

Corso Enaip
Rimini 22/02/2023

Chi sono



Giuseppe Trisciuoglio

CTO e Enterprise Architect

Classe '83

Lavoro nel IT dal 2003

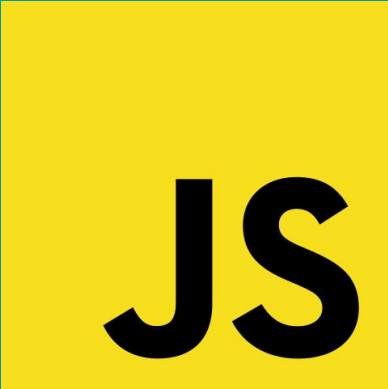
I miei contatti sono:

email: giuseppe@laborinto.io

telegram: eurotrip11

Conosciamoci

Introduzione a

A solid yellow square containing the letters 'JS' in a bold, black, sans-serif font.

JS

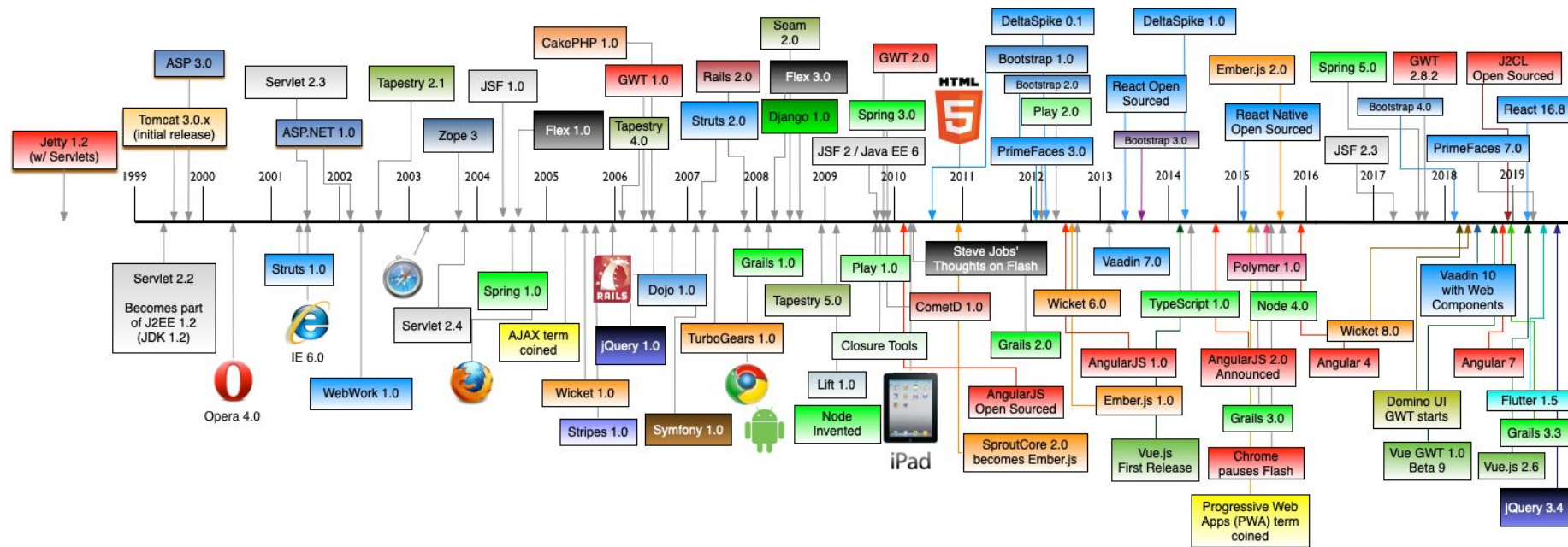
Javascript

JavaScript è uno dei linguaggi di programmazione più usati al mondo JavaScript è stato ideato nel 1995 da Netscape e rilasciato con la versione 2.0 del suo browser, *Netscape Navigator*, dapprima con il nome **LiveScript** e subito dopo con l'attuale nome, creando all'inizio non poca confusione con Java che proprio in quell'anno debuttava con grande attenzione da parte del mondo del software

JavaScript aggiunse alle pagine HTML la possibilità di essere modificate in modo dinamico, in base all'interazione dell'utente con il browser (lato client). Questo grazie alle funzionalità di calcolo e di manipolazione dei documenti che era possibile effettuare anche senza coinvolgere il server.

