

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>1ME322, Laboration 3 – Objekt i JavaScript</title>
6     <link rel="stylesheet" type="text/css" href="css/layout.css">
7     <link rel="stylesheet" type="text/css" href="css/style.css">
8     <script type="text/javascript" src="js/script.js"></script>
9   </head>
10  <body>
11    <header>
12      <h1>1ME322 Webbteknik 2</h1>
13      <h3>Laboration 3 – Objekt i JavaScript</h3>
14    </header>
15    <main>
16      <h2>Spelet Hänga gubbe</h2>
17      <p>Välkommen till spelet Hänga gubbe. Datorn väljer ett ord slumpmässigt och du gissar ordet genom att trycka på knapparna på tangentbordet. När du har gjort en felaktig gissning kommer det att trivas en hängande gubbe i hängmattan. Spelet är över när du har gjort sju felaktiga gissningar.</p>
18      <button>Starta spelet</button>
19      <table border="1">
```
- Browser Preview:** A separate window shows the running application. The title bar says "1ME322, Laboration 3 - Objekt i JavaScript". The main content area displays the game's header ("1ME322 Webbteknik 2" and "Laboration 3 - Objekt i JavaScript"), the game title ("Spelet Hänga gubbe"), a descriptive paragraph about the game rules, a "Starta spelet" button, and a keyboard input field containing letters A through Ö.

2a. Planering och struktur för programmet

I denna laboration ska du skapa ett program för spelet Häng gubbe. Det går ut på att ett ord väljs slumpmässigt och du kan sedan gissa på bokstäver. Varje gång du gissar på en bokstav som inte ingår i ordet, utökas en bild, så att det till slut blir en hängd gubbe. Du kan totalt gissa fel sex gånger, innan gubben hängs. Men lyckas du få alla bokstäver rätt innan dess, klarar sig gubben.

Programmet består av ett antal steg och det blir då också några funktioner för dessa. Innan man börjar skriva koden, är det bra att planera vad som ska göras och skapa en struktur för programmet. Det beskrivs här, innan vi kommer in på övningarna med kodskrivandet.

Studera gränssnittet

- Öppna filen *index.htm*
- Det finns ett antal delar i gränssnittet:
 - Startknapp, för att starta spelet.
 - Div-element "board" som omger de delar som används under spelet:
 - Bokstavsknappar, för att gissa på bokstäver.
 - Div-element "letterBoxes", där det kommer finnas en ruta per bokstav i det ord som ska gissas.
 - Då spelet startas läggs dessa in som span-element. I CSS-koden utformas de som vita rutor. Då man sedan gissat rätt på en bokstav, skrivs den in i rutan. Se stilsättning i filen *style.css*.
 - Img-element för att visa bilden med galgen och gubben.
 - Div-element "message" med meddelande till användaren, då spelet är över.

Spelet Häng gubbe

Välkommen till spelet Häng gubbe. Datorn väljer ett ord slumpmässigt och du gissar ordet genom att trycka på bokstavsknapparna.

Starta spelet

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	Ä	Å	Ö	

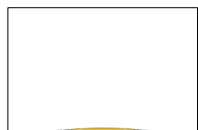
S A

Tyvärr, gubben hängdes. Rätt svar är SOMMAR
Det tog 10.8 sekunder.

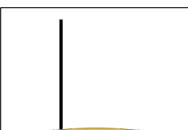
Bildfiler

För att bygga upp bilden med den hängda gubben, ska bilden i img-elementet bytas ut.

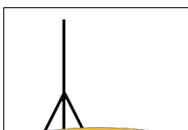
Följande bilder finns i mappen *img*.



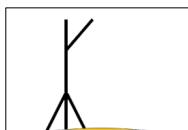
h0.png



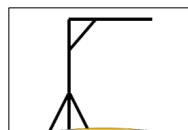
h1.png



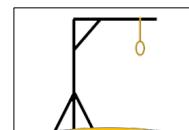
h2.png



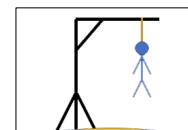
h3.png



h4.png



h5.png



h6.png

2b. Planering och struktur för programmet

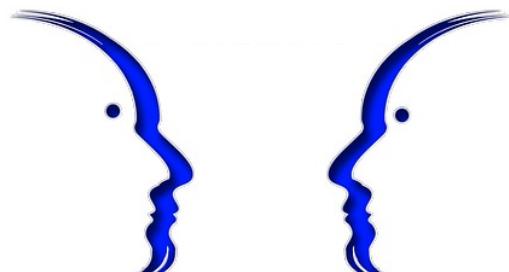
På nästa sida beskrivs ett förslag på struktur.

Men, titta inte
på det ännu.

Stanna upp ett tag och fundera
själv på hur programmet skulle
kunna struktureras i olika
funktioner.



