

# Modulo 2

# Web Tier

JavaScript – Day 2

Presentato da:

*Vitale Esca*

IBM Client Innovation Center - Italy

**21/11/2023**

IBM Client Innovation Center  
**Italy**

# Agenda Day 2

- 1 Introduzione a JavaScript
- 2 Integrazione di JavaScript in una pagina HTML
- 3 Interazione con gli elementi HTML
- 4 Manipolazione delle date in JavaScript

## Day 2 – Introduzione a JavaScript

### Cos'è JavaScript?

**JavaScript** è il linguaggio di programmazione orientato al web tra le sue caratteristiche c'è la possibilità di eseguire codice lato client.

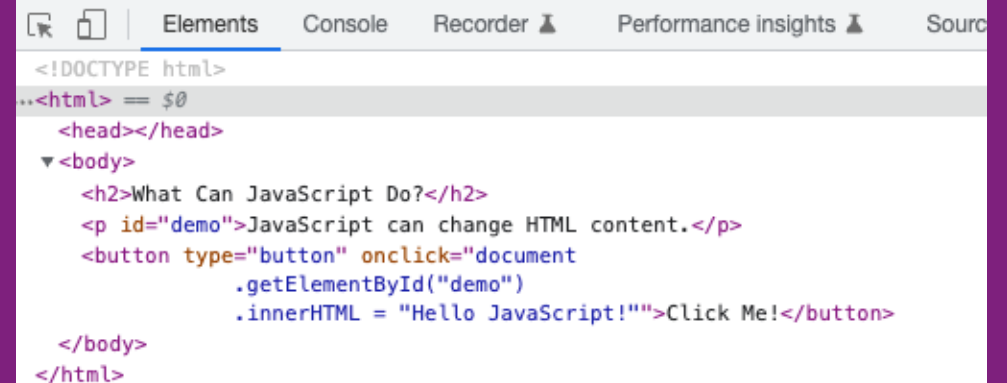
Tra le altre cose **JavaScript** consente di interagire dinamicamente con gli elementi della pagina, anche dopo il rendering/caricamento della stessa.

```
<!DOCTYPE html>
<html>
<body>
  <h2>What Can JavaScript Do?</h2>
  <p id="demo">JavaScript can change HTML content.</p>
  <button
    type="button"
    onclick='document
      .getElementById("demo")
      .innerHTML = "Hello JavaScript!"'>Click Me!</button>
</body>
</html>
```

### What Can JavaScript Do?

JavaScript can change HTML content.

Click Me!



```
<!DOCTYPE html>
..<html> == $0
  <head></head>
  <body>
    <h2>What Can JavaScript Do?</h2>
    <p id="demo">JavaScript can change HTML content.</p>
    <button type="button" onclick="document
      .getElementById("demo")
      .innerHTML = "Hello JavaScript!">Click Me!</button>
  </body>
</html>
```

Per approfondimenti: [https://www.w3schools.com/js/js\\_intro.asp](https://www.w3schools.com/js/js_intro.asp)

# Cos'è JavaScript?

- JavaScript è stato progettato per aggiungere interattività alle pagine HTML
- E' un linguaggio di scripting
- Un linguaggio di scripting è un linguaggio di programmazione 'alleggerito' ed efficiente, utilizzabile in modo semplice senza grandi configurazioni
- Uno script JavaScript **incorporato in una pagina web**, funziona **client-side**
- Un file JavaScript incorporato direttamente in una pagina HTML all'esecuzione viene interpretato (cioè gli script vengono eseguiti senza una compilazione preliminare)
- Tutti possono utilizzare JavaScript senza acquistare una licenza
- Java e JavaScript sono la stessa cosa?



## Day 2 – Introduzione a JavaScript

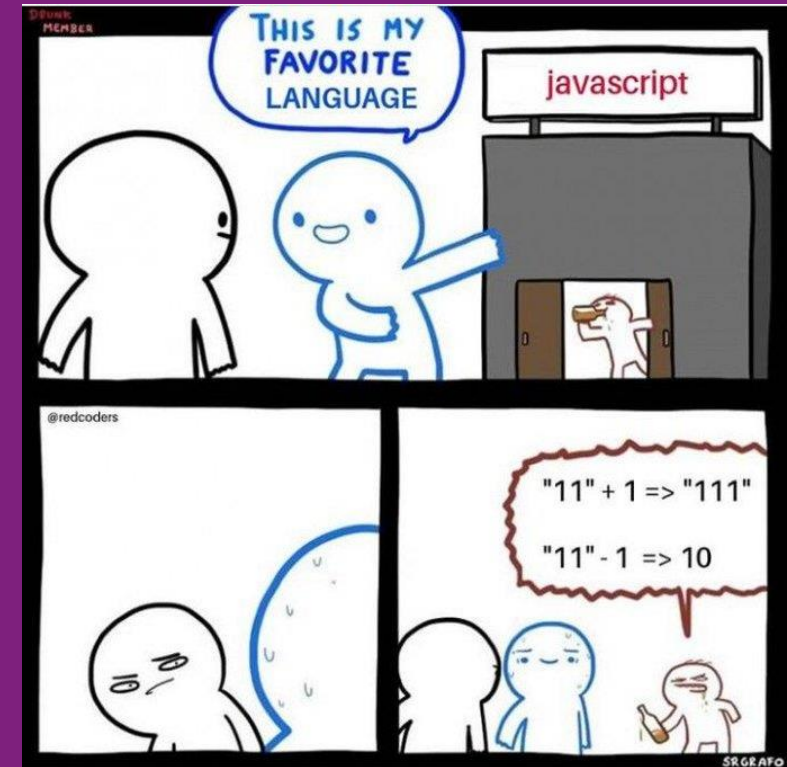
### Cos'è JavaScript?

