Introduzione a JavaScript

Lamberto Ballan, Ombretta Gaggi

Università degli Studi di Padova

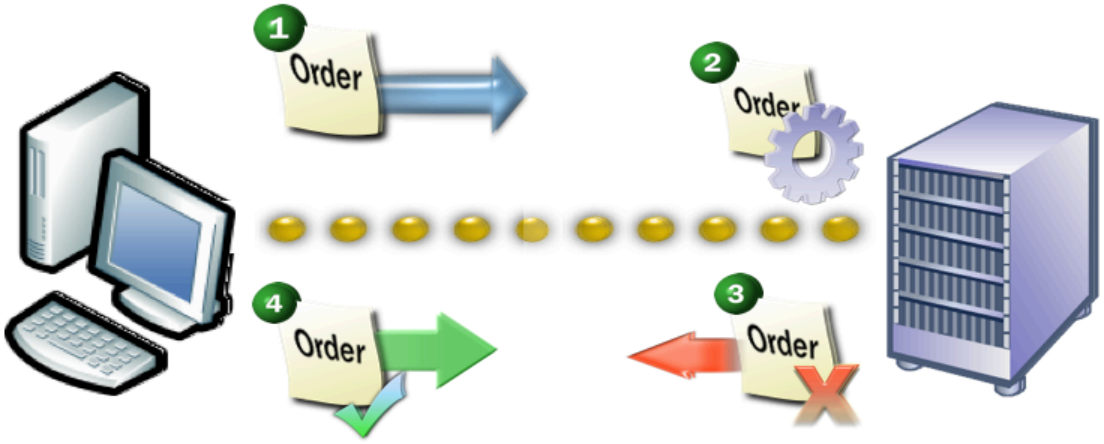
Tecnologie Web
A.A. 2017/18

Introduzione

- JavaScript viene introdotto nel 1995 da Netscape con il nome LiveScript
  - successivamente si unisce *Sun Microsystems*, creatrice del linguaggio Java, e il nome viene aggiornato
  - Microsoft sviluppò una sua versione proprietaria, Jscript
  - L'ente internazionale ECMA si è preoccupato di standardizzare i vari "dialetti" del linguaggio

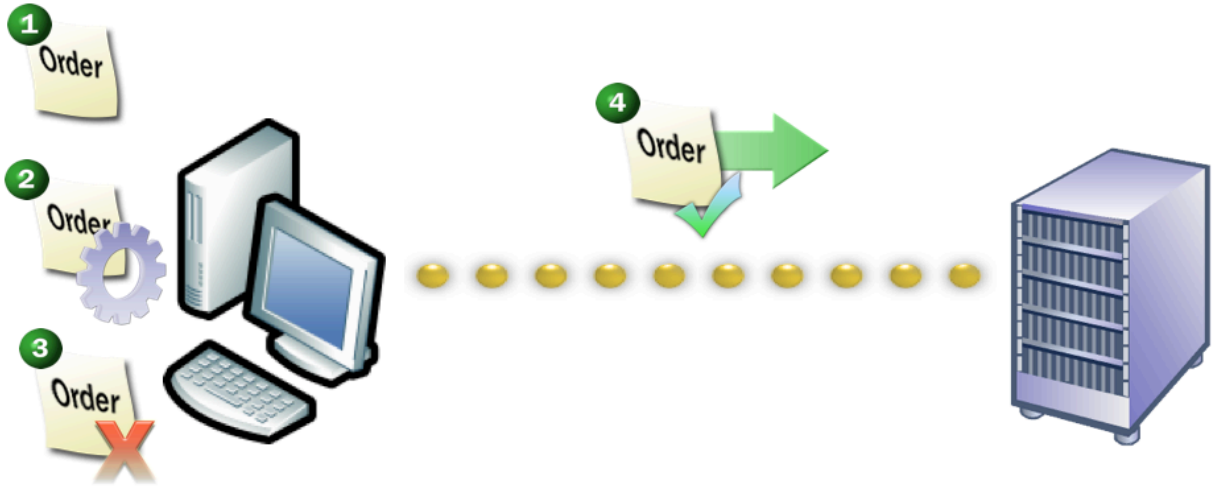
- JavaScript è un linguaggio di scripting di tipo *client side*
  - agisce su lato client per rendere dinamiche ed interattive le pagine web
  - permette lo sviluppo e il controllo di programmi e logiche di controllo "complesse"