Diskutera gärna
denna del med
en kurskamrat.



2c. Planering och struktur för programmet

Innan man skriver kod, är det ofta bra att tänka igenom vilka funktioner som behövs och skriva korta beskrivningar. Dessa beskrivningar kan man sedan ha som kommentarer i koden, för att förklara funktionerna. När man sedan börjar skriva koden, kanske det tillkommer en del funktioner (man kanske delar upp en funktion i flera), men det underlättar att först ha en genomtänkt planering och struktur.

Först skapades listan i den vänstra kolumnen. Det är *init*-funktionen och funktioner som ska anropas, då man klickar på knapparna. Sedan kompletterades det med funktionerna i den högra kolumnen.

Funktionen *init*

- Initiering av globala variabler.
- Koppling av funktioner till knappar.

Funktionen *startGame*

- Anropas då man klickar på knappen "Starta spelet".
- Välj ord slumpmässigt.
- Visa bokstavsruor.
- Visa första bilden, h0.png.

Funktionen *guessLetter*

- Anropas då man klickar på en bokstavsknapp.
- Avläs vald bokstav ur button-elementets value-attribut.
- Gå igenom alla bokstäver i ordet och kontrollera om vald bokstav finns (kan finnas flera gånger i ordet):
 - I så fall skrivs den in i motsvarande ruta.
- Om bokstaven ej finns, byts bilden till nästa bild.
 - Om man då visar den sista bilden (h6.png), avslutas spelet med hängd gubbe.
- Annars kontrolleras om alla bokstäver är klara.
 - I så fall avslutas spelet.

Funktionen *randomWord*

- Ta fram ett slumptal mellan 0 och antal ord i en lista av ord.
- Indexera listan med slumptalet och spara valt ord i en global variabel.

Funktionen *showLetterBoxes*

- Gå igenom valt ord och skapa en HTML-kod med ett span-element för varje bokstav. (I CSS-koden finns redan kod som gör att ett span-element visas som en ruta.)
- Lägg in HTML-koden i elementet med id "letterBoxes".

Funktionen *endGame*

- Parameter för att veta hur spelet slutade.
- Om gubben blev hängd, skrivs ett meddelande med det rätta ordet.
- Annars skrivs ett meddelande med en gratulation.

2d. Planering och struktur för programmet

Utifrån planeringen med uppdelning i funktioner på föregående sida, tar man fram en lista på vilka globala variabler som behövs. Även här kan det sedan tillkomma flera, men en första planering är alltid bra att göra, innan man börjar skriva koden.

Globala variabler

- wordList
 - Array med ett antal ord, där man sedan väljer ett slumpmässigt.
- selectedWord
 - Det ord som valts slumpmässigt och som användaren ska gissa på.
- letterBoxes
 - Array med referenser till de span-taggar som utgör rutor för bokstäverna i ordet.
- hangmanImg
 - Referens till img-elementet med bilden för galgen och gubben.
- hangmanImgNr
 - Nummer för aktuell bild (0-6), för den bildfil som visas (så man sedan kan veta vilket som blir nästa bild).
- msgElem
 - Referens till div-elementet för meddelanden.

Variabeln *selectedWord* tilldelas ett nytt värde i funktionen *randomWord*.

Variabeln *letterBoxes* tilldelas i funktionen *showLetterBoxes*, då *span*-elementen lagts in i HTML-koden.

Variabeln *hangmanImgNr* tilldelas värdet 0, då bilden byts till *h0.png* i funktionen *startGame*.

Övriga variabler initieras i *init*-funktionen.

När vi nu har en struktur med en uppdelning i funktioner och en lista på globala variabler, kan koden börja skrivas.

All programkod skriver
du i filen *script.js*.



script.js

3. Globala variabler och init-funktionen

Börja med att lägga in listan med globala variabler och gör initieringar i *init*-funktionen.

Globala variabler

- I filen *script.js* finns redan *wordList* som en konstant.
Övriga variabler i listan i övning 2d deklarerar du med *var* eller *let*.
- Glöm inte att också lägga in kommentarer.

