Corso Front End Developer JavaScript

Emanuele Galli

www.linkedin.com/in/egalli/



JavaScript

- F
- Linguaggio di programmazione interpretato, multiparadigma, imperativo, funzionale, event-driven
- Nato nel 1995 (Brendan Eich @ Netscape) per aggiungere funzionalità alla coppia HTML-CSS, è ora utilizzato un po' ovunque
- Dal 1997 ECMA ne coordina lo sviluppo, con il nome ufficiale di ECMAScript
- Nonostante il nome, è sostanzialmente diverso da Java

HTML – JavaScript

- Elemento script, nella head del documento
- Il codice può essere:
 - Scritto direttamente nell'elemento script (sconsigliato in produzione)
 - Caricato da un file JS esterno, specificato nell'attributo src

```
<head>
<head>
<!-- -->
                       <!-- -->
                       <script src="js/basic.js">
<script>
  <!-- codice JS -->
                       </script>
</script>
                       <!-- ... -->
<!-- ... -->
                       </head>
</head>
et target = document.getElennentById('target');
target.textContent = 'Hello!'
console.log('hello!');
```

Web Developer Tools

- Firefox / Chrome (DevTools)
- Scorciatoia comune per l'attivazione: ctrl+shift+i
 - Settings (F1), Advanced settings, Disable HTTP cache
 - Tab Debugger, accesso al codice
 - Tab Console, visualizzazione log
 - Tab Inspector, HTML widget
 - Tab Style Editor, CSS

Variabili

- Per dichiarare una variabile si usa let (o var)
- Non si esplicita il tipo, che può essere:
 - string: let name = 'Bob'; // apice singolo o doppi apici
 - <mark>六 number</mark>: let value = 42; // sia interi sia float
 - boolean: let flag = true; // o false
 - object: let dog = { name : 'Zip', breed : 'Alsatian' };
 array: let data = [1, 'Tom', false];



- Una variabile può cambiare il suo tipo associato nel corso della sua vita
- 🔁 L'operatore <mark>typeof()</mark> ritorna la stringa che descrive il tipo dedotto da JS (o <mark>undefined)</mark>
- Per dichiarare constanti si usa const
 - const z = 42;

Operatori aritmetici

- + addizione: 2 + 3
- - sottrazione: 2 3
- * moltiplicazione: 2 * 3
- / divisione: 2 / 3
- modulo o resto: 2 % 3 se io ho 1.23 e voglio solo 1 faccio 1.23%0.23;
- ** esponente: 2 ** 3 // vecchio stile: Math.pow(2, 3)
- ++ / -- incremento / decremento (sia prefisso sia postfisso)

Operatori di assegnamento

- Operatori che assegnano alla variabile sulla sinistra ...
 - = il valore sulla destra
 - = += la somma dei valori a sinistra e destra
 - -= la differenza tra il valore di sinistra e quello di destra
 - *= il prodotto del valore di sinistra per quello di destra
 - /= la divisione del valore di sinistra per quello di destra

Operatori relazionali

- Operatori che ritornano un booleano
 - ==== stretta uguaglianza (tesso tipo e valore)
 - !== di stretta disuguaglianza (diverso tipo o valore)
 - valore sulla sinistra è minore del valore sulla destra
 - <= minore o uguale
 - > il valore sulla sinistra è maggiore del valore sulla destra
 - >= maggiore o uguale
 - 📑 !! conversione a booleano, equivalente alla funzione Boolean() 📁 💥
- Gli operatori non-strict == e != vanno usati con cautela

Operatori logici (e bitwise)

F

ritornano un booleano se lo applico ai booleni, altrimenti li valuta e ritorna il valore delle variabili stessi

```
let alpha = true;
                           let beta = false;
                       console.log(alpha && beta); // false
      AND
& &
                      console.log(alpha || beta); // true console.log(!alpha); // false console.log(alpha & beta); // 0 console.log(alpha | beta); // 1
      OR
     NOT
     AND
                          let gamma = 101; // 5
      OR
                           let delta = 0b110; // 6
      XOR
                        console.log(gamma & delta); // 4 == 0100
console.log(gamma | delta); // 7 == 0111
console.log(gamma ^ delta); // 3 == 0011
console.log(gamma && delta); // 6
```

