JavaScript Basics

JavaScript è un linguaggio di programmazione versatile che consente di creare applicazioni web dinamiche e interattive. Questa guida copre i concetti fondamentali e include esempi pratici per aiutarti a padroneggiare le basi.

Che cos'è JavaScript?

JavaScript è un linguaggio di programmazione di alto livello, interpretato e orientato agli oggetti. È utilizzato principalmente per migliorare l'esperienza utente delle pagine web, consentendo funzionalità come:

- Menu a tendina dinamici
- Validazioni dei moduli
- Animazioni
- Interazioni in tempo reale (es. chat)

Differisce da **HTML** (che definisce la struttura delle pagine) e **CSS** (che ne definisce lo stile) in quanto controlla il **comportamento**.

Variabili

Le variabili memorizzano dati che possono essere utilizzati e manipolati nel programma. Puoi dichiarare variabili con `var`, `let` o `const`.

- `var`: Scope globale o di funzione (meno utilizzato nei progetti moderni).
- `let`: Scope di blocco, più sicuro rispetto a `var`.
- `const`: Per variabili che non devono essere riassegnate.

Esempio:

```
let age = 25; // Variabile modificabile
const name = "John"; // Costante
var isStudent = true; // Variabile tradizionale
```

Tipi di Dati

I tipi di dati in JavaScript sono suddivisi in **primitivi** e **oggetti**.

Tipi primitivi:

1. `string` - Testo
2. `number` - Numeri interi e decimali (incluso il tipo float)
3. `boolean` - Valori logici (`true`/`false`)
4. `null` - Valore nullo intenzionale

5. `undefined` - Valore non definito
6. `symbol` - Identificatori unici (introdotto in ES6)

Esempio:

```
let text = "Hello"; // Stringa
let num = 42; // Numero intero
let floatNum = 3.14; // Numero decimale (float)
let isActive = true; // Booleano
let value = null; // Null
let notAssigned; // Undefined
```

Operatori

Gli operatori eseguono operazioni sui dati. Alcuni esempi:

1. **Aritmetici**: `+`, `-`, `*`, `/`, `%`
2. **Assegnazione**: `=`, `+=`, `-=`
3. **Confronto**: `==`, `===`, `!=`, `<`, `>`
4. **Logici**: `&&`, `||`, `!`

Esempio:

```
let a = 10,
  b = 5;
console.log(a + b); // 15
console.log(a === b); // false
console.log(a > b && b > 0); // true
```

Funzioni

Le funzioni sono blocchi riutilizzabili di codice.

Dichiarazione di funzione:

```
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("Alice"));
```

Arrow function (ES6):

```
const greet = (name) => `Hello, ${name}!`;
console.log(greet("Bob"));
```

Oggetti e JSON

Gli **oggetti** memorizzano dati come coppie chiave-valore, mentre **JSON** (JavaScript Object Notation) è un formato per strutturare dati.

Esempio di oggetto:

```
let person = {
  name: "Alice",
  age: 30,
  greet: function () {
    console.log(`Hi, I'm ${this.name}`);
  },
};

console.log(person.name); // Alice
person.greet(); // Hi, I'm Alice
```

Esempio di JSON:

```
{
  "name": "Alice",
  "age": 30,
  "isStudent": false
}
```

Per convertire un oggetto in JSON o viceversa:

```
let jsonString = JSON.stringify(person); // Oggetto → JSON
let parsedObject = JSON.parse(jsonString); // JSON → Oggetto
```

Classi

Le classi sono un modello per creare oggetti con proprietà e metodi comuni.

Esempio:

```
class Animal {
  constructor(name, type) {
```

```
    this.name = name;
    this.type = type;
  }

  speak() {
    console.log(`${this.name} says hello!`);
  }
}

let dog = new Animal("Buddy", "Dog");
dog.speak(); // Buddy says hello!
```

Manipolazione del DOM

Il DOM rappresenta la struttura HTML di una pagina. JavaScript può interagirvi per modificarne il contenuto o lo stile.

Esempio:

```
let header = document.getElementById("header");
header.textContent = "New Title";
header.style.color = "red";
```

Eventi

Gli eventi permettono di reagire a interazioni dell'utente.

Esempio:

```
document.getElementById("btn").addEventListener("click", () => {
  alert("Button clicked!");
});
```

Array

Gli array memorizzano liste di elementi.

Esempio:

```
let fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // apple

fruits.push("date");
console.log(fruits); // ["apple", "banana", "cherry", "date"]
```

Strutture di Controllo

1. **Condizionali:** `if`, `else if`, `else`

```
if (score > 80) {
  console.log("Excellent");
} else {
  console.log("Good");
}
```

2. **Cicli:** `for`, `while`, `do-while`

Esempio di ciclo WHILE

Esegue un blocco di codice **finché** la condizione rimane vera.

```
let i = 0;

while (i < 3) {
  console.log(`Il valore di i è: ${i}`);
  i++;
}
```

Esempio di ciclo DO-WHILE

Esegue il blocco di codice **almeno una volta**, e poi controlla la condizione.

```
let j = 0;

do {
  console.log(`Il valore di j è: ${j}`);
  j++;
} while (j < 3);
```

Esempio di ciclo FOR

Esegue un blocco di codice **n** volte.

```
for (let i = 0; i < 3; i++) {
  console.log(i);
}
```

Error Handling

Gestione degli errori con `try-catch`.

```
try {
  throw new Error("Qualcosa è andato storto");
} catch (error) {
  console.error(error.message);
}
```

Caratteristiche ES6

1. **Template Literals:**

   ```
   let greeting = `Hello, ${name}!`;
   ```

2. **Moduli:**

   ```
   export function greet() {
     console.log("Hi");
   }
   import { greet } from "./greet.js";
   ```

3. **Spread Operator:**

   ```
   let arr1 = [1, 2];
   let arr2 = [...arr1, 3, 4];
   ```