SEMINARIO JAVASCRIPT: INTRODUZIONE AL LINGUAGGIO

Linguaggi per il Web

Ingegneria Informatica, Ingegneria dell'Informazione,
Sapienza Università di Roma, sede di Latina

02 maggio 2024

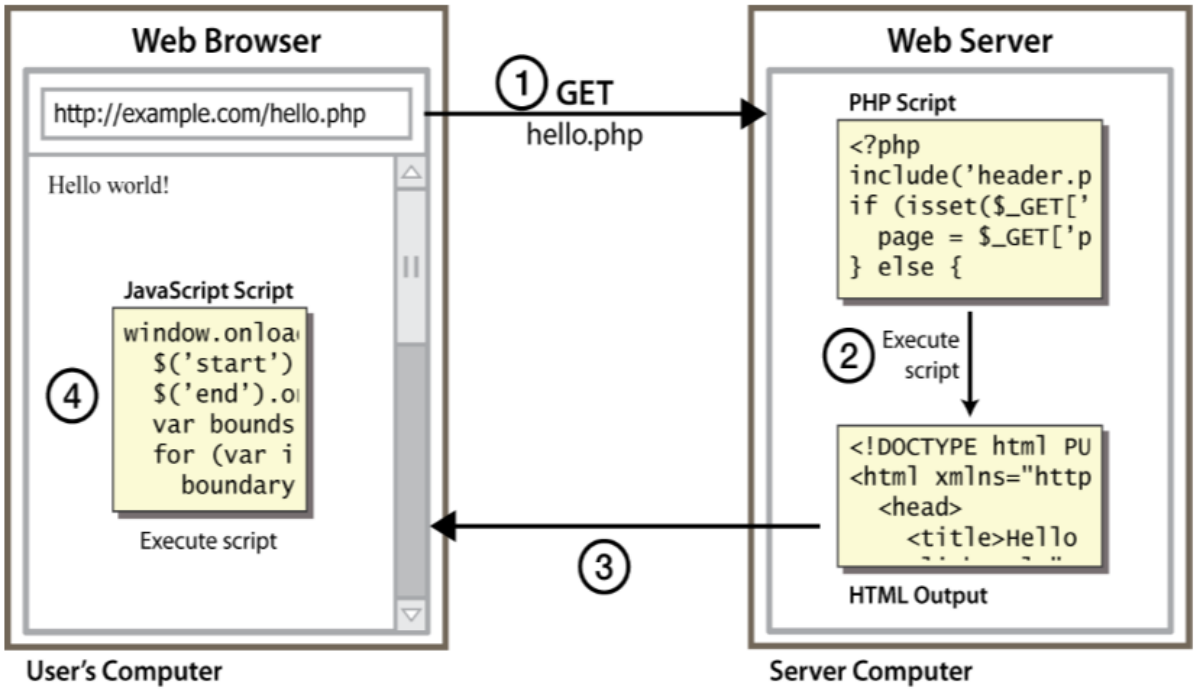
Corrado Di Benedetto

ARGOMENTI

- Introduzione
- Linguaggio
- Browser Object Model (BOM)
- Document Object Model (DOM)
- Esercizi

PROGRAMMAZIONE CLIENT-SIDE E SERVER-SIDE

La programmazione server-side permette la creazione di pagine web dinamiche, perché usare programmazione client-side?



BENEFICI CLIENT-SIDE

- **Usabilità**: si può modificare il contenuto di una pagina web senza *post back* con il server (UI più veloci)
- **Efficienza**: può fare piccoli e veloci cambiamenti di una pagina senza aspettare la risposta del server
- **Event-driven**: può rispondere ad azioni dell'utente come click e pressione di tasti

BENEFICI SERVER-SIDE

- **Sicurezza**: ha accesso a dati privati del server; il client non può vedere il codice sorgente
- **Compatibilità**: non è soggetto a problemi di compatibilità del browser
- **Potente**: può scrivere file, aprire connessioni verso altri server, connessione verso a database, …

CHE COSA È JAVASCRIPT ?

- Javascript (JS) è un linguaggio **client-side scripting** usato per:
  - Rendere interattive le pagine web
  - Inserire testo dinamico nel HTML (es. date)
  - Event-driven (es. click dell'utente su un button)
  - Recuperare informazioni riguardo il computer dell'utente (es. tipo di web browser)
  - Fare calcoli sul computer dell'utente (es. validazione in una form HTML)
- E' uno standard web (nel 1998 ECMAScript-262 è diventato standard ISO), ma non è supportato allo stesso modo da tutti i browser

INFORMAZIONI RIGUARDO L'UTENTE

Tipo di web browser utilizzato dall'utente

```
function recuperaInfoBrowser(){
 document.getElementById("Form1").Name.value=navigator.appName;
 document.forms["Form1"]["Version"].value=navigator.appVersion;
 document.forms["Form1"]["Code"].value=navigator.appCodeName;
 document.getElementById("Form1").Agent.value=navigator.userAgent;
}
```

