Creative Web Development

Guarise, Degl'Innocenti, Rossi - 2018

Lezione 4

A cura di: **Prof. Degl'Innocenti**

Teoria

I have not failed 1,000 times. I have successfully discovered 1,000 ways to NOT make a light bulb.

Nello sviluppo software ci si trova spesso a fronteggiare ostacoli e percorsi ardui e alle volte ripetitivi per riuscire ad arrivare all'obiettivo: codice funzionante e che assolve allo scopo preposto.

Prima regola: continuare a provare!

Ricerca e sviluppo: le giuste risorse

Nel mondo dell'informatica e della programmazione, o comunque dello sviluppo software è importante:

- Conoscere gli strumenti e le procedure giuste
- Saper risolvere le problematiche quando si presentano
- Conoscere i giusti punti di riferimento
  - Corsi specifici
  - Documentazione online
  - Libri e testi
  - Community
- Poter affidarsi ad informazioni certe e condivise
- Mantenere alta la motivazione anche di fronte a difficoltà

Il Web per aumentare la conoscenza

- Una persona di riferimento già skillata che trasmetta la propria conoscenza è uno dei modi migliori di formarsi
  - Crea un percorso su misura
  - Indirizza sulle giuste tematiche
  - Risponde a domande specifiche o anche su altri argomenti correlati
  - Permette di espandere il proprio “campo visivo”
- In momenti di necessità è possibile fruire di ulteriori strumenti
  - Sempre disponibili
  - Gratuiti
  - Meno dedicati e meno chiari
  - Di veloce fruizione

La documentazione Online

- Relativa ad un linguaggio, ambiente, framework **specifico**
- Vasta quantità di informazioni nelle sue varie sfaccettature
- Utile per esplorazione od approfondimento
- Spesso creata da azienda od ente ufficiale di riferimento

- JS by W3C: https://www.w3schools.com/js/default.asp
- JS by MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- Angular2: https://angular.io/docs
- Ionic: https://ionicframework.com/docs/
- Node: https://nodejs.org/it/docs/

Ricerce efficaci con Google

- Cercare in **inglese**
- Utlizzare in testa le keyword del linguaggio e successivamente le richieste del problema
  - Es. “**js** for loop”
  - Es. “**html** input form”
  - Es. “**js** string object methods”
  - Es. “**npm** modal support”
- Ricerca compatta (max 5-6 parole)
- Evitare di usare stop words (a, the, of, at, and, or, ...)
- Provare i primi due o tre risultati prima di modificare la ricerca
- E’ possibile trovare un valido aiuto nei momenti di difficoltà in solitaria

Ricerce con Google: risoluzione errori

- Capita a volte di incorrere in **errori lanciati dal proprio programma sconosciuti** o di cui non siamo consapevoli fino in fondo
- Possiamo cercare l'errore per trovare **qualcuno che lo ha già risolto per noi**
- Ridurre il messaggio di errore ai **minimi termini e eliminare i nomi** da noi assegnati specifici del nostro software
    - es. *Uncaught ReferenceError: user is not defined*
      at <anonymous>:1:13
      diventa nella nostra ricerca ➜ "*ReferenceError: is not defined*"
    - *Uncaught TypeError: document.wite is not a function*
      at <anonymous>:1:10
      diventa nella nostra ricerca ➜ "*TypeError: is not a function*"

- Meglio che passare ore a fare tentativi quando non abbiamo alcun supporto!

Come leggere la documentazione

- Se arriviamo da una ricerca, possiamo leggere la parte che ci interessa nello specifico per risolvere il problema attuale
- Se ci stiamo formando o approfondendo, conviene **dare un'occhiata all'indice** e farsi un idea di **quali parti ci interessano**
- Non leggere solo il pezzettino che ci interessa ma **cercare di capire tutto il contesto** anche leggendo tutto il capitolo o la pagina, nel lungo termine ci si guadagna in knowledge!!!
- Leggere gli esempi di codice e se possibile provarli sul proprio PC

I network di supporto

- Community di persone o di professionisti che in cambio di visibilità, per lavoro o per passione aiutano altri diffondendo le proprie conoscenze
- Di vari tipi:
  - Generali (multi-argomento) o project/product-oriented a seconda dei casi
  - Gratuiti, o a pagamento relativamente ad un prodotto
- Attenzione che non sempre la soluzione proposta è quella giusta, guardare sempre la reazione della community in merito
- StackOverflow: https://stackoverflow.com/ - Community open in merito al Web Development
- Forum dedicati per prodotto
- GitHub: issues report e wiki - https://github.com/

Pratica

Programma Pratica

- Array
  - Sintassi
  - Lettura
  - Scrittura
  - Lunghezza
  - Push
  - Pop
  - indexOf
- Oggetti
  - Sintassi
  - Proprietà
    - Lettura
    - Scrittura
  - Metodi
  - this
- Esercizi
- Soluzione esercizi precedenti

Array

Un array è una lista di elementi ordinati all'interno di una variabile.
E' possibile accedere agli elementi dell'array tramite il loro indice di riferimento.

