1. Cos'è JavaScript?

JavaScript è un linguaggio di programmazione che permette di rendere interattive le pagine web. È utilizzato per validare moduli, creare animazioni, manipolare elementi HTML e CSS, comunicare con server e creare applicazioni web complesse (come Single Page Applications o SPAs) grazie a framework come **React**, **Vue**, e **Angular**.

Dove eseguire JavaScript

JavaScript può essere eseguito:

- **Nel Browser**: tutti i browser moderni supportano JavaScript.
- **In un Ambiente Server**: come con Node.js, un runtime che permette di eseguire JavaScript lato server.

2. Primi Passi in JavaScript

Inserire JavaScript in una Pagina HTML

JavaScript può essere aggiunto direttamente all'interno di un file HTML usando i tag `<script>`.

html
Copia codice

```
<!DOCTYPE html>
<html lang="it">
<head>
    <title>JavaScript Guida</title>
</head>
<body>
    <h1>Ciao, mondo!</h1>
    <script>
        console.log("Ciao dal console log!");
    </script>
</body>
</html>
```

Console del Browser

La console del browser (disponibile con **F12** o **Ctrl+Shift+J** su Chrome) permette di eseguire codice JavaScript, vedere errori e messaggi di debug con `console.log()`.

3. Variabili e Tipi di Dati

Dichiarazione di Variabili

Le variabili si dichiarano con `let`, `const`, o `var` (sebbene `var` sia usato meno frequentemente oggi).

javascript
Copia codice
```
let nome = "Mario";   // Variabile modificabile
const età = 30;       // Costante, non può essere cambiata
```

Tipi di Dati

JavaScript ha diversi tipi di dati, inclusi:

- **String**: testo
- **Number**: numeri interi e decimali
- **Boolean**: true/false
- **Object**: oggetti complessi
- **Array**: liste di valori
- **Null e Undefined**: rappresentano un valore assente

Esempio:

javascript
Copia codice
```
let prezzo = 19.99;     // Number
let nomeProdotto = "Penna"; // String
let disponibile = true; // Boolean
let colori = ["rosso", "blu", "verde"]; // Array
let prodotto = { nome: "Penna", prezzo: 1.99 }; // Oggetto
```

4. Operatori

JavaScript supporta vari operatori:

- **Aritmetici**: `+`, `-`, `*`, `/`, `%`
- **Assegnamento**: `=`, `+=`, `-=`, `*=`, `/=`
- **Confronto**: `==`, `!=`, `===`, `!==`, `<`, `>`, `<=`, `>=`
- **Logici**: `&&`, `||`, `!`

Esempio:

javascript
Copia codice
```
let x = 10;
let y = 5;
let somma = x + y;          // 15
let confronto = (x === y);   // false
let logico = (x > 5 && y < 10); // true
```

5. Strutture di Controllo

Condizionali

Le strutture `if`, `else if`, e `else` controllano il flusso del programma.

javascript
Copia codice
```
let età = 18;
if (età >= 18) {
    console.log("Sei maggiorenne");
} else {
    console.log("Sei minorenne");
}
```

Cicli

JavaScript offre vari cicli per eseguire codice ripetutamente:

- **for**: per un numero noto di iterazioni.
- **while** e **do-while**: per cicli basati su una condizione.

Esempio:

javascript
Copia codice
```
for (let i = 0; i < 5; i++) {
    console.log(i);
}

let count = 0;
while (count < 5) {
    console.log(count);
    count++;
}
```

6. Funzioni

Le funzioni in JavaScript possono essere dichiarate con `function`, come funzioni freccia (`=>`), o come funzioni anonime.

