JavaScript

Per Sviluppatori Python

Per chi è pensato questo Corso Rapido su JavaScript?

- **Sviluppatori Python che vogliono acquisire familiarità con JavaScript**
- Sviluppatori Django/Flask che *devono* imparare JavaScript *rapidamente*
- Sviluppatori Web Python che vogliono apprendere le basi necessarie ad imparare i moderni Framework JavaScript (React, Angular, Vue.js ...)

Prerequisiti Fondamentali

- Buona conoscenza delle basi della Programmazione (con Python)
  Concetto di variabile, strutture di controllo e logica booleana, scope, tipi di dato, funzioni, classi...
- Conoscenza base di HTML e CSS
  Buona familiarità con la creazione e personalizzazione di pagine web
- Capacità di effettuare ricerche e approfondire in autonomia
  Link a pagine di approfondimento, guide complete in Italiano e codice del corso disponibili nelle slide finali
- Tanta voglia di imparare!

Link ai corsi completi sullo Sviluppo Web con Python, Django, Django REST Framework e Vue.js in descrizione

JavaScript

Per Sviluppatori Python

Argomenti Trattati

- Introduzione a JS lato client
- Differenze tra JS e Python
- Tipi di dato
- Sintassi (ES6)
- Variabili
- Funzioni e Classi
- DOM e Eventi
- Fetch API

Come seguire questo crash course?

Questo crash course è suddiviso in *due parti* principali.

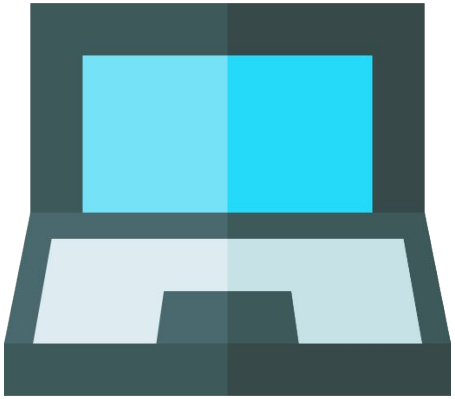
Nella prima parte del video useremo delle slide per evidenziare le differenze sintattiche e comportamentali più rilevanti tra JavaScript e Python, e nella seconda parte passeremo alla scrittura di codice, introducendo concetti nuovi e più avanzati.

Nella descrizione di questo video lascerò un link alla pagina del blog dove troverete ulteriori link a guide, materiale di studio ed esercizi di approfondimento per JS in italiano e codice che vi consiglio di visionare ed esplorare con calma ed attenzione.

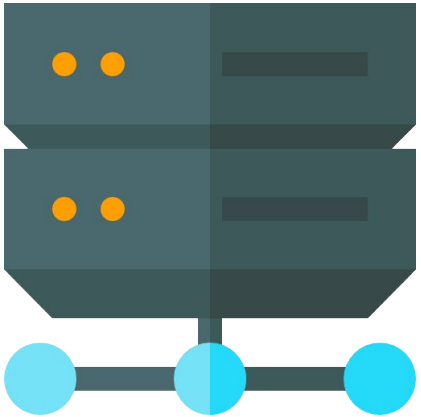
Potete quindi seguire e tenere questo video come guida introduttiva principale al linguaggio, ai suoi utilizzi nello sviluppo front-end e alle differenze con Python.

Volendo, potete scrivere il codice assieme a me o prendere appunti per massimizzare la comprensione dei contenuti!

JavaScript: Introduzione



Client



Server

Con **JavaScript** abbiamo la possibilità di far girare **codice lato client**, ottenendo vantaggi in termini di velocità e qualità dell'esperienza di navigazione per i nostri utenti. JavaScript viene usato anche in altri contesti, ma noi lo utilizzeremo principalmente all'interno dei **Browser** per ottenere effetti dinamici e per comunicare con le Web API.

Introduzione a JavaScript

Nel tempo sono uscite svariate versioni di JavaScript che hanno portato con sé aggiornamenti della sintassi e l'aggiunta di tante nuove funzionalità.

