## Uge8 Workshop: GitHub Copilot som VS Code Plugin

## Del 1: Introduktion til GitHub Copilot plugin (30 min)

* **Hvad er GitHub Copilot plugin til VSCode?**
* **Hvordan fungerer det?** (AI-genereret kode)
  + Husk, AI er en co-pilot, ikke en udvikler.
  + Det øger din produktivitet, men kan også hæmme den.
* **Hvorfor bruge GitHub Copilot?**
  + Automatisering af gentagende kodeopgaver.
  + Forbedret produktivitet ved hurtigere kodning.
  + Inspiration til løsninger, når man sidder fast.

## Del 2: Opsætning af GitHub Copilot i VS Code (15 min)

* Installation af VS Code og Copilot-plugin.
* Aktivering af GitHub Copilot.
* En rundtur i grænsefladen

## Del 3: Hands-on øvelser med Copilot (15 min)

1. Opret en mappe kaldet AI og opret en ny fil test.js.
2. Start med at skrive en funktionsdefinition, f.eks.:

function sum(a, b) {

1. Stop med at skrive og vent et par sekunder. Copilot bør foreslå resten af funktionen automatisk.###
2. Generering af kodeblokke1. Begynd at skrive en kommentar, der beskriver den ønskede funktion, f.eks.:

// create an HTML document with JavaScript function to display Fibonacci numbers.

## Del 4: JavaScript øvelser

**IF Else**

Opret en ny doc hedder ifelse.html

**Trin 1: Opret en HTML-fil**

1. Åbn din . Visual Studio Code

2. Opret en ny fil og navngiv den **index.html**.

**Trin 2: Tilføj den medfølgende kode**

1. Kopiér den medfølgende HTML- og JavaScript-kode.

2. Indsæt koden i **index.html**-filen inde i <script>.

3. Gem filen.

let number = 10;

if (number > 0) {

console.log("The number is positive.");

} else if (number < 0) {

console.log("The number is negative.");

} else {

console.log("The number is zero.");

}

**Trin 3: Åbn HTML-filen i en browser**

1. Find **index.html**-filen, du har oprettet, på din computer.

2. Højreklik på **index.html**-filen.

3. Vælg **“Åbn med”** og vælg din foretrukne webbrowser (f.eks. Google Chrome).

Trin 4: Åbn browserens konsol

1. Når **index.html**-filen er åbnet i browseren, højreklik på siden.

2. Vælg **“Inspicer”** eller **“Inspicer element”** fra kontekstmenuen. Dette åbner udviklerværktøjerne.

3. I udviklerværktøjerne skal du klikke på fanen **“Konsol”**.

**Trin 5: Se resultatet i konsollen**

**Spørgsmål**

 **Forståelse af else if:**

* Forklar hvad else if-sætningen gør i denne kode.
* Hvad vil der ske, hvis du fjerner else if-sætningen og kun har if og else?

**Console-metoder:**

* Hvad gør metoden console.log() i denne kode?

**Praktisk anvendelse:**

* Skriv en kode snippet, der beder brugeren om at indtaste et tal og derefter bruger den samme logik til at bestemme, om tallet er positivt, negativt eller nul.

**Løkker**

Sum of Even Numbers

function sumOfEvenNumbers(arr) {

let sum = 0;

for (let i = 0; i < arr.length; i++) {

if (arr[i] % 2 === 0) { // Check if the number is even

sum += arr[i]; // Add the even number to the sum

}

}

return sum;

}

// Example usage:

const numbers = [1, 2, 3, 4, 5, 6];

console.log(sumOfEvenNumbers(numbers)); // Output: 12

**Trin-for-trin guide**

Gør det samme som med **if/else** øvelser

**Spørgsmål**

1. **Forståelse af Løkker:**
   * Hvordan itererer løkken over hvert element i arrayet arr?
2. **Løkkekontrol:**
   * Hvad ville der ske, hvis løkkens betingelse blev ændret fra i < arr.length til i <= arr.length?
   * Hvordan ville løkken opføre sig anderledes, hvis inkrementerklæringen i++ blev ændret til i += 2?
3. **Løkkens Variationer:**
   * Hvordan ville du ændre løkken til at bruge en while-løkke i stedet for en for-løkke for at opnå det samme resultat?
4. **Praktiske Anvendelser:**
   * Kan du tænke på et scenarie fra den virkelige verden, hvor du måske skal bruge en løkke til at behandle hvert element i en liste eller et array?
   * Hvordan ville du ændre koden for at finde summen af alle tal i arrayet, uanset om de er lige eller ulige?
5. **Forståelse af Løkkeoptimering:**
   * Hvordan kan du optimere en løkke for at reducere antallet af iterationer og forbedre ydeevnen? \*\*\*