HÖGSKOLAN DALARNA

GIK339 – Labb 1

JavaScript intro, variabler och funkioner

Uppdaterad

2025-11-19

GIK339 – LABB 1

JAVASCRIPT INTRO, VARIABLER OCH FUNKIONER

För 4,5hp i kursen Dynamiska Webbapplikationer ska tre labbar och ett onlineprov utföras. Detta är den första labben. Denna labb utförs individuellt och med kamratrespons.

Läs igenom instruktioner för förberedelser, uppgifter och inlämning noggrant och hör av dig direkt till mie@du.se om du har några frågor/behöver förtydligande.

Lycka till!

HÖGSKOLAN DALARNA

GIK339 – Labb 1

JavaScript intro, variabler och funkioner

Uppdaterad

2025-11-19

Innehåll

HÖGSKOLAN DALARNA

GIK339 – Labb 1

JavaScript intro, variabler och funkioner

Uppdaterad

2025-11-19

Förberedelser

Labben ska helst lämnas in som GitHub-länk. För att förbereda för det, utför följande steg:

1. Skapa ett lokalt repository på din dator, exempelvis enligt instruktioner i *Git, GitHub, & GitHub Desktop for beginners*.
2. Döp det till **gik339-[ditt-du-användarnamn]-labb1**.

Litteraturanvisningar

Att ha varit med på/tagit del av Föreläsning 1, inklusive dess kodexempel, bör ta dig ganska långt på vägen till att klara denna labb. Har du dessutom varit med på övningen har du fått öva ytterligare på dessa saker och har ännu större chans att klara den utan problem.

Läsning för att klara labben

- **Flanagan (2020), JavaScript: The Definitive Guide**:
  - **Avsnitt 1.2** komma igång med script
  - **Avsnitt 3.4–3.5** Booleans (***truthy***/***falsy***) samt specialvärdena `null` och `undefined`
  - **Avsnitten 3.9.0–3.9.1** Typkonverteringar och `NaN` (implicit konvertering bakom `==`)
  - **Avsnitten 3.10.1-3.10.2** Skapa variabler med `let`, `const`, `var`
  - **Avsnitt 4.9.1** Jämförelseuttryck (`==` vs `===`)
  - **Avsnitt 4.13.1** Villkorsoperatorn (`? :`)
  - **Avsnitten 8.1.0–8.1.3, 8.2.1 och 8.3.0** Funktioner: deklarationer, uttryck, arrowfunktioner argument och parametrar.
  - **Avsnitt 15.1.1** JavaScript i Webbläsare

Fördjupning/referenslitteratur (frivillig)

- **Flanagan (2020), JavaScript: The Definitive Guide**:
  - **Avsnitten 3.9.2–3.9.3** Fördjupning om jämförelser och typkonverteringar
  - **Avsnitt 15.15.5** Prompt etc.
- **Rauschmayer (2025), Exploring JavaScript:**
  - **Kap. 13** Variabler och assignment
  - **Kap. 15** Operatorer (inkl. `==` vs `===` och Object.is)
  - **Kap. 17** Booleans (***truthy***/***falsy***)
  - **Kap. 27** Callable values (olika funktionsformer)

HÖGSKOLAN DALARNA

GIK339 – Labb 1

JavaScript intro, variabler och funkioner

Uppdaterad

2025-11-19

Uppgifter

Nedan följer de uppgifter som ni ska ha utfört för att få godkänt på labben. Uppgifterna är *minimum*, vilket innebär att du får labba runt med egna tillägg om du vill, så länge *instruktionerna följs*. Instruktionerna i uppgifterna kanske inte alltid är helt logiska, men det finns en poäng med dem.

Uppgiften följer det du har lärt dig under kursens första vecka, dvs. grunderna i JavaScript. Allt material, föreläsningar, övningar och läsanvisningar finns i kursrummet.

Struktur och upplägg gemensamt för uppgifterna

På vissa ställen uppmanas du att ta ställning till saker rörande din kod. Dessa svar skriver du inom kommentarer i din HTML- eller script-fil. Du får gärna skriva andra förtydligande kommentarer rörande ytterligare resonemang eller tydligare struktur i dina filer om du vill.

> Du kommer att prova dig fram med olika lösningar under uppgifternas gång och under processen kommer scriptet då och då att krascha. Se till att du i den slutgiltiga version av labben har ett script som är helt körbart och fritt från felmeddelanden.

Notering rörande GAI

I denna uppgift är det inte tillåtet att använda AI i uppgiften *Nivå 0 enligt Högskolans riktlinjer för användning av AI i uppgifter*.

