LEZIONE 4

JAVASCRIPT COME

JAVASCRIPT È UN LINGUAGGIO INTERPRETATO

Lo script viene eseguito immediatamente: uno script javascript viene interpretato dal browser e da un output sul monitor, sulla stampante, un output audio, ecc.

JAVASCRIPT È UN LINGUAGGIO INTERPRETATO

- L'interprete elabora, risolve:
  - espressioni separate da un punto e virgola
  - blocchi di espressioni contenute tra due parentesi graffe
- Il programma è l'effetto collaterale di questa valutazione

LA FUNZIONE EVAL

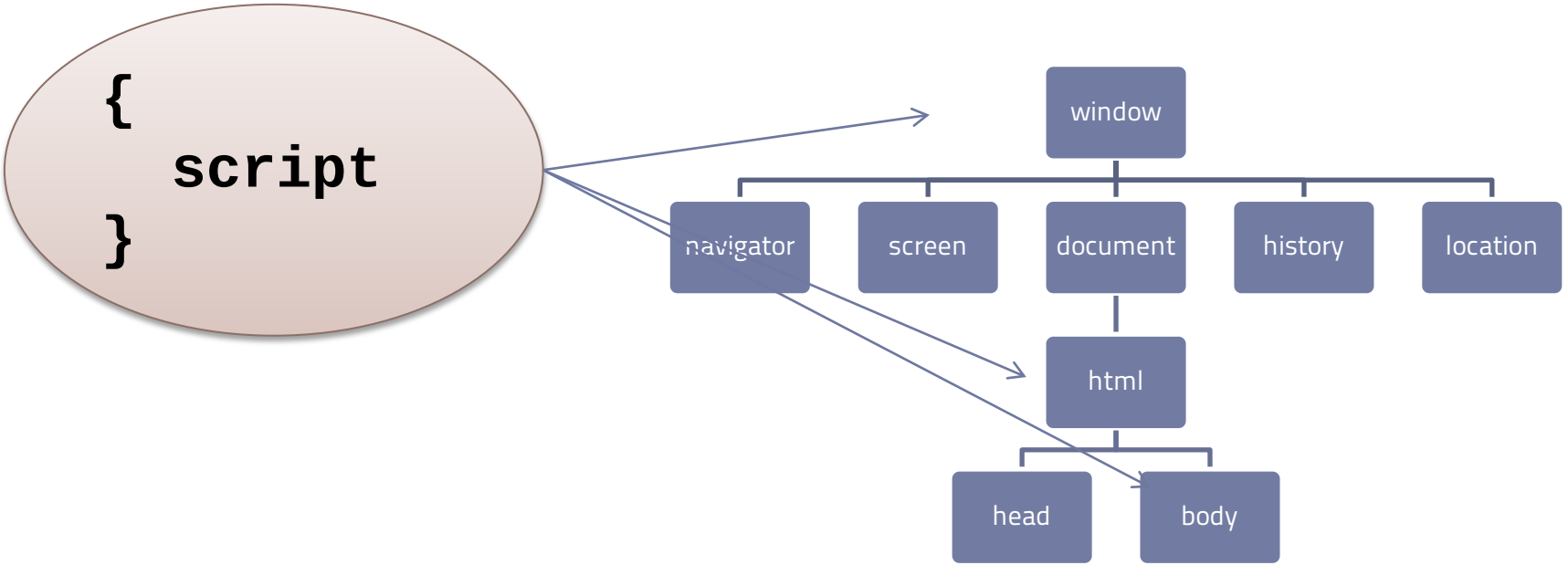
```
var risultato;
risultato = eval(script);
//  script rappresenta un pezzo
    di script in formato stringa
```

LA FUNZIONE EVAL

- `script` viene valutato (risolto) dall'interprete e se esiste viene restituito un valore
- l'effetto collaterare è l'esecuzione del programma

JAVASCRIPT COSA

JAVASCRIPT AGISCE SUL DOM



JAVASCRIPT AGISCE SUL DOM

- Ogni elemento di una pagina HTML è un oggetto collocato nella struttura gerarchica del Document Object Model.
- Possa intervenire sulle proprietà degli oggetti per modificarne l'aspetto
- Posso applicare i metodi che il DOM mi mette a disposizione per ogni oggetto per compiere azioni o ricavare informazioni.

