
boolean

JS + HTML

Utilizziamo le nostre conoscenze di JS per interagire con HTML

Quali sono le possibilità?

Come si può interagire con l'HTML tramite JS?

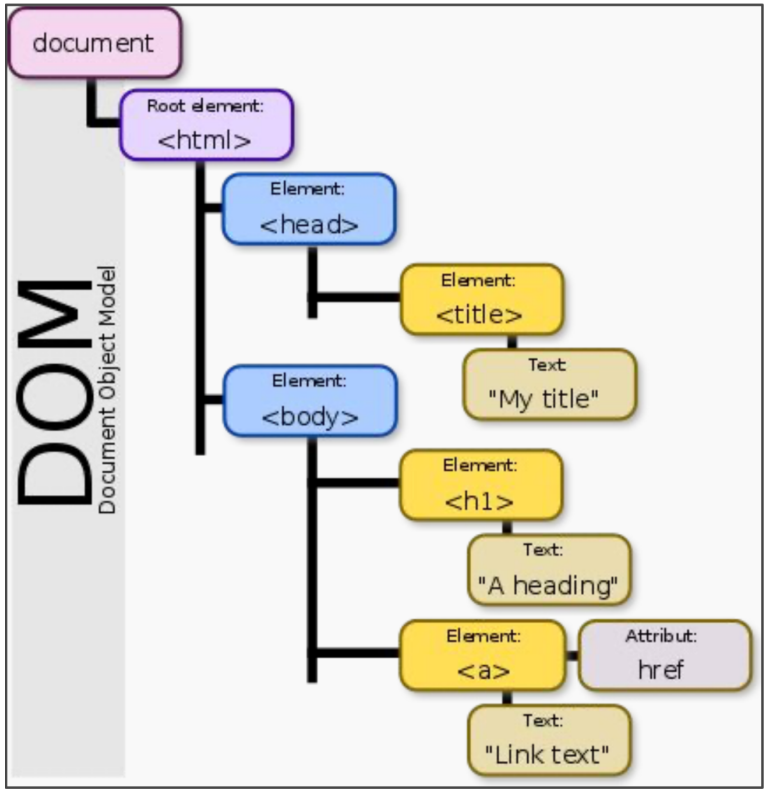
Come sempre gli strumenti non sono tanti, ma combinandoli abbiamo infinite possibilità

- **Modificare l'HTML**
  - Aggiungere/rimuovere classi css
  - Cambiare *l'inner html* di un elemento
  - Nascondere elementi
- **Reagire a eventi**
  - Al click
  - A hover
  - Al caricamento completo dalla pagina
  - ....
- **Interagire con il browser**
  - Impostare i cookie
  - Leggere lo user-agent
  - ...

Un po’ di Terminologia

- **DOM (Document Object Model)**: è una modalità di rappresentazione ad albero di dati strutturati.
  È un termine generico ma in ambito web si riferisce all’insieme di elementi/tag che formano una pagina web
- **Element**: la rappresentazione dell’unità che forma il DOM
- **Outer HTML**: il codice che rappresenta un element
- **Inner HTML**: “il contenuto” dell’elemento, cioè che è racchiuso tra apertura e chiusura

```
1 <p style="color:red">Hello World</p>
```



In Plain Javascript

Cosa possiamo fare con JavaScript puro?
Un sacco di cose!

Selezionare un elemento del DOM cercandolo **per ID**

```html
<div id="saluto">CIAO</div>
```

```javascript
const element = document.getElementById("saluto");

// element è un DOM Object Element
// typeof(element) -> object
```

In Plain Javascript

Cosa possiamo fare con JavaScript puro?
Un sacco di cose!

Selezionare solo **il primo elemento** del DOM che corrisponde ad un selettore css.

```html
<div class="red">Red</div>
<div class="red">Red</div>
<div class="red">Red</div>
```

```
const element = document.querySelector('.red');
```

DOM Element Object

Cosa si può fare su un Element?

