

# Una Breve Storia

**1995:** Creato da Brendan Eich presso Netscape Communications.

**Originariamente chiamato Mocha, poi LiveScript, infine JavaScript.**

**Progettato per rendere le pagine web più interattive.**

**Non direttamente correlato a Java (nonostante il nome).**

**Standardizzato come ECMAScript.**

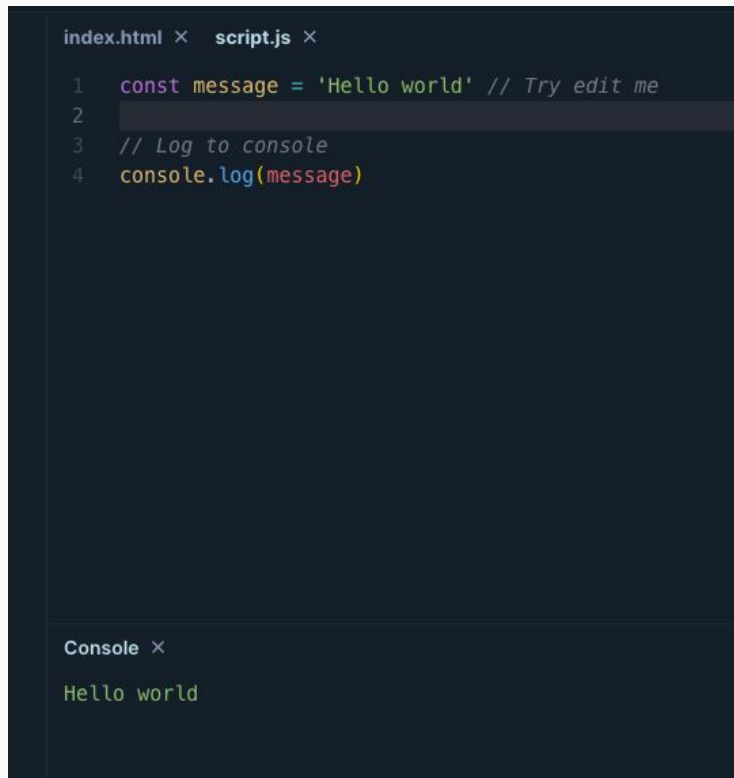
## Il Tuo Primo Assaggio di Codice

`console.log()` è una funzione per mostrare messaggi nella console del browser (strumenti per sviluppatori).

"Hello, World!" è una stringa, un testo da visualizzare.

Ogni istruzione JavaScript termina spesso con un punto e virgola (;), anche se non sempre obbligatorio.

<https://playcode.io/javascript>



The image shows a code editor with two tabs: 'index.html' and 'script.js'. The 'script.js' tab is active, displaying the following code:

```
1  const message = 'Hello world' // Try edit me
2
3  // Log to console
4  console.log(message)
```

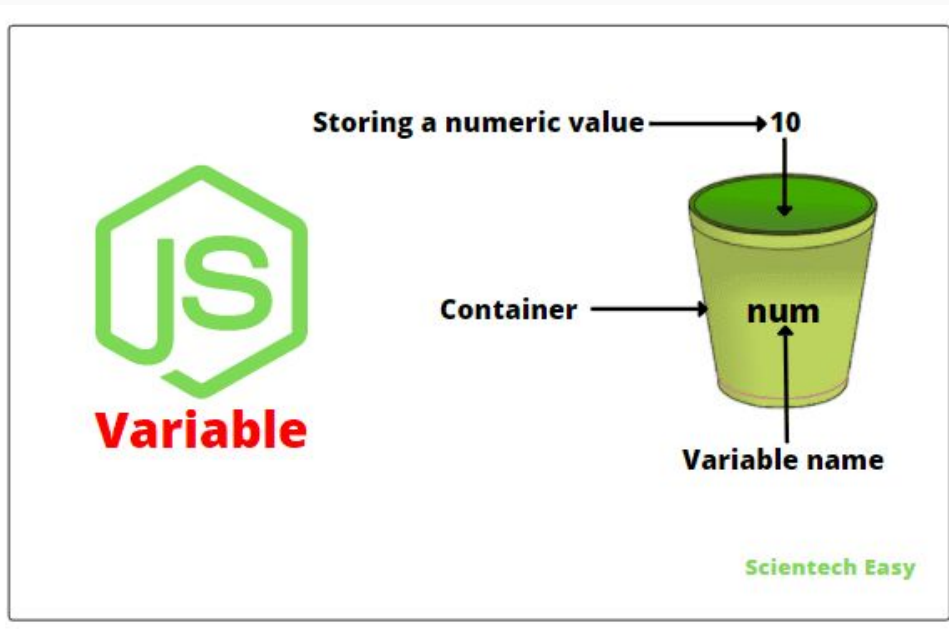
Below the code editor, there is a 'Console' tab. It shows the output of the code: 'Hello world'.