Versione Attuale JS EcmaScript 13

Storia di Javascript e delle tecnologie web



Gli strumenti di lavoro

Come per qualsiasi linguaggio di programmazione, per iniziare a lavorare con JavaScript c'è bisogno di almeno tre strumenti:

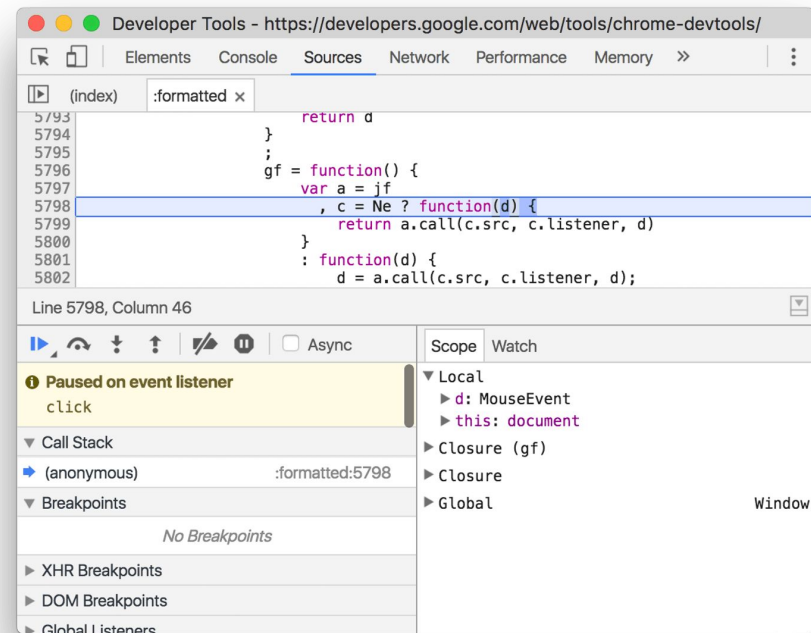
- un editor;
- un interprete o compilatore;
- un debugger.

Come editor è naturalmente possibile utilizzare un comune editor di testo. Molti hanno ormai funzionalità di supporto allo sviluppo con vari linguaggi di programmazione, tra cui anche JavaScript: evidenziazione della sintassi, completamento automatico, snippet di codice. I maggiori IDE di sviluppo per JavaScript sono:

- Microsoft Visual Studio Code
- JetBrains WebStorm
- Eclipse

Strumenti di sviluppo nei browser

Gli ambienti di sviluppo che prevedono un engine hanno in genere un debugger integrato che consente di analizzare il codice durante l'esecuzione. In assenza di un ambiente di sviluppo integrato, possiamo comunque ricorrere a un comune browser, dal momento che i più diffusi prevedono ormai un ambiente di debugging. Come ad esempio Chrome con i devtools.



Strumenti di sviluppo nei browser

Naturalmente anche **Safari** e **Firefox** hanno ambienti analoghi ai DevTools di Chrome e tutti e tre i browser prevedono un ambiente per l'esecuzione diretta di codice JavaScript, la cosiddetta console.

In questa finestra speciale possiamo scrivere codice JavaScript per vederne l'immediata esecuzione. Tutti gli esempi di codice che faremo durante il corso potranno essere eseguiti all'interno della console del vostro browser preferito per verificarne il funzionamento.

La stessa console può essere utilizzata come ambiente per la visualizzazione dell'output di uno script, grazie all'omonimo oggetto console.



