# 03\_01 Javascript nel browser

Importare javascript nell’head con **defer.**

*Nel caso in cui siano usati async o defer, lo script va inserito nell’head.*

* *Script caricato in parallelo all’html ma esecuzione blocca il parsing e il rendering dell’html.*
* **DEFER**: Script viene caricato ad inizio pagina html ma viene eseguito dopo che la pagina html è pronta
  + Semplifica e migliora le cose perché dopo il caricamento del body, c’è solo l’esecuzione

**Dove gira il codice js?**

Il codice js gira all’interno del browser che è una **sendbox** → ambiente protetto rispetto al resto; quindi, una pagina web non può scrivere su un’altra pagina web o su disco tranne se vengono fornite determinate funzioni.  
Tutti gli script agiscono all’interno di una window → accedono solo a metodi e funzioni qui presenti:

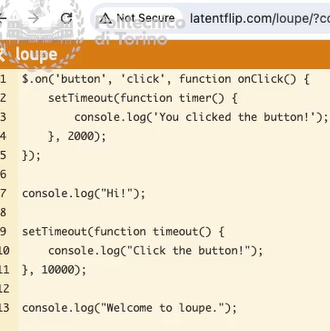
* Libreria standard di javascript → tipi, async, awayt, promise…
* BOM → opzioni che il browser può mettere a disposizione alla pagina
  + Accedere ai preferiti
  + Barra di navigazione (andare avanti e indietro nella navigazione)
  + Location: url attuale
  + History: per accedere ad intera history
  + Screen: per accedere a proprietà di schermo e finestra
  + XMLHttpRequest: usare protocollo http
* DOM → funzionalità oggetto dalla pagina in cui ci troviamo

### Eventi e Event loop

Tre tipi:

* Predefiniti
* Definiti da utente
* Ignorati

**EventLoop**:

* heap di memoria: tiene oggetti
* call stack: per fare le chiamate una riga alla volta
  + inserisco in stack
  + eseguo
  + rimuovo da stack
* API offerte da execution environment, da node, altro
  + Sincrone: vanno in stack
  + Asincrone: entra in gioco l’event loop
    - Es: Timeout 5s
      * Preso in carico
      * Rimosso stack
      * Messo in callback queue
      * Callback da essere eseguita dopo 5s
      * Riga dopo dello stack viene eseguita
        + Continua ad andare avanti nello stack finché non deve eseguire qualcosa nella callback queue
      * 5secondi passati → chiama callback setTimeout che stampa a schermo
        + Messa su stack e eseguita

### DOM:

**Albero** che rappresenta una **struttura** di una pagina WEB.

Attraverso javascript si può chiedere all’albero di prendere un determinato figlio, navigando sull’albero.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, algebra

Descrizione generata automaticamente

**NodeList**: lista di nodi simile ad array

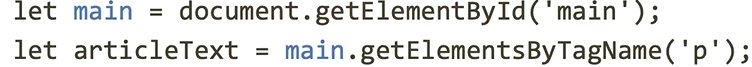
**Manipolazione del DOM**: attraverso document:

**Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente**

*css:*

* *.row*
* *.row > table*



I metodi per manipolazione, si possono **concatenare**:

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteesempi**:

**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteNavigare l’albero:**

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Quando si ha un nodo, gli attributi di un elemento diventano proprietà, body, del’oggetto. Se si ha un body id → proprietà di body id.

**Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamenteGestione degli attributi:**

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteCreare elementi:**

* document.createElement(tag) → crea elemento del tipo tag
* document.createTextNode(text) → nodo testuale

quando si crea un elemento, **bisogna inserirlo nel DOM**:

* document.body.appendChild(div) → aggiunge in coda al body

**metodi per elementi html:**

* Immagine che contiene testo, Carattere, linea, bianco

  Descrizione generata automaticamenteaggiungere un figlio
* aggiungere prima di un nodo
* rimpiazzare un nodo con uno nuovo.

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente**Metodi** applicabili su elementi ma anche su **nodi**:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteRecuperare contenuto presente in html in maniera testuale → .**innerHTML**:

permette di impostare o recuperare l’html di un nodo o di un elemento in forma testuale.

Permette di aggiungere elementi o testo.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, algebra

Descrizione generata automaticamentePermette di clonare: utile per replicare e poi modificare

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamenteSi può cambiare lo stile attraverso attirbuto classList

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteEsiste anche il metodo elem.style:

### EVENT HANDLING

La modalità di programmazione è **event-driven** → ogni cosa è causata da un evento.

Gli eventi sono determinati da:

* **target**: l’elemento DOM che genera l’evento
* **type**: il tipo di evento generato

**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteaddEventListener**

**Esistono varie categorie di eventi:**

* User Interface events (load, resize, scroll, etc.)
* Focus/blur events
* Mouse events (click, dblclick, mouseover, drag, etc.)
* Keyboard events (keyup, etc.)
* Form events (submit, change, input)
* Mutation events (DOMContentLoaded, etc.)
* HTML5 events (invalid, loadeddata, etc.)
* CSS events (animations etc.)

Immagine che contiene schermata, testo, Carattere

Descrizione generata automaticamenteMolti eventi hanno un comportamento di default (click su link → vai ad url). Si può modifcare il comportamento di default rimpiazzandolo con uno proprio:

**Eventi relativi al ciclo di vita della pagina HTML:**

* DOMContentLoaded (definito su documento)
  + Il browser carica tutto l’html, e l’albero del DOM è pronto
  + Risorse esterne non sono caricate
* Load
  + Il browser è finito caricando tutto le risorse esterne
* beforeunload/unload → da usare a proprio rischio e pericola perché non sempre funzionano