JS

INFORMAZIONI RIGUARDO L'UTENTE

Creare Cookie

```
function impostaCookie (nome, valore, scadenza) {
 if (scadenza == "") {
       var oggi = new Date();
       oggi.setMonth(oggi.getMonth() + 3);
       //restituisce la data nel formato necessario
       scadenza = oggi.toGMTString(); }
 valore = escape(valore);
 document.cookie = nome + "=" + valore + " , expires = " + scadenza;
}
```

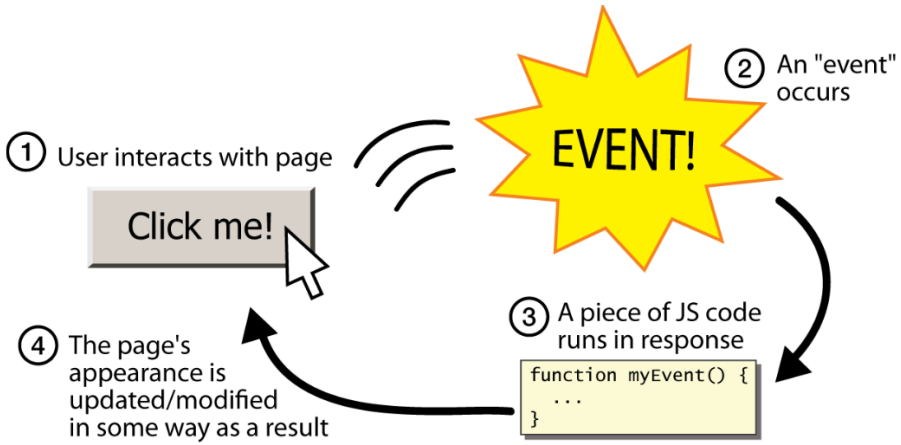
JS

EVENT-DRIVEN

Evento click su button

```
function myEvent() {
  alert ("Ciao!") ;
}
```

JS



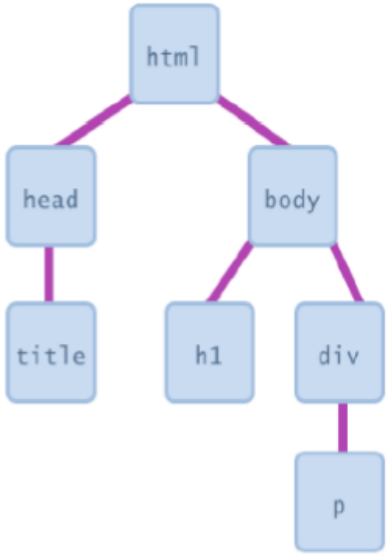

```
<input type="button" value="Click me!" onClick="myEvent()">
```

HTML

Gli oggetti di una pagina web hanno associati dei gestori di eventi (es. `onClick`) attraverso i quali è possibile eseguire comandi JS o chiamata a funzioni personalizzate

MANIPOLARE OGGETTI DI UNA PAGINA WEB

- Il codice JS è in grado, attraverso il DOM, di manipolare gli elementi di una pagina web:
  - Esaminare lo stato dell'elemento (es. *checked box*)
  - Cambiare lo stato dell'elemento (es. inserire nuovo testo in un `div`)
  - Cambiare lo stile di un elemento (es. rendere rosso un paragrafo `p`)



Il **DOM** rappresenta il modello ad oggetti di una pagina web e possiede metodi per la navigazione e la modifica della sua struttura

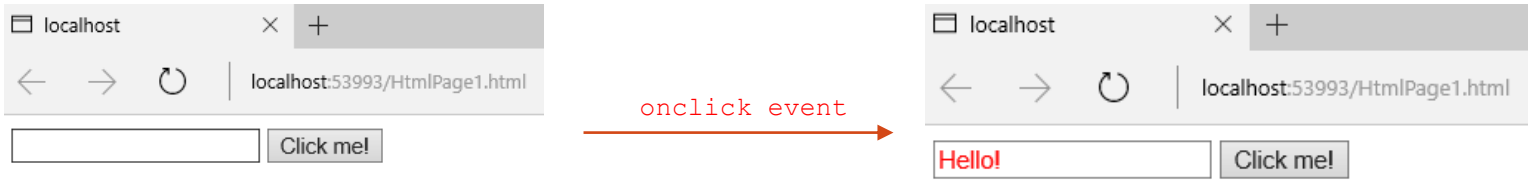
ESEMPIO DI MANIPOLAZIONE DOM

```html
<html>
    <body>
        <input id="textbox" type="text" />
        <input type="button" onclick="changeText()" value="Click me!" />
    </body>
</html>
```

HTML

```
function changeText() {
        var textBox = document.getElementById("textbox");
        textbox.value = "Hello!";
        textbox.style.color = "red";
        }
```

JS



VALIDAZIONE FORM HTML

Prima dell'invio di una form HTML

```
function validaUserForm() {
  var x=document.forms["userForm"]["userName"].value;
  if (x==null || x=="")
  {
    alert("Devi inserire un nome.");
    return false;
  }
 }
```

JS

DOVE USARLO ?