# Variabili: Cosa Sono?

**Contenitori per memorizzare dati.**

Puoi pensare a loro come etichette che dai a un valore.

Utili per riutilizzare e modificare i dati nel tuo codice.



# Dichiarare le Variabili

In JavaScript, usiamo parole chiave per dichiarare le variabili:

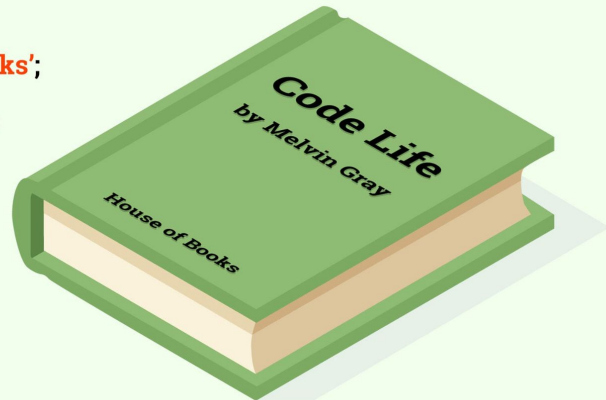
- **let**: Per variabili il cui valore può cambiare.
- **const**: Per variabili il cui valore non dovrebbe cambiare (costanti).
- **var**: (Evitare se possibile, ha un comportamento più complesso).

## JavaScript Variables

```
var title = 'Code Life';
```

```
let publisher = 'House of Books';
```

```
const author = 'Melvin Gray';
```



# Assegnare Valori alle Variabili

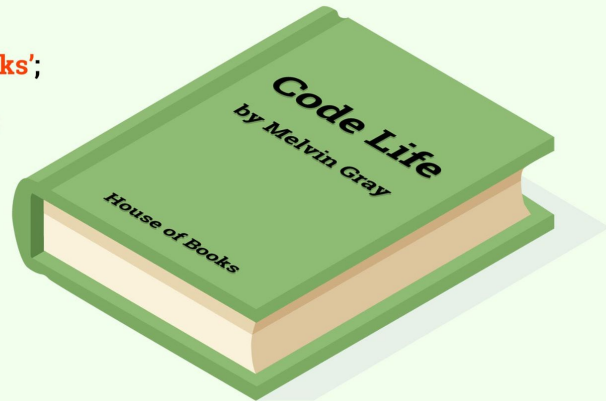
Usiamo l'operatore di assegnazione (=) per dare un valore a una variabile:

## JavaScript Variables

```
var title = 'Code Life';
```

```
let publisher = 'House of Books';
```

```
const author = 'Melvin Gray';
```