Introduzione (2)



Server side

vs



Client side

Introduzione (3): JavaScript *vs* Java

- Nonostante il nome, JavaScript non è una derivazione del linguaggio Java (anche se hanno sintassi molto simili)

- Mentre Java è un linguaggio orientato agli oggetti, JavaScript non lo è pienamente
  - JavaScript supporta gli oggetti ma non le classi ("al posto" delle classi abbiamo i *prototipi*
  - JavaScript non supporta l'ereditarietà

- A differenza di Java, JavaScript non richiede la dichiarazioni delle variabili e permette una tipizzazione dinamica
  - È comunque opportuno dichiarare le variabili esplicitamente attraverso l'utilizzo della keyword var
  - Java è fortemente tipizzato, JavaScript lo è debolmente

Introduzione (4)

- JavaScript è un linguaggio interpretato (analogamente a quanto visto precedentemente per PHP)
  - essendo client side, l'interprete JavaScript è "contenuto" nel browser che visualizza la pagina (X)HTML

- Comportamento generale:
  - Se un documento XHTML NON contiene script, allora il browser elabora il documento linea per linea e ne presenta il contenuto
  - Se invece il documento XHTML contiene script:
    - il browser chiama l'interprete JavaScript per eseguirlo
    - finita l'elaborazione il browser torna al documento XHTML

Scripting non intrusivo

- Lo *scripting non intrusivo* è focalizzato sull'utente ed è progettato per migliorare una struttura di markup già di per sé semantica ed accessibile

- In particolare:
  - non attira l'attenzione dell'utente, è un'aggiunta funzionale al sito (migliora usabilità)
  - non attira l'attenzione dell'utente quando non funziona (degrado aggraziato)
  - non modifica le funzionalità della pagina, se non funziona l'utente non deve accorgersi della mancanza (accessibilità)
  - non modifica la struttura della pagina (separazione struttura – comportamento)

JavaScript

- È un linguaggio piuttosto semplice, che permette di creare documenti dinamici, in grado di interagire con l'utente
  - Es. dare un messaggio se l'utente fa un click con il tasto destro del mouse

- JavaScript non supporta il networking e le operazioni sui file, anche se queste ultime trovano parziale supporto con le File API di HTML5
  - esempio: cookies

- Document Object Model (DOM): permette agli script JavaScript di avere accesso ai contenuti del documento HTML in cui sono contenuti

Inserire gli script in pagine web

- Possono apparire sia nell'header di un file HTML che nel corpo, con funzioni molto diverse:
  - header → servono per produrre contenuto su richiesta o si occupano dell'interazione con l'utente. In generale, definizioni di funzioni che vengono riutilizzate più volte
    - Es. codice associato agli elementi di un form
  - body → script da interpretare una volta sola
    - Es. controllo su un dato specifico
- Come nel caso dei CSS, gli script inseriti nell'intestazione vanno inseriti tra commenti

```
<!--
      codice JavaScript
//-->
```

  - Nota: i commenti in JavaScript sono // oppure /* commento */

Esempio script all'interno del body

```
<html>
<head>
  <title>Pagina di Esempio</title>
</head>

<body onLoad="alert('Messaggio di apertura');">
  <p>Pagina di esempio con un alert.</p>
</body>
</html>
```

Browser che non supportano gli script

```
<script type="text/javascript">
  <!--
    codice dello script
  //-->
</script>
<noscript>
    <meta http-equiv=“refresh” content="0;
                                        url=altrapagina.htm">
</noscript>
```

