<TAG> in HTML e metodi() in JavaScript

⬜ Principali tag HTML

1. `<html>`: Definisce l'inizio e la fine di un documento HTML.
2. `<head>`: Contiene informazioni di intestazione del documento, come il titolo della pagina, collegamenti a fogli di stile e script.
3. `<body>`: Contiene il contenuto principale del documento, come testo, immagini, video, link e altri elementi.
4. `<h1>` `<h6>`: Definiscono i titoli di diversi livelli, con `<h1>` come il titolo principale e `<h6>` come il meno importante.
5. `<p>`: Definisce un paragrafo di testo.
6. `<a>`: Crea un link a un'altra pagina o a una risorsa esterna.
7. `<img>`: Inserisce un'immagine nel documento.
8. `<ul>`: Crea un elenco non ordinato.
9. `<ol>`: Crea un elenco ordinato.
10. `<li>`: Definisce un elemento dell'elenco.
11. `<div>`: Crea una sezione o un contenitore generico.
12. `<span>`: Definisce una sezione di testo o un contenuto in linea.

13. `<table>`: Crea una tabella per visualizzare dati tabulari.
14. `<tr>`: Definisce una riga di una tabella.
15. `<td>`: Definisce una cella di dati in una tabella.
16. `<form>`: Crea un modulo per la raccolta di input dall'utente.
17. `<input>`: Definisce un campo di input all'interno di un modulo.
18. `<button>`: Crea un pulsante cliccabile.
19. `<textarea>`: Definisce un'area di testo di input multiriga.
20. `<select>`: Crea un menu a discesa o una lista di opzioni selezionabili.
21. `<header>`: Definisce l'intestazione di una sezione o di un documento.
22. `<footer>`: Definisce il piè di pagina di una sezione o di un documento.
23. `<nav>`: Definisce una sezione di navigazione.
24. `<section>`: Definisce una sezione logica all'interno di un documento.
25. `<article>`: Definisce un contenuto autonomo e significativo che può essere distribuito e riutilizzato individualmente.
26. `<aside>`: Definisce un contenuto che è separato dal contenuto principale e può essere considerato autonomo.
27. `<label>`: Definisce una descrizione per un elemento di input.
28. `<fieldset>`: Raggruppa elementi di input correlati in un modulo.
29. `<legend>`: Fornisce una legenda o un titolo per un elemento `<fieldset>`.
30. `<iframe>`: Incorpora un'altra pagina HTML all'interno del documento corrente.
31. `<video>`: Incorpora un video nel documento.
32. `<audio>`: Incorpora un file audio nel documento.
33. `<script>`: Incorpora o fa riferimento a uno script JavaScript.
34. `<style>`: Definisce regole di stile CSS per il documento.
35. `<meta>`: Fornisce metadati sul documento HTML, come l'encoding dei caratteri, la descrizione e le parole chiave.
36. `<span>`: Definisce una sezione di testo o un contenuto in linea.
37. `<strong>`: Rende il testo in grassetto.

38. `<em>`: Rende il testo in corsivo.
39. `<hr>`: Inserisce una linea orizzontale nel documento.
40. `<canvas>`: Crea un'area disegnabile per grafica o animazioni.

Sintassi

1. Tag di **apertura** e **chiusura**:

```
<tag>Contenuto del tag</tag>
```

   - Esempio:

```
<h1>Titolo</h1>
```

2. Tag **auto-chiusura**:

```
<tag />
```

   - Esempio:

```
<img src="immagine.jpg" alt="Immagine" />
```

3. **Attributi**:

```
<tag attributo="valore">Contenuto del tag</tag>
```

   - Esempio:

```
<a href="<https://www.example.com>">Link</a>
```

4. **Commenti**:

```
<!-- Commento -->
```

- Esempio:

```
<!-- Questo è un commento -->
```

5. Tag per la **struttura del documento**:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titolo del documento</title>
  </head>
  <body>
    Contenuto del documento
  </body>
</html>
```

6. Tag per il **testo**:

```
<h1>Titolo di livello 1</h1>
<p>Paragrafo di testo</p>
<strong>Testo in grassetto</strong>
<em>Testo in corsivo</em>
<span>Testo generico</span>
```

7. Tag per le **liste**:

```
<ul>
  <li>Elemento di una lista non ordinata</li>
</ul>
<ol>
  <li>Elemento di una lista ordinata</li>
</ol>
```

8. Tag per i **link**:

```
<a href="<https://www.example.com>">Testo del link</a>
```

9. Tag per le **immagini**:

```
<img src="immagine.jpg" alt="Testo alternativo" />
```

10. Tag per i **form**:

```
<form>
  <input type="text" name="nome" placeholder="Nome" />
  <input type="submit" value="Invia" />
</form>
```

⚪ Principali metodi e funzioni in JavaScript

1. `getElementById()`: Restituisce un elemento del documento HTML in base all'ID specificato.
2. `querySelector()`: Restituisce il primo elemento nel documento HTML che corrisponde al selettore CSS specificato.
3. `addEventListener()`: Aggiunge un gestore di eventi a un elemento HTML, consentendo di rispondere a interazioni come clic, hover, input, etc.
4. `createElement()`: Crea un nuovo elemento HTML.
5. `appendChild()`: Aggiunge un elemento figlio a un elemento padre esistente.
6. `removeChild()`: Rimuove un elemento figlio da un elemento padre.
7. `setAttribute()`: Imposta un attributo su un elemento HTML.
8. `getAttribute()`: Restituisce il valore di un attributo di un elemento HTML.
9. `classList.add()`: Aggiunge una classe CSS a un elemento.
10. `classList.remove()`: Rimuove una classe CSS da un elemento.
11. `classList.toggle()`: Aggiunge una classe se non è presente, altrimenti la rimuove.
12. `querySelectorAll()`: Restituisce tutti gli elementi nel documento HTML che corrispondono al selettore CSS specificato.
13. `setTimeout()`: Esegue una funzione dopo un ritardo specificato.
14. `setInterval()`: Esegue una funzione in modo ripetuto a intervalli specificati.

15. `JSON.parse()`: Analizza una stringa JSON e restituisce un oggetto JavaScript.
16. `JSON.stringify()`: Converte un oggetto JavaScript in una stringa JSON.
17. `Math.random()`: Restituisce un numero casuale compreso tra 0 e 1.
18. `Math.floor()`: Arrotonda un numero verso il basso all'intero più vicino.
19. `Math.ceil()`: Arrotonda un numero verso l'alto all'intero più vicino.
20. `Math.round()`: Arrotonda un numero all'intero più vicino.
21. `parseInt()`: Converte una stringa in un numero intero.
22. `parseFloat()`: Converte una stringa in un numero a virgola mobile.
23. `toFixed()`: Formatta un numero con un numero specificato di cifre decimali.
24. `toLowerCase()`: Converte una stringa in minuscolo.
25. `toUpperCase()`: Converte una stringa in maiuscolo.
26. `substring()`: Restituisce una sottostringa di una stringa, in base agli indici specificati.
27. `split()`: Divide una stringa in un array di sottostringhe in base a un delimitatore.
28. `join()`: Unisce gli elementi di un array in una stringa, separati da un delimitatore.
29. `push()`: Aggiunge uno o più elementi alla fine di un array.
30. `pop()`: Rimuove l'ultimo elemento di un array e lo restituisce.
31. `shift()`: Rimuove il primo elemento di un array e lo restituisce.
32. `unshift()`: Aggiunge uno o più elementi all'inizio di un array.
33. `length`: Restituisce la lunghezza di una stringa o di un array.
34. `typeof()`: Restituisce il tipo di dati di una variabile.
35. `isNaN()`: Verifica se un valore è NaN (Non a Number).
36. `Date()`: Crea un nuovo oggetto data per la gestione delle date e degli orari.
37. `toLocaleString()`: Restituisce una stringa che rappresenta una data e un'ora, formattate in base alle impostazioni locali.
38. `XMLHttpRequest()`: Crea un nuovo oggetto XMLHttpRequest per effettuare richieste HTTP asincrone.

Sintassi metodi e funzioni generiche

1. **Dichiarazione** di una **funzione**:

```
function nomeFunzione(parametro1, parametro2) {
  // corpo della funzione
}
```

   - Esempio:

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

2. **Funzione anonima**:

```
var nomeFunzione = function(parametro1, parametro2) {
  // corpo della funzione
};
```

   - Esempio:

```
var saluta = function(nome) {
  console.log("Ciao, " + nome + "!");
};
```

3. **Arrow function**:

```
var nomeFunzione = (parametro1, parametro2) => {
  // corpo della funzione
};
```

   - Esempio:

```
var saluta = (nome) => {
  console.log("Ciao, " + nome + "!");
};
```

4. **Chiamata** di una **funzione**:

```
nomeFunzione(argomento1, argomento2);
```

- Esempio:

```
saluta("Mario");
```

5. Utilizzo di **return** per **restituire un valore** dalla funzione:

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

- Esempio:

```
var risultato = somma(2, 3);
console.log(risultato); // Output: 5
```

6. Utilizzo di **funzioni callback**:

```
function eseguiOperazione(a, b, callback) {
  var risultato = a + b;
  callback(risultato);
}
```

- Esempio:

```
function visualizzaRisultato(risultato) {
  console.log("Il risultato è: " + risultato);
}

eseguiOperazione(2, 3, visualizzaRisultato); // Output: Il risultato è: 5
```

Sintassi metodi e funzioni predefinite

1. Metodo `alert()` per mostrare un messaggio di avviso:

```
alert("Questo è un messaggio di avviso!");
```

2. Metodo `console.log()` per stampare un messaggio sulla console del browser:

```
console.log("Messaggio di log");
```

3. Metodo `confirm()` per mostrare una finestra di conferma con i pulsanti OK e Annulla:

```
var conferma = confirm("Sei sicuro di voler procedere?");
if (conferma) {
  // Codice da eseguire se l'utente conferma
} else {
  // Codice da eseguire se l'utente annulla
}
```

4. Metodo `prompt()` per mostrare una finestra di input con un messaggio:

```
var nome = prompt("Inserisci il tuo nome:");
if (nome) {
  // Codice da eseguire se l'utente inserisce un valore
} else {
  // Codice da eseguire se l'utente annulla o lascia vuoto l'input
}
```

5. Metodo `parseInt()` per convertire una stringa in un numero intero:

```
var numeroStringa = "10";
var numero = parseInt(numeroStringa);
```

6. Metodo `parseFloat()` per convertire una stringa in un numero decimale:

```
var numeroStringa = "3.14";
var numero = parseFloat(numeroStringa);
```

7. Metodo `toFixed()` per arrotondare un numero decimale a un numero specifico di decimali:

```
var numero = 3.14159;
var numeroArrotondato = numero.toFixed(2); // Restituisce "3.14"
```

8. Metodo `toUpperCase()` per convertire una stringa in maiuscolo:

```
var stringa = "ciao";
var stringaMaiuscola = stringa.toUpperCase(); // Restituisce "CIAO"
```

9. Metodo `toLowerCase()` per convertire una stringa in minuscolo:

```
var stringa = "CIAO";
var stringaMinuscola = stringa.toLowerCase(); // Restituisce "ciao"
```

10. Metodo `length` per ottenere la lunghezza di una stringa o la dimensione di un array:

```
var stringa = "Hello";
var lunghezzaStringa = stringa.length; // Restituisce 5

var array = [1, 2, 3, 4, 5];
var dimensioneArray = array.length; // Restituisce 5
```

11. Metodo `push()` per aggiungere un elemento alla fine di un array:

```
var array = [1, 2, 3];
array.push(4); // Aggiunge il numero 4 all'array
```

12. Metodo `pop()` per rimuovere l'ultimo elemento di un array e restituirlo:

```
var array = [1, 2, 3, 4];
var ultimoElemento = array.pop(); // Rimuove il numero 4 dall'array e lo restituis
ce
```

13. Metodo `join()` per unire gli elementi di un array in una stringa, separati da un delimitatore:

```
var array = ["Ciao", "come", "va?"];
var stringaUnita = array.join(" "); // Restituisce "Ciao come va?"
```

14. Metodo `indexOf()` per trovare l'indice di un elemento in un array:

```
var array = [10, 20, 30, 40, 50];
var indice = array.indexOf(30); // Restituisce 2 (indice dell'elemento 30 nell'arr
ay)
```

15. Metodo `substring()` per estrarre una sottostringa da una stringa:

```
var stringa = "Hello, world!";
var sottostringa = stringa.substring(7, 12); // Restituisce "world"
```

16. Metodo `replace()` per sostituire una sottostringa con un'altra all'interno di una stringa:

```
var stringa = "Ciao mondo!";
var nuovaStringa = stringa.replace("mondo", "amico"); // Restituisce "Ciao amico!"
```

17. Metodo `charAt()` per ottenere il carattere in una posizione specifica di una stringa:

```
var stringa = "Ciao";
var carattere = stringa.charAt(2); // Restituisce "a" (carattere in posizione 2)
```

18. Metodo `Math.random()` per generare un numero casuale compreso tra 0 e 1:

```
var numeroCasuale = Math.random(); // Restituisce un numero casuale compreso tra 0
e 1
```

19. Metodo `Math.round()` per arrotondare un numero all'intero più vicino:

```
var numero = 3.7;
var numeroArrotondato = Math.round(numero); // Restituisce 4
```

Differenza tra funzioni e metodi in javascript

In JavaScript, sia le funzioni che i metodi sono blocchi di codice riutilizzabili che eseguono determinate operazioni. Tuttavia, c'è una differenza fondamentale tra le due:

1. **Funzioni**: Una funzione in JavaScript è un blocco di codice autonomo che può essere richiamato o eseguito in qualsiasi punto del programma. Le funzioni possono essere definite in modo indipendente, al di fuori di qualsiasi oggetto, e

possono essere invocate con i loro nomi seguiti da parentesi tonde. Le funzioni possono ricevere argomenti (valori di input) e restituire valori di output.

- Esempio di definizione e invocazione di una funzione:

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

saluta("Mario"); // Output: "Ciao, Mario!"
```

2. **Metodi**: Un metodo in JavaScript è una funzione che è associata a un oggetto specifico. I metodi vengono definiti all'interno di un oggetto e possono accedere e manipolare le proprietà dell'oggetto stesso. Per richiamare un metodo, viene utilizzata la notazione punto, in cui si specifica l'oggetto seguito dal nome del metodo.

   - Esempio di definizione e invocazione di un metodo:

```
var persona = {
  nome: "Mario",
  saluta: function() {
    console.log("Ciao, " + this.nome + "!");
  }
};

persona.saluta(); // Output: "Ciao, Mario!"
```

Nell'esempio sopra, `saluta` è un metodo dell'oggetto `persona` che può accedere alla proprietà `nome` dell'oggetto utilizzando la parola chiave `this`.

In breve, la differenza principale tra funzioni e metodi in JavaScript è che le funzioni sono blocchi di codice autonomi, mentre i metodi sono funzioni associate agli oggetti e possono accedere alle proprietà dell'oggetto.