Övningar 1 - javascript och ES6

Projekt

- Skapa en folder för projektet och öppna i VS Code
- I terminalen i VS Code kör kommandot `npm init -y` för att skapa en package.json i denna folder
- Skapa en eller flera filer (allt eftersom) för att göra övningarna, filerna behöver ha extensionen `.js`
- Kör filerna med `node filnamn.js`

Prettier i VS Code

- installera detta extension om ni vill, det är hjälpsamt för formattering av koden

Installera Prettier Extension

- Öppna VS Code och gå till **Extensions** (Ctrl+Shift+X).
- Sök efter **Prettier - Code formatter**.
- Installera extensionen.

Ställ in Prettier som Standardformatterare

- Öppna inställningarna (Ctrl+,).
- Sök på **default formatter**.
- Välj `Prettier` som standardformatterare.

Aktivera Format on Save

- I inställningarna, sök på **format on save** och bocka i alternativet.

ES6 Övningar

I dessa övningar får du öva på moderna ES6-koncept genom att skriva egen kod. Varje uppgift följs av en slide med en korrekt lösning som du kan jämföra med. Observera att datan, variabelnamnen och instruktionerna i övningarna är annorlunda än i genomgången.

Övning: let och const

Uppgift:

1. Skapa en variabel med **let** med värdet `"Erik". Ändra sedan värdet
2. Skapa en variabel med **const** med värdet `42`. Försök sedan ändra värdet

Lösning: let och const

```
// Med let (kan omassignas)
let firstName = "Erik";
firstName = "Oskar"; // Ok

// Med const (kan inte omassignas)
const fixedNumber = 42;
// fixedNumber = 50; // Detta skulle ge ett fel – kommenterat ut
```

Övning: Arrow Functions

Uppgift:

Skapa en arrow function med namnet `sumNumbers` som tar två tal som argument och returnerar deras summa.

Lösning: Arrow Functions

```
const sumNumbers = (x, y) => x + y;
console.log(sumNumbers(4, 9)); // Förväntat utdata: 13
```

Övning: Template Literals

Uppgift:

Skapa en variabel med namnet `userName` med värdet `"Oskar"`. Använd sedan en template literal för att skapa en hälsningssträng och lagra den i en variabel med namnet `welcomeMessage` så att den blir "Hej, Oskar!".

Lösning: Template Literals

```
const userName = "Oskar";
const welcomeMessage = `Hej, ${userName}!`;
console.log(welcomeMessage); // Förväntat utdata: "Hej, Oskar!"
```

Övning: Destructuring

Uppgift:

Skapa ett objekt `device` med egenskaperna **model** och **productionYear**. Använd destructuring för att extrahera dessa värden till variabler.

Logga variablernas värden.

Lösning: Destructuring

```
const device = { model: "Alpha", productionYear: 2018 };
const { model, productionYear } = device;
console.log(model); // "Alpha"
console.log(productionYear);        // 2018
```

Övning: Default Parameters

Uppgift:

Skriv en funktion med namnet `greetUser` med en parameter som har ett standardvärde `"Anonym"`. Funktionen ska returnera en hälsningssträng. Testa funktionen både utan argument och med ett argument

Lösning: Default Parameters

```
const greetUser = (username = "Anonym") => `Välkommen, ${username}!`;
console.log(greetUser());          // "Välkommen, Anonym!"
console.log(greetUser("Maja"));    // "Välkommen, Maja!"
```

Övning: reduce() – Summera en Array

Uppgift:

Använd **reduce()** för att summera alla tal i arrayen `[3, 5, 7, 9]`. Logga resultatet

Lösning: reduce() – Summera en Array

```
const numArray = [3, 5, 7, 9];
const totalSum = numArray.reduce((acc, cur) => acc + cur, 0);
console.log(totalSum); // Förväntat utdata: 24
```

Övning: reduce() – Hitta Största Talet

Uppgift:

Använd **reduce()** för att hitta det största talet i arrayen `[12, 8, 29, 4]`. Logga resultatet.

Lösning: reduce() – Hitta Största Talet

```
const valueArray = [12, 8, 29, 4];
const maxValue = valueArray.reduce((max, val) => (val > max ? val : max), valueArray[0]);
console.log(maxValue); // Förväntat utdata: 29
```

Övning: Rest och Spread Operators

Uppgift:

1. Skriv en funktion med namnet `sumAllNumbers` som använder **rest-operatorn** för att ta emot ett obestämt antal tal och returnerar summan av dem. Testa med värdena 2, 4 och 6.
2. Använd **spread-operatorn** för att kopiera arrayen `[5, 15]` och lägg till talen 25 och 35 i slutet.