- Spesso il codice JS viene scritto all'interno del tag `script` posto nel `body` o `head` (come per il CSS), ma tale pratica è **sconsigliata** perché *contenuto*, *presentazione* e *comportamento* dovrebbero sempre rimanere separati

```
<html>
      <head>
           <title>…</title>
            <script>…</script>
      </head>
      <body>
             <script>…</script>
      </body>
 </html>
```
HTML

- In genere le istruzioni sono eseguite in ordine sequenziale, ad eccezione eventi/funzioni, e se presente del codice all'interno del tag `head` sarà eseguito prima del completo caricamento della pagina HTML

LINKING A UN FILE JS

- E' buona pratica usare un file separato di estensione `.js` per contenere tutto il codice JS utilizzato, come avviene in una libreria JS
- Per utilizzarlo in una pagina HTML è necessario usare il tag `script`

```HTML
<script src="filname.js" type="text/javascript"></script>
```

posto nel tag `head` della pagina HTML

- L'attributo `type` è opzionale e specifica il tipo di codice in uso

LIBRERIE

- È possibile usare l'attributo `src` per definire il *path* di una libreria JS

```HTML
<script src="…" type="text/javascript"></script>
```

in locale

```HTML
src="script/jquery-1.10.2.js"
src="script/myLib.js"
```

in rete

```HTML
src="http://code.jquery.com/jquery-1.10.2.js"
```

ARGOMENTI

- Introduzione
- Linguaggio
- Browser Object Model (BOM)
- Document Object Model (DOM)
- Esercizi

LE VARIABILI

- In una variabile vengono salvati i dati durante l'esecuzione del codice
- JS è un linguaggio **debolmente tipizzato** ed è quindi facoltativo indicare il tipo della variabile
- Per usare una variabile è necessario indicare al *parser* il suo nome utilizzando l'istruzione `var`:

```
var nomeVariabile;                                    JS
```

Una variabile non ha valore fino a quando non viene inizializzata

LE VARIABILI

- E' possibile **inizializzare** una variabile contestualmente alla dichiarazione:

```
var nomeVariabile = espressione;                    JS
```

- Per `espressione` si intende una sequenza di operatori, variabili e/o dati che restituisca a un valore

- L'operazione fondamentale delle variabili è **l'assegnazione**, attribuzione di un valore ad una variabile:

```
var nomeVariabile;          // definizione
nomeVariabile = valore;     // assegnazione          JS
```

dove `valore` è un'espressione

LE VARIABILI

Esempi

```JS
var1 = "ciao!";
var2 = 3.8;
var3 = false;
var4 = var3;  //var4 copia del valore di var3
```

I tipi non sono specificati, ma JS utilizza

`Number, Boolean, String, Array, Object, Function, Null, Undefined`

utilizzando `typeof()` si può verificare il tipo di variabile

OPERATORI MATEMATICI

Operatori	Descrizione	Esempi	Valore	Risultati
+	Addizione	x=y+2	y=5	x=7
-	Sottrazione	x=y-2	y=5	x=3
*	Moltiplicazione	x=y*2	y=5	x=10
/	Divisione	x=y/2	y=5	x=2.5
%	Modulo	x=y%2	y=5	x=1
++	Incrementa di uno	x=++y	y=6	x=7
		x=y++	y=6	x=6
--	Decrementa di uno	x=--y	y=4	x=3
		x=y--	y=4	x=4

OPERATORI STRINGA

- In JS il + è l'operatore concatenazione, unione di più stringhe

Esempio

```
var nome = "Mario";
var cognome = "Rossi";

var messaggio = " Mi chiamo " + nome +" "+ cognome + ".";
// messaggio: Mi chiamo Mario Rossi.
```

JS

OPERATORI BOOLEANI

Sono quelli che confrontano due valori dello stesso tipo e restituiscono un valore booleano (`true` o `false`)

Operatori	Descrizione	Esempi	Risultati
<	minore	3 < 4	true
<=	minore o uguale	3 <= 3	true
==	uguale	4 == 5	false
>=	maggiore o uguale	5 >= 9	false
>	maggiore	3 > 4	false
!=	diverso	5 != 5	false

OPERATORI LOGICI

Gli operatori logici permettono di comporre insieme più espressioni booleane e restituiscono un valore booleano

Operatori	Descrizione	Esempi	Risultati	Valore
&&	congiunzione logica	(a > 0) && (a < 100)	true	devono essere vere entrambe le espressioni
||	disgiunzione logica	(a < 0) || (b > 20)	true	se almeno una delle due espressioni è vera
!	negazione logica	!(a < 0)	true	se il valore di a non è minore di 0

STRUTTURE DI CONTROLLO

```
if (condizione) {
  istruzioni;
}
```
JS

```
if (condizione) {
  istruzioni;
}else{
  istruzioni;
}
```
JS

```
if (condizione1)
{
  istruzioni;
}else if (condizione2){
  istruzioni;
}else{
  istruzioni;
}
```
JS

```
switch (espressione)
{
  case valore1:
    istruzioni;
    break;
  case valore2:
    istruzioni;
    break;
  default :
    istruzioni;
}
```
JS

CICLI

```
for (variabile in oggetto) {
  istruzioni;
}
```
JS

```
do {
   istruzioni;
}
while(condizione)
```
JS

```
while(condizione) {
  istruzioni;
}
```
JS

```
for ([iniziale]; [condizione]; [incremento]) {
   istruzioni;
 }
```
JS