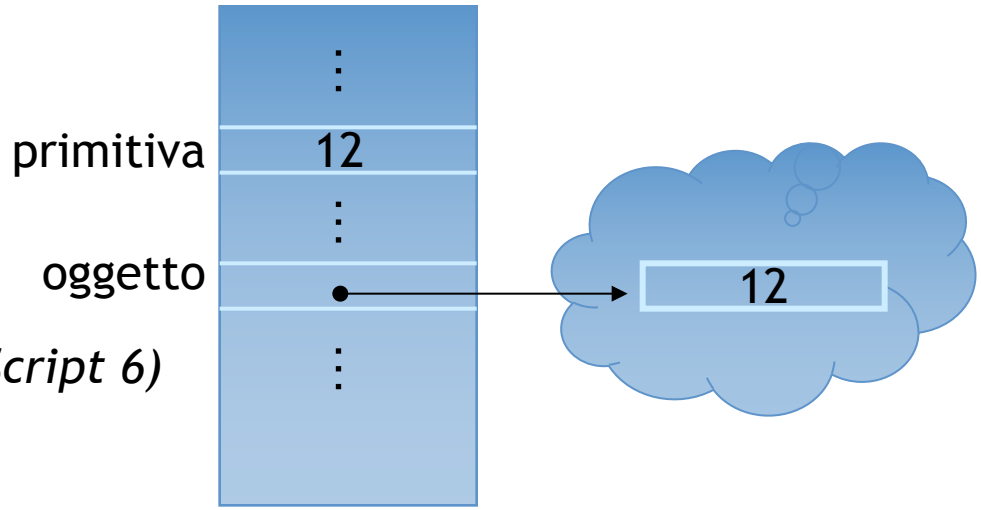
Gli oggetti e variabili JavaScript

- Ogni oggetto ha un insieme di proprietà
  - proprietà di dati
  - proprietà di metodi
- I tipi che non sono oggetti vengono chiamati primitive
- Per riferirsi alle proprietà di un oggetto si usa la forma nome_variabile.nome_proprietà
  - automobile.modello
  - automobile.gira(90)

- I nomi di variabili possono contenere lettere, cifre (non al primo posto), _, $, e non devono essere uguali alle stringhe utilizzate per i comandi (parole riservate)
  - per convenzione non si usano lettere maiuscole e il simbolo $

Tipi, primitive ed oggetti

- number
- string
- boolean
- undefined
- null
- symbol *(a partire da ECMAScript 6)*



- Letterali numerici
  - 12 .12 12.12 … come in PHP
- Letterali stringa
  - “questa è una stringa”, ‘anche questa e\’ una stringa’
- Operatori numerici
  - + - * / ++ --
- Oggetti specifici includono una serie di operazioni e costanti di uso frequente
  - Oggetti Math e Number

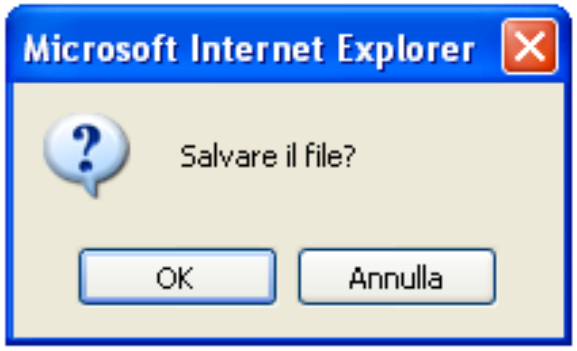
Tipi (2)