L'array si dichiara utilizzando le parentesi quadre [ ] e separando gli elementi con virgola.

```
var array = [ 1, 7, 2, 4, 10];
var array2 = ["italia","spagna","germania","francia"];
var array3 = [12.4, 23.5, 7.9];
var array4 = [ ];
var array5 = [ null, null, undefined ];
var array6 = [37, "svizzera", true, null];
```

Array

Gli **elementi interni all'array** sono sempre **ordinati**, ed ad ogni elemento è associato un **numero indice**, significante la sua **posizione all'interno dello stesso**. L'indice parte sempre **da 0**, e arriva **fino ad `n-1`**, con `n = dimensione dell'array`.


```
var array = ["italia","spagna","germania","francia"];
```

Array: lettura

Per accedere ad uno specifico elemento dell'array è necessario conoscere il suo indice di riferimento...

```
var array = ["italia","spagna","germania","francia"];

console.log( array[0] ); //stampa italia
console.log( array[2] ); //stampa germania
```

```
<nome array>[<indice>]
```

Array: lettura in ciclo iterativo

...oppure eseguire un ciclo iterativo su tutti gli elementi al fine di trovarlo

```
var array = ["italia","spagna","germania","francia"];

var elementToFind = “francia”;
for(var index in array){
    if( array[index] === elementToFind ){
        console.log("L’indice dell’elemento è:",index);
    }
}
```

Array: scrittura di un elemento

Per scrivere un elemento in un array, è necessario specificare la posizione dell'indice in cui scrivere quell'elemento.

```
var myArray = [ 1, 2, 3, 10 ];
console.log( myArray ); // stampa [1, 2, 3, 10]
myArray[2] = 17;
console.log( myArray ); // stampa [1, 2, 17, 10]
```

Array: l'oggetto Array

L'array in Javascript, è in realtà rappresentato dall'oggetto nativo **Array**

```
var myArray = new Array();
```

Supporta la sua dichiarazione e allocazioni in due modi diversi:

Un solo parametro → numero di elementi *(inizialmente vuoti)*

```
var myArray = new Array(3);
```

→ *crea un array di 3 elementi tutti undefined, equivale a:*

```
var myArray = [undefined, undefined, undefined];
```

Più parametri → gli elementi interni

```
var myArray = new Array(3,7,9,12);
```

→ *crea un array contenente i valori specificati, equivale a:*

```
var myArray = [3,7,9,12];
```

Array: numero degli elementi

Per conoscere il numero totale degli elementi, o dimensione dell'array, o lunghezza dell'array, si utilizza la proprietà **`lenght`** dell'oggetto array.

```
var myArray = new Array(3, 7, 9, 12);
console.log( myArray.length ); //stampa 4

var myArray2 = new Array(3, 7, 9, 12, 43, 67, 48);
console.log( myArray2.length ); //stampa 7
```

Array: push(...)

Il metodo `push` dell'oggetto Array ci consente di a**ggiungere un elemento alla fine dell'array** (e restituisce la nuova lunghezza)

```
var myArray = [ "ciao", "hello", "hola" ];
var newLenght = myArray.push("salut");
console.log( myArray ); //stampa [ "ciao", "hello", "hola", "salut"];
console.log( newLenght ); //stampa 4
```

Array: pop()

Il metodo `pop` dell'oggetto Array ci consente di **rimuove l'elemento alla fine dell'array** (e restituisce l'elemento rimosso)

```
var myArray = [ "ciao", "hello", "hola", "hallo" ];
var deletedElement = myArray.pop();
console.log( deletedElement ); //stampa hallo
console.log( myArray ); //stampa [ "ciao", "hello", "hola"];
```

Array: indexOf(...)

Il metodo indexOf dell'oggetto array, ci consente di conoscere l'**indice della posizione di un determinato elemento specificato** all'interno dell'array.

```
var myArray = [ "ciao", "hello", "hola", "hallo" ];
var position = myArray.indexOf("hello");
console.log(position); //stampa 1
console.log( myArray[position] ); //stampa hello
```

Esercitazione

1. Creare un array di 4 numeri a piacere e stampare il secondo numero
2. Creare un array di 5 numeri a piacere e stamparlo in un ciclo FOR o FOREACH
3. Creare un array di 4 numeri a piacere e stamparlo come sopra, sostituire il terzo elemento e stamparlo di nuovo
4. Creare un vettore di 4 elementi a piacere con il costruttore dell'oggetto Array e stamparlo
5. Creare un vettore di 6 elementi a piacere e stampare la dimensione
6. Creare un vettore vuoto, leggere con PROMPT 6 elementi e inserirli con PUSH, e stamparlo
7. Creare un vettore di 5 elementi a piacere e stamparli tutti usando POP
8. Creare un vettore con [1 , 5, 7, 9, 15] e stampare l'indice del valore 7 con INDEXOF

Oggetti (mappe o array associativi)

In JS, una **mappa, hash** o **array associativo**, o più semplicemente **oggetto**, è una specie di array ma che utilizza **indici con un nome**, invece che con un numero.

JS non supporta realmente array associati, ma li riconduce al tipo oggetto.