Tenete bene a mente che JavaScript è un linguaggio di programmazione **debolmente tipizzato!**

#### Il meme spiegato...

In JavaScript, l'operatore di somma (+) esegue la concatenazione di stringhe quando uno dei due operandi è una stringa. Quando si scrive `"11" + 1`, il numero 1 viene automaticamente convertito in una stringa e concatenato con la stringa `"11"`. Pertanto, l'espressione viene valutata come `"111"`.

D'altra parte, l'operatore di sottrazione (-) esegue la sottrazione quando entrambi gli operandi sono numeri. Nell'espressione `"11" - 1`, la stringa `"11"` viene automaticamente convertita in un numero (11), quindi l'espressione viene valutata come la differenza tra i numeri 11 e 1, che è 10.



## Cosa posso fare con JavaScript?

- Posso inserire codice HTML in una pagina
- Posso reagire ad un “evento”, come ad esempio il click di un pulsante
- Posso leggere e scrivere/sovrascrivere elementi HTML partendo dal loro id, tipo, classe...
- Può essere molto utile per validare dati o campi!





# Alcune regole base di JavaScript

- Di design, ogni asserzione in JavaScript deve terminare con un punto e virgola.
- Ricordate che JavaScript è case sensitive!
- Come si commenta il codice di JavaScript:
  - I commenti su linea singola cominciano con il doppio slash (//)
  - I commenti multi linea invece cominciano con “/\*” e terminano con “\*/”

Every time I write  
documentation



## Day 2– Integrazione di JavaScript in una pagina HTML

### Come integrare JS in una pagina HTML?

- Tramite il tag `<script></script>`
- Includendo il file `.js` nel documento HTML con il tag `<script></script>` e l'attributo `src`

E' consigliabile importare uno script nella parte finale della pagina, poco prima del tag `</body>`, in modo da non rallentare il rendering della pagina ed essere sicuri che quest'ultima sia stata caricata del tutto prima di eseguire del codice javascript

In che punto del documento inserisco il tag? :  
<https://stackoverflow.com/questions/436411/where-should-i-put-script-tags-in-html-markup>

### Integrazione diretta

```
<!DOCTYPE html>
<html>
  <body>
    <p id="demo"></p>
    <script>
      document.getElementById("demo").innerHTML = "Hello JavaScript";
    </script>
  </body>
</html>
```

Hello JavaScript!

### Integrazione tramite risorsa esterna

#### Documento HTML

```
<!DOCTYPE html>
<html>
  <body>
    <h1>The script src attribute</h1>
    <script src="demo_script.js">
    </script>
  </body>
</html>
```

#### File .js indicato in src

```
document.write("This text comes
from an external script.");
```

#### The script src attribute

This text comes from an external script.



## Day 2 – Interazione con gli elementi HTML

### Richiamare uno script da un bottone

Gli eventi permettono di interagire con gli elementi HTML oppure invocare intere funzioni in JavaScript. Per adesso limitiamoci ad invocare uno script cliccando un bottone.

- Con la funzione **onclick**: è possibile fare in modo che al click del mouse su uno specifico elemento, venga invocata una specifica funzione

Vedremo in seguito alcuni esempi interessanti...