Lösning: Rest och Spread Operators

```
// Rest operator
const sumAllNumbers = (...nums) => nums.reduce((acc, num) => acc + num, 0);
console.log(sumAllNumbers(2, 4, 6)); // Förväntat utdata: 12

// Spread operator
const originalList = [5, 15];
const extendedList = [...originalList, 25, 35];
console.log(extendedList); // Förväntat utdata: [5, 15, 25, 35]
```

Övning: map() – Transformera Array

Uppgift:

Använd **map()** för att skapa en ny array där varje tal i arrayen `[2, 4, 6]` multipliceras med 4. Logga den nya arrayen.

Lösning: map() – Transformera Array

```
const baseNums = [2, 4, 6];
const quadrupled = baseNums.map(n => n * 4);
console.log(quadrupled); // Förväntat utdata: [8, 16, 24]
```

Övning: filter() – Filtrera Array

Uppgift:

Använd **filter()** för att skapa en ny array med endast de tal i arrayen `[1, 7, 3, 9, 2]` som är större än 4. . Logga resultatet.

Lösning: filter() – Filtrera Array

```
const mixedNums = [1, 7, 3, 9, 2];
const filteredNums = mixedNums.filter(n => n > 4);
console.log(filteredNums); // Förväntat utdata: [7, 9]
```

Övning: sort() – Sortera en Array

Uppgift:

Använd **sort()** för att sortera arrayen `[23, 8, 47, 15]` i stigande ordning. Logga resultatet. Not: den befintliga arrayen ändras med sort(), du behöver inte skapa en ny array.

Lösning: sort() – Sortera en Array

```
const unsortedNums = [23, 8, 47, 15];
unsortedNums.sort((a, b) => a - b);
console.log(unsortedNums); // Förväntat utdata: [8, 15, 23, 47]
```

Övning: Default Export

Uppgift:

Skapa en fil `operations.js` där du exporterar en funktion med namnet `calculateModulus` som **default export**. Funktionen ska returnera resten vid division av två tal.

Lösning: Default Export

```
// operations.js
export default (x, y) => x % y;
// Alternativt med ett funktionsnamn:
// const calculateModulo = (x, y) => x % y;
// export default calculateModulo;
```

Övning: Import

Importera funktionen från default-exporten i en annan fil och anropa den med två värden och logga resultatet

Lösning:

```
import calculateModulo from "./operations.js";

const result = calculateModulo(10, 3);
console.log(result);
```

Övning: Promises

Uppgift:

Skapa ett **Promise** med namnet `asyncTask` som simulerar en asynkron operation med `setTimeout`. Skapa const-variabeln `taskSuccess` - om du sätter den till true, lös upp Promiset med meddelandet `"Task completed!"`; annars avvisa det med `"Task failed!"`. Logga initialt `asyncTask`-status.

Lösning: Promises

```
const asyncTask = new Promise((resolve, reject) => {
  setTimeout(() => {
    const taskSuccess = true; // Ändra till false för att testa reject
    if (taskSuccess) {
      resolve("Task completed!");
    } else {
      reject("Task failed!");
    }
  }, 1500);
});

console.log(asyncTask); // Visar Promise-status (Pending initialt)
```

Övning: then() och catch()

Uppgift:
Använd det **Promise** `asyncTask` du skapade ovan med `then()` för att logga meddelandet vid success, och `catch()` för att logga fel vid reject.

Lösning: then() och catch()

```
asyncTask
  .then(result => {
    console.log(result); // "Task completed!"
  })
  .catch(error => {
    console.error(error); // "Task failed!"
  });
```

Övning: async/await

Uppgift:

Skriv en asynkron funktion med namnet `executeTask` som använder `await` för att vänta på ditt Promise `asyncTask` från föregående övning. Hantera eventuella fel med en `try/catch`-struktur och logga resultatet.

Lösning: async/await

```
async function executeTask() {
  try {
    const result = await asyncTask;
    console.log(result); // "Task completed!"
  } catch (error) {
    console.error(error); // "Task failed!"
  }
}

executeTask();
```

Övning: fetch() – Hämta Data från ett API

Uppgift:

Skriv en asynkron funktion med namnet `fetchUserData` som hämtar data från `https://jsonplaceholder.typicode.com/users/2` med `fetch()`. Omvandla svaret till JSON och logga resultatet. Hantera fel med `try/catch`.

Lösning: fetch() – Hämta Data från ett API

```
async function fetchUserData() {
  try {
    const response = await fetch("https://jsonplaceholder.typicode.com/users/2");
    const userData = await response.json();
    console.log(userData);
  } catch (error) {
    console.error("Fel vid hämtning:", error);
  }
}

fetchUserData();
```

Övning: Event Listener

Uppgift:

Skap en event listener med JavaScript som visar en console.log med meddelandet `"Knappen trycktes!"` när knappen klickas.

Lösning: Event Listener

```
const actionBtn = document.getElementById("actionBtn");
    actionBtn.addEventListener("click", () => {
      console.log("Knappen trycktes!");
});
```