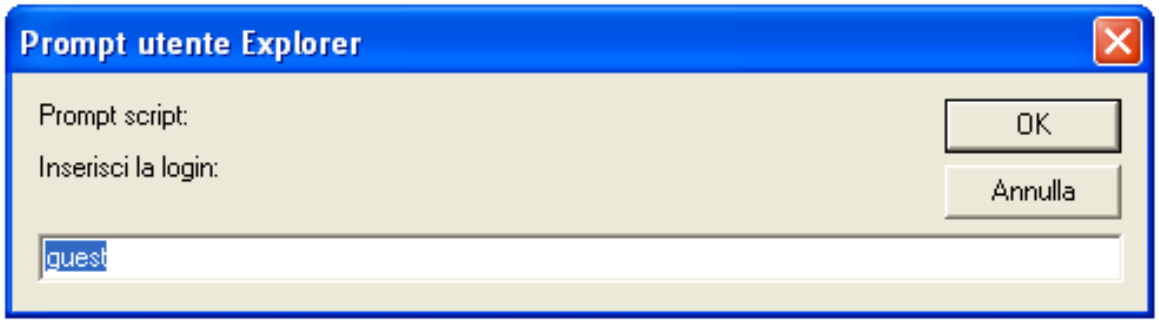
- In JavaScript le variabili possono contenere diversi tipi di dato, come numeri, stringhe e oggetti
- JavaScript supporta la tipizzazione dinamica
  - Esempi:
    - 1 + “Aprile” + 2005
    - 14 * “3”
    - 1 * “Aprile” → NaN
  - quando si somma un numero ad una stringa, JavaScript interpreta il numero come stringa
  - *.toString* converte numeri in stringa, ove è necessario
  - le espressioni sono valutate da sinistra verso destra
    - Es. 2 + 50 + “BC” produce un risultato diverso da “BC” + 2 + 50, i.e. 52BC invece di BC250

Output

- document.write(“ Testo paragrafo ”);
- alert(“Messaggio \n su più righe”);

- var question =
  confirm(“Salvare il file?”);
- input = prompt(“Inserisci la login:”,
  “guest”);





Istruzioni condizionali

- Sono uguali a quelle di PHP, ma non è obbligatorio l'uso dei blocchi
  - if (espressione di controllo) istruzione
  - nome_variabile=(condizione)?valore_se_vero:valore_se_falso
- switch (espressione) {

```
 case valore1:
      …
  case valore2:
      …
 [default:
      …]
}
```

Operatori booleani
&&, and
||, or
!, not

Operatori relazionali
==, !=
===, !==
>, >=
<, <=

Cicli

- while (espressione di controllo){
  //istruzioni del ciclo
  }

- for (inizializzazione; espr. controllo; incremento){
  //istruzioni del ciclo
  }
- do {
  //istruzioni del ciclo
  } while (espressione di controllo)

Creazione e modifica di oggetti

- Quando viene creato, un oggetto è vuoto e privo di proprietà
  - var occhiale = new Object();
- Le proprietà vengono istanziate dinamicamente
  - occhiale.tipo = “solari”;
  - occhiale.marca = “Rayban”;
- Accesso alle proprietà:
  - for (var prop in occhiale)
    istruzione
- Le proprietà possono anche essere eliminate
  - delete occhiale.marca;

```
function marca_occhiale(){document.write(this.marca);}
function occhiale(ntipo,nmarca){
    this.tipo =ntipo; this.marca = nmarca;
    this.print_marca = marca_occhiale;}
```

Array

- Sono oggetti che svolgono alcune funzioni speciali
  - var lista = new Array(1, 2, “tre”, “quattro”);
  - var lista_vuota = new Array(100);
  - var lista_spesa = [“pane”, “latte”, “birra”];
  - lista_spesa[1] → “latte”
  - lista.length → 4
- Altri metodi:
  - lista_spesa.join(“;”); → “pane;latte;birra”
  - lista_spesa.sort(); → [“birra”, “latte”,“pane”];
  - var nuova_lista = lista_spesa.concat(5,6);
  - slice: come substring per le stringhe
  - pop, push, shift, unshift