FUNZIONI

- Una funzione è un blocco di istruzioni identificato da una nome, che può:
  - accettare o meno parametri
  - restituire un valore
- La sintassi per la creazione di una funzione è la seguente:

```
function nomeFunzione (arg1, arg2,…, argN) {

   istruzioni;

  // return …; se restituisce un valore
}
```
JS

ESEMPI DI FUNZIONI

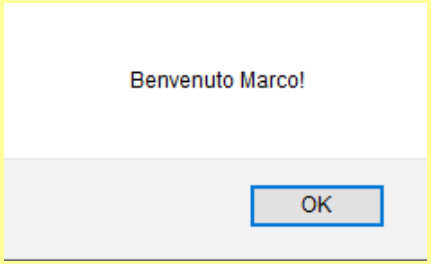
Definizione

```
function visualizzaMessaggio(nome){
 alert("Benvenuto, " +nome+ "!");
}
```
JS

```
function sommaDueNumeri(a, b){
   var somma = a + b;
   return somma;
}
```
JS

Utilizzo

```
visualizzaMessaggio("Marco!");
```
JS




```
var c = sommaDueNumeri(3,5);
```
JS

```
//  c assume il valore 8
```

GESTIONE DEGLI ERRORI

- Uno script JS può terminare normalmente o *sollevare un'eccezione*
- Un'eccezione rappresenta una situazione anomala o di errore
- Viene *lanciata* in un punto del codice e può essere *catturata* e gestita utilizzando il costrutto

```
try {
// Istruzioni in cui possono essere lanciate delle eccezioni
}
catch (error) {
// Istruzioni da eseguire in caso di eccezione
}
finally {
// Istruzioni da eseguire successivamente in entrambi i casi
}
```
JS

L'INTERAZIONE CON L'UTENTE

- È possibile interagire con l'utente, utilizzando uno codice JS all'interno di una pagina web, nei seguenti modi:
  - finestra di avviso
  - finestra di conferma
  - finestra di richiesta input
  - standard output

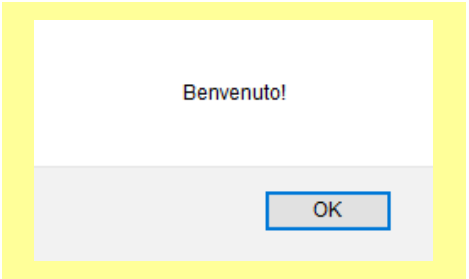
ESEMPIO DI FINESTRA DI AVVISO

Definizione

```
function visualizzaMessaggio(){
   alert("Benvenuto!");
}
```
JS

Utilizzo

```
visualizzaMessaggio();
```
JS



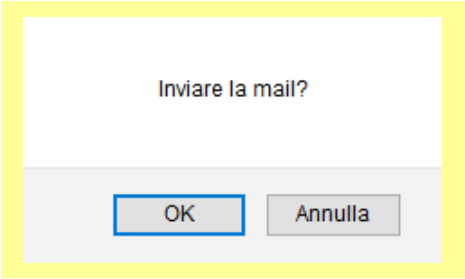
ESEMPIO FINESTRA DI CONFERMA

Definizione

```
function mostraConferma(){
     var r = confirm("Inviare la mail?");
     if (r == true)     {
       alert("Mail inviata!");
     }else{
       alert("Invio mail annulato!");
     }
   }
```
JS

Utilizzo

```
mostraConferma();
```
JS



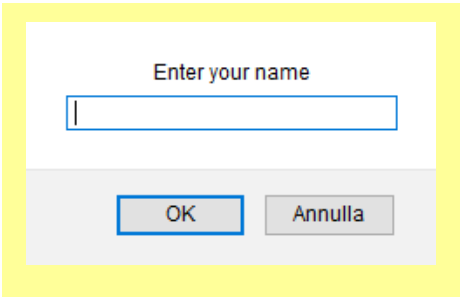
ESEMPIO FINESTRA DI RICHIESTA INPUT

Definizione

```
function mostraRichiestaInput(){
     var name = prompt("Enter your name", "");
     if (name != null && name != ""){
       document.write("Ciao " + name + "!");
     }
   }
```
JS

Utilizzo

```
mostraRichiestaInput();
```
JS



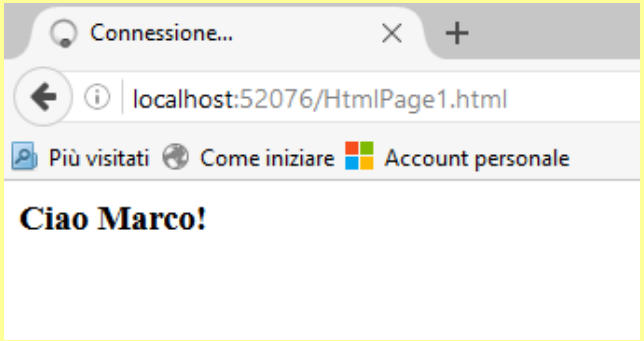
ESEMPIO DI STANDARD OUTPUT

Definizione

```
function mostraStandardOutput(){
        var name = "Marco";
        document.write("<h3>Ciao " + name + "!</h3>");
   }
```
JS