JAVASCRIPT DOVE

IL TAG SCRIPT

- Possiamo inserire il codice JavaScript nella sezione **head** oppure nella sezione **body** della pagina HTML utilizzando un'apposito TAG il tag script.
- Se il tag viene inserito nella sezione **head** in codice verrà eseguito prima che qualsiasi elemento della pagina venga visualizzato .
- Se il tag viene inserito nella sezione **body** in codice verrà eseguito immediatamente, non appena il browser incontrerà il tag durante l'elaborazione della pagina.

IL TAG SCRIPT

- È sufficiente creare il tag (**<script></script>**) e aggiungere l'attributo **type**="text/javascript"
- Il codice JavaScript va inserito tra l'apertura e la chiusura del tag. Così:

```
<script type="text/javascript">
  alert("ciao");
</script>
```

IL TAG SCRIPT

- Se si usa come DTD XHTML strict. La sisntassi corretta diventa questa:

```
<script type="text/javascript">
/* <![CDATA[ */
  alert("ciao");
/*]]>*/
</script>
```

NO SCRIPT

- Il tag **noscript** serve per visualizzare messaggi specifici quando javascript non è supportato o disabilitato.

```
<noscript>
<div align="center">
  <h3><font face="Verdana,Arial,Helvetica,sans-serif">
    Per visualzzare correttamente il contenuto della
    pagina occorre avere JavaScript abilitato.
  </font></h3>
</div>
</noscript>
```

FILE ESTERNO

- In molti casi il codice javascript è inserito in un file esterno (normalmente con estensione **.js**).
- In questo caso il tag script prende questa forma:

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

- Quando si carica un file esterno il tag **script** va **SEMPRE** inserito nella sezione **head** del documento.

JAVASCRIPT QUANDO

QUANDO VIENE ESEGUITO UNO SCRIPT

- I comandi presenti in uno script vengono eseguiti immediatamente, quando il browser incontro il **tag** "script" che li contiene.
- I comandi incorporati in una funzione non vengono eseguiti immediatamente. Vengono eseguiti:
  - quando la funzione viene richiamata da uno script;
  - quando viene lanciato (fired) l'evento a cui la funzione è associata;
- In questo modo è possibile rinviare, delegare l'esecuzione di un gruppo di comandi ad un evento.

ESEMPIO 1 (A)

```
<head>
  <script type="text/javascript">
      var nome;
      var oggi;
      nome = window.prompt("Inserisci il
             tuo nome", "");
      oggi = new Date();
  </script>
</head>
```

ESEMPIO 1 (B)

```
<div class="article">
   <script type="text/javascript">
       document.write("<p>Benvenuto " + nome +
              " in questo sito. " +"Oggi &egrave; il " +
              oggi.getDate() + "/" + (oggi.getMonth() + 1)+
              "/" + oggi.getFullYear() + " e sono le " +
              oggi.getHours() + " e " + oggi.getMinutes() +
              " minuti.");
   </script>
</div>
```

ESEMPIO 2

```
<head>
    <script>
    function eseguiCodice ()
    {
          window.document.getElementById("eval_txt").value =
          eval(window.document.getElementById("espressione_txt").value);
    }
    function caricamentoPagina()
    {
          window.document.getElementById("eval_btn").onclick = eseguiCodice;
    }
    window.onload = caricamentoPagina;

    </script>
</head>
```

COME PROCEDEREMO

1. Creeremo una pagina html e assegneremo ad ogni elemento da programmare un attributo **id** univoco.
2. Inseriremo in un tag **<script>** collocato nella sezione **<head>** della pagina oppure in un **file di script esterno** una funzione che assoceremo all'evento windows.onload. Daremo a questa funzione un nome di nostro gradimento, ad esempio "fineCaricamento" o "documentoPronto".
3. In questa funzione inseriremo tutti i comandi di inizializzazione:
   - Daremo un valore iniziale alle proprietà degli elementi che ci interessano
   - Associaremo dei gruppi di comandi (delle funzioni) agli eventi che gestiremo