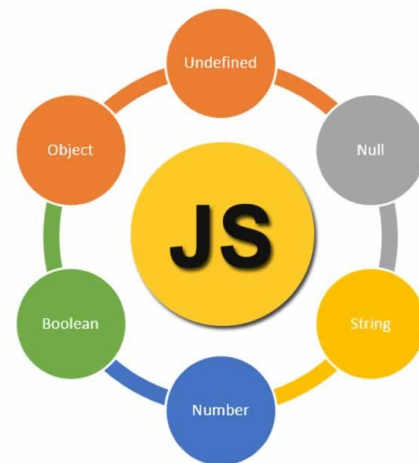


# Tipi di Dati Fondamentali (Parte 1)

**Stringa (String):** Sequenza di caratteri (testo) racchiusa tra virgolette singole ( ' ) o doppie ( " ).

**Numero (Number):** Rappresenta valori numerici (interi o decimali).

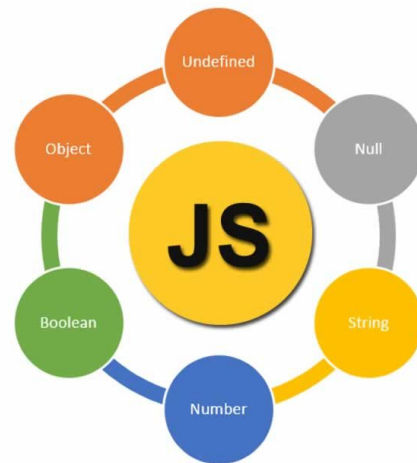
```
let messaggio = "Ciao mondo!";  
let nomeUtente = 'Giovanni';  
let numeroIntero = 10;  
let numeroDecimale = 3.14;  
let negativo = -5;
```



# Tipi di Dati Fondamentali (Parte 2)

- **Booleano (Boolean)**: Rappresenta un valore di verità: **true** (vero) o **false** (falso).
- **null**: Rappresenta l'assenza intenzionale di un valore.
- **undefined**: Indica che una variabile è stata dichiarata ma non ha ancora ricevuto un valore.

```
let isUtenteLoggato = true;  
let isMaggiorenne = false;  
let utenteCorrente = null;  
let indirizzo;  
console.log(indirizzo); // Output: undefined
```



# Data Types in JS

```
graph TD; A[Data Types in JS] --> B[Primitive Types]; A --> C[Reference Types]; B --> D[String]; B --> E[Number]; B --> F[Boolean]; B --> G[Null]; B --> H[Undefined]; C --> I[Object]; C --> J[Array]; C --> K[Function];
```

## Primitive Types

String

Number

Boolean

Null

Undefined

## Reference Types

Object

Array

Function



# Array: Collezioni Ordinate di Elementi

Un array è una **lista ordinata** di valori.

Ogni elemento ha un **indice**, che inizia da 0.

Possono contenere elementi di **qualsiasi tipo** (anche altri array o oggetti).

Si definiscono usando le **parentesi quadre** `[]`.

Si accede agli elementi utilizzando l'**indice** tra parentesi quadre.

```
let numeri = [1, 2, 3, 4, 5];  
let nomi = ["Alice", "Bob", "Charlie"];  
let misto = [1, "ciao", true, null];
```

```
console.log(numeri[0]); // Output: 1 (il primo elemento)  
console.log(nomi[1]);  // Output: Bob (il secondo elemento)
```

# Oggetti: Insiemi di Coppie Chiave-Valore

Un oggetto è una **collezione di proprietà**, dove ogni proprietà ha un **nome** (chiave) e un **valore**.

Le chiavi sono generalmente stringhe (o simboli).

I valori possono essere di **qualsiasi tipo** (primitivo o non primitivo).

Si definiscono usando le **parentesi graffe** `{ }`

Si accede alle proprietà usando la **notazione a punto** `(.)` o la **notazione a parentesi quadre** `[ ]`.

La notazione a parentesi quadre è utile quando il nome della proprietà è memorizzato in una variabile o contiene spazi o caratteri speciali.

```
5
6 let persona = {
7   nome: "Alice",
8   età: 30,
9   città: "Roma"
10 };
11
12 let chiave = "città";
13 console.log(persona[chiave]); // Output: Roma
```

```
chiave = "città";
console.log(persona[chiave]); // Output: Roma
```

# Funzioni: Blocchi di Codice Riutilizzabili

Anche le funzioni in JavaScript sono considerate oggetti di "prima classe".

