Edgemony

Coding Bootcamp

Funzioni complesse

Funzioni freccia, funzioni che tornano indietro, funzioni al sugo…

*Una funzione è come una skill, un'**abilità**, che permette di **fare ogni cosa**… anche di lanciare incantesimi*

Funzioni complesse

Funzioni freccia, funzioni che tornano indietro, funzioni al sugo…

Lezione dei anatomia

Dichiarazione

Keyword Name Parameters

```
function makePizza(flavor) {
    // … code …
    return pizza;
}
```

Return Keywork

Invocazione

```
function makePizza(margherita);
```

Il ciclo di vita di un funzione:

1. **Dichiarazione -> si definisce nome, parametri e contenuto**
2. **Invocazione -> la si richiama tramite il nome e si inseriscono i parametri**

Arrow function

Argomento di ES6

Una **arrow function** non è altro che una funzione definita in modo più ‘contenuto’.
Per esempio potremmo scrivere:

```
Function sum(numOne, numTwo) {
   return numOne + numTwo;
}
```

In una singola riga, così:

```
const sum = (numOne, numTwo) => numOne + numTwo
```

Spread syntax

Argomento di ES6

È possibile **inserire** gli elementi di un array (ma anche di un oggetto) come **parametri**.

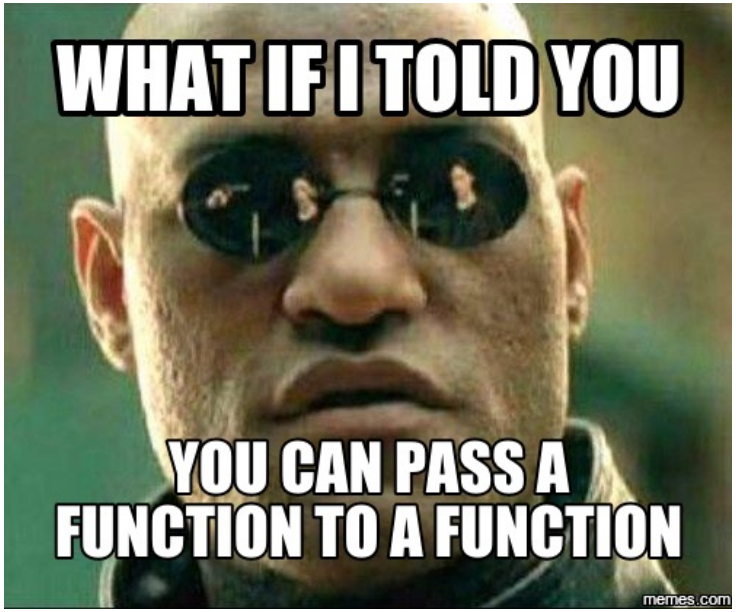
```
const evenNumbers = [2,4,6,8];
Function sum([ ...evenNumbers ]) {
   // ... code ...
}
```

Strumento potentissimo con i metodi, come per es. ***math.min(...evenNumbers)***

A proposito... Cosa è un metodo?

Funzioni complesse

Funzioni freccia, funzioni che tornano indietro, funzioni al sugo…



Callback

È possibile anche **inserire** come **parametro** un'altra funzione

```
const sum = (numOne, numTwo) => numOne + numTwo;

const yourSum = (yourName, yourFunc) => {
   return (yourName + ` ` + yourFunc);
}

console.log(yourSum(`Casi`, sum(7, 9)));
```

Funzioni pure, IIFE e altri concetti audaci

Le funzioni pure stanno al **cuore** del concetto di **programmazione funzionale.**
Una funzione per essere definita pura deve soddisfare questi **punti**:

1. In relazione ad **un dato input** produrrà sempre **lo stesso output**
2. Non produce **effetti collaterali** (come modifiche allo scope esterno ad essa, chiamate HTTP o interazione col DOM)

Piccola nota: l'apprendimento non è mai un processo lineare, ci saranno giorni in cui sembrerà non aver appreso nulla e altri in cui si realizza di aver appreso tutto. In entrambi i casi vi sbaglierete!

L'apprendimento è un processo lento e costante, poiché necessario dare il tempo adeguato alla nostra testa di fare amicizia con ciò che ci mettiamo dentro. Nessuna premura dunque, solo tanta sete!