Inserire codice JavaScript in una pagina HTML

Per utilizzare Javascript in una pagina web:

- inserire codice inline;
- scrivere blocchi di codice nella pagina;
- importare file con codice JavaScript esterno.

```
<button type="button" onclick="alert('Ciao Mondo!')">Cliccami</button>  
<script>alert('Ciao Mondo!')</script>  
<script src="codice.js"></script>
```

Codice inline

Il primo approccio, l'inserimento di codice inline, consiste nell'inserire direttamente le istruzioni JavaScript nel codice di un elemento HTML, assegnandolo ad un attributo che rappresenta un evento.

```
<button type="button" onclick="alert('Ciao Mondo!')">Cliccami</button>
```

Abbiamo assegnato all'attributo **onclick** dell'elemento **button** la stringa **alert('Ciao Mondo!')**. L'attributo **onclick** rappresenta l'evento del clic sul pulsante del mouse, quindi in corrispondenza di questo evento verrà analizzato ed eseguito il codice JavaScript assegnato. Nel caso specifico verrà visualizzato **un box** con la scritta **Ciao Mondo!**

```
<a href="javascript:alert('Ciao Mondo!')">Cliccami</a>
```

Blocchi di codice, il tag <script>

L'approccio inline può risultare immediato ma spesso si ha la necessità di definire variabili e funzioni. Per questo motivo si può utilizzare il tag **<script>** per inserire blocchi di codice in una pagina HTML.

Quando il **parser HTML del browser** esamina la pagina, riconosce il tag **<script>** come blocco di codice e ne passa il contenuto all'engine JavaScript, che lo **esegue**.

```
<!doctype html>
<html lang="it">
<head><title>Ciao Mondo!</title></head>
<body>
<h1>Ciao Mondo!</h1>
<p>Questa è la nostra prima pagina HTML!</p>
<script>
    apriModale('Ciao Mondo');

    function apriModale(messaggio){
        alert(messaggio);
    }
</script>
</body>
</html>
```

JavaScript esterno

Il terzo approccio consiste nel collegare alla pagina HTML, codice JavaScript presente **in un file esterno**. Questo permette di agganciare script e librerie (anche di terze parti) in modo non intrusivo, con il vantaggio di una separazione netta tra la struttura del documento html e il codice javascript.

Per inserire un file JavaScript esterno ci serviamo sempre del tag **<script>** in cui specificando l'attributo **src**.

```
<!doctype html>
<html lang="it">
<head><title>Ciao Mondo!</title></head>
<body>
<h1>Ciao Mondo!</h1>
<p>Questa è la nostra prima pagina HTML!</p>
<script src="https://www.adriasonline.it/codice.js"></script>
</body>
</html>
```

Gli elementi di base del linguaggio

Regole sintattiche di base (;)

Il codice JavaScript è composto da una **sequenza di istruzioni** che viene interpretata ed eseguita dall'engine. In questa sequenza, ciascuna istruzione (o blocco di istruzioni) è **delimitata da un punto e virgola (;)**

```
var i = 10;
```

```
i = i + 1;
```

In molti casi il ; finale non è obbligatorio, **ma è consigliato**, in alcuni casi potrebbero verificarsi comportamenti non previsti.

Regole sintattiche di base (case sensitive)

Un altro aspetto sintattico da tenere in considerazione è il fatto che JavaScript è **case sensitive**, cioè fa distinzione tra maiuscole e minuscole nei nomi di istruzioni, variabili e costanti.

Dimenticare questo aspetto può essere fonte di problemi quando, ad esempio, andiamo a dichiarare una variabile con il nome **MiaVariabile** e poi la richiamiamo con **Miavariabile**. Tanto più che normalmente un engine JavaScript non evidenzia alcun errore.

Regole sintattiche di base (case sensitive)

Per questo motivo è consigliato l'utilizzo delle annotazioni CamelCase o snake_case.