Per approfondimenti: [https://www.w3schools.com/jsref/event\\_onclick.asp](https://www.w3schools.com/jsref/event_onclick.asp)

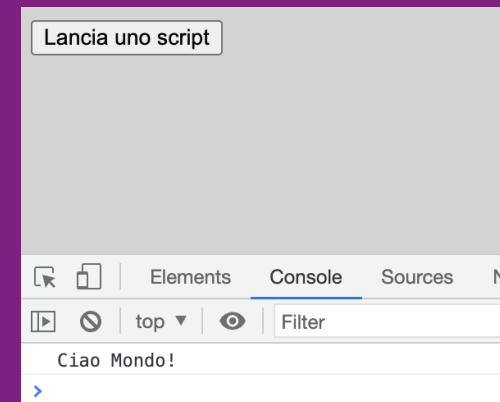
## HTML

```
<!doctype html>
<html lang="en">
  <head>
    <title>Hello, world!</title>
    <style>
      body {
        background-color: lightgray
      }
    </style>
  </head>
  <body>
    <button onclick="myProcedure()">Lancia uno script</button>
    <script src="./assets/test.js"></script>
  </body>
</html>
```

## Contenuto del file JavaScript

```
function myProcedure() {
  console.log("Ciao Mondo!");
}
```

## Output nella console del browser



## Alcune regole base di JavaScript – Valori aritmetici

Dato che  $y=5$ , lo schema qui sotto spiega come agiscono gli operatori aritmetici in JavaScript

Sign	Description	Example	Result
+	Addition	$x=y+2$	$x=7$
-	Subtraction	$x=y-2$	$x=3$
*	Multiplication	$x=y*2$	$x=10$
/	Division	$x=y/2$	$x=2.5$
%	Modulus (division remainder)	$x=y\%2$	$x=1$
++	Increment	$x=++y$	$x=6$
--	Decrement	$x=--y$	$x=4$

# Alcune regole base di JavaScript – Stringhe e numeri

Come anticipato precedentemente, JavaScript è un linguaggio debolmente tipizzato, quindi a differenza di altri linguaggi con tipizzazione forte, è possibile sommare elementi di tipo diverso e non ottenere errori:

Ecco alcuni esempi di addizioni tra tipi diversi:

```
> x=5+5;  
< 10  
  
> x="5"+"5"  
< '55'  
  
> x=5+"5"  
< '55'  
  
> x="5"+5  
< '55'
```



## Alcune regole base di JavaScript – comparatori logici

Ecco alcuni esempi di comparatori logici in JavaScript... Attenzione alle differenze con java!

Dato x=5

Sign	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x==5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

### Math Class

1 = 1  
1 ≠ 2



### Normal Coding Languages

1 == 1  
1 != 2



### Javascript

1 === 1  
1 !== 2



## Gli array in JavaScript

### Come si dichiara un array in JavaScript?

Il metodo più corretto per dichiarare un array in Javascript è il seguente:

```
const test = []
```

E' consigliabile dove possibile utilizzare la definizione const se l'array non dovrà essere ri assegnato, sarà sempre possibile modificarne il contenuto (non è immutabile)

E' anche possibile inizializzare array con dei dati in questo modo:

```
const test = ["a", "b", "c"]
```

*Altri tipi di inizializzazione come ad esempio:*

```
const test = new Array()
```

*Sono possibili ma disincentivati!*

Per approfondimenti: [https://www.w3schools.com/js/js\\_arrays.asp](https://www.w3schools.com/js/js_arrays.asp)



# Gli array in JavaScript

### Alcune funzioni utili sugli array:

- Metodo `.push()` aggiunge uno o più elementi alla fine dell'array
- Metodo `.unshift()` aggiunge uno o più elementi all'inizio dell'array
- Ottenere un dato elemento dell'array sapendo la posizione
- Metodo `.forEach()` per ottenere tutti gli elementi di un array
- Metodo `.map()` crea un nuovo array con i risultati di una funzione su ogni elemento
- Metodo `.filter()` crea un array formato dagli elementi che rispettano la funzione filtro

```
const fruits = ['apple', 'banana'];  
fruits.push('peach', 'orange');  
console.log(fruits);
```

```
► (4) ['apple', 'banana', 'peach', 'orange']
```

```
const fruits = ['apple', 'banana'];  
fruits.unshift('peach', 'orange');  
console.log(fruits);
```

```
► (4) ['peach', 'orange', 'apple', 'banana']
```

```
const fruits = ['apple', 'banana', 'cherry'];  
console.log(fruits[1]);
```

```
banana
```

```
const fruits = ['apple', 'banana', 'cherry'];  
fruits.forEach(fruit => console.log(fruit));
```

```
apple
```

```
banana
```

```
cherry
```

```
const fruits = ['apple', 'banana', 'cherry'];  
const uppercaseFruits = fruits.map(fruit => fruit.toUpperCase());  
console.log(uppercaseFruits);
```

```
► (3) ['APPLE', 'BANANA', 'CHERRY']
```

```
const fruits = ['apple', 'banana', 'cherry'];  
const longFruits = fruits.filter(fruit => fruit.length > 5);  
console.log(longFruits);
```

```
► (2) ['banana', 'cherry']
```