Su questo tipo di oggetto non funzionano i normali metodi array.

```
var person = [];
person["firstName"] = "Mario";
person["lastName"] = "Rossini";
person["age"] = 42;
```

Oggetti

L’esempio precedente è equivalente a dichiarare un oggetto con le normali parentesi graffe { }

```
var person = {
        "firstName": "Mario",
        "lastName":   "Rossini",
        "age":        42
};
```

Oggetti

```
var <nome oggetto> = {
        <nome proprietà>: <valore>,
        <nome proprietà 2>: <valore 2>,
        <nome proprietà 3>: <valore 3>,
        ...
};
```

Oggetti: nome delle proprietà

Se il nome della proprietà contiene solo lettere, è possibile scriverlo senza apici:

```
var person = {
        firstName:    "Mario",
        lastName:     "Rossini"
};
```

Altrimenti è necessario utilizzare gli apici:

```
var person = {
        "first-name": "Mario",
        "last name":   "Rossini"
};
```

Oggetti: accedere a proprietà

Per accedere alle proprietà, è possibile fare in due modi (sia in lettura che scrittura):

```
var person = { firstName: "Mario", lastName: "Rossini" };
```

Come per gli array, utilizzando **stringhe in parentesi quadre**.

```
var nome = person["firstName"];

person["firstName"] = "Fabio";
```

Utilizzando la **dot-notation**, cara alla programmazione ad oggetti.

```
var nome = person.firstName;

person.firstName = "Fabio";
```

Oggetti: funzioni → metodi

E’ possibile aggiungere delle funzioni all’interno di un oggetto (ma anche di un array), la funzione all’interno dell’oggetto in questo caso di chiama **metodo**.+

```
function getFullName() {
    return "Mario Rossini";
}
```

```
person.getFullName = getFullName;
```

e poi richiamarla come una normale funzione utilizzando le parentesi tonde **( )** con eventuali parametri

```
person.getFullName(); // ritorna Mario Rossini
```

Oggetti: metodi

E’ possibile assegnare un metodo ad un oggetto anche assegnandolo direttamente o in fase di dichiarazione direttamente nel corpo dell’oggetto.

```
var person = {};
person.getFullName = function () { return "Mario Rossini"; }
```

oppure

```
var person = {
    firstName: "Mario",
    lastName: "Rossini",
    getFullName: function () { return "Mario Rossini"; }
};
```

Oggetti: il riferimento this

Con la keyword `this` è possibile per un metodo all'interno di un oggetto, riferirsi ad altri metodi o proprietà all'interno dell'oggetto stesso. Questo è utile spesso per maneggiare dati ed eseguire operazioni di vario tipo.

```
var person = {
    name: "",
    lastName: "",
    getFullName: function(){
        return this.name + " " + this.lastName;
    }
}
```

Esercitazione

1. Dichiarare e assegnare un oggetto di tipo “animale” con delle proprietà specifiche a piacere (es. colore, n° zampe, famiglia animale, habitat, ...)
2. Stampare singolarmente tutte le proprietà dell’oggetto sopra
3. Leggere da prompt un valore relativo ad una delle caratteristiche e sostituirla nell’oggetto
4. Creare un metodo che stampi il verso dell’animale
5. Creare un metodo che stampi almeno due proprietà in una stringa concatenata

Esercizi per casa

1. Creare un array vuoto, riempirlo con prompt e push in un ciclo for, e stampare tutti gli elementi dal fondo usando POP
2. Creare un oggetto “automobile” con 4 caratteristiche tipiche come proprietà di cui una “maxSpeed”, e un metodo che letto un numero in ingresso, se minore di “maxSpeed” stampi “ok”, altrimenti “too fast!!!”, ed utilizzarlo

Soluzioni Esercizi Lezione 3

1)

```
var input = prompt("Inserisci un numero:");
var number = parseInt(input);
```

2)

```
var number1 = parseInt(prompt("Inserisci 1° numero:"));
var number2 = parseInt(prompt("Inserisci 2° numero:"));
console.log(number1+number2);
console.log(number1-number2);
console.log(number1*number2);
console.log(number1/number2);
```

Soluzioni Esercizi Lezione 3

3)

```
var number1 = parseInt(prompt("Inserisci 1° numero:"));
var number2 = parseInt(prompt("Inserisci 2° numero:"));
if(number1>number2){
    console.log("Il primo numero è maggiore del secondo.");
}
if(number1<=number2){
    console.log("Il primo numero è minore-uguale del sec.");
}
if(number1!==number2){
    console.log("I due numeri sono diversi tra di loro.");
}
```

Soluzioni Esercizi Lezione 3

4)

```
var boolean1 = true;
var boolean2 = false;
if(boolean1 === true && boolean2 === true){
    console.log("Entrambi veri");
}else if(boolean1 === true || boolean2 === true){
    console.log("Uno dei due vero");
}else if(boolean1 === false && boolean2 === false){
    console.log("Entrambi falsi");
}
```

Soluzioni Esercizi Lezione 3

5)

```
var string1 = prompt("Scrivi una stringa:");
var string1 = prompt("Scrivi un'altra stringa:");

console.log(string1 + string2);
```