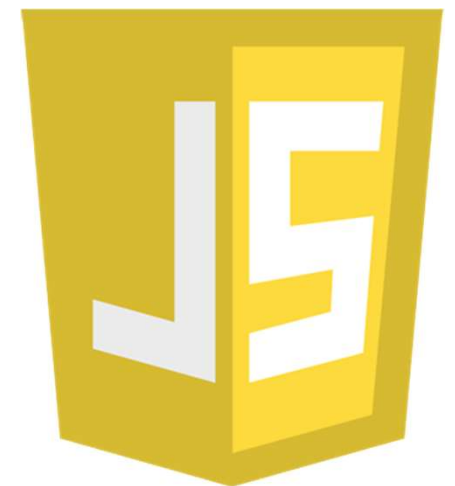


JAVASCRIPT

Riccardo Cattaneo

Lezione 5



Le Funzioni

E' un blocco di codice nel quale sono contenute delle istruzioni. La sintassi è molto semplice :

```
<script>  
    function nomeFunzione(){  
        ... istruzioni ...  
    }  
</script>
```

- Una funzione può ricevere dei parametri e ritornare un valore (se non torna nulla... torna undefined)
- Le funzioni ci aiutano a non ripetere blocchi di codice ma organizzarli in modo funzionale.

Facciamo subito un esempio e andiamo a scrivere una funzione che fa la somma di due numeri :

```
<script>  
    function somma(num1,num2){  
        document.write(num1 + num2);  
    }  
</script>
```

Se provo ad eseguire la pagina non visualizzo nulla... perché? Perché nessuno ha chiamato la funzione... come faccio a richiamare una funzione? Attraverso il suo nome :

```
<script>  
    function somma(num1,num2){  
        document.write(num1 + num2);  
    }  
</script>
```

```
somma(5,6);  
somma(3,44);
```

Come possiamo notare il vantaggio di una funzione è che può essere utilizzata più volte all'interno di una pagina.

Esempio di ritorno

```
function testProva(){  
  console.log('corso js');  
  return 'benvenuto';  
}  
  
testProva();    // corso js  
console.log(testProva()); // corso js benvenuto
```

E' anche possibile memorizzare direttamente il risultato, o meglio il «ritorno» della funzione dentro una variabile in questo modo :

```
function testProva(){  
    console.log('corso js');  
    return 'benvenuto';  
}  
  
var risultato = testProva(); // corso js  
console.log(risultato); // benvenuto
```

Facciamo un altro esempio :

```
<script>  
    function saluto(nome1,nome2,nome3){  
        document.write("Benvenuto" + nome1);  
        document.write("Benvenuto" + nome2);  
        document.write("Benvenuto" + nome3);  
    }  
</script>
```

```
saluto("Mario","Luca","Giovanna"); // cosa succede ?  
saluto("Mario","Luca"); // cosa succede ?
```


arguments

L'oggetto **arguments** è una variabile locale disponibile in tutte le funzioni. È possibile fare riferimento agli argomenti di una funzione all'interno della funzione utilizzando l'oggetto arguments.

Questo oggetto contiene una voce per ogni argomento passato a una funzione, che ci aiuterà a conoscere il numero di questi argomenti e ad accedervi.

Come abbiamo detto, l'oggetto arguments è un oggetto (array), ed ha una proprietà **length**. Quindi possiamo accedere ai valori individuali usando la notazione di indicizzazione dell'array arguments[i].

```
function myFunction(a, b, c){  
  console.log(arguments[0]);  
  console.log(arguments[1]);  
  console.log(arguments[2]);  
}  
myFunction(1,2,3);
```

Variabili locali e globali

Possiamo dichiarare le variabili all'interno del corpo di una funzione. Queste variabili sono accessibili soltanto all'interno della funzione e non vengono viste fuori di essa o, in termini tecnici, hanno uno **scope locale**.

Lo scope o ambito di visibilità di una variabile è la parte di uno script all'interno del quale si può fare riferimento ad essa. Le variabili dichiarate all'interno di una funzione sono dette locali alla funzione dal momento che sono accessibili soltanto all'interno del suo corpo.

Le variabili dichiarate fuori da qualsiasi funzione sono dette globali e sono accessibili da qualsiasi punto dello script, anche all'interno di funzioni.

Espressione di Funzione

Un modo alternativo per scrivere una funzione è attraverso la tecnica di «espressione di funzione» cioè memorizzare una funzione all'interno di una variabile in questo modo :

```
function test1 (){  
    
}
```

```
var test1 = function (){  
    
};
```

Prima cosa da notare è che abbiamo messo il punto e virgola alla fine, in quanto è una assegnazione (`nomeVariabile = valore`) con l'unica differenza che il valore è una funzione.

Per richiamare la funzione è sufficiente usare il nome della variabile seguito dalle parentesi tonde (quindi nello stesso modo della dichiarazione di una funzione normale). La domanda quindi è... effettivamente cosa cambia tra le due modalità ?