## Alcuni esempi di comparazione

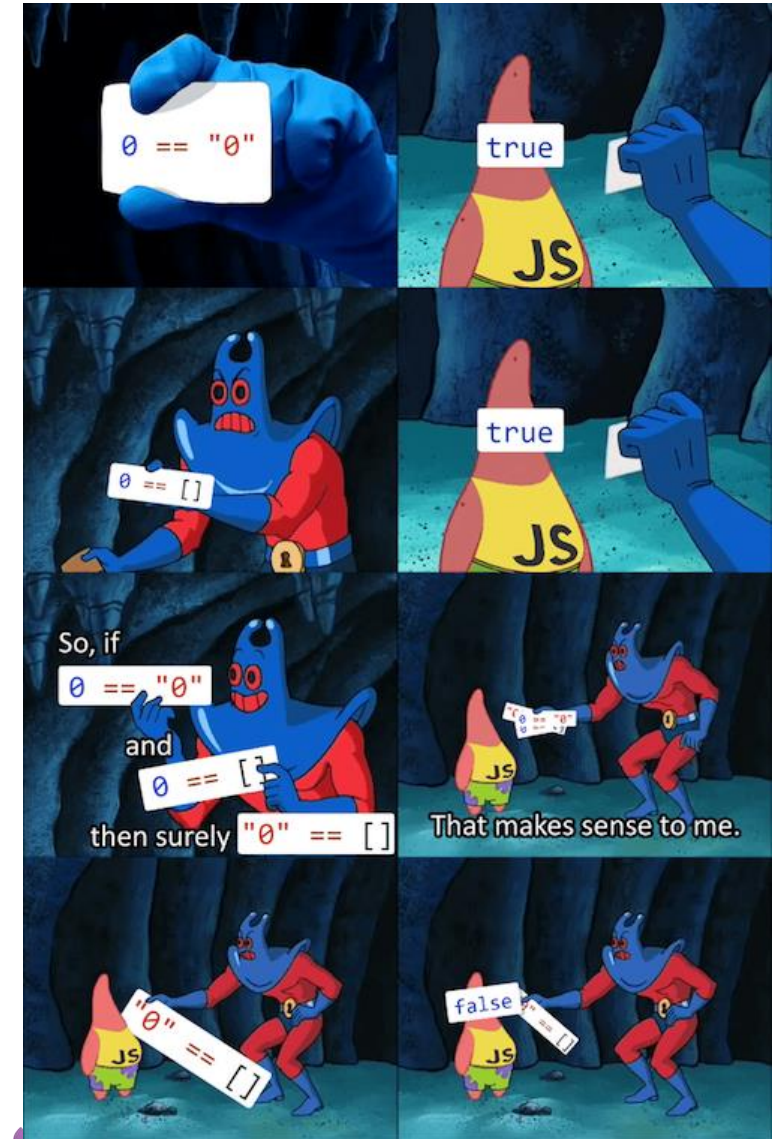
Con:

- `==` i valori vengono convertiti prima di essere comparati

```
0 == "";           // true
1 == "1";          // true
1 == true;         // true
```

- `===` viene confrontato il valore e il tipo

```
0 === "";          // false
1 === "1";         // false
1 === true;        // false
```





## Day 2 – Introduzione a JavaScript

# I valori NaN, null e undefined

Le variabili o gli oggetti in JavaScript possono assumere anche i seguenti valori:

### NaN (Not a Number)

Quando il valore di un'operazione dovrebbe essere un numero, ma non lo è, il valore di ritorno è NaN. NaN è un numero

```
> typeof NaN
< "number"

> "cucumber" - 10
< NaN

> 12 / "R2D2"
< NaN
```

Però ci sono alcuni casi che JavaScript riesce ad interpretare, anche se gli addendi non sono numeri

```
> 20000 + "miles"
< "20000miles"

> true + "false"
< "truefalse"

> "R" + 2 + "D" + 2
< "R2D2"

> NaN == NaN
< false

> NaN === NaN
< false

> true + 10
< 11

> false - 15
< -15

> true * 55
< 55

> false / 999
< 0

> true * false
< 0
```

### null

Rappresenta una referenza ad un oggetto invalido. Di solito è intenzionale.

```
> typeof null
< "object"

>
```

Nelle operazioni matematiche null è considerate come 0

```
> null + 5
< 5

> null - 2
< -2

> null * 27
< 0

> null/10
< 0

> null%3
< 0

> !null
< true

> null == false
< false

> null === false
< false

> null == true
< false

> null === true
< false

>
```

### undefined

Quando una variabile non ha un valore assegnato, il valore della variabile è undefined, che è una primitiva.

```
> var a;
< undefined

> a
< undefined

> typeof undefined
< "undefined"

>
```

Nelle operazioni aritmetiche, se si include undefined, il risultato dell'operazione sarà NaN

```
> !undefined
< true

> undefined == false
< false

> undefined === false
< false

> undefined == true
< false

> undefined === true
< false
```