Stringa

- Una stringa è una sequenza di caratteri delimitata da apici singoli o doppi
- Per concatenare stringhe si usa il metodo concat() o l'operatore +
 - Conversione implicita da numero a stringa
 'Solution' + 42 === 'Solution42'
- Conversione esplicita da numero a stringa via toString()
 a.toString() === '42' // se a === 42
- Conversione esplicita da stringa a numero via Number()
 Number('42') === 42

Lavorare con stringhe <

- Lunghezza: s.<mark>length</mark> 📃
- Accesso ai caratteri: s[i] // i in [0, s.length-1]
- Ricerca di substr: s.indexOf(sub) // -1 not found
- Estrazione di substr: s.<mark>substr</mark>(i, sz), s.<mark>slice</mark>(i, j)
- Minuscolo: s.toLowerCase()
- Maiuscolo: s.toUpperCase()
- Modifica: s.replace(sub, other)
- Estrazione di componenti: s.split(',') // da stringa ad array

✓Array

- Collezione di oggetti di qualunque tipo
- Numero di elementi nella proprietà length
- Accesso agli elementi in lettura e scrittura

posso modificarli come voglio

- Scansione di tutto l'array via for loop
- Da array a string via join(), toString()
- Per aggiungere un elemento: push(), unshift()
- Per eliminare un elemento: pop(), shift(), splice()
- ordine alfabetico dei dati: sort()
- inversione dell'ordine: reverse() =

```
let data = [1, 'hello', [true, 42.24]];
console.log(data.length);
console.log(data], data[2][1]);
data[2] = false;
for(let i = 0; i < data.length; i++) {
   console.log(data[i]);
console.log(data.join(), data.toString());
data.pop();
data.shift();
data.push('push');
data.unshift('unshift');
```

Condizioni

- if else if else
 - se la condizione è vera, si esegue il blocco associato
 - altrimenti, se presente, si esegue il blocco "else"
- switch case default
 - Scelta multipla su valore
- Operatore ternario ?:
 - Ritorna la prima scelta se la condizione è vera, altrimenti la seconda

```
if (condition) {
    doSomething();
    else if (other) {
        doOther();
    } else {
        doAlternative();
    }
```

```
switch (value) {
   case 1:
     doOther();
     break;
   default:
     doStuff();
     break;
}
```

```
let result = condition ? choice1 : choice2;
```

- Preferito l'uso degli operatori strict === e !==
- Conversione implicita a boolean che ritorna true per valori che non sono false, undefined, null, 0, NaN, " (la stringa vuota)



Loop

```
while (condition) {
    // ...
    if (something) {
        condition = false;
    }
}
```

```
for (let i = 0; i < 5; i++) {
    // ...
    if (i == 2) {
        continue;
    }
    // ...
}</pre>
```

```
for (;;) {
    // ...
    if (something) {
        break;
    }
    // ...
}
```

Funzione

- Blocco di codice a cui è associato un nome, definite indicando
 - la keyword function
 - il nome (opzionale: funzioni anonime, notazione classica e "freccia")
 - una lista di parametri tra parentesi tonde

```
default x = 0, parametro 'rest' ...va =
```

- una lista di statement tra parentesi graffe
- In JavaScript sono oggetti, e dunque possono
 - essere assegnate a variabili, proprietà di oggetti, elementi di array
 - essere passate ad altre funzioni
 - contenere altre funzioni (metodi)
- Si invoca una funzione specificando
 - il suo nome
 - i valori da associare ai parametri se non specificati, default o undefined

```
function t() {
   console.log('hello');
}

function g(a, b) {
   return a + b;
}
```

```
let f1 = function(a, b) {
    return a + b;
}
let f2 = (a, b) => a + b;
```