```
var selectedWord;  
var letterBoxes;  
var hangmanImg;  
var hangmanImgNr;  
var msgElem;
```

Funktionen *init*

- Startknappen:
 - Deklarera variabeln *startGameBtn* med *let* och lägg in en referens till knappen "*startGameBtn*" i den.
 - Egentligen behövs ingen variabel endast för att göra nedanstående, men du ska i en senare övning lägga till mer kod för startknappen, så spara referensen i en variabel redan nu.
 - Lägg in kod, så att funktionen *startGame* anropas, då man klickar på knappen.
- Bokstavsknapparna:
 - Ta fram en array med referenser till till alla knappar i *div*-elementet "*letterButtons*".
 - Gå igenom allihop i en loop och se till att funktionen *guessLetter* anropas då man klickar på dem.
 - Samma funktion ska alltså anropas oavsett vilken av knapparna man klickar på.

```
let letterButtons = document.getElementById("letterButtons").getElementsByTagName("button");  
for (let i = 0; i < letterButtons.length; i++)  
    letterButtons[i].onclick = guessLetter;
```

- Elementen för bild och meddelanden:
 - Lägg in en referens till *img*-elementet "*hangman*" i variabeln *hangmanImg*.
 - Lägg in en referens till *div*-elementet "*message*" i variabeln *msgElem*.
 - Dessa två variabler är de globala variablerna som du redan deklarerat, så här använder du inte *let* eller *var*, eftersom det då skulle bli nya variabler.

4. Skal till alla funktioner

Nu lägger du in deklaration av de övriga funktionerna.

Börja här med att lägga in "skalet" med *function*, namnet och en kommentar.

Övriga funktioner

- Lägg in skalet för funktionen *startGame*.

```
// Initiera nytt spel. Välj ord, visa bokstavsrum,  
// visa första bilden (tom bild), sätt bildnummer till 0.  
function startGame() {  
  
} // End startGame
```

- Lägg på samma sätt in skal för funktionerna:

- *randomWord*
- *showLetterBoxes*
- *guessLetter*
- *endGame*
 - Funktionen *endGame* ska också ha en parameter, som du kan kalla *manHanged*.
 - Då man sedan anropar funktionen skickar man med *true* eller *false*, så att man i funktionen kan avgöra vilket meddelande som ska skrivas ut.

Testa i webbläsaren

- Öppna Verktyg för webbutvecklare och se till att du har fliken Konsol och att det är markerat att fel ska visas.
- Kontrollera att du inte får några felmeddelanden.
- I övrigt ska inget hända, eftersom funktionerna ännu så länge är tomta.

5a. Funktionerna startGame och randomWord

I övning 2c och 2d beskrevs funktionerna enligt rutorna till höger. Detta ska nu översättas till kod.

Kod i funktionen *startGame*

- Lägg in ett anrop av funktionen *randomWord*.
- Lägg också in ett anrop av funktionen *showLetterBoxes*.
- Lägg in bilden "img/h0.png" i img-elementet som refereras av *hangmanImg*.
- Lägg in 0 i variabeln *hangmanImgNr*.

- Välj ord slumpmässigt.
- Visa bokstavsruor.
- Visa första bilden, h0.png.
- Variabeln *hangmanImgNr* tilldelas värdet 0

Testa programmet i webbläsaren

- Kontrollera att den första bilden (en liten kulle) visas, då du klickar på startknappen.
- Kontrollera också att du inte får några felmeddelanden i konsolen.



Kod i funktionen *randomWord*

- Generera ett slumptal mellan 0 och antal ord i *wordList*.
Spara slumptalet i variabeln *wordIndex*.
- I *selectedWord* sparar du det ord i *wordList* som
indexeras med *wordIndex*.
 - Variabeln *wordIndex* ska vara lokal, så använd *let*, medan *selectedWord* är den globala variabeln,
som du redan deklarerat i början av programmet. För den ska du alltså inte använda *let* i funktionen.

- Ta fram ett slumptal mellan 0 och antal ord i en lista av ord.
- Indexera listan med slumptalet och spara vart ord i en global variabel.
- Variabeln *selectedWord* tilldelas ett nytt värde.

Testa programmet i webbläsaren

- Det händer inget mer nu, men kontrollera att du inte får något felmeddelande i konsolen.

5b. Funktionen showLetterBoxes

Nu tar vi nästa funktion som anropas från *startGame*.

Kod i funktionen *showLetterBoxes*

- Definiera variabeln *newCode* med *let* och lägg in en tom sträng (två citattecken, utan något mellan dem) i den.
 - Variabeln ska bli en textsträng med HTML-koden för bokstavsrutorna.
- Lägg in en *for-loop*, där du går igenom alla tecken i *selectedWord*.
 - I loopen lägger du till strängen " " till *newCode*.
 - Använd operatorn *+=* för att lägga strängen till *newCode*.
 - För att CSS-koden ska funka och en "tom" ruta visas, kan inte span-elementet vara tomt eller endast innehålla ett vanligt blanktecken, så därför använder vi nonbreakable space ().
- Efter loopen lägger du in *newCode* i elementet med id "*letterBoxes*".
- I variabeln *letterBoxes* tar du fram en array med referenser till alla *span*-taggar. Börja med *getElementById* och fortsätt sedan med *getElementsByName*.