# Come definire una funzione

Una funzione in JavaScript può essere dichiarata in vari modi:

- Tramite la keyword `function` come dichiarazione

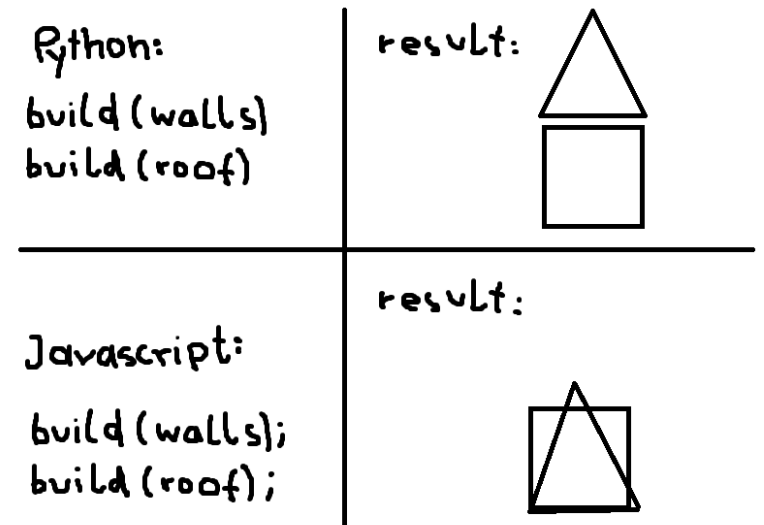
```
function multiply(x, y) {  
  return x * y;  
}
```

- Come espressione

```
const multiply = function (x, y) {return x * y};
```

- Inline (*Arrow function*)

```
const multiply = (x, y) => x * y;
```



# Gli oggetti in JavaScript

Un oggetto in JavaScript può contenere sia dati che logiche di business (come metodi). Un oggetto può contenere più variabili a cui sono associati i rispettivi valori.

```
const car = {type:"Fiat", model:"500", color:"white"};
```

Una variabile dell'oggetto può essere valorizzata anche con un metodo

```
const person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

## Day 2 – Introduzione a JavaScript

### Variabili globali e locali

In JavaScript è possibile definire variabili con diverso *scope*:

- Globale: ha effetto in tutto il *lifecycle* della web page
- Locale: ha effetto solo nel blocco di codice in cui la variabile è dichiarata
- **let**: è una keyword con cui è possibile dichiarare una variabile il cui scope è delimitato dalle parentesi del blocco di codice "{ }"
- **var**: è una keyword con cui è possibile dichiarare una variabile il cui scope **non** è delimitato dalle parentesi del blocco di codice "{ }"  
Se si ridefinisce la variabile con **var**, questa viene sovrascritta
- **const**: si comporta come **let**, ma la variabile è valorizzabile solo in fase di dichiarazione

```
{  
  let x = 2;  
}  
  
// x NON può essere utilizzata qui
```

```
{  
  var x = 2;  
}  
  
// x può essere utilizzata qui
```



Per approfondimenti: <https://www.tutorialspoint.com/difference-between-var-and-let-in-javascript>

## Day 2 – Integrazione di JavaScript in una pagina HTML

### Differenza pratica tra let e var

La variabile **var** ha una **portata “globale”** ed il suo valore può essere modificato in blocco di codice interno (delimitato da {})

La variabile **let**, invece, ha una **portata “locale”** ed il suo valore è limitato al blocco in cui è dichiarata.

*Ciò rende let più adatta alle variabili locali, mentre var alle variabili globali.*

Per approfondimenti: [https://www.w3schools.com/js/js\\_let.asp](https://www.w3schools.com/js/js_let.asp)

```
function esempioVar(){
    var x = 10;