Oggetto •

- Struttura, delimitata tra parentesi graffe, che contiene una lista di proprietà (attributi e metodi) separate da virgola
- Array associativo di proprietà definite come coppie chiave-valore
- Accesso proprietà per mezzo dell'operatore . o specificando il nome della proprietà fra parentesi quadre
- È possibile
 - aggiungere proprietà per assegnamento
 - rimuoverle via delete
 - usare un costruttore per semplificare la creazione

```
function Person(first, last) {
    this.first = first;
    this.last = last;
}
let p = riew Person('Tom', 'Jones');
```

Math **5**

Costanti e funzioni matematiche di uso comune

- Math.E, Math.PI, Math.SQRT ...
- Math.abs()
- Math.ceil() Math.floor()
- Math.cos(), Math.sin(), Math.tan(), ...
- Math.exp(), Math.pow(), Math.sqrt(...
- Math.max(), Math.min()

Date 🖪

- Data + ora fino al secondo
 - new Date()
 - new Date(2019, 10, 15, 20, 58, 51)
 - new Date("15 October 2019 12:23:15")
- Differenza: millisecondi tra due date
- Getter e setter per leggere o modificare componenti
 - getDate(), setDate(), ...

Destrutturazione

Estrazione di informazioni da array o oggetti in variabili distinte

```
let data = [1, 2, 3, 4, 5];
let [firs = second] = data; // i primi due elementi dell'array
let [a, , c, ...va] = data; // primo, terzo, e tutti gli altri

operatore spread metterà i "va" dalla posizione 3 in avanti
```

```
let x = 12;
let y = 24;
[x, y] = [y, x]; // swap
```

```
let obj = { a: 42, b: true };
let { a, b } = obj;
```

```
let obj = { a: 42, b: true };
let { a: age, b: flag } = obj; // estrazione con nuovi nomi
```

Template literals (o strings)

- Stringhe che gestiscono espressioni interne e in cui possiamo andare a capo esplicitamente invece di usare '\ n'
- Delimitate da accenti gravi (backtick alt-96 '`')
- Possono contenere placeholder, nel formato \${expr}

```
let x = 12;
let y = 24;
console.log(`Sum is ${x + y}`);
```

BOM: Browser Object Model

- Una pagina web viene visualizzata in un oggetto window
 - outerHeight, outerWidth, innerHeight, innerWidth
- alert(message)
 - confirm(message) // true = OK
 - Navigazione nella cronologia via history
 - back()
 - forward()

DOM: Document Object Model

- La pagina corrente è document
 - bgColor, fgColor: colore dello sfondo e del testo
 - title, URL
 - forms: array dei form nella pagina
 - ogni form è accedibile per indice o per 'name'
 - Getter di elementi
 - getElementById()
 - getElementsByClassName()=
 - getElementsByTagName()
 - getElementsByName()

Eventi su documento

- Associazione di eventi su elemento a codice JavaScript via attributo on...
- Se il risultato è false il comportamento standard viene annullato

Eventi & attributi

- Click del button submit in form: onsubmit posso metterlo anche in HTML
- Input prende/perde focus: onfocus, onblur
- Input blur + cambiamento: onchange
- Click su un elemento: onclick
- Mouse entra/esce: onmouseover, onmouseout

dal boxmodel dell'elemento

•

JSON ...

- JavaScript Object Notation
- Formato per lo scambio di dati basato su
 - Coppie nome-valore (oggetto JS)
 - Array di valori
- Da JSON a stringa
 - JSON.stringify()
- Da stringa a JSON
 - JSON.parse()

```
{
    name: "tom",
    job: {
        title: "developer",
        languages: ["JavaScript", "HTML", "CSS"]
    }
}
```

AJAX e XMLHttpRequest

- Asynchronous JavaScript And XML
- Uso dell'oggetto XMLHttpRequest per comunicare con il server (XML, JSON, testo semplice, ...) senza lasciare la pagina corrente
- Dopo aver creato un oggetto XMLHttpRequest
 - Si definisce una callback in onload (o onreadystatechange)
 - Si invoca open() per definire la risorsa richiesta sul server 📃
 - E infine send()