```
document.getElementById("letterBoxes").innerHTML = newCode;  
letterBoxes = document.getElementById("letterBoxes").getElementsByTagName("span");
```

Testa programmet i webbläsaren

- Klicka på knappen "Starta spelet".
Du ska då få ett antal rutor under bokstavsknapparna.



6a. Funktionen guessLetter

I övning 2c beskrevs funktionen enligt rutan till höger. Detta ska nu översättas till kod.

Kod i funktionen *guessLetter*

- Lägg in en rad för att avläsa knappens bokstav ur *button*-elementet och sparar i den lokala variabeln *letter*.
 - En referens till den knapp som användaren klickat på finns i *this*.
 - Bokstaven finns i *value*-attributet, se filen *index.htm*.

```
let letter = this.value;
```

- Avläs vald bokstav ur *button*-elementets *value*-attribut.
- Gå igenom alla bokstäver i ordet och kontrollera om vald bokstav finns (kan finnas flera gånger i ordet):
 - I så fall skrivs den in i motsvarande ruta.
- Om bokstaven ej finns, byts bilden till nästa bild.
 - Om man då visar den sista bilden (h6.png), avslutas spelet med hängd gubbe.
- Annars kontrolleras om alla bokstäver är klara.
 - I så fall avslutas spelet.

- Lägg in en *for*-loop där du går igenom alla tecken i *selectedWord*.
 - I loopen ska du ha en *if*-sats där du jämför *letter* med det tecknen i *selectedWord* som bestäms av loopvariabeln *i*.
 - Använd *charAt* för att ta ut tecknet ur textsträngen.
 - Om de är lika lägger du in *letter* i motsvarande textruta.
 - Använd *letterBoxes* och indexera med *i*, för att få en referens till textrutans element.
 - Glöm inte *innerHTML*.

Testa i webbläsaren

- Även om du nu endast gjort de första punkterna i funktionen, är det tillräckligt för att kunna testa spelet.
- Klicka på "Starta spelet".
- Klicka sedan på bokstavsknappar.
- Då bokstaven finns i ordet, ska den skrivas ut i rutorna.



6b. Funktionen guessLetter

Du fortsätter här med kod i funktionen *guessLetter*.

Kod i funktionen *guessLetter*

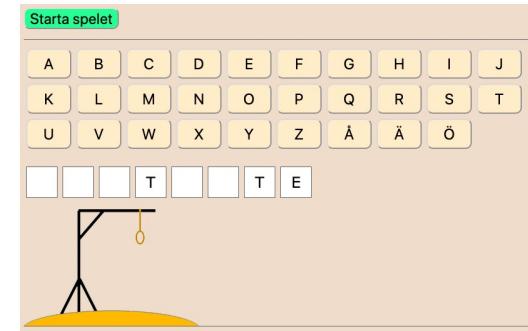
- För att avgöra om bokstaven finns i ordet, behövs en "flagga":
 - Före loopen lägger du in *false* i variabeln *letterFound*.
 - Inuti *if*-satsen i loopen lägger du in *true* i *letterFound*.
 - Använd *let* då du inför variabeln före loopen, men sedan refererar du till den utan *let*.
- Efter loopen lägger du in en ny *if*-sats där du kontrollerar om bokstaven ej fanns, dvs *letterFound* är fortfarande *false* efter loopen
 - I så fall ska nästa bild ska visas.
 - Nästa bild bestäms av *hangmanImgNr*, så räkna först upp den variabeln med 1.
 - Lägg sedan in "*img/h*" + *hangmanImgNr* + ".png" i *img*-taggen som refereras av *hangmanImg*. (Glöm inte *src*)
 - I den inre *if*-satsen kontrollerar du om man nu kommit fram till den sista bilden, dvs om *hangmanImgNr* är 6.
 - Denna *if*-sats ska alltså ligga inuti den första.
 - I så fall ska spelet avslutas, genom att du anropar funktionen *endGame*, med parametern *true*.

- Avläs vald bokstav ur button-elementets value-attribut.
- Gå igenom alla bokstäver i ordet och kontrollera om vald bokstav finns (kan finnas flera gånger i ordet):
 - I så fall skrivs den in i motsvarande ruta.
- Om bokstaven ej finns, byts bilden till nästa bild.
 - Om man då visar den sista bilden (h6.png), avslutas spelet med hängd gubbe.
- Annars kontrolleras om alla bokstäver är klara.
 - I så fall avslutas spelet.