COSA POSSO FARE

- Definire o inizializzare una variabile
- Dichiarare e definire una funzione
- Assegnare un valore a una proprietà o variabile
- Modificare il valore di una proprietà o variabile
- Invocare il metodo di un oggetto
- Richiamare una funzione
- Prendere una decisione

DEFINIRE E/O INIZIALIZZARE UNA VARIABILE

```
var adesso;
var adesso = new Date();
```

DICHIARARE E DEFINIRE UNA FUNZIONE

```
function somma(n1, n2){
    return n1 + n2;
}
```

funzione con nome

DICHIARARE E DEFINIRE UNA FUNZIONE

```
var somma = function(n1, n2){
    return n1 + n2;
}
```

funzione anonima

ASSEGNARE UN VALORE A UNA PROPRIETÀ O VARIABILE

```
var adesso = new Date();
document.getElementById("oggi_data").innerHTML = adesso.getDate();
```

RICHIAMARE UNA FUNZIONE

```
document.getElementById("eval_txt").value =
    eval(document.getElementById("espressione_txt").value);
```

TIPI IN JAVASCRIPT

Tipo di dati	Spiegazione	Esempio
Number	Qualsiasi valore numerico	miaVariabile=300;
Number	Numeri con virgola	miaVariabile=12.5;
String	Qualsiasi valore letterale. È una sequenza di caratteri, racchiusa tra virgolette.	miaVariabile="Wolfgang";
Null	È uno speciale tipo di dato che indica l'assenza di alcun valore ('è il nulla"). Non è lo zero.	miaVariabile=null;

Boolean	È uno tipo di dato che indica uno stato. Di fatto un valore booleano può assumere solo due valori: acceso (vero), spento (falso). È il classico 'interruttore della luce".	//Vero:
		miaVariabile=true;
		
		//Falso:
		miaVariabile=false;

Object	Array (Elenco di valor)i	miaVariabile=['lunedì','martedì','mercoledì', 'giovedì','venerdì', 'sabato', 'domenica']
Object	Informazione complessa	miaVariabile = {nome:"Mario", cognome:"Rossi", eta:25}

Function	Blocco di comandi	miaVariabile = fucntion()
		{
		//blocco comandi
		}

PRENDERE DECISIONI

LE STRUTTURE DI CONTROLLO

- Le strutture di programmazione che mi consentono di prendere decisioni sono essenzialmente due:
  - **condizionale**: faccio una determinata cosa se una condizione risulta vera altrimenti ne faccio un'altra
  - **iterativa** (o loop): ripeto una determinata operazione finche una condizione risulta vera

LE TABELLE DI VERITÀ

- Prendiamo questi enunciati:
  - esco se il tempo è bello ed è caldo
  - esco se il tempo è bello o è caldo
  - non esco se il tempo non è bello e non è caldo
  - non esco se il tempo non è bello o non è caldo

LE TABELLE DI VERITÀ

– esco se il tempo è bello ed è caldo

enunciato 1	congiunzione	enunciato 2	risultato
tempo è bello	ed	temperatura è caldo	esco
true	and	true	true
false	and	true	false
true	and	false	false
false	and	fasle	false

LE TABELLE DI VERITÀ

– esco se il tempo è bello o è caldo

enunciato 1	congiunzione	enunciato 2	risultato
tempo è bello	o	temperatura è caldo	esco
true	or	true	true
false	or	true	true
true	or	false	true
false	or	fasle	false

LE TABELLE DI VERITÀ

– non esco se il tempo non è bello e non è caldo

	enunciato 1	congiunzione	enunciato 2	risultato
non	tempo è bello	e	temperatura è caldo	non esco
not	true	and	true	false
not	true	and	false	false
not	false	and	true	false
not	false	and	false	true

LE TABELLE DI VERITÀ

– non esco se il tempo non è bello o non è caldo

	enunciato 1	congiunzione	enunciato 2	risultato
non	tempo è bello	o	temperatura è caldo	non esco
not	true	or	true	false
not	true	or	false	true
not	false	or	true	true
not	false	or	false	true