La differenza sta nel fatto che se la dichiaro come variabile non posso richiamarla ovunque, ma solo dopo averla dichiarata e valorizzata, mentre se la dichiaro nel modo «classico» posso chiamarla ovunque anche prima di dichiararla.

Questo perché javascript all'avvio fa una cosa «strana» (tanto per cambiare) chiamata «**HOIST**» : cioè prende tutte le funzioni e le mette in testa al codice in modo da poterle richiamare ovunque.

```
test1();    // ERRORE
test2();    // OK

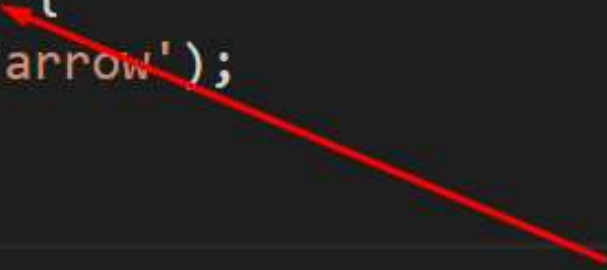
var test1 = function (){
  console.log('espressione di funzione');
};

function test2(){
  console.log('funzione classica');
}
```

Arrow function

Dalla versione 6 di javascript sono state introdotte le cosiddette arrow function. La sintassi delle arrow function è la seguente :

```
var test3 = () => {  
  console.log('arrow');  
}  
  
test3();
```



Le parentesi tonde () stanno ad indicare che posso passare opzionalmente anche dei parametri alla arrow function. Ad esempio :

```
var test3 = (nome, cognome) => {  
  console.log('arrow');  
  console.log(nome + ' ' + cognome);  
}  
  
test3('Riccardo', 'Cattaneo');
```

Parametri di default

Prima di ES6 non era possibile dare dei valori di default ai parametri di una funzione. Quindi se avevamo una funzione con dei parametri in ingresso dovevamo controllarli con una if ed eventualmente se vuoti assegnargli manualmente dei valori in questo modo :

```
function prova(a,b){  
  a = a || 0;  
  b = b || 0;  
  console.log(a);  
  console.log(b);  
}
```

```
prova(); // 0 0
```

Invece con ES6 è possibile dare dei valori di default direttamente in ingresso alla funzione in questo modo :

```
function prova(a = 0, b = 0){  
  console.log(a);  
  console.log(b);  
}  
  
prova(); // 0 0
```

```
<script>
    function saluta(){
        var messaggio = "Buongiorno";
        document.write(messaggio);
    }
</script>
```

```
// saluta();
window.alert(messaggio); // NON FUNZIONA
```

Proviamo a rifare lo stesso esempio dichiarando la variabile al di fuori della funzione e vediamo cosa succede...

Esercizio 5.1

Scrivi una funzione di uguaglianza che prenda in input due argomenti e restituisca TRUE se i due argomenti sono IDENTICI, altrimenti FALSE.

Esercizio 5.2

Scrivi un programma con due funzioni, la prima prende in ingresso un intero e verifica se sia compreso nel range da 1 a 7. Se è compreso, la funzione restituirà TRUE, se non è compreso restituirà FALSE.

La seconda funzione:

- nel caso la prima restituisca TRUE visualizza il giorno della settimana corrispondente, considerando la seguente corrispondenza:

 - 1 = Lunedì

 - ...

 - 7 = Domenica

- nel caso il giorno non sia compreso nel range, la funzione dovrà restituire un messaggio d'errore simile a questo: 'Peccato, non posso indovinare il giorno.'

Esercizio 5.3

ESERCIZIO TRE: INDOVINA IL NUMERO

Creare una pagina per il gioco di Indovina un numero:

Gioco: indovina il numero!

input text
id="tentativo" →

Controlla

Tentativi: ← **input text**
id="conteggio"

Nuovo Gioco

All'avvio il programma genera un numero casuale compreso fra 0 e 100. Il giocatore deve indovinarlo scrivendo il proprio tentativo e premendo il tasto Controlla. Il programma dice al giocatore se il tentativo fatto è più piccolo o più grande del numero da indovinare.

Esercizio 5.4

Scrivere un pagina che tramite una funzione javascript realizza queste linee tenendo presente che hanno una lunghezza che parte da 0% fino al 100% con passo 5. (utilizzare il tag html `<hr>` per fare le righe) ed utilizzare il dom per modificare lo stile.



Esercizio 5.5

Scrivere un pagina che tramite una funzione javascript stampa a video 10 asterischi, poi 11 poi 12 ... fino a 30 (utilizzare il ciclo for)

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Esercizio 5.6

Calcolatrice : Scrivere una pagina html con 2 input di tipo number e far scegliere tramite una select il tipo di operazione (+ - / *) ed un bottone calcola che, in base all'opzione scelta chiama una funzione javascript che calcola e visualizza a video il risultato.