Queste diverse versioni seguono uno standard chiamato **ECMAScript**, di cui JavaScript è senza dubbio l'implementazione più famosa.

Noi utilizzeremo una delle versioni più popolari dello standard: **ECMAScript 2015**.

Conosciuta anche come ES2015 o **ES6**, è al momento una delle versioni più ampiamente supportate dalle varie versioni dei browser in circolazione, rendendola perfetta per noi.

P.S. JavaScript e Java sono due linguaggi distinti!

Tipi di Dato

Tipi di Dato in JavaScript

- **Numeri**
- **Stringhe**
- **Booleani**
- **Oggetti**
  - **Funzioni**
  - **Array**
  - **Date**
  - **RegEx**
- **Null**
- **Undefined**

```
> typeof 3
"number"
> typeof 3.0
"number"
> typeof "3"
"string"
> typeof "ciao"
"string"
> typeof [5, 2, 1]
"object"
```

```
> typeof {name:'Alice', age:25}
"object"
> typeof null
"object"
> typeof undefined
"undefined"
> typeof function miaFunzione(){}
"function"
> typeof true
"boolean"
```

*Con l'operatore **typeof** possiamo verificare il tipo di dato associato a un'espressione*
Il fatto che null venga considerato come object è una delle "stranezze" di JavaScript

Tipi di dato

Attenzione: combinare tipi di dato diversi tra di loro può produrre risultati inaspettati!

1 + 1 *da come risultato* **2**

1 + “1” *da come risultato* “**11”**

1 + “1” - “1” *da come risultato* **10**

1 + "1" - "1" + "1" *da come risultato* “**101”**

Alla seconda operazione, Python avrebbe invece restituito:
TypeError: unsupported operand type(s) for +: 'int' and 'str'

Sintassi:
differenze con Python

Operatori e Valori Booleani

- and
- or
- not

- &&
- ||
- !

- True
- False

- true
- false

A sinistra la sintassi Python, a destra la sintassi JavaScript.

Operatori di Comparazione

Per quanto riguarda gli operatori di comparazione ci sono alcune differenze sostanziali tra Python e JavaScript che vanno evidenziate.

In JavaScript esistono due operatori per l'uguaglianza: **==** e **===**

Il primo effettua comparazioni *abstract*, dove gli operandi vengono prima convertiti allo stesso tipo e poi viene effettuata la comparazione, per cui **1 == "1"** da risultato **true**.

Il secondo effettua comparazioni *strict*, dove true viene restituito solo se tipi e valori degli operandi combaciano, per cui **1 === "1"** da risultato **false**, **1 === 1** da risultato **true**

Lo stesso discorso può essere fatto per **!=** e **!==**

Operatori di Comparazione

Questo comportamento apparentemente *contro intuitivo* può essere notato *anche* con l'utilizzo degli altri operatori di comparazione.

In Python, provando ad effettuare 5 > "3" otteniamo invece il seguente errore:

TypeError: '>' not supported between instances of 'int' and 'str'

```
> 5 > "3"
< true
> 22 < "11"
< false
> 5 >= "4"
< true
> 5 == "5"
< true
> 5 === "5"
< false
> 5 != "5"
< false
> 5 !== "5"
< true
```

r/ProgrammerHumor

Variabili

Variabili

Da JavaScript ES6, le variabili possono essere dichiarate usando 3 parole chiave:

1. **var**: usata per la definizione di variabili *con* ***Function Scope*** *e* ***Global Scope***
2. **let**: usata per la definizione di variabili con ***Block Scope***
3. **const**: usata per la definizione di ***Costanti***

Prima di ES6, le variabili potevano essere dichiarate solo con var, il che comportava una serie di svantaggi ed errori potenziali.