CamelCase: Il nome deriva dai "*sal**ti*" all'interno di una parola, che fanno venire in mente le gobbe di un cammello. La prima lettera può essere sia maiuscola (es. CamelCase), come il nome delle classi in PHP, che minuscola (es. camelCase), come le proprietà di un oggetto in PHP.

snake_case: è la pratica di scrivere gli identificatori separando le parole che li compongono tramite trattino basso (o underscore: `_`), solitamente con le prime lettere delle singole parole in minuscolo, e la prima lettera dell'intero identificatore minuscola o maiuscola (es. "foo_bar" e "Hello_world").

Regole sintattiche di base (commenti)

Come ogni linguaggio, JavaScript prevede delle sequenze di caratteri per inserire commenti nel codice. Tutto ciò che viene marcato come commento non viene preso in considerazione dall'interprete JavaScript. Possiamo inserire due tipi di commenti nel codice:

- commento per singola riga;
- commento multiriga.

Regole sintattiche di base (commenti)

```
/* Questo è un commento sulla prima riga  
Questo è un commento sulla seconda rig  
Questa è l'ultima riga del commento */
```

```
/**  
 * Questo è un commento sulla prima riga  
 * Questo è un commento sulla seconda rig  
 * Questa è l'ultima riga del commento */
```

```
//Questo è un commento sulla linea
```

Strutture dati in JavaScript

- Numeri (interi, float)
- Boolean (true, false)
- Stringhe(“esempio” o ‘esempio’)
- Oggetti
- Array
- null e undefined

Numeri in JavaScript

JavaScript ha un unico tipo di dato numerico, cioè **non c'è distinzione formale**, ad esempio, tra intero e decimale. Internamente tutti i valori numerici sono rappresentati come numeri in virgola mobile, ma se non è specificata la parte decimale il numero viene trattato come intero.

L'insieme dei numeri rappresentabili in JavaScript cade nell'intervallo compreso tra $-1.79769 \cdot 10^{308}$ e $1.79769 \cdot 10^{308}$, con una precisione pari a $5 \cdot 10^{-324}$.

Ogni valore che ha una precisione maggiore di quella rappresentabile viene considerata uguale a zero.

Un altro valore numerico speciale è **NaN**, acronimo di **Not a Number**, che indica un valore numerico non definito.

Numeri in JavaScript

```
var interoNegativo = -10;  
var zero = 0;  
var interoPositivo = 123;  
  
var numeroDecimale = 0.52;  
var altroNumeroDecimale = 12.34;  
var decimaleNegativo = -1.2;  
var decimaleZero = 1.0;  
  
var primoNumero = 12e3; // equivalente a  $12 \times 10^3$  cioè 12.000  
var secondoNumero = 3.5e-4; // equivalente a  $3.5 \times 10^{-4}$  cioè 0,00035  
  
var numeroOttale = 0123; // sistema in base 8 (equivalente a 83)  
var numeroEsadecimale = 0x123; // equivalente a 291
```

Boolean in JavaScript

Il tipo di dato booleano prevede due soli valori: **true** (vero) e **false** (falso).

```
var vero = true;  
var falso = false;
```

Stringhe in JavaScript

Una **stringa** in JavaScript è una **sequenza di caratteri delimitata da doppi o singoli apici**. Le seguenti sono esempi di stringhe:

```
"stringa aperta con i doppi apici"  
'stringa aperta con l\' apice singolo'
```

Non c'è una regola per stabilire quale delimitatore utilizzare. **L'unica regola è che il delimitatore finale deve essere uguale al delimitatore iniziale**. Questo ci consente di scrivere stringhe come le seguenti senza incorrere in errori:

```
'stringa aperta con un "solo" apice'  
"stringa aperta con l' apice doppio"
```


