Funzioni e Scope

Definizione e utilizzo delle funzioni

In JavaScript, una funzione è un blocco di codice che esegue un compito specifico o calcola un valore. Le funzioni ci permettono di evitare la ripetizione di codice, organizzare meglio i nostri programmi e riutilizzare sezioni di codice.

Sintassi di una funzione

La sintassi di base per dichiarare una funzione è la seguente:

```
function nomeFunzione(parametro1, parametro2) {
  // Corpo della funzione
  return risultato;
}
```

- **`nomeFunzione`**: Nome della funzione, che la identifica.
- **`parametro1, parametro2`**: Parametri che la funzione riceve. Possono essere zero o più di uno.
- **`return`**: Restituisce un valore al termine dell'esecuzione della funzione.

Esempio di una funzione semplice

Supponiamo di voler creare una funzione che sommi due numeri:

```
function somma(a, b) {
  return a + b;
}

let risultato = somma(5, 3); // risultato = 8
console.log(risultato);
```

Questa funzione prende due numeri come parametri e restituisce la loro somma.

Funzioni anonime e arrow functions

In JavaScript, possiamo anche creare funzioni anonime (cioè senza nome) e assegnarle a una variabile:

```
const moltiplica = function(x, y) {
  return x * y;
};

console.log(moltiplica(4, 5)); // 20
```

Le **arrow functions** sono una sintassi alternativa, introdotta con ES6:

```
const sottrai = (a, b) => a - b;

console.log(sottrai(10, 4)); // 6
```

Differenza tra scope locale e globale

Lo **scope** (ambito) in JavaScript determina dove le variabili sono accessibili. Ci sono due tipi principali di scope:

1. **Scope Globale**: Variabili dichiarate all'esterno di qualsiasi funzione. Sono accessibili ovunque nel codice.

   ```
   let nome = "Mario";

   function saluta() {
     console.log("Ciao, " + nome);
   }

   saluta(); // "Ciao, Mario"
   ```

2. **Scope Locale**: Variabili dichiarate all'interno di una funzione o di un blocco di codice. Sono accessibili solo all'interno di quel blocco.

   ```
   function mostraMessaggio() {
     let messaggio = "Hello, World!";
     console.log(messaggio);
   }

   mostraMessaggio(); // "Hello, World!"
   console.log(messaggio); // Errore: messaggio non è definito
   ```

Esercizio: Creazione di funzioni che manipolano contenuti HTML

Supponiamo di voler creare una funzione che cambia il contenuto di un elemento HTML con id "testo" e un'altra funzione che cambia il colore del testo.

HTML:

```
<div id="testo">Testo originale</div>
<button onclick="cambiaTesto()">Cambia Testo</button>
<button onclick="cambiaColore()">Cambia Colore</button>
```

JavaScript:

```
function cambiaTesto() {
  document.getElementById("testo").innerHTML = "Testo aggiornato!";
}

function cambiaColore() {
  document.getElementById("testo").style.color = "red";
}
```

- **`cambiaTesto`** modifica il contenuto dell'elemento HTML.
- **`cambiaColore`** cambia il colore del testo in rosso.

Array e Oggetti

Creazione e utilizzo di array

Un **array** è una struttura dati che permette di memorizzare una collezione di valori. In JavaScript, gli array sono molto flessibili e possono contenere diversi tipi di dati.

Creare un array

```
let numeri = [1, 2, 3, 4, 5];
let misto = [1, "ciao", true];
```

- Gli array possono contenere qualsiasi tipo di dato.
- Gli elementi di un array sono accessibili tramite indici numerici, partendo da 0.

Utilizzare gli array

Per accedere o manipolare gli elementi di un array, usiamo i metodi e proprietà come `length`, `push`, `pop`, ecc.

```
let frutti = ["mela", "banana", "pera"];

// Aggiungere un elemento alla fine
frutti.push("arancia");

// Rimuovere l'ultimo elemento
frutti.pop();

// Accesso al primo elemento
console.log(frutti[0]); // "mela"

// Lunghezza dell'array
console.log(frutti.length); // 3
```

Introduzione agli oggetti e loro proprietà

Gli **oggetti** in JavaScript sono strutture dati complesse che permettono di definire dati tramite coppie chiave-valore. Gli oggetti sono fondamentali perché permettono di raggruppare dati e funzioni.

Creare un oggetto

```
let persona = {
  nome: "Luca",
  eta: 30,
  saluta: function() {
    console.log("Ciao, sono " + this.nome);
  }
};
```

- `nome`, `eta`, e `saluta` sono **proprietà** dell'oggetto `persona`.
- `saluta` è una funzione (o metodo) che accede alle altre proprietà usando `this`.

Accesso alle proprietà di un oggetto

Esistono due modi principali per accedere alle proprietà di un oggetto:

```
console.log(persona.nome); // "Luca"
console.log(persona["eta"]); // 30
persona.saluta(); // "Ciao, sono Luca"
```

Esercitazione su array e oggetti in contesti pratici

Esempio pratico: Array di oggetti

Supponiamo di voler creare una lista di oggetti che rappresentano persone, ognuno con nome ed età, e di voler stampare solo i nomi delle persone maggiorenni.

```
let persone = [
  { nome: "Luca", eta: 30 },
  { nome: "Anna", eta: 17 },
  { nome: "Marco", eta: 22 }
];

persone.forEach(persona => {
  if (persona.eta >= 18) {
    console.log(persona.nome + " è maggiorenne");
  }
});
```

In questo esempio, usiamo un array di oggetti e il metodo `forEach` per iterare tra gli elementi. La condizione `if (persona.eta >= 18)` filtra solo le persone maggiorenni.

Esempio pratico: Oggetto e manipolazione dei dati

Supponiamo di avere un oggetto `carrello` che rappresenta un carrello di acquisti con una lista di articoli. Aggiungeremo una funzione per calcolare il totale degli articoli nel carrello.

```
let carrello = {
  articoli: [
    { nome: "Maglietta", prezzo: 20 },
    { nome: "Pantaloni", prezzo: 40 },
    { nome: "Scarpe", prezzo: 60 }
  ],
  calcolaTotale: function() {
    let totale = 0;
    this.articoli.forEach(articolo => {
      totale += articolo.prezzo;
    });
    return totale;
  }
};

console.log("Totale del carrello: €" + carrello.calcolaTotale()); // Totale del
carrello: €120
```

In questo esempio:

- L'oggetto `carrello` contiene una proprietà `articoli`, un array di oggetti che rappresentano i singoli articoli.
- La funzione `calcolaTotale` calcola la somma dei prezzi di tutti gli articoli nel carrello.