Variabili - var keyword

```
var x = 1945;

// questo codice può usare x

console.log(x)

function miaFunzione() {

    // anche il codice qui può usare x

     console.log(x)

}
```

Le variabili dichiarate globalmente (fuori da una funzione) hanno Global Scope.

```
// il codice qui NON può usare pi

function miaFunzione() {

    var pi = 3.14;

    // il codice qui può usare pi

}

// il codice qui NON può usare pi
```

Le variabili dichiarate localmente (dentro a una funzione) hanno Function Scope.

Variabili - let keyword

Le variabili dichiarate con **let** possono avere **Block Scope**.
Variabili dichiarate con let all'interno di blocchi di codice delimitati da **{ }** non possono essere usate all'esterno del blocco.

```
// z NON può essere usata qui

{

    let z = "orange";

}

// z NON può essere usata qui
```

Variabili - let keyword

```
var x = 10;

// qua fuori a x corrisponde 10

{

    var x = 2;

    // qui dentro a x ora corrisponde 2

}

// anche qua fuori a x ora corrisponde 2!
```

```
var x = 10;

// qua fuori a x corrisponde 10

{

    let x = 2;

    // qui dentro a x ora corrisponde 2

}

// qua fuori a x corrisponde ancora 10 ;)
```

Grazie ad un buon impiego della parola chiave let possiamo mantenere un codice più ordinato ed aggirare fastidiosi problemi che si manifestano con l'uso di var

Variabili - const keyword

```
> const pi = 3.14
<· undefined
> pi
<· 3.14
> pi = 1936.27
✕ ▶ Uncaught TypeError: Assignment to constant variable.
```

Il valore di una variabile dichiarata con const ***non può*** essere mutato

Indentazione

Indentazione

In JavaScript l'indentazione *non* è obbligatoria.

È comunque sicuramente consigliabile mantenere uno stile uniforme e leggibile!

La "minimizzazione" del codice (eng. minify) è una pratica comune dove una volta terminato uno script, tutti gli spazi non necessari vengono rimossi per ridurre le dimensioni del file e velocizzare i tempi di caricamento delle pagine web.

Funzioni

Funzioni - Python e JavaScript a confronto

```python
def somma(a, b):
    return a + b
```

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

Un esempio di funzione per calcolare la somma tra due numeri a e b. A sinistra la sintassi Python, a destra la sintassi JavaScript.

JavaScript: Anonymous Functions e Arrow Functions

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

```
(a, b) => { return a + b };
(a, b) => a + b;
```

```
let somma = function(a, b) {
    return a + b;
}
```

```
let somma = (a, b) => { return a + b };
let somma = (a, b) => a + b;
```

Le funzioni senza un nome sono chiamate Anonymous Functions; Se assegnate a una variabile, possiamo usare la stessa variabile per richiamarle.
Gli esempi nella parte inferiore della slide sono esempi di “arrow functions”, nuovi modi di definire funzioni introdotti in ES6

Classi

Classi

```python
class Persona:

    def __init__(self, nome, cognome):
        self.nome = nome
        self.cognome = cognome

    def profilo(self):
        print(f"Profilo: {self.nome} {self.cognome}")



p = Persona("Marco", "Aurelio")
p.profilo()
```

```javascript
class Persona {

    constructor(nome, cognome) {
        this.nome = nome;
        this.cognome = cognome;
    }

    profilo() {
        console.log(`Profilo: ${this.nome} ${this.cognome}`)
    }
}

p = new Persona("Marco", "Aurelio");
p.profilo()
```

Classi in Python e JavaScript a confronto. A sinistra la sintassi di Python 3, a destra quella di JavaScript ES6.