```
if ( ... bokstaven ej fanns ... ) {  
    ... byt bild ...  
    if ( ... sisa bilden ... ) {  
        ... avsluta med true ...  
    }  
}
```

Testa i webbläsaren

- Nu kan du testa om bilderna byts ut. Klicka på knappen "Starta spelet" och sedan på bokstavsknappar
- Varje gång du trycker på en bokstav som inte finns i ordet, ska bilden bytas ut.



6c. Funktionen guessLetter

Nu återstår den sista punkten i funktionen.

Kod i funktionen *guessLetter*

- För att avgöra om alla bokstäver är korrekta, ska du kontrollera om alla bokstavsrum är ifyllda. Du behöver då en räknare, som du sedan kan jämföra med antalet bokstäver i ordet.
 - Före loopen lägger du in 0 i variabeln *correctLettersCount*.
 - Inuti loopen lägger du in ytterligare en *if*-sats under den första. I denna *if*-sats kontrollerar du om rutan ej är tom, dvs om *letterBoxes[i].innerHTML* är skilt ifrån " ".
 - I så fall innehåller den en bokstav och du räknar upp *correctLettersCount* med 1.
- Nu kommer vi in på den inringade punkten i rutan ovan.
 - Lägg in en *else-if*-del till den yttre av *if*-satserna som ligger under loopen.
 - Strukturen blir alltså nu enligt vidstående bild.**
 - I *if*-satsen jämför du *correctLettersCount* med antal tecken i *selectedWord*.
 - Är de lika, anropar du funktionen *endGame*, med parametern *false*.

- Avläs vald bokstav ur button-elementets value-attribut.
- Gå igenom alla bokstäver i ordet och kontrollera om vald bokstav finns (kan finnas flera gånger i ordet):
 - I så fall skrivs den in i motsvarande ruta.
- Om bokstaven ej finns, byts bilden till nästa bild.
 - Om man då visar den sista bilden (h6.png), avslutas spelet med hängd gubbe.
- Annars kontrolleras om alla bokstäver är klara.
 - I så fall avslutas spelet.

```
for (...) {  
    if (...) {  
        ...  
    }  
    if (...) {  
        ...  
    }  
}
```