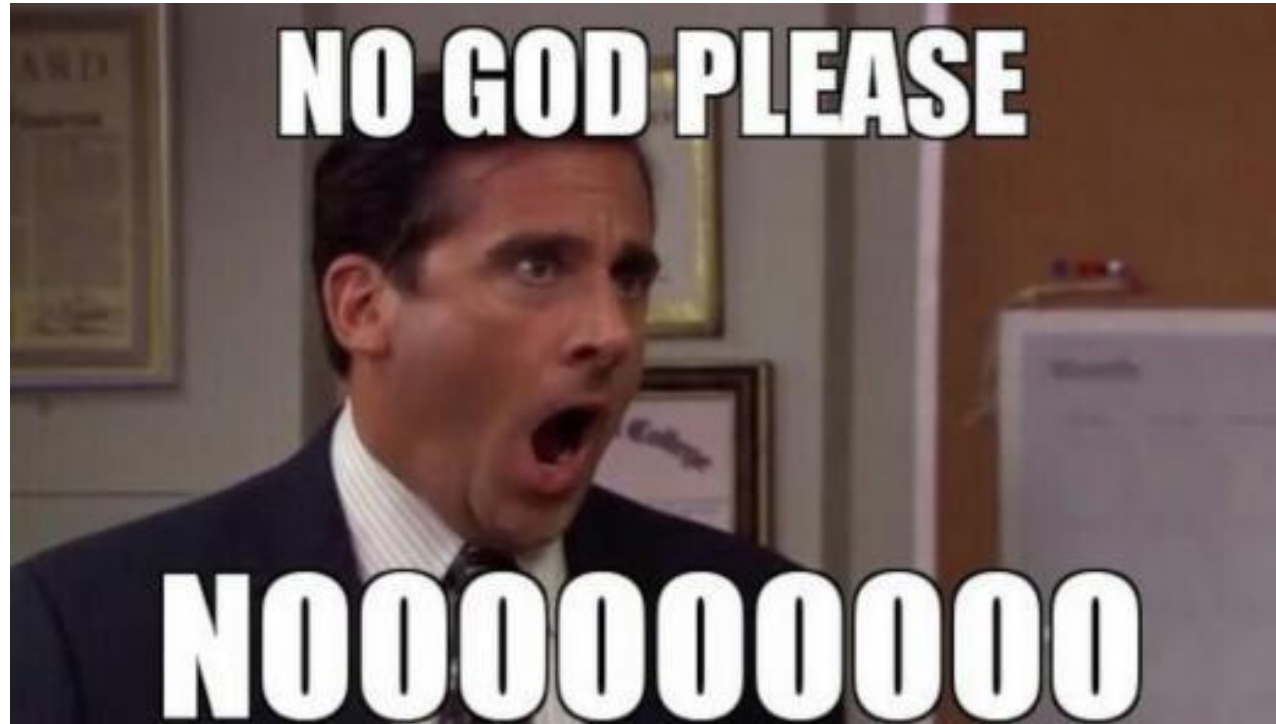
    if (true) {
        var x = 20;
        console.log(x); // Output 20
    }

    console.log(x); // Output 20
}
```

```
function esempioLet(){
    let x = 10;

    if (true) {
        let x = 20;
        console.log(x); // Output 20
    }

    console.log(x); // Output 10
}
```



## Day 2 – Interazione con gli elementi HTML

### Trovare un elemento HTML da JS

Nell'esempio voglio trovare tutti gli elementi *p* e richiamare il contenuto.

Poi stampo il contenuto in un altro *p*.

Come per CSS, possiamo avere diversi tipi di selettori. Nell'esempio il selettore è l'elemento HTML, ma possiamo anche selezionare un elemento tramite:

- **Tag name:**  
`document.getElementsByTagName("p");`
- **Identificativo:**  
`document.getElementById("main");`
- **Classe:**  
`document.getElementsByClassName("intro");`

```
<!DOCTYPE html>
<html>
  <body>
    <h2>JavaScript HTML DOM</h2>
    <p>Finding HTML Elements by Tag Name.</p>
    <p>This example demonstrates the <b>getElementsByTagName</b> method.</p>
    <p id="demo"></p>
    <script>
      const element = document.getElementsByTagName("p");
      document.getElementById("demo").innerHTML = 'The text in first paragraph (index 0) is: ' +
element[0].innerHTML;
    </script>
  </body>
</html>
```

Finding HTML Elements by Tag Name.

This example demonstrates the `getElementsByTagName` method.

-----

The text in first paragraph (index 0) is: Finding HTML Elements by Tag Name.



## Day 2 – Interazione con gli elementi HTML

### Cosa si può fare con un elemento del DOM

Partendo dall'esempio precedente, proviamo a manipolare un paragrafo partendo dal suo **id** univoco appositamente assegnato.

HTML:

```
<body>
  <p id="paragrafo">Questo è un paragrafo.</p>
  <button onclick="modificaParagrafo()">Modifica il paragrafo</button>
<script src="./assets/test.js"></script>
</body>
```

JavaScript:

```
function modificaParagrafo() {
  document.getElementById('paragrafo').innerHTML = 'Il testo è stato modificato.';
}
```

Questo è un paragrafo.

Modifica il paragrafo

## Day 2 – Interazione con gli elementi HTML

### Modifiche nel DOM?

Come faccio a cambiare la proprietà di un elemento del DOM, tipo la sorgente di una immagine?

HTML:

```
<body>

  <button onclick="injection()">Inject</button>

  <img id="meme" src=""></img>

<script src="./assets/test.js"></script>
</body>
```

JavaScript:

```
function injection() {
  var image = document.getElementById("meme");
  image.src = "https://tinyurl.com/56nwx87w";
}
```



## Day 2 – Interazione con gli elementi HTML

# Passare dati da una pagina all'altra... Due vie, due filosofie...

Come faccio a cambiare la proprietà di un elemento del DOM, tipo la sorgente di una immagine?

### *Un possible metodo: Url Query String*

La pagina 1 invia la variabile tramite anchor:

```
<body>
  <!-- voglio passare come parametro url, la stringa "http://photo.com"-->
  <a href="/pag2.html?url=http://photo.com">Go To Pag2</a>
  <script src="/assets/test.js"></script>
</body>
```

La pagina 2 intercetta il valore e lo stampa:

```
<body>

  <h1>PAGINA 2</h1>
  <span id="span"></span>

  <script>
    //Prendo i parametri dall'url
    const urlParams = new URLSearchParams(window.location.search);
    //Prendo il parametro con nome url
    const url = urlParams.get('url');
    //Scrivo nella pagina web
    document.write("L'url passato è: " + url);
  </script>
</body>
```

[Go To Pag2](#)

Per approfondimenti: <https://www.w3docs.com/snippets/javascript/how-to-get-url-parameters.html>

## Day 2 – Interazione con gli elementi HTML

# Passare dati da una pagina all'altra... Due vie, due filosofie...

Come faccio a cambiare la proprietà di un elemento del DOM, tipo la sorgente di una immagine?

### *Un altro metodo: Local Storage*

La pagina 1 invia la variabile tramite button ed effettua il redirect:

```
<body>
  <!-- voglio passare come parametro url, la stringa "http://photo.com" -->
  <button onclick="goToPag2()">Vai alla pagina 2</button>
<script>
  function goToPag2(){
    localStorage.setItem("url", "http://photo.com"); //salva la variabile
    window.location.href = "./pag2.html"; //effettua il redirect
  }
</script>
</body>
```

La pagina 2 intercetta il valore e lo stampa:

```
<body>
  <h1>PAGINA 2</h1>
  <span id="span"></span>
  <script>
    //Prendo il valore url dal Local Storage
    let url = localStorage.getItem('url');
    //Scrivo nella pagina web
    document.write("L'url passato è: " + url);
  </script>
</body>
```

Vai alla pagina 2

### La differenza tra i due metodi

#### Url Query String

##### Pro:

- Semplice da utilizzare; le variabili possono essere passate tramite l'aggiunta di parametri all'url.
- Accessibile: le variabili sono facilmente accessibili a livello di codice.
- Visibile: le variabili possono essere viste e condivise dagli utenti, poiché fanno parte dell'URL.

##### Contro:

- Dimensione limitata: la quantità di dati che è possibile passare tramite l'URL è limitata.
- Sicurezza: i dati passati tramite l'URL non sono sicuri, poiché possono essere facilmente manipolati o visualizzati da terze parti.
- Non adatto ai dati sensibili: non è consigliabile passare dati sensibili, come le password, tramite l'URL.

#### Local Storage

##### Pro:

- Maggiore capacità di archiviazione: il local storage offre una maggiore capacità di archiviazione rispetto all'URL query string.
- Sicurezza: i dati archiviati nel local storage sono più sicuri, poiché possono essere crittografati e protetti da password.
- Persistente: i dati nel local storage sono persistenti, ovvero rimangono disponibili anche dopo che l'utente ha chiuso il browser o riavviato il dispositivo. (cache dati browser)

##### Contro:

- Accessibilità limitata: i dati nel local storage possono essere accessibili solo tramite codice JavaScript, rendendoli meno accessibili per alcune operazioni.
- Dimensione limitata: anche se maggiore rispetto all'URL query string, la dimensione del local storage è comunque limitata.

## Day 2 – Manipolazione delle date in JavaScript

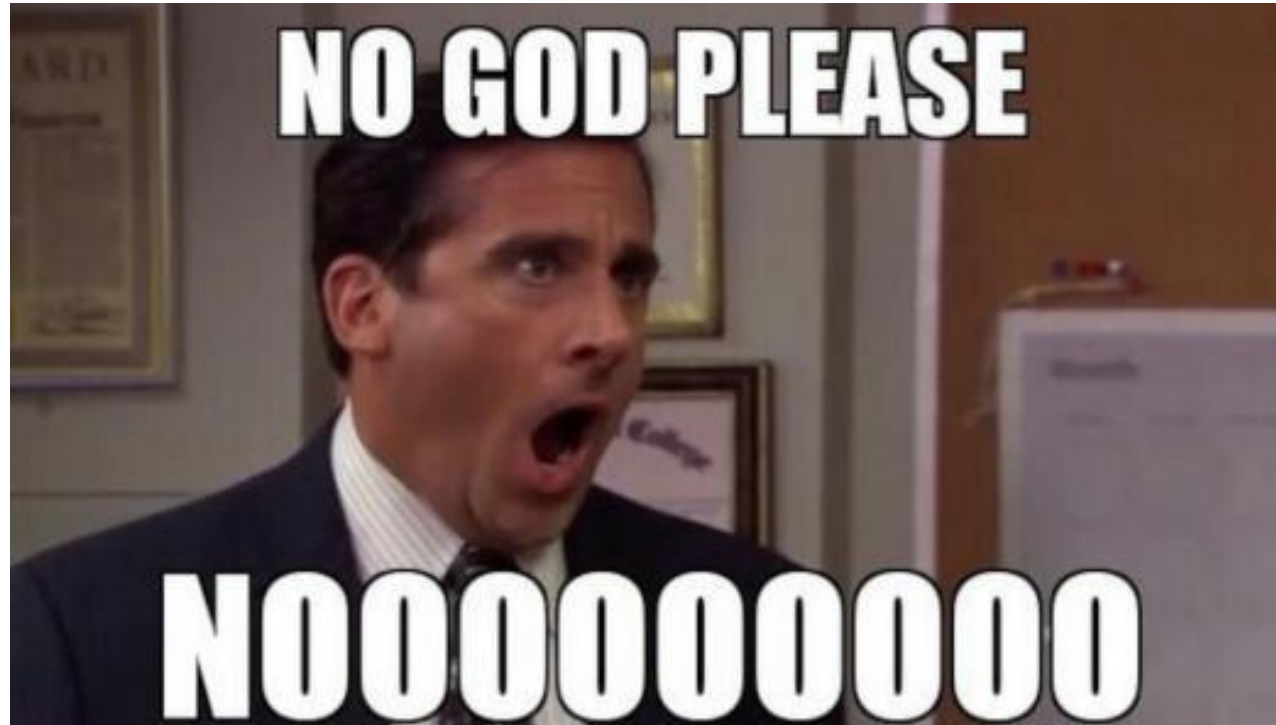
### Dichiarazione e formattazione del Type Date

- L'oggetto **Date** permette di definire e manipolare una data
- La data è un oggetto composto da più elementi che la compongono, quale ad esempio il giorno della settimana, il mese, il giorno, l'anno e l'ora
- È possibile selezionare ogni elemento singolarmente tramite gli specifici metodi. Eccone alcuni:
  - `getDate()` restituisce il numero del giorno del mese nell'intervallo 1-31
  - `getMonth()` restituisce il numero del giorno del mese nell'intervallo 0-11