Uppgift 1 - Skapa HTML-fil och koppla script

1. Skapa filen index.html och sätt upp en grundläggande HTML-struktur.
   ⇒ *Tips:* I VS Code kan man skriva utropstecken (`!`) följt av tangenten **tab** i en fil med filändelsen `.html` för att få en grundläggande HTML-struktur.
2. Skapa en fil som heter **script.js**.
1. Koppla **script-filen** till **HTML-filen** på valfritt sätt.
2. Öppna **index.html** i webbläsaren med tillägget Live Server.
3. Öppna konsolen i webbläsaren för att kunna se framtida output.
4. Skriv en kort slutsats i en kommentar i **index.html** () där du resonerar kring:

Reflektion uppgift 1

Skriv i en kommentar (inom `<!-- ... -->`) i **index.html** där du:

1. Beskriver vilken betydelse placeringen av `<script>`-taggen har.
2. Beskriver hur attributet `defer` fungerar.

Uppgift 2 - Variabler och scope

1. I **script.js**: deklarera några variabler med `let`, `cons` och `var`.
2. Lägg samtliga i block (inom `{ ... }`).

3. Testa att skriva ut dessa variabler med `console.log()`. Gör flera olika `console.log` på flera ställen i koden (innan blocket, inuti blocket, efter blocket)
4. Kontrollera i **DevTools** så att koden inte kraschar eller ger felmeddelanden.

Reflektion uppgift 2
Skriv i en kommentar (inom `/* … */`) i **script.js** där du:

1. Beskriver hur `var`, `let` och `const` skiljer sig gällande block.
2. Reflekterar över vad som händer vid olika placering av `console.log()` (före blocket, i blocket, efter blocket).

Uppgift 3 - Jämförelser och specialvärden

1. Testa `==` och `===` med olika värden, t.ex. `'3' == 3` och `'3' === 3`.
   ⇒ Kontrollera `NaN` === `NaN`, `null == undefined`, `null === undefined`.
   ⇒ Kontrollera värdena genom `console.log()`.
2. Använd en ***ternary operator*** (`?:`) för att avgöra om `undefined` är **truthy** eller **falsy**.
3. Kontrollera i **DevTools** så att koden inte kraschar eller ger felmeddelanden.

Reflektion uppgift 3
Skriv i en kommentar (inom `/* … */`) i **script.js** där du:

1. Förklarar output för de olika värdena som du testade med `==` och `===` och på vilket sätt de skiljer sig åt.
2. Förklarar vad som händer när ett uttryck står för sig självt i exempelvis en **tenerary operator** eller inom parenteserna hos en `if`-sats.
3. Förklarar vad `NaN`, `undefined` och `null` representerar.

Uppgift 4 - Funktioner

1. Skapa en funktion vid namn `greet` som tar en ***parameter*** (`name`). Funktionen ska returnera strängen `"Hej"` följt av det som står i parametern `name`.
   ⇒ Returvärdet ska alltså vara något i stil med `"Hej Mikaela"`, beroende på vad du skickar in till funktionen.
   ⇒ Välj själv om du använder en ***funktionsdeklaration***, ett ***funktionsuttryck*** eller en ***arrowfunktion***.
2. Anropa din funktion med valfritt ***argument***.
   ⇒ Skriv ut resultatet genom att göra funktionsanropet direkt i en `console.log()`, eller
   ⇒ Spara returvärdet till en variabel som du sedan skriver ut med `console.log();`
3. Skapa en variabel (med `let`) som också heter `name` (samma som parametern till funktionen, alltså) i roten av **script.js** (dvs. *inte* inuti något tidigare block eller i din funktion) och tilldelar den värdet av valfri sträng.

⇒ Skriv ut värdet i `name` vid olika positioner i koden (inuti/utanför funktionerna) med `console.log()`.

4. Kontrollera i **DevTools** så att koden inte kraschar eller ger felmeddelanden.

Reflektion uppgift 4

Skriv i en kommentar (inom `/* … */`) i **script.js** där du:

1. Beskriver skillnaden mellan de olika sätten att skapa en funktion (***funktionsdeklaration***, ***funktionsuttryck*** och ***arrowfunktion***), samt varför du valde den varianten som du valde.
2. Reflekterar över vad du behöver tänka på gällande *varifrån du kan anropa dina funktioner* (innan/efter funktionerna har skapats).
3. Förklarar vad som händer om du ändrar variabeln/parametern `name` i de olika situationerna och hur det påverkar utskrifterna på de olika ställena i koden.
4. Förklarar skillnaden mellan ***parameter***, ***variabel*** och ***argument***.