```
if ( ... bokstaven ej fanns ...) {  
    ... byt bild ...  
    if ( ... sisa bilden ... ) {  
        ... avsluta med true ...  
    }  
}  
else if ( ... antal korrekta bokstäver är lika med antal bokstäver i ordet ... ) {  
    ... avsluta med false ...  
}
```

Testa i webbläsaren

- Du får inget meddelande, då spelet är slut, eftersom *endGame* ännu så länge är tomt.
- Men testa programmet ändå, för att se att du inte får några felmeddelanden i debuggern.

7. Funktionen endGame

I övning 2c beskrevs funktionen *endGame* enligt vidstående lista.

Då du skapade skalet till funktionen i övning 4, la du också in en parameter kallad *manHanged*. Då du anropat funktionen i övning 6, har du skickat med *true* eller *false* till denna parameter.

Du ska nu skriva koden i funktionen.

- Parameter för att veta hur spelet slutade.
- Om gubben blev hängd, skrifs ett meddelande med det rätta ordet.
- Annars skrifs ett meddelande med en gratulation.

Kod i funktionen *endgame*

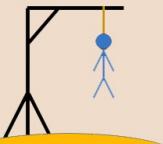
- I en *if*-sats kontrollerar du om *manHanged* är *true*.
 - I så fall skriver du i elementet för meddelanden (som du refererar till med *msgElem*) att gubben hängdes. Skriv också ut vad rätt ord är (dvs variabeln *selectedWord*).
- I en *else*-del lägger du in följande
 - Skriv ett meddelande som gratulerar användaren till att ha klarat av hela ordet.

Testa i webbläsaren

- Kontrollera att du får ut ovanstående meddelanden, då gubben blir hängd eller då du lyckas få fram hela ordet.

Starta spelet

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	Å	Ä	Ö	
S	K		I			O			



Tyvärr, gubben hängdes. Rätt svar är SKRIVBORD

Starta spelet

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	Å	Ä	Ö	
K	L	Ä	D	S	K	Å	P		



Gratulerar. Du kom fram till rätt ord.

Klart?

Är nu programmet klart?

Nja, du har nu grundstommen klar, så att all funktionalitet är klar.

Men det återstår en del tillägg, för att fixa till gränssnittet.

Nu kan man ju klicka på samma bokstav flera gånger. Då man klickat på en bokstav, ska knappen inaktiveras, så att man inte kan klicka på den igen. Det kan göras med egenskapen *disabled*.

Man kan också börja klicka på bokstäverna innan man startat spelet och något ord valts.
Det går också klicka på knappen för att starta spelet, innan man är klar med det spel som pågår.
Så alla dessa knappar behöver också aktiveras och inaktiveras vid en del ställen i programmet.

Ett annat tillägg som ska göras är att i funktionen *randomWord*, se till att inte samma ord väljs två gånger i rad.

Ett tredje tillägg är att mäta tiden, för att se hur lång tid spelet tar.

Dessa tillägg ska du nu göra i de kommande övningarna.

8a. Aktivera och inaktivera knappar

Förändringar ska ske på följande ställen:

- Då sidan laddats in, alltså i funktionen *init*.
 - Startknappen aktiveras och bokstavsknapparna inaktiveras.
- Då man startar spelet, alltså i funktionen *startGame*.
 - Startknappen inaktiveras och bokstavsknapparna aktiveras.
- Då spelet avslutas, alltså i funktionen *endGame*.
 - Startknappen aktiveras och bokstavsknapparna inaktiveras.
- Då man klickar på en bokstavsknapp, alltså i funktionen *guessLetter*.
 - Den knapp man klickat på ska inaktiveras.

Man ändrar aktivering genom att lägga in *true* eller *false* i egenskapen *disabled*:

- `knappReferens.disabled = true;` // Knappen inaktiveras
- `knappReferens.disabled = false;` // Knappen aktiveras

Globala variabler

För att kunna komma åt referenser till knapparna i de olika funktionerna, måste variablerna *startGameBtn* och *letterButtons* vara globala variabler. De finns nu som lokala variabler i *init*-funktionen.

- Flytta deklarationen av dessa två variabler från *init* till början av filen, där du har dina globala variabler.
 - Ta alltså bort nyckelordet *let* från raderna där de införs i *init*-funktionen och deklarera variablerna med *var* eller *let* i listan med övriga globala variabler.

```
var startGameBtn;  
var letterButtons;
```

Funktionen *init*

- Sist i funktionen lägger du in kod för att sätta startknappens *disabled* till *false*, vilket aktiverar den.
- I en *for-loop* går du igenom alla *letterButtons* och sätter *disabled* till *true*, vilket inaktiverar dem.