Utilizzo

```
mostraRichiestaInput();
```
JS

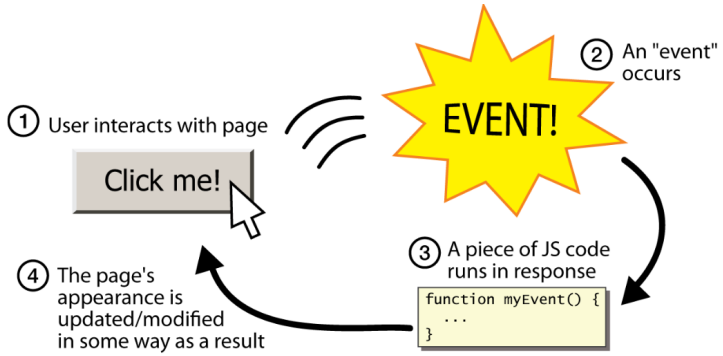


EVENTI

- Gli oggetti HTML possono generare eventi ai quali corrisponderà un'azione, se il corrispondente **gestore di eventi (event handler)** referenzia un codice JS

```
// funzione in script/myLib.js
function myEvent(){
  alert ("Ciao!") ;
}
```

JS




```
<html>
   <head>
<script src="script/myLib.js" type="text/javascript"></script>
   </head>
    <body>
       <input type="button" onclick="myEvent()" value="Click me!"/>
   </body>
</html>
```

HTML

VALIDAZIONE FORM HTML

Prima di inviare una form HTML al server è possibile validare i dati contenuti utilizzando un codice JS

I controlli più comuni sono relativi a:

- campi obbligatori
- campi data
- campi numerici
- validità formale di un pattern (e-mail, codice fiscale,…)

ESEMPIO DI VALIDAZIONE FORM HTML

```
function validaUserForm (){
            var x = document.forms["userForm"]["userName"].value;
            if (x == null || x == ""){
                  alert("Devi inserire un nome.");
                  return false;
             }
             return true;
}
```

JS

```
<form name="userForm" action="userForm.php" onsubmit="return
         validaForm();" method="post">
         Nome: <input type="text" name="userName">
         <input type="submit" value="Invia">
</form>
```

HTML

OGGETTI NATIVI

Gli oggetti nativi (built-in) JS più usati sono:

- String
- RegExp
- Array
- Date
- Math
- Boolean
- Number

OGGETTO STRING

- L'oggetto `String` permette di effettuare operazioni sulle stringhe, come la ricerca, l'isolamento di un carattere e altro
- Esempi

```
var testo = "Sono una stringa"; // creazione oggetto String
testo.length;                   // restituisce  16

var t1 = new String ("Roma");  // creazione oggetto String
var t2 = t1.toLowerCase();     // restituisce "roma"
var t3 = t1.toUpperCase();     // restituisce "ROMA"
```

JS

OGGETTO REGEXP

- L'oggetto `RegExp` permette di usare *regular expressions* per fare *pattern-matching*
- Il *pattern* è definito utilizzando le *regular expressions*
- Un esempio d'uso: controllo sintattico della mail

```
function demoShowMatchClick() {
  var re = new RegExp(document.demoMatch.regex.value);
  var m = re.exec(document.demoMatch.subject.value);
  if (m == null) {
    alert("No match");
  } else {
    var s = "Match at position " + m.index + ":\n";
    for (i = 0; i < m.length; i++) {
      s = s + m[i] + "\n";
    }
    alert(s);
  }
}
```

JS

OGGETTO ARRAY

- L'oggetto `Array` permette l'organizzazione dei dati secondo un indice

```
// Costruttori oggetto Date
var corsi = new Array(); // Array vuoto


var corsi = new Array(3); // Array vuoto con lunghezza 3
corsi[0]="Geometria";
corsi[1]="Matematica1";
corsi[2]="Fisica1";


var corsi=new Array("Geometria","Matematica1","Fisica1");
var corsi = ["Geometria","Matematica1","Fisica1"];
```

JS

ESEMPIO OGGETTO ARRAY

```JS
// Array
var corsi = ["Geometria", "Matematica1","Fisica1"];

// Proprietà
corsi.length;           // restituisce la lunghezza di corsi

// Metodi
corsi.push("Fisica2"); // inserisce un elemento in corsi
corsi.pop();           // elimina l'ultimo elemento e restituisce il valore eliminato
corsi.shift();         // elimina il primo elemento e restituisce il valore eliminato
corsi.sort();          // ordina secondo l'ordine alfabetico
corsi.reverse();       // inverte l'ordine degli elementi
```

OGGETTO DATE

- L'oggetto `Date` permette di gestire le date (aggiungere / sottrarre anni, mesi, giorni, recuperarne il valore, …)

```
// Costruttori oggetto Date
new Date();
new Date(millisecondi);// millisecondi dal 1970/01/01
new Date(dateString); // con le impostazioni del client
new Date(anno,mese,giorno,[ora],[minuti],[secondi],[millisecondi]);
// gennaio è il mese 0 e non 1
```

