Roni Chaikho
Systemvetenskap 1
Dynamisk webbapplikation
Hej!

Tack för att du granskar min Labb 1. Jag har försökt följa alla instruktioner noga och har inkluderat de obligatoriska reflektionerna som kommentarer (`` i HTML och /* */ i JavaScript) i respektive fil.

Här är en steg-för-steg-genomgång av vad jag har gjort:

Uppgift 1: HTML-koppling och Defer

1. **Filstruktur:** Jag skapade index.html och script.js och kopplade dem.
2. **Koppling och Defer:** Jag placerade <script src="script.js" defer></script> i <head>.
    - **Reflektion:** Jag valde defer för att visa att skriptet laddas samtidigt som HTML-dokumentet parsas, men att exekveringen väntar tills hela DOM-trädet är klart. Detta är ett säkert sätt att koppla skript utan att blockera sidans rendering.
3. **HTML-reflektion:** Reflektionen för Uppgift 1 hittar du som en HTML-kommentar i slutet av index.html.

Uppgift 2: Variabler och Scope

1. **Scope-test:** I script.js deklarerade jag globala variabler (globalLet, globalConst, globalVar) och sedan lokala variabler (blockLet, blockConst, blockVar) inuti ett block ({}).
2. **Utskrifter:** Jag lade till console.log() före, inuti och efter blocket för att testa synligheten (scope).
3. **Resultat:** Jag noterade att:
    - let och const från blocket inte var tillgängliga efter blocket (de är **block-scopade**).
    - var från blocket **var** tillgängligt efter blocket, vilket bevisar att var ignorerar block-scope.
4. **Reflektion:** Reflektionen för Uppgift 2 (som beskriver skillnaden mellan var, let, const och vad som händer vid olika utskriftsplatser) finns som en flerradig kommentar (/* */) i script.js.

Uppgift 3: Jämförelser och Specialvärden

1. **Strikt vs Lös Jämförelse:** Jag testade == och === med '3' och 3.
    - == (lös) gav true på grund av implicit typkonvertering.

Roni Chaikho
Systemvetenskap 1
Dynamisk webbapplikation

- === (strikt) gav false eftersom typen (string vs number) inte matchade.

2. **Specialvärden:** Jag testade också att NaN === NaN blir false, och att null och undefined är löst ekvivalenta (== true) men inte strikt ekvivalenta (=== false).
3. **Falsy:** Jag använde en **ternary operator** (undefined ? "Truthy" : "Falsy") för att visa att undefined är ett Falsy-värde.
4. **Reflektion:** Reflektionen för Uppgift 3 finns i script.js och förklarar varför === är att föredra och vad de olika specialvärdena (NaN, null, undefined) representerar.

Uppgift 4: Funktioner och Parametrar

1. **Funktion:** Jag skapade funktionen greet(name) som en **funktionsdeklaration** (vald variant) som returnerar en hälsningssträng.
2. **Scope-demonstration:** Jag deklarerade en **global variabel** let name = "Globala Namn" och skickade in ett annat värde som **argument** till funktionen (greet("Mikaela")).
3. **Resultat:** Utskrifterna bekräftade att den globala variabeln **inte ändrades** av funktionsanropet, eftersom parametern name fungerar som en oberoende, lokal variabel inuti funktionen.
4. **Reflektion:** Reflektionen för Uppgift 4 finns i script.js och täcker skillnaden mellan funktionstyper, varför hoisting är viktigt, och skillnaden mellan termerna *parameter*, *argument* och *variabel*.

Koden ska vara helt körbar och fri från felmeddelanden i konsolen. Låt mig veta om du hittar något konstigt eller om jag missat något i instruktionerna!

Tack på förhand!