Array associativi

- JavaScript prevede anche la definizione di array associativi:

```
voti = new Array();
voti["Mario"] = 7;
voti["Gianni"] = 4;
voti["Monica"] = 4;
```

oppure

```
var voti = { "Mario":"7", "Gianni":"4", "Monica":"4" };
```

Funzioni

```
function scriviNome() {
  // inizializzo le variabili all'interno delle funzioni
  var nome=prompt("inserisci qui il tuo nome","il tuo nome");
}

scriviNome();
nome="Gianni"; // in assenza di var questa è una (nuova)
               // variabile globale
```

- I parametri di una funzione possono variare nel numero
  - array arguments

Scope delle variabili

- Le variabili vengono dichiarate con la parola chiave var
  - var x=5, y=7, mese = ‘Aprile’;

- Lo scope di una variabile è legato alle funzioni
  - se definita all’interno di una funzione, indipendentemente da dove è definito lo scope è l’intera funzione
  - Se definito fuori da una funzione la variabile è globale
  - Se una variabile non viene dichiarata (tramite la parola chiave var) questa è automaticamente un variabile globale

Corrispondenza dei pattern

- Ripresi dal linguaggio Perl ma utilizzando i metodi dell'oggetto String
  - search
  - replace
  - I modificatori vengono usati come parametri per i metodi
- Esempi:

Blindspot-S03E09

  - var stringa = "Tecnologie Web";
  - var pos = stringa.search(/c/); → pos = 2
  - stringa.replace(/e/, "E"); → stringa = "TEcnologie Web"
  - stringa.replace(/e/g, "E"); → stringa = "TEcnologiE WEb"
  - var parole = stringa.split(" "); → ["Tecnologie", "Web"]

Introduzione a JavaScript

Lamberto Ballan, Ombretta Gaggi

Università degli Studi di Padova

Tecnologie Web
A.A. 2017/18

Browser Object Model (BOM)

- Il BOM è un modello ad oggetti, non standardizzato e privo di specifica che consente di interagire con il browser

Elemento	Oggetto
Browser	navigator
Finestra	window
Frame	window.frames[“ID Frame”]
Barra indirizzi	location
Barra di stato	status

Metodi BOM

- L’oggetto window rappresenta la finestra (o scheda) del browser (può essere omesso nella chiamata alle sue proprietà o metodi perché usato implicitamente)
- La funzione open permette di aprire una nuova finestra

Apertura di una nuova finestra - 1

- Questa soluzione si trasforma elegantemente perchè se javascript non è abilitato apre il link nella stessa pagina

```
<a href="http://www.example.com/"
onclick="popUp(this.href); return false;">Example</a>
```

```
function popUp(winURL) {
  window.open(winURL,"popup","width=320,height=480");
}
```

```
 window.open(url, nome, lista_di_features);
```

- Per dispositivi touch: onKeyPress

Apertura di una nuova finestra - 2

- Questa soluzione preserva la separazione comportamento – struttura

```
<a href="http://www.example.com/"
                                    class="popup">Example</a>
```

```
var links = document.getElementsByTagName("a");
for (var i=0; i<links.length; i++) {
  if (links[i].getAttribute("class") == "popup") {
    links[i].onclick = function() {
      popUp(this.getAttribute("href"));
      return false;
    }
  }
```

Connessione script – pagina HTML

```
window.onload = linkNuovaFinestra;
function linkNuovaFinestra(){
    var links = document.getElementsByTagName("a");
    for (var i=0; i<links.length; i++) {
      if (links[i].getAttribute("class") == "popup") {
        links[i].onclick = function() {
          popUp(this.getAttribute("href"));
          return false;
        }
      }
    }
}
function popUp(url){
    window.open(url, “nuovaFinestra”, “width=320,height=480”);
}
```

Document Object Model (DOM)