Esempio AJAX

```
<textarea id="target"></textarea>
<button onclick="getInfo():">Get programmer info</button>
function getInfo() {
                                                        function callback() {
   let request = new XMLHttpRequest();
                                                           let target = document.getElementById('target');
   request.onload = callback; 5
                                                           if (this.status != 200) {
   request.open("GET", "tom.json");
                                                              target.value += "[" + this.status + "]\n";
   request.send();
                                                              return:
                                                         let json = JSON.parse(tills.responseText);
   "name": "tom",
                                                           target.value += json.name + '\n';
   "job": {
                                                           target.value += json.job.title + '\n';
     "title": "developer",
                                                           target.value += json.job.languages + '\n';
     "languages": ["JavaScript", "HTML", "CSS"]
```

JQuery 🖪

- Libreria JavaScript progettata per semplificare la gestione del DOM
- Creata da John Resig nel 2006
- Download da https://jquery.com/download/ <script src="js/jquery-3.4.1.min.js"></script>
- Documentazione https://api.jquery.com/

Passo a ready 'function' che me la esegue quando la pagina (il DOM) viene caricata Ou con (1)

```
iQuery(document).ready(function() {
  // ...
});
$(document).ready(function() {
$(function() {
```

- Vogliamo eseguire funzioni appena il documento corrente è caricato dal browser
- Il metodo ready() di jQuery ha come parametro una funzione in cui possiamo mettere il nostro codice
- Il dollaro è l'alias standard per la funzione ¡Query()
- Forma abbreviata equivalente
- Alternative "pure JavaScript", via eventi load e DOMContentLoaded di window

```
window.addEventListener('DOMContentLoaded', (event) => {
```

Selezione di elementi

Wrap jQuery di elementi via selettore CSS

```
tag: $('textarea')
id: $('#myId')
classe: $('.myClass')
lista di selettori: $('div,span')
...
```

questi selettori ritornano un array ecco perchè possiamo usare length per vedere quanti elementi sono stati trovati

- Numero di elementi selezionati: length 🗔
 - Esempio: numero di div nella pagina: \$('div').length

Creazione di elementi

- Passando il relativo codice HTML si può creare un elemento, arricchirlo e inserirlo nel documento
- Esempio:
 - Crea un div contenente 'Hello'
 - Stilalo assegnando un colore al suo testo
 - Appendi l'elemento al body della pagina

```
$('<div>Helio</div>').css({colorered'}).appendTo('boo');
```

click e dblclick 🗸

Risposta a evento click e double click

```
// override del comportamento dei link in una pagina $('a').c k(function(e nt) { alert("You should not use any link on this page!"); event.preventDefault() non fare il default (non fa una GET) });
```

```
// double-click detector 
$('html').dblclick(function(e) {
    console.log('Double-click detected at ' + e.pageX +  + e.pageY + '\n');
});
```

L'attributo class

```
addClass()
    $('#msg1').addClass('red'); ma la classe red deve essere definita in CSS
removeClass()
    $('#msg1').removeClass('red');

    toggleClass()

    $('#msg2').toggleClass('red');
hasClass()
    $('#msg3').hasClass('red');
```

Getter e setter

- html() Mantiene la formattazione HTML mi permette di inserire elementi mantenendo la formattazione
- text() Testo puro
 \$('#signature').text('Hello by JQuery'); se ho un mi andrà a sostiuire il testo in p, se non metto niente legge quello che c'è
- val() Accesso al valore in input
 \$('#msg').val('Something');
- css()
- let cur = parseInt(\$('#msg').css('font-size')); \$('#msg').css('font-size', cur* 2);

Node JS

- Piattaforma per server app in JavaScript
 - Ben supportata da VS Code
- https://nodejs.org/en/download/ (LTS)
 - Verifica installazione (versione): node -v
- In una nuova directory
 - Crea: app.js
 - Esegui: node app.js

app.js

let message = 'hello';
console.log(message);

Node + Express

- https://nodejs.org/
- Da una nuova directory:
 - pnpm init

package.json

- npm install express --save
- crea il file index.js
- esegui l'app
 - node index.js
- Accedi all'app via browser, porta 3000

index.js

```
let express = require('express');
let app = express();
app.get('/', function (req, res) {
    res.send('Hello World');
});
app.listen(3000, function () {
    console.log('Listening on port 3000');
});
```