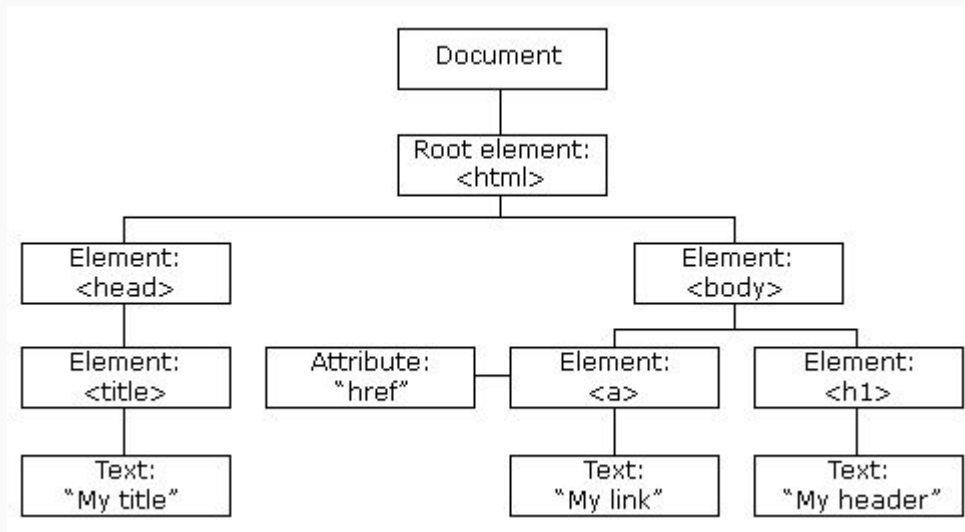
Rappresentano **blocchi di codice** che possono essere definiti e poi **invocati** (eseguiti) più volte.

Possono ricevere **parametri** (input) e restituire un **valore** (output).

```
5
6
7  function saluta(nome) {
8    |   console.log("Ciao, " + nome + "!");
9  }
10
11  saluta("Bob"); // Invoca la funzione
```

# Cos'è il DOM (Document Object Model)?

- Rappresentazione **strutturata** (ad albero) di un documento HTML.
- Ogni elemento HTML, attributo e testo all'interno del documento è un **nodo** nell'albero del DOM.
- JavaScript utilizza il DOM per **accedere e manipolare** gli elementi della pagina web.



# Accedere agli Elementi del DOM

- `document.getElementById('id')`: Seleziona un elemento tramite il suo attributo `id` (unico).
- `document.getElementsByClassName('classe')`: Seleziona tutti gli elementi con una determinata `class` (restituisce un `HTMLCollection`).
- `document.getElementsByTagName('tag')`: Seleziona tutti gli elementi con un determinato tag HTML (restituisce un `HTMLCollection`).
- `document.querySelector('selettore CSS')`: Seleziona il **primo** elemento che corrisponde a un selettore CSS (come `div`, `.classe`, `#id`, `p` `span`).
- `document.querySelectorAll('selettore CSS')`: Seleziona **tutti** gli elementi che corrispondono a un selettore CSS (restituisce un `NodeList`).

Ripassa i selettori:  
<https://flukeout.github.io/>

# Modificare gli Stili CSS

- La proprietà `elemento.style` consente di accedere e modificare gli stili CSS inline di un elemento.

La proprietà `elemento.classList` fornisce metodi utili per manipolare le classi CSS di un elemento:

- `elemento.classList.add('nuovaClasse')`: Aggiunge una o più classi.
- `elemento.classList.remove('classeDaRimuovere')`: Rimuove una o più classi.
- `elemento.classList.toggle('classeDaAlterare')`: Aggiunge la classe se non esiste, la rimuove se esiste.
- `elemento.classList.contains('classeDaVerificare')`: Restituisce `true` se l'elemento ha la classe specificata.
- 

```
let bottone = document.getElementById('mioBottone');  
bottone.style.backgroundColor = 'lightblue';  
bottone.style.color = 'white';  
bottone.style.padding = '10px 20px';
```

```
let divDinamico = document.getElementById('mioDiv');  
divDinamico.classList.add('importante', 'evidenziato');  
divDinamico.classList.remove('importante');  
divDinamico.classList.toggle('nascosto');
```