```
const d0 = new Date()
const d1 = new Date(2018, 11, 24, 10, 33, 30, 0) // parametro data e orario specifici
const d2 = new Date(1000000000) // parametro in ms (parte dalla Unix epoch)
const d3 = new Date("2015-03-25") // parametro in formato ISO 8601

// console.log(d0): Mon May 30 2022 18:14:58 GMT+02:00 (Ora legale dell'Europa centrale)
// console.log(d1.getDate()): 24
// console.log(d2.getDay()): 6
// console.log(d3.getSeconds()): 0
```

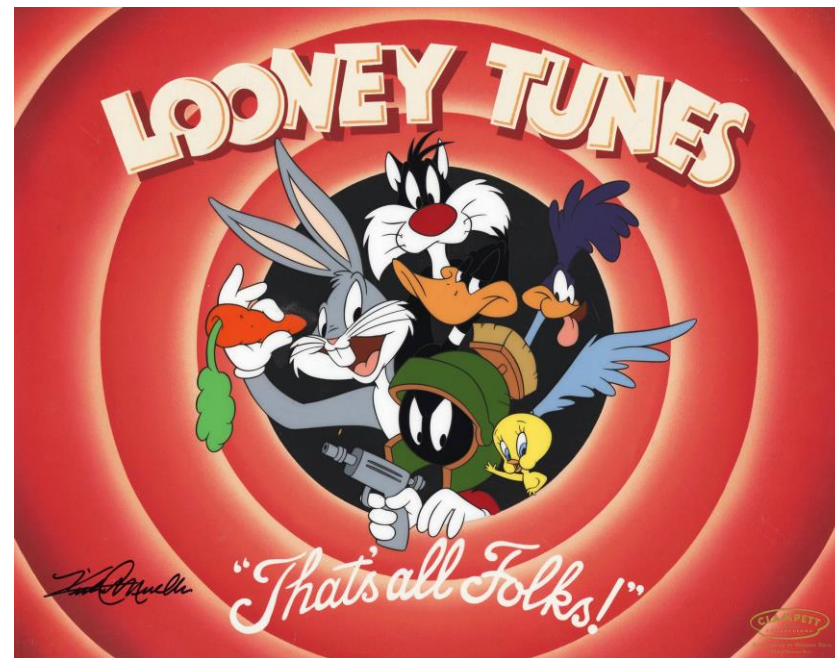
Per approfondimenti: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date)







Experience.  
Create.  
Inspire.



# Thank You

Vitale Esca

IBM Client Innovation Center  
Italy