Extrauppgift (frivillig)

1. Bygg en liten räknare som uppmanar en användare att mata in två tal via *prompt*.
   ⇒ Be om ett tal åt gången (visa alltså prompten igen efter att första inmatningen gjorts). Input kommer att komma till JavaScript som sträng.
   ⇒ Säkerställ att användarinput innehåller ett värde och att värdet faktiskt är siffror, annars skicka ett meddelande till användaren om att input är ogiltig
2. Låt sedan funktion utföra följande:
   ⇒ Adderar talen
   ⇒ Multiplicerar talen
   ⇒ Kontrollerar om talen är samma (`===`).
3. Returnera dessa uträkningar genom
   ⇒ En ***sträng*** med dessa värden konkatenerade, t.ex. `"Summan är x, produkten är y och talen är lika)`.
   ⇒ En ***array*** med dessa tre värden
   ⇒ Ett ***objekt*** med egenskap-värde-par, t.ex. `summa: x`, `produkt: y`.
4. Skriv ut värdena till konsolen.
   ⇒ Om du returnerar ett objekt, testa `console.table` för tydligare output.

HÖGSKOLAN DALARNA

GIK339 – Labb 1

JavaScript intro, variabler och funkioner

Uppdaterad

2025-11-19

Inlämning

Inlämning sker i kursrummet och uppgiften Labb 1. Lämna i första hand in din uppgift som en länk till ett GitHub-repository.[1]

Kod

- Lämna in länk till GitHub-repository som skapades och lades upp på GitHub under *förberedelserna*. Använd inlämningstypen: Text
    - Länken ska se ut något i stil med: https://github.com/[**ditt-github-användarnamn]/**gik339-[**ditt-du-användarnamn**]-labb1.
- Det är ok, men inte rekommenderat, att lämna in samtliga filer i **zip:at** format
    - Använd **Infoga** → **Dokument** i uppgiftens textinmatning för att inkludera en **zip-fil**.

Kamratrespons

Genom kamratrespons får du chansen att ta del av dina kurskamraters lösningar och idéer. Du får också chansen att öva på din bedömningsförmåga.

Regler för kamratrespons

1. Kamratrespons sker endast på första inlämningen. Om du inte lämnar in i tid får du endast lärarens återkoppling vid senare försök.
2. Du kan inte bli underkänd av en kamrat. Kamratrespons är formativ – endast läraren examinerar och sätter betyg.
3. Om du lämnar in ogiltigt material (t.ex. trasig GitHub-länk eller fil):
   ⇒ Det är ditt eget ansvar att omedelbart åtgärda felet.
   ⇒ Du ska själv kontakta både läraren och din kamrat för att rätta till problemet.
   ⇒ Din kamrat är inte skyldig att jaga rätt på din kod.
4. Om problemet inte löses inom kamratresponsperioden får du ingen kamratrespons och går istället vidare till nästa försök med lärarfeedback.
   ⇒ Korrekt inlämning (fungerande länk eller fil) är en del av uppgiften.

Att ge kamratrespons

1. I Canvas kommer du att bli tilldelad **2 labbar** att ge respons på
2. Respons görs i inlämningens kommentarfält
3. Stöd rörande vad du kan titta efter finns i en dold sida som du kommer kunna se när du har lämnat in din labb.
   ⇒ *OBS!* Denna matris är inte på något sätt betygsättning för arbetet som du granskar, bara en checklista för dig att gå igenom med tips vad du kan kolla efter.

[1] Senare labbar kommer att kräva GitHub, så det är lämpligt att komma igång, men det ska inte vara ett hinder för utförandet av själva labben.

4. Om arbetet har en så bristfällig kvalitet att du inte upplever att du har kunnat utföra din respons enligt instruktion, skriv **[FLAGGA]** överst i kommentarsfältet för att flagga till lärare att extra kontroll behövs.
   ⇒ Du kan som riktlinje ha att flagga arbetet du för en eller flera av punkterna A-C har angett "Ej bedömningsbart".
   ⇒ *Notera (igen)* att den bedömning som du gör *inte* är ett betyg utan endast ett verktyg för dig att få material till din kommentar, samt att kunna flagga till lärare om något är helt på tok.
5. Skriv en **kommentar** på ca 120 ord där du anger:
   ⇒ Om du bedömer att arbetet behöver kontrolleras av lärare, ange **[FLAGGA]** högst upp i kommentaren. *Obs! Ange detta överst i kommentaren.*
   ⇒ Övergripande kommentar om arbetet
   ⇒ 2 styrkor
   ⇒ 2 områden som kan förbättras