Oggetti in JavaScript

Un **oggetto** è un **contenitore** di valori eterogenei, messi insieme a formare una struttura dati unica e tale da avere generalmente una particolare identità. Normalmente, infatti, un oggetto è utilizzato per rappresentare un'entità specifica come ad esempio una persona, un ordine, una fattura, una prenotazione, etc. tramite un'aggregazione di dati e di funzionalità.

```
var oggettoVuoto = {};  
var persona = { "nome": "Mario", "cognome": "Rossi"};
```

Array in JavaScript

Gli **array** consentono di associare più valori ad un unico nome di variabile (o identificatore). In genere i valori contenuti in un array hanno una qualche affinità (ad esempio l'elenco dei giorni della settimana). L'uso degli array evita di definire più variabili e semplifica lo svolgimento di operazioni cicliche su tutti i valori.

```
var colori = ["rosso", "verde", "blue", "giallo", "magenta"];
```

Array in JavaScript

```
var myArray = [,"rosso"];  
var myArray = ["rosso", "verde",];  
var myArray = [123, "rosso", true, null];  
var myArray = [123, "stringa", ["verde", "rosso", 99]];
```

Array in JavaScript

```
var blue    = colori[2];  
var novantanove = myArray[2][2];
```

Array in JavaScript

La presenza di array come elementi di altri array ci consente di definire **array multidimensionali o matrici**.

```
var matrice = [[24, 13, 1], [48, 92, 17], [8, 56, 11]]
```

I tipi di dato null, undefined

JavaScript prevede due tipi di dato speciali per rappresentare valori nulli e non definiti.

Il tipo di dato **null** prevede il solo valore null, che rappresenta un valore che non rientra tra i tipi di dato del linguaggio, cioè non è un valore **numerico valido**, **né una stringa**, **né un oggetto**.

Il tipo di dato **undefined** rappresenta un valore che non esiste. Anche questo tipo di dato contiene un solo valore: **undefined**. Questo è il valore di una variabile non inizializzata, a cui non è stato assegnato nessun valore, nemmeno null.

Variabili in JavaScript

In JavaScript possiamo utilizzare le **variabili** per memorizzare valori o oggetti durante l'esecuzione degli script. Una variabile è identificata da un nome che deve rispettare alcune regole:

- non deve coincidere con una delle parole chiave del linguaggio;
- non può iniziare con un numero;
- non può contenere caratteri speciali come ad esempio:
 - lo spazio,
 - il trattino (-),
 - il punto interrogativo (?),
 - il punto (.),
 - ecc.
- Sono però ammessi:
 - l'underscore (_)
 - il simbolo del dollaro (\$).

Variabili in JavaScript

```
var nomeVariabile = "";  
var nome_variabile = "";  
var $nomeVariabile = null;  
$corsoEnaip = 10;
```


Costanti in JavaScript

Fino alla versione 5 di ECMAScript **non era presente il concetto di costante**, cioè di un valore non modificabile. Era necessario ricorrere alle variabili, eventualmente con una notazione convenzionale per indicare che si tratta di valori da non modificare nel corso dell'esecuzione dello script. A partire dalla versione 6 dello standard viene introdotta la possibilità di dichiarare costanti tramite la parola chiave **const**. Questo **garantisce l'utilizzo in sola lettura del valore**, evitando rischi di modifica durante l'esecuzione.

Costanti in JavaScript

```
const SCUDETTI_VINTI_DALLA_AS_ROMA = 3;
```

Operatori in JavaScript

- Matematici (+, -, *, /, %)
- Boolean (&&, ||, !)
- Stringa (+)
- Confronto (<, <=, >, >=, ==, !=, ===, !==)
- Assegnazione (=, *=, /=, %=, +=, -=)
- Incremento (++ , —)
- Condizionale(?:)

Setup ambiente di sviluppo

Ambiente di sviluppo su Windows

Requisiti di sistema

- Windows 10 o superiori (64-bit), x86-64 based.
- Almeno 5GB di spazio su disco
- Tool di sviluppo richiesti:
 - Power Shell (versione 5 o superiori)
 - Chocolatey (<https://chocolatey.org/install>)
 - Git per Windows (versione 2.x)
 - Visual Studio Code
 - Node (versione 16)



Windows

PowerShell

Fai clic su **Start**, fai clic su Tutti i programmi, fai clic su **Accessori**, fai clic su Windows PowerShell e poi fai clic su **Windows PowerShell**.