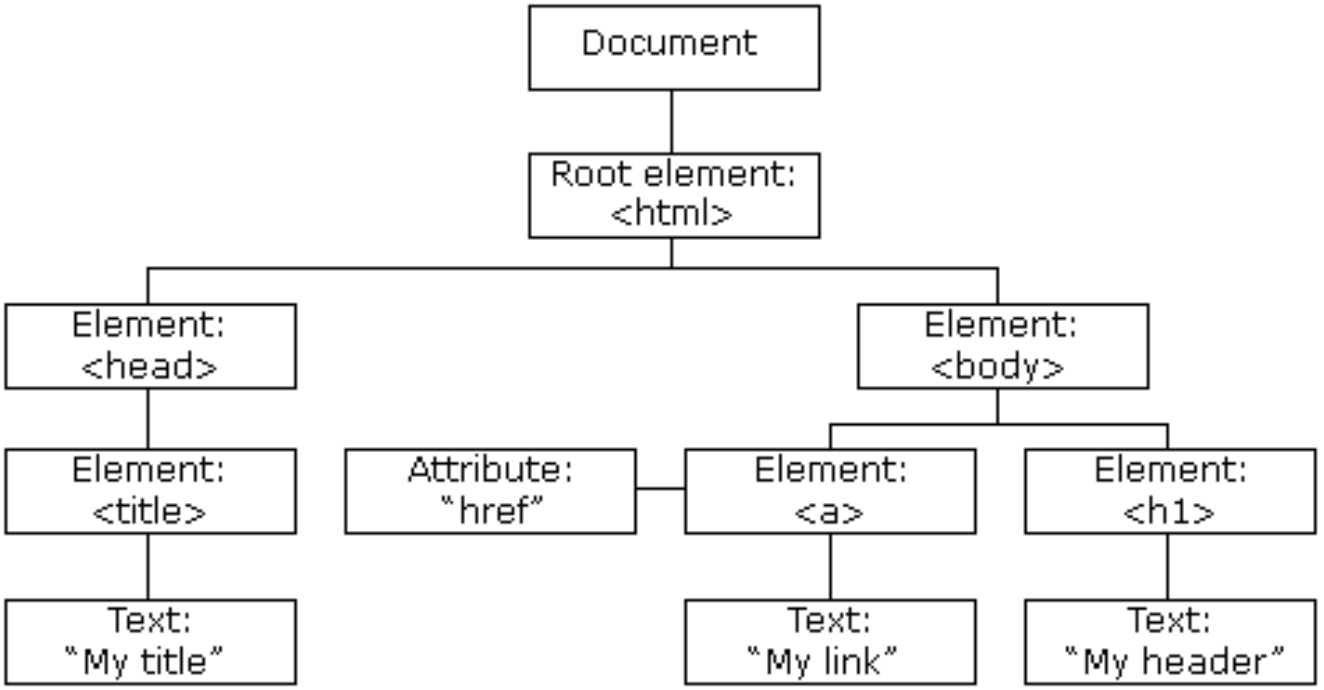
- Il DOM permette di accedere ai diversi elementi di una pagina web. È uno standard, ma il supporto non è ancora completo.
- La pagina è divisa in vari elementi in relazione tra loro

Elemento	Oggetto
Pagina web	window.document o document
Form	document.forms[“ID form”]
Immagini	document.images[“ID immagine”]

http://www.quirksmode.org/dom/core/

Document Object Model (DOM)

