Edgemony

Coding Bootcamp

Scope

Gestire l'accessibilità delle variabili

*Lo **scope** è quel luogo entro il quale le **variabili** sono 'visibili' e possono essere 'richiamate'*

Global & Local scope

Fondamentalmente ci sono due tipi di scope:

GLOBAL => riguarda l'intero file `script.js` (la Terra intera)

LOCAL => riguarda invece blocchi ben definiti (gli stati)

Una simpatica analogia: immaginiamo la Terra (global scope), come l'insieme di tutte le nazioni del mondo. Dove avremo popolazioni che parlano lingue diverse. Data la vastità del pianeta, troveremo certamente qualcuno che parli l'Inglese o il Coreano, per es.
Se invece ci focalizziamo su una precisa nazione (local scope), la situazione certamente cambia. Immaginate di andare in pizzeria e ordinare una bella pizza parlando il Coreano!

Lo scope e le funzioni

Lo scope riguarda anche (soprattutto!) le funzioni.

- La variabile *pizza* è una variabile globale
- La variabile *ìngredienteBase* è una variabile locale

Perché questo codice ci solleva un errore? Eppure sembra scritto correttamente…

```
> const pizza = 'margherita'

  function faiPizza(gusto) {
     const ingredienteBase = 'farina'
     // ... tanto altro codice qui ...
     // ... ma alla fine ...
     return pizza
  }

  console.log(pizza)
  console.log(ingredienteBase)
  margherita
  ▸ Uncaught ReferenceError: ingredienteBase is not defined
      at <anonymous>:11:13
```

ERRORE! Siamo fuori scope!!!

Lo scope e le variabili

Javascript permette di utilizzare ben 3 tipi diversi di variabili:

- ***var*** *– deprecated – scope GLOBALE*
- ***let*** *– mutabili – scope LOCALE*
- ***const*** *– immutabili – scope LOCALE*

Let e const sono concetti introdotti in **ES6** (ECMAScript 6), seppur nei giorni successivi vedremo altri dettagli su questo argomento, consiglio una bella ricerca su Google fin da adesso!

Piccola nota: è consigliatissimo evitare l'utilizzo di `var`, in quanto potrebbe generare dei conflitti nel codice più questo va crescendo di complessità.
Inoltre, se la scelta permette, preferire `const` a `let`.

Ultime considerazioni, accenni e recap

- *Global scope*
- *Local scope*
- *Hoisting*
- *Block scope – { … }*
- *Variable scope*
- *Function scope*

Piccola nota: questo è uno degli argomenti più teorici (che pratici) di Javascript. Non abbiate timore di cercare sulla rete. Comunque, con l'esperienza, e scrivendo sempre più codice, tanto di quello che oggi vi sembra incomprensibile diventerà ovvio ;)

E ricorda che se usi 'var' qualcuno potrebbe offendersi…