JS

ESEMPIO OGGETTO DATE

```
var oDate = new Date("10-7-2007");
// Attenzione:
// per un utente USA invece che il 10 luglio 2007
// la data sarà 7 ottobre 2007.

var oDate = new Date(2007, 7, 10); // non ambiguo

var giornoS = oDate.getDay();
var giornoM = oDate.getDate();
var mese = oDate.getMonth();
var anno = oDate.getFullYear();
var nuovadata = oDate.setDate(data.getDate()+5);
```

JS

OGGETTO MATH

- L'oggetto `Math` mette a disposizione numerose funzioni e costanti matematiche e si accede ai metodi e alle proprietà senza istanziare l'oggetto

```
// Proprietà
Math.E;          // restituisce il valore della costante matematica E
Math.PI;         // restituisce il valore approssimato Pi greco
Math.LN2;        //restituisce valore del logaritmo naturale di 2

// Metodi
Math.abs(x);      // restituisce il valore assoluto di x
Math.sqrt(x);     // restituisce la radice quadrata di x
Math.pow(b, n);   // restituisce l'enesima potenza del numero b
```

JS

OGGETTO BOOLEAN

- L'oggetto `Boolean` permette di convertire in un booleano un valore

```JS
// Costruttori oggetto Boolean
var oBoolean = new Boolean(valore);
var oBoolean = Boolean(valore);
```

se il `valore` è: *omesso*, *`false`*, *`0`*, *`null`*, *`""`*, allora l'oggetto sarà `false`, altrimenti sarà `true`

- Esempi

```JS
var oBoolean = true;
var oBoolean = !!(valore);
var oBoolean = "valore" > 0; // false
```

OGGETTO NUMBER

- L'oggetto `Number` permette di effettuare operazioni di calcolo e di accedere a valori di costanti (numero massimo e minimo rappresentabile, + infinito, – infinito e NaN)

```
// Costruttore oggetto Number
var oNumber = new Number(valore);                    JS
```

il costruttore prende come parametro un `valore` numerico

- Esempi

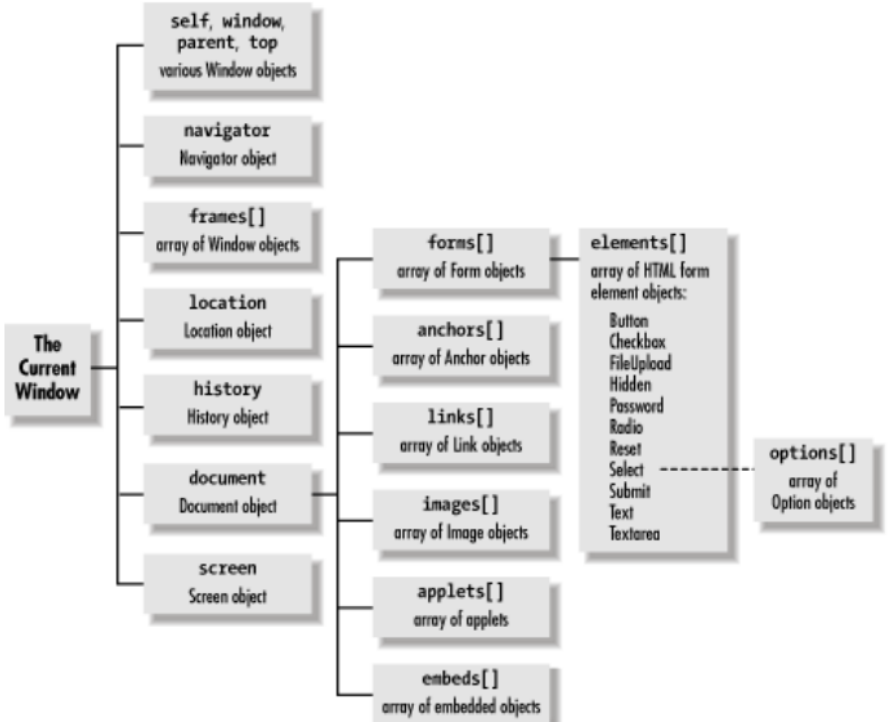
```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);                       JS
```

ARGOMENTI

- Introduzione
- Linguaggio
- Browser Object Model (BOM)
- Document Object Model (DOM)
- Esercizi

BROWSER OBJECT MODEL (BOM)

- Il **BOM** rappresenta il modello ad oggetti del web browser e possiede metodi e proprietà utilizzabili per recuperare informazione o interagire con esso



nome	descrizione
window	la finestra del browser
navigator	informazioni circa il web browser in uso
location	URL della pagina web corrente
history	lista delle pagine visitate dall'utente
document	pagina web corrente e suoi contenuti (DOM)
screen	informazioni circa l'area dello schermo occupata dal web browser

L'OGGETTO WINDOW

- Rappresenta la finestra del browser
- Tutti gli oggetti JS globali, le funzioni e le variabili sono automaticamente membri (proprietà e metodi) dell'oggetto finestra

```JS
window.document.getElementById("Form1");
// sono equivalenti
document.getElementById("Form1");
```

proprietà:

`document, history, location, name, innerHeight, innerWidth,...`

metodi:

- `alert(), confirm(), prompt()`
- `setInterval(), setTimeout(), clearInterval(),clearTimeout()` (**timer**)
- `open(), close()` (**nuove window del browser**)
- `blur(), focus(), moveBy(), moveTo(), print(), resizeBy(), resizeTo(), scrollBy(), scrollTo()`

L'OGGETTO NAVIGATOR

- Contiene informazioni riguardo il browser in uso
- L'oggetto `window.navigator` può essere scritto senza il prefisso `window`

proprietà:
`appName,appVersion, browserLanguage, cookieEnabled, platform, userAgent, language, ...`

metodi:
`javaEnabled(), ...`

Esempio

```
document.getElementById("demo").innerHTML = navigator.language;          JS
```

L'OGGETTO LOCATION

- Usato per recuperare l'indirizzo web della pagina corrente (URL) e per direzionare il browser su una nuova pagina
- L'oggetto `window.location` può essere scritto senza il prefisso `window`

proprietà:
`host, hostname, href, pathname, port, protocol, search`

metodi:
`assign(), reload(), replace()`

Esempio

```
document.getElementById("demo").innerHTML = "Page location: " + location.href;
```

JS

L'OGGETTO HISTORY

- Contiene la lista dei siti visitati dall'utente
- L'oggetto `window.history` può essere scritto senza il prefisso `window`
- Per tutelare la privacy dell'utente, esistono delle limitazioni riguardo come JS può accedere a tale oggetto

proprietà:
`length`

metodi:
`back(), forward(), go()`

Esempio

```
history.back();                    JS
```

L'OGGETTO SCREEN

- Contiene informazioni riguardo lo `screen` in uso
- L'oggetto `window.screen` può essere scritto senza il prefisso `window`

proprietà:
`availHeight, availWidth, colorDepth, height, pixelDepth, width, ...`

Esempio

```
document.getElementById("demo").innerHTML = "Screen Width: " + screen.width;
```
JS

L'OGGETTO DOCUMENT

- Rappresenta la pagina web corrente e i suoi contenuti (DOM)
- L'oggetto `window.document` può essere scritto senza il prefisso `window`

proprietà:

`anchors, body, cookie, domain, forms, images, links, referrer, title, URL`

metodi:

`getElementById(), getElementsByName(), getElementsByTagName(), close(), open(), write(), writeln(), firstChild(), lastChild(), childNodes(), nextSibling(), previousSibling(), parentNode(), createElement(), createTextNode(), appendChild(), insertBefore(), removeChild(), replaceChild(), ...`

Esempio

```
document.getElementById("demo").innerHTML = navigator.language;        JS
```

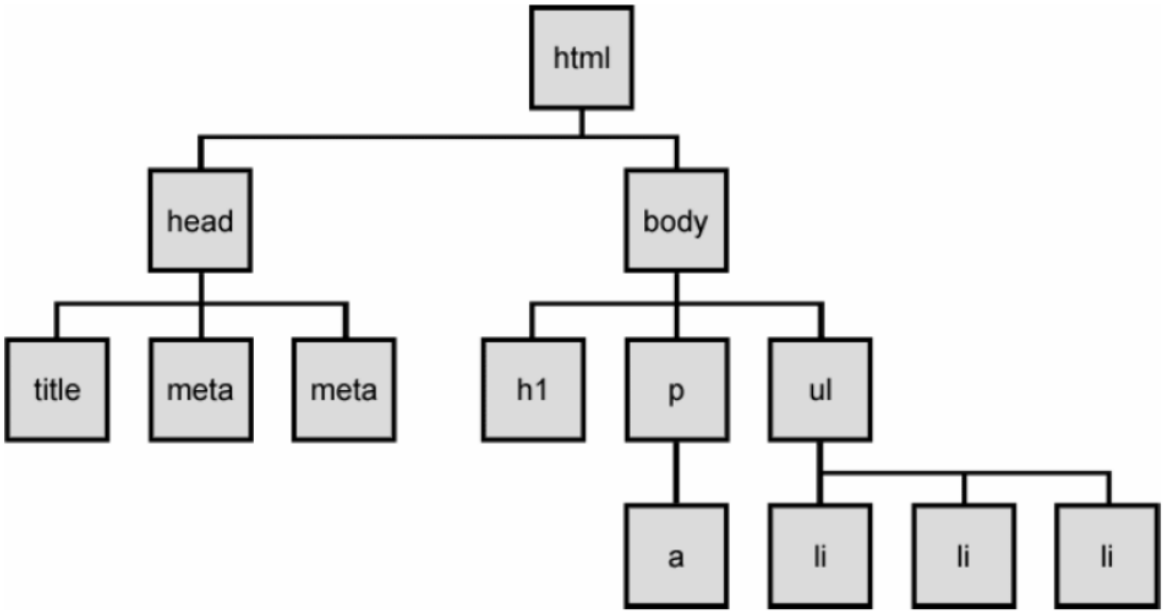
ARGOMENTI

- Introduzione
- Linguaggio
- Browser Object Model (BOM)
- Document Object Model (DOM)
- Esercizi

DOCUMENT OBJECT MODEL (DOM)

- Il **DOM** rappresenta il modello ad oggetti di una pagina HTML e possiede metodi per la navigazione e modifica della sua struttura

