DOM

Document Object Model.
E’ la rappresentazione ad oggetto di tutta la nostra pagina e con Javascript è possibile manipolare il DOM (*DOM manipulation)*

```
<script src="script.js"></script>
```

Inseriamo lo script a fondo <body> dopo quelli di Bootstrap.

Tutto ciò che vediamo in una pagina web viene chiamato *document*.
Per richiamare in JS un elemento usiamo il *getElementBy*

```
let nomeVariabile = document.getElementById('nomeId')
let nomeVariabile = document.getElementByTagName('p')
let nomeVariabile = document.getElementByClassName('myClass')
```

Altro metodo è il *querySelector*

```
let nomeVariabile = document.querySelector('h1#titolo.two')
let nomeVariabile = document.querySelectorAll('p')
```

Il *querySelector* semplice serve per richiamare un elemento singolo, e in tal caso si da un Id allo stesso per identificarlo. E’ possibile concatenare anche classi per renderlo ancora più selettivo.

Il *querySelectorAll* è per indicare tutti gli elementi di una determinata classe.

PROPRIETA' e METODI del DOM
Dopo aver richiamato l'elemento che vogliamo modificare, possiamo modificarlo grazie a delle proprietà.

.inner HTML *proprietà che cambia il contenuto di un elemento.*
Se utilizzato con gli apici greci è possibile inserire nel DOM direttamente del codice HTML.

```
let button = document.querySelector('#custom-btn')
button.innerHTML = "Sono un bottone custom";
```

```
div.innerHTML = `
      <div class="card">
            <a href"#"> Vai in vacanza in ${..}
      </div>`
```

.classList *proprietà che restituisce il nome dell'elemento indicato. Usato con:*
- ***.add()*** *metodo aggiungere una classe ad un elemento*
- ***.remove()*** *metodo rimuovere una classe ad un elemento*
- ***.contains()*** *metodo restituisce un true o false, una condizione*
- ***.toggle()*** *metodo per switchare due stili sull'elemento selezionato o creare una sticky navbar.*

```
nomeVariabile.classList.add("classe-1", "classe-2")
```

.parentNode *proprietà riferito al padre dell'elemento*
.firstChild *proprietà riferito al primo figlio dell'elemento*
.lastChild *proprietà riferito all'ultimo figlio dell'elemento*
.nextSibiling *proprietà riferito al fratello successivo all'elemento indicato*
.previousSibiling *proprietà riferito al fratello precedente all'elemento indicato*

```
elementoIndicato.parentNode.metodo()
```

.createElement() *metodo per creare un tag. All'interno delle parentesi si specifica "div" o "button" o "p", o comunque un tipo di tag html.*

```
let button = document.createElement('button')
```

.appendChild() *metodo per inserire un elemento creato come ultimo figlio di un padre.*

```
nomePadre.appendChild('nomeFiglio')
```

.getAttribute() *metodo che restituisce l'attributo di un elemento indicato*
.removeAttribute() *metodo che rimuove un attributo da un elementi indicato*
.setAttribute() *metodo che aggiunge un attributo all'elemento indicato. Il primo valore è il tipo dell'attributo, il secondo il valore*

```
elementoIndicato.setAttribute("href" , "www.google.com/")
```

STILE

Possiamo aggiungere dello Stile agli elementi che modifichiamo/creiamo nel DOM, e la sintassi differisce leggermente da quella di CSS.
Per tutti i comandi disponibili vai al link.
https://www.w3schools.com/jsref/dom_obj_style.asp

```
elementoIndicato.style.color =  "red";
elementoIndicato.style.backgroundColor = "green";
item.style.fontSize = "20px";
```

UTILIZZO delle FUNZIONI e CICLI nel DOM

Possiamo implementare nel DOM Funzioni e Cicli.
A seguito degli esempi.

.forEach() *possiamo per esempio utilizzare questo metodo per creare una serie di bottoni che differenziano l'uno dagli altri per un testo diverso specificato in un Array.*
In generale il ForEach è ottimo anche per effettuare una modifica per tutti gli elementi con una classe selezionata con una variabile.

```
let offerte = [ { 'country' : 'italia' }, { 'country' : 'spagna' }, {
'country' : 'germania'} ]
offerte.forEach( elemento =>{
      let button = document.createElement('button')
      button.innerHTML = 'Vai in ${country.elemento}'
      button.classList.add('btn' , 'btn-primary' , 'mx-3')

      btnWrapper.appendChild(button)
}
```