GLI OPERATORI LOGICI

operazione	javascript	precedenza
uguaglianza	==	1
disuguaglianza	!=	1
maggiore	>	1
maggiore o uguale	>=	1
minore	<	1
minore o uguale	<=	1
and	&&	2
or	||	2
not	!	2

SINTASSI DELL'ISTRUZIONE IF

- L'istruzione if può avere due forme:

  if (espressione) blocco di istruzioni

  if (espressione) blocco di istruzioni **else** blocco di istruzioni
- L'espressione che compare dopo la parola chiave **if** deve essere di tipo logico, se la condizione risulta vera viene eseguita l'istruzione subito seguente; nel secondo caso, invece, se la condizione risulta vera si esegue l'istruzione seguente, altrimenti si esegue l'istruzione subito dopo la parola chiave **else**.
- Per più scelte invece si può usare l'**else if** che permette di porre una condizione anche per le alternative, lasciando ovviamente la possibilità di mettere l'else (senza condizioni) in posizione finale.

BLOCCO IF

```
If (condizione)
{
  //comandi se condizione è vera
}
// il programma continua qui
```

BLOCCO IF ELSE

```
If (condizione)
{
  comandi se condizione è vera
}
else
{
  comandi se condizione è falsa
}
// il programma continua qui
```

ESEMPIO 1

```
/**
 * Funzione che formatta ore minuti e secondi
 * @param {Number} n
 */
function zeroPrima(n)
{
   //converto n in stringa concatenandolo a str
   var str = "";
   str = str + n;
   // se la lunghezza della stringa n è minore di 2
   // aggiungo uno 0 in testa
   if (str.length < 2){
     str = "0" + str;
   }
   return str;
}
```

ESEMPIO 2

```
var confronta = function ()
{
   var n = parseFloat(document.getElementById("numero").value);
   var c = parseFloat(document.getElementById("confronta").value);
   var message = "";
   if (isNaN(c) || isNaN(n))
   {
      message = "Errore: almeno uno dei valori inseriti non è un numero."
   }
   else if (c > n)
   {
      message = "Il numero inserito (" + c +") è maggiore del numero di riferimento."
   }
   else if (c == n)
   {
      message = "Il numero inserito (" + c +") è uguale del numero di riferimento."
   }
   else
   {
      message = "Il numero inserito (" + c +") è minore del numero di riferimento."
   }
   document.getElementById("messaggio_confronto").innerHTML = message;
}
```

LA PROGRAMMAZIONE ITERATIVA

- **Flusso naturale del programma:**
  - viene eseguita un'istruzione dopo l'altra fino a che non si incontra l'istruzione di fine programma.
- **Programmazione iterativa:**
  - un'istruzione (o una serie di istruzioni) vengo eseguite continuamente, fino a quando non sopraggiungono delle condizioni che fanno terminare il ciclo.

for

- Il **for** inizializza una variabile, pone una condizione e poi modifica (normalmente incrementa o decrementa) la variabile iniziale.

```
for (inzializzazione; condizione; modifica)
  blocco istruzioni;
```

- Il codice <blocco istruzioni> viene eseguito fino a che l'espressione <condizione> risulta vera, poi si passa la all'istruzione successiva al **for**.

ESEMPIO

```
for (var i = 0; i < valoreMassimo; i++)
{
   // faccio qualcosa utilizzando in valore di
   // che incrementa ad ogni ciclo fino a che
   // non raggiunge il valore massimo
}
// quando i raggiunge il valore massimo il
// programma continua qui
```

ESEMPIO

```
var cerca = function()
{
  var str = document.getElementById("ricerca").value;
  for (var i = 0; i < mesi.length; i++)
  {
    if (mesi[i] == str)
    {
         document.getElementById("messaggio_ricerca").innerHTML =
            "La stringa " + str + " è stata trovata al posto " + i;
         return;
    }
  }
  document.getElementById("messaggio_ricerca").innerHTML = "La stringa " +
         str + " non è stata trovata.";
}
```