Esempio **albero DOM**

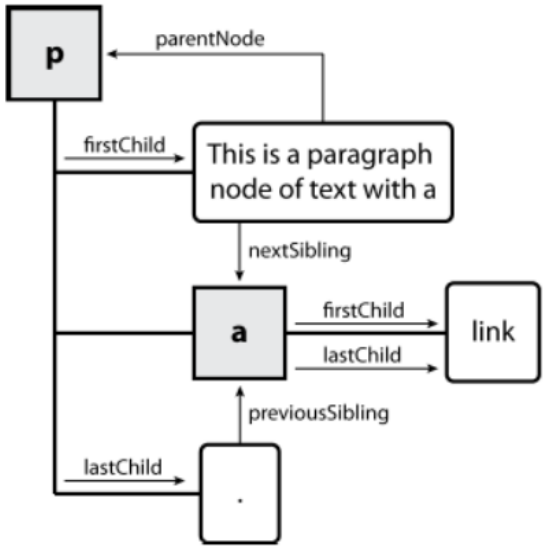


TIPI DI NODO DOM

- Elemento (tag HTML)
  - può avere figli e/o attributi
  - può contenere testo
- Attributo (coppia attributo/valore)
  - è figlio in un elemento
  - non può avere figli/attributi
  - in genere non è mostrato nell'albero DOM

Esempio

```
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```
HTML



ARGOMENTI

- Introduzione
- Linguaggio
- Browser Object Model (BOM)
- Document Object Model (DOM)
- Esercizi

ESERCIZI

Esercizio 1

Creare una pagina HTML contenente un elenco di link.

Quando l'utente passa sopra (onmouseover) un link mostrare un messaggio sulla barra di stato. Il messaggio deve essere diverso per ogni link.

Esercizio 2

Creare una funzione JS che cambi il colore di sfondo (document.bgcolor) ogni due secondi.

La funzione deve essere mandata in esecuzione dopo il caricamento della pagina (onload).

ESERCIZI

Esercizio 3

Modificare l'esercizio precedente in modo che il cambio di colore di sfondo viene attivato con la pressione (onclick) di un bottone.

Inserire un secondo bottone per interrompere l'animazione.

Esercizio 4

Creare uno script JS che chieda il nome e il cognome all'utente prima che la pagina venga caricata.

Provare a chiamare una finestra di dialogo che contenga la scritta "Benvenuto"+ nome + cognome.

In fondo alla pagina deve essere scritto "Benvenuto"+ nome + cognome.

ESERCIZI

Esercizio 5

Creare una pagina che chieda all'utente di fornire la propria autorizzazione al trattamento dei dati.

Nel caso non sia data, scrivere "Autorizzazione non concessa".

Nel caso contrario scrivere "Autorizzazione concessa".

Esercizio 6

Scrivere uno script JS che stampi il fattoriale di un numero ricevuto in input (metodo prompt) dall'utente.

Lo script deve verificare che il numero si compreso tra 1 e 15, in caso contrario deve segnalare l'errore all'utente.

ESERCIZI

Esercizio 7

Scrivere uno script JS che stampi la tabellina di un numero, compreso tra 1 e 10, ricevuto in input dall'utente.

Esercizio 8

Scrivere uno script JS che chieda una serie di numeri all'utente. Si interrompa quando l'utente inserisce al posto di un numero la parola chiave "end". Interagisce con l'utente nel caso abbia inserito dei valori non validi. Chieda conferma prima di calcolare la media dei numeri introdotti. Stampi a video il risultato in una tabella.

ESERCIZI

Esercizio 9

Chiedere tre colori in input all'utente: uno per il testo, uno per lo sfondo e uno per il titolo.

Chiedere all'utente la dimensione del carattere di base (da 1 a 7).

Determinare la dimensione e le caratteristiche di H1 (+3, grassetto), H2 (+2) e H3 (+1, grassetto, corsivo) rispetto alle caratteristiche inserite dall'utente per il carattere.

Creare un testo in cui siano evidenti queste caratteristiche.

Controllare che il colore dello sfondo, quello del titolo e quello del testo siano diversi.

Nel caso due o più dei colori assegnati dall'utente siano uguali, devono essere presenti regole che determinino che colore devono assumere titolo, testo e sfondo.

ESERCIZI

Esercizio 10

Implementare la funzione celsius che restituisce l'equivalente Celsius di una temperatura Fahrenheit usando la formula

C=5.0/9.0 * (F -32)

Implementare la funzione fahrenheit che restituisce l'equivalente Fahrenheit di una temperatura Celsius usando la formula

F=9.0/5.0 * C +32

Le funzioni devono verificare se il parametro immesso è di tipo numerico. Testare le funzioni inserendo le loro definizioni in un documento HTML contenente uno script JS che chiede all'utente di immettere un valore e visualizza il risultato dell'invocazione della funzione sul valore immesso.

RISORSE

JavaScript

- https://it.wikibooks.org/wiki/JavaScript/Introduzione
- https://it.wikipedia.org/wiki/JavaScript
- https://www.w3schools.com/js/default.asp
- https://www.w3schools.com/jsref/default.asp
- ECMA-262 - Ecma International (ecma-international.org)

DOM

- https://www.w3schools.com/js/js_htmldom.asp