Ereditarietà

```python
class Studente(Persona):

    # resto del codice
```

```javascript
class Studente extends Persona {

    // resto del codice

}
```

Esempi di ereditarietà. A Sinistra la sintassi Python 3, a destra quella di JavaScript ES6

Strutture di Controllo

Strutture di Controllo

```python
età = 26
patente = True

if età >= 18 and patente == True:
    print('Puoi noleggiare una Ducati!')
elif età >= 18 and patente == False:
    print('Niente patente, niente Ducati!')
else:
    print('Per ora niente moto!')
```

```javascript
var età = 26;
var patente = true;

if (età >= 18 && patente === true) {
    console.log('Puoi noleggiare una Ducati!');
} else if (età >= 18 && patente === false) {
    console.log('Niente patente, niente Ducati!');
} else {
    console.log('Per ora niente moto!');
}
```

Istruzioni di controllo. A sinistra la sintassi Python, a destra quella di JavaScript

Strutture di Controllo

```python
contatore = 0

while contatore < 10:
    print(contatore)
    contatore += 1
```

```javascript
var contatore = 0;

while (contatore < 10) {
    console.log(contatore);
    contatore += 1;
}
```

Cicli while a confronto. A sinistra la sintassi Python, a destra quella di JavaScript

Strutture di Controllo

```python
for numero in range(11):
    print(numero)
```

```
0
1
2
...
...
10
```

```javascript
var i;

for (i = 0; i <= 10; i++) {
    console.log(i);
}
```

```
0
1
2
...
...
10
```

Cicli for a confronto. A sinistra la sintassi Python, a destra quella di JavaScript

Ora che abbiamo appreso le basi della sintassi possiamo iniziare con la scrittura del codice!

Vi ricordo che in queste slide troverete i link al codice utilizzato e a guide utili ed esercizi di approfondimento.

Link, Guide, Corsi e Referenze

JavaScript per Sviluppatori Python- Link Utili

Repository Codice Usato: https://github.com/pymike00/javascript-per-sviluppatori-python-code

Wikipedia: ***https://it.wikipedia.org/wiki/JavaScript***

Guida per Principianti - MDN: ***https://developer.mozilla.org/it/docs/Web/JavaScript***

Reintroduzione a JavaScript - MDN: ***https://developer.mozilla.org/it/docs/Web/JavaScript/Una_reintroduzione_al_JavaScript***

ES6 New Features - Babel: ***https://babeljs.io/docs/en/learn#ecmascript-2015-features***

ES6 New Features - Gaearon: ***https://gist.github.com/gaearon/683e676101005de0add59e8bb345340c***

Variabili - Let Keyword: ***https://www.w3schools.com/js/js_let.asp***

DOM: ***https://www.w3schools.com/js/js_htmldom.asp***

JavaScript HTML DOM Events: ***https://www.w3schools.com/js/js_htmldom_events.asp***

JavaScript per Sviluppatori Python- Link Utili

querySelector: ***https://www.w3schools.com/jsref/met_document_queryselector.asp***

Fetch API: ***https://developer.mozilla.org/it/docs/Web/API/Fetch_API***

Fetch API Support: ***https://caniuse.com/#feat=fetch***

Using Fetch: ***https://developer.mozilla.org/it/docs/Web/API/Fetch_API/Using_Fetch***

Promises: ***https://developer.mozilla.org/it/docs/Web/JavaScript/Reference/Global_Objects/Promise***

Esercizi JavaScript: ***https://github.com/AlbertoOlla/esercizi-di-programmazione-javascript***

Vue.JS: ***https://vuejs.org/***

I Nostri Corsi Completi su Udemy

Guida Completa a Django 2, Python 3 e Bootstrap 4

In questo corso imparerai tutte le basi della programmazione e del Web Development con Python, e diventerai abile nell'utilizzo di un framework per lo sviluppo web tra i più potenti e avanzati, Django, usato anche da Instagram e Mozilla.
Scopri di più => https://www.programmareinpython.it/video-corso-python-django-bootstrap-completo/

Guida Per Sviluppatori a Django REST Framework e Vue.JS

Impara come creare potenti Web API con Python, Django e Django REST Framework, e usale in combinazione col framework frontend Vue.JS per la creazione di moderne Single Page Applications!
Scopri di più => https://www.programmareinpython.it/blog/nuovo-corso-guida-sviluppatori-django-rest-framewo/

JavaScript

Per Sviluppatori Python