Funzioni Classiche

javascript
Copia codice
```
function saluta(nome) {
    return `Ciao, ${nome}!`;
}

console.log(saluta("Mario"));
```

Funzioni Freccia

javascript
Copia codice
```
const saluta = (nome) => `Ciao, ${nome}!`;
console.log(saluta("Luigi"));
```

7. Array e Oggetti

Array

Gli array in JavaScript memorizzano liste di elementi. Offrono molti metodi integrati come `.push()`, `.pop()`, `.map()`, e `.filter()`.

javascript
Copia codice
```
let numeri = [1, 2, 3, 4, 5];
numeri.push(6);      // Aggiunge 6 alla fine
console.log(numeri[0]); // Stampa 1
```

Oggetti

Gli oggetti memorizzano coppie chiave-valore e sono estremamente flessibili.

javascript
Copia codice

```javascript
let persona = {
    nome: "Mario",
    età: 30,
    saluta: function() {
        return `Ciao, mi chiamo ${this.nome}`;
    }
};

console.log(persona.saluta());  // "Ciao, mi chiamo Mario"
```

8. Manipolazione del DOM

DOM (Document Object Model) è la struttura della pagina web. JavaScript permette di manipolare il DOM per creare pagine interattive.

Selezione degli Elementi

Usiamo `document.getElementById()`, `document.querySelector()` e altri metodi per selezionare elementi HTML.

javascript
Copia codice
```javascript
let titolo = document.getElementById("titolo");
let paragrafi = document.querySelectorAll("p");
```

Modifica degli Elementi

Possiamo aggiornare il contenuto e lo stile degli elementi.

javascript
Copia codice
```javascript
titolo.innerText = "Nuovo Titolo";
titolo.style.color = "blu";
```

Eventi

JavaScript gestisce eventi come `click`, `mouseover`, e `submit`.

javascript
Copia codice
```javascript
let bottone = document.getElementById("mioBottone");
bottone.addEventListener("click", function() {
```

```
    alert("Bottone cliccato!");
});
```

9. Programmazione Asincrona

JavaScript supporta operazioni asincrone, come richieste di rete, usando **promises** e **async/await**.

Promises

javascript
Copia codice

```
let promessa = new Promise((resolve, reject) => {
    let successo = true;
    if (successo) resolve("Operazione completata");
    else reject("Errore");
});

promessa.then((risultato) => console.log(risultato))
        .catch((errore) => console.log(errore));
```

Async/Await

javascript
Copia codice

```
async function fetchData() {
    try {
        let risposta = await
fetch("https://jsonplaceholder.typicode.com/todos/1");
        let dati = await risposta.json();
        console.log(dati);
    } catch (errore) {
        console.log("Errore:", errore);
    }
}

fetchData();
```

10. JSON e Comunicazione con API

JavaScript supporta **JSON (JavaScript Object Notation)**, un formato leggero per lo scambio di dati. Molte API restituiscono dati in formato JSON.

javascript
Copia codice
```
let persona = { nome: "Mario", età: 30 };
let jsonString = JSON.stringify(persona); // Conversione a JSON
let oggetto = JSON.parse(jsonString);     // Conversione da JSON
```

11. Strutture Avanzate: Classi e Moduli

Classi

Le classi permettono di creare oggetti con proprietà e metodi.

javascript
Copia codice
```
class Persona {
    constructor(nome, età) {
        this.nome = nome;
        this.età = età;
    }

    saluta() {
        return `Ciao, mi chiamo ${this.nome}`;
    }
}

let mario = new Persona("Mario", 30);
console.log(mario.saluta());
```

Moduli

I moduli permettono di organizzare il codice e separare funzionalità.

javascript
Copia codice
```
// In file funzione.js
export function saluta(nome) {
    return `Ciao, ${nome}!`;
}
```

```
// In file main.js
import { saluta } from './funzione.js';
console.log(saluta("Mario"));
```

Questa guida copre i concetti principali di JavaScript, dalle basi alle funzionalità avanzate. Approfondire concetti come **event loop**, **closure**, **scope**, e **higher-order functions** ti aiuterà a padroneggiare JavaScript.

4o