Più di 80 operazioni, vediamone alcune:

- element.classList: restituisce i nomi delle classi CSS associate all'elemento
- element.className: permette di impostare il/le classi CSS che l'elemento ha associate
- element.innerHTML: permette di modificare l'innerHTML dell'elemento
- element.style: permette di leggere o impostare singole proprietà CSS per l'elemento
- element.addEventListener(): aggancia un funzione che sarà chiamata quando accade un evento (es. click)
- element.value: per ottenere il valore di un element, se ne ha uno

Aggiungere una classe

Cosa possiamo fare con JavaScript puro?

Aggiungere una classe

```html
<!-- elemento iniziale -->
<div id="saluto">CIAO</div>

<!-- elemento dopo l'esecuzione dello script -->
<div id="saluto" class="titolo">CIAO</div>
```

```js
const element = document.getElementById("saluto");

element.className = "titolo";

// ma se non vogliamo perdere le altre classi...
element.className = element.classList + " titolo";
```

Cambiare lo style

Cosa possiamo fare con JavaScript puro? Un sacco di cose

Accedere al CSS di un elemento

```
1  <div id="saluto" style="color: red;">CIAO</div>
```

```
1 const element = document.getElementById("saluto");
2
3 element.style.color = "red";
```

Cambiare qualunque proprietà

Cosa possiamo fare con JavaScript puro? Un sacco di cose

Accedere a qualunque proprietà di un elemento

```html
<img id="pic" src="miafoto.png" alt="foto">
```

```js
const element = document.getElementById('pic');

element.src = 'miafoto.png';

element.alt = 'foto'
```

Aggiungere un elemento al DOM

Cosa possiamo fare con JavaScript puro? Un sacco di cose

Possiamo inserire del testo o html all'interno di elementi esistenti.

```html
<div class="red">Ciao</div>
```

```js
const div = document.querySelector("div.red");
//metodo che già conosciamo
div.innerHTML += '<p>Mio testo</p>';
//oppure
div.append('Mio testo');
//con questo metodo
//non possiamo usare tag
```

🤔 Se **append** inserisce **alla fine** di un elemento, **prepend cosa farà**?

LIVE CODING

Click me

Cosa possiamo fare con JavaScript puro?

Scatenare un evento al click

```html
<div id="saluto">CIAO</div>
```

```js
const element = document.getElementById("saluto");

element.addEventListener('click',

 function() {
   // codice
 }

);
```

Click me

Cosa possiamo fare con JavaScript puro?

Scatenare un evento al click

Rimaniamo in **ascolto di un evento** su quell'elemento.
Quale evento? '**click**' (ma ce ne sono tanti altri)

Vediamo qui anche una **funzione**, che viene invocata solo quando avviene l'evento '**click**'

Quando una funzione viene inserita in un event listener per essere richiamata più tardi (ad esempio al click del bottone) viene detta "funzione di richiamo" o **callback function**

```
1 const button = document.querySelector('button');
2
3 button.addEventListener('click',
4
5  function() {
6    // codice
7  }
8
9 );
```

Click me

Callback functions

Le callback functions possono essere **anonime**. Ossia: non gli diamo nome e le dichiariamo direttamente nell'event listener.

```
1 const button = document.querySelector('button');
2
3 button.addEventListener('click', function() {
4    console.log('tasto cliccato!')
5  });
```

E' anche possibile utilizzare come callback function una funzione dichiarata altrove, utilizzandola nell'event listener

```
1 const button = document.querySelector('button');
2
3 function myFunction(){
4    console.log('tasto cliccato!')
5 }
6
7 button.addEventListener('click', myFunction);
```

Click me

Callback functions

Se la funzione che intendiamo utilizzare non prevede parametri, bastera inserire il suo nome come secondo argomento nel listener, senza le parentesi tonde alla fine, altrimenti sarebbe invocata immediatamente!

```
1 const button = document.querySelector('button');
2
3 function myFunction(){
4    console.log('tasto cliccato!')
5 }
6
7 button.addEventListener('click', myFunction);
```

Se invece dovesse prevedere parametri, allora dovremo includerla **all'interno di una funzione anonima**. In questo modo potremo poi invocarla con le tonde ed inserire al loro interno gli argomenti.

```
1 const button = document.querySelector('button');
2
3 function logNumber(num){
4    console.log('Il numero è: ' + num)
5 }
6
7 button.addEventListener('click', function(){
8   logNumber(3);
9 });
```

LIVE CODING

ESERCITAZIONE