```
startGameBtn.disabled = false;  
for (let i = 0; i < letterButtons.length; i++)  
    letterButtons[i].disabled = true;
```

8b. Aktivera och inaktivera knappar

Här fortsätter du med koden från föregående sida.

Funktionen *startGame*

- Sist i funktionen lägger du in en likadan kod som den du la in i *init*, fast du sätter nu startknappen till *true* och *letterButtons* till *false*.

Funktionen *endGame*

- Sist i denna funktion lägger du också in samma kod som du la in i *init*, och här har du samma värden, dvs startknappen sätts till *false* och *letterButtons* till *true*.

Funktionen *guessLetter*

- I denna funktion har du en referens till knappen i *this*. I början av funktionen sätter du *disabled* för den till *true*.

```
this.disabled = true;
```

Testa i webbläsaren

- Startknappen ska först vara aktiv, men bokstavsknapparna inaktiva. De är "grå" och det händer inget om du klickar på dem.
- Klicka på knappen "Starta spelet", så inaktiveras startknappen, men alla bokstavsknappar blir aktiva.
- Klicka på knappar för bokstäver, så inaktiveras den knapp du klickar på.
- Gå igenom spelet tills gubben blir hängd eller du fått fram rätt ord. Då ska startknappen bli aktiv och bokstavsknapparna inaktiva.



9. Ta bort gammalt meddelande

Då man startar spelet igen, ska det gamla meddelandet från föregående spel tas bort.

Funktionen *startGame*

- Sist i funktionen lägger du till en programsats, där du lägger in en tom sträng i elementet för meddelanden.

Testa i webbläsaren

- Kör igenom spelet och klicka på startknappen igen.
- Då tas meddelandet längst ner på sidan bort.

10. Välj ej samma ord två gånger i rad

Då man startar spelet igen, ska inte samma ord som förra gången kunna väljas igen.

Detta sker i funktionen *randomWord*, som nu ska modifieras lite.

Förändringar i koden i funktionen *randomWord*

- Först i funktionen inför du en variabel som du kallar *oldWord* och lägg in *selectedWord* i den.
- De två programsatser som du redan hade i funktionen, lägger du i en *while*-loop.
 - Villkoret i loopen ska vara att *oldWord* är lika med *selectedWord*
 - Man går alltså runt i loopen och genererar ett nytt slumptal och ord, så länge det är likadant som det gamla ordet. Men då man får ett annat ord, lämnas loopen.

Testa i webbläsaren

- Kontrollera att du inte får något felmeddelande i debuggern, då du klickar på startknappen.

11. Mäta tiden för spelet

Det sista tillägget du ska göra är att mäta hur lång tid det tar för användaren att gissa ordet eller få gubben hängd.

Detta görs genom att vi först tar fram tiden då spelet startas och sedan tiden då spelet avslutas. Skillnaden blir då speltiden.

Tiden räknas ut i funktionen *endGame*, men starttiden måste vi spara i funktionen *startGame*. Det måste då läggas i en global variabel.

För tidsmätningen kommer funktionen *getTime* i *Date*-objektet användas. Den ger tiden i millisekunder från den 1 januari 1970. Detta har dock ingen betydelse här, utan vi kommer ta fram *getTime* vid två olika tillfällen och sedan räkna ut skillnaden mellan dem.

Globala variabler

Inför en global variabel som du kallar *startTime*.

```
var startTime;
```

Funktionen *startGame*

- I slutet av funktionen skapar du ett nytt *Date*-objekt, som du lägger i variabeln *now*. Sedan avläser du *getTime* och sparar i *startTime*.

```
let now = new Date();
startTime = now.getTime();
```

Funktionen *endGame*

- I början av funktionen lägger du in kod för att skapa ett nytt *Date*-objekt och sedan beräkna skillnaden mellan *getTime* i objektet och *startTime*. Tiden blir i millisekunder, så dividera med 1000, för att få det i sekunder.

```
let runTime = (new Date().getTime() - startTime) / 1000;
```

- I slutet av funktionen lägger du till en rad där du skriver ut *runTime* i elementet för meddelanden.

- Använd *toFixed(1)*, för att avrunda till en decimal.

```
msgElem.innerHTML += "<br>Det tog " + runTime.toFixed(1) + " sekunder.;"
```

Testa i webbläsaren

- Kontrollera att du får ut tiden.

Gratulerar. Du kom fram till rätt ord.
Det tog 19.9 sekunder.

12. Städa koden och lägg till kommentarer

Då du nu är klar med programmet, ser du över det och snyggar eventuellt till indenteringar, m.m.

Kontrollera också att du skrivit en kommentar för varje funktion och varje ny variabel som införs i koden.