Verificare che la versione sia uguale maggiore di 5 con il comando:

Get-Host | Select-Object Version

Se la versione è minore di 5, installare la versione più aggiornata al link:

<https://www.microsoft.com/download/details.aspx?id=54616>



PowerShell

Chocolatey

Eseguire PowerShell come Amministratore e digitare il seguente comando:

Set-ExecutionPolicy AllSigned

per consentire l'installazione di pacchetti esterni. Sempre nella stessa finestra digitate il seguente comando:

```
Set-ExecutionPolicy Bypass -Scope Process -Force;  
[System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor  
3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

Una volta terminata l'installazione verificare che il comando: **choco -?** ci indichi l'ultima versione del software.



Visual Studio Code

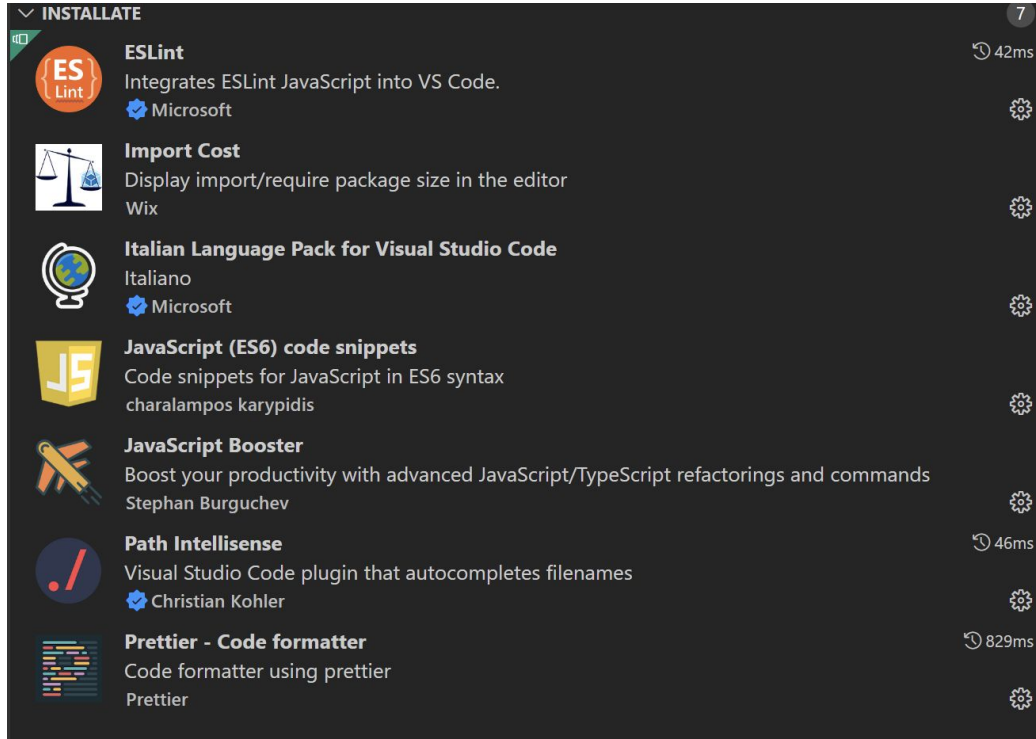
Eseguire PowerShell come Amministratore e digitare il seguente comando:

choco install vscode



Visual Studio Code

Visual Studio Code Estensioni



Git per Windows (versione 2.x)

Eseguire PowerShell come Amministratore e digitare il seguente comando:

choco install git



NodeJS per Windows (versione 16.x)

Eseguire PowerShell come Amministratore e digitare
il seguente comando:

```
choco install nodejs-lts
```



Domande?