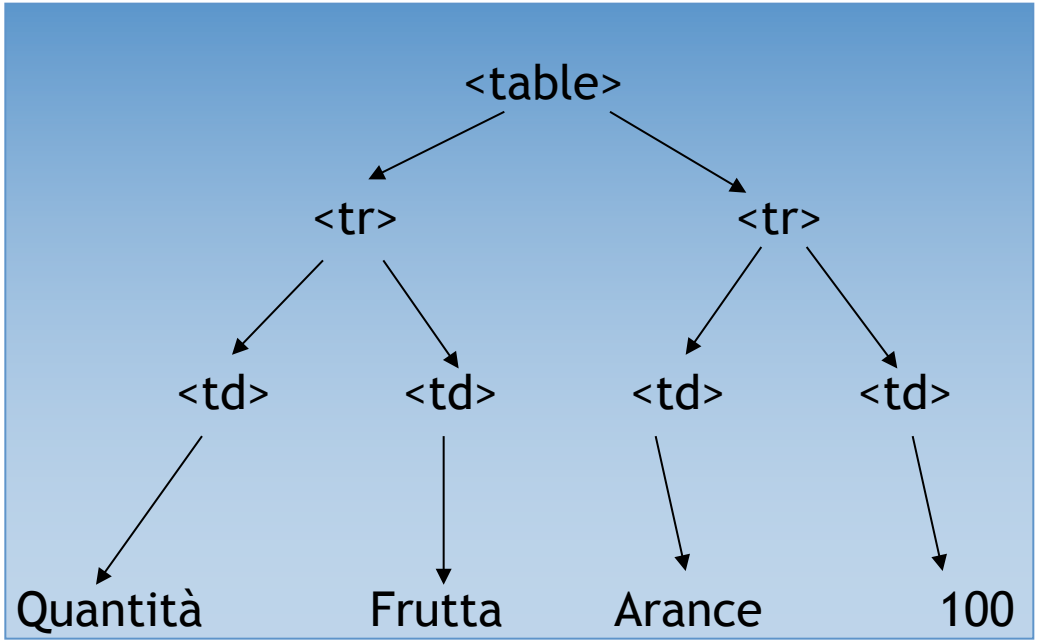
- Il DOM HTML permette di modificare, aggiungere o rimuovere elementi HTML in modo standard.



JavaScript e HTML

- Posso accedere ad un tag tramite getElementsByTagName o getElementById
- Ogni elemento dell'array forms, contiene un array elements con gli elementi del form (pulsanti, caselle di testo, etc.)
- Un documento DOM ha una struttura ad albero

```
<table>
 <tr>
   <td> Frutta </td>
   <td> Quantità </td>
 </tr>
 <tr>
   <td> Arance </td>
   <td> 100 </td>
 </tr>
</table>
```



Proprietà e metodi

- Proprietà
  - x.innerHTML → testo contenuto nell'elemento x
  - x.nodeName, x.nodeValue → nome/valore del nodo x
  - x.parentNode → nodo padre del nodo x
  - x.childNodes → array contenente i figli di x
  - x.firstChild, x.lastChild → primo/ultimo figlio di x
  - x.attributes, x.getAttribute(y) → attributi di x, attributo y
  - x.nextSibling, x.previousSibling
- Metodi
  - x.getElementById(*id*)
  - x.getElementsByTagName(*name*) → restituisce tutti i tag di un certo tipo
  - x.appendChild(*node*) → inserisce un figlio in coda
  - x.removeChild(*node*)

Gli eventi

- Nella programmazione “tradizionale” il codice contiene in sé l’ordine in cui viene scritto

- Nella programmazione ad eventi si specificano funzioni (event handler) da eseguire all’occorrenza di un determinato evento
  - simile alla gestione delle eccezioni:
    - function unload_saluto() {
      alert(“Grazie per aver visitato il nostro sito!”);}
  - event handler o chiamata di funzione su di un attributo specifico:
    - onclick=“alert(‘Stai uscendo da questo sito.’);”

Attributi ed eventi

Attributo	Tag
onabort	<img>
onblur	<body>, <form>, <frameset>, <frame>, etc.
onchange	<input>, <textarea>, <select>
onclick	<a>, <input>
onerror	<img>, <body>, <frameset>
onfocus	<body>, <frameset>, <frame>, <input>, etc.
onload	<img>, <body>, <frameset>
onmouseover	<a>, <area>

Eventi JavaScript

Mouse	Keyboard	Touch
mousedown	keydown	touchstart
mousemove	keypress	touchmove
mouseup	keyup	touchend
mouseover	focus	–
mouseout	blur	–

Esempio: event handler

```
<script type="text/javascript">
    <!--
    function voto(){
        if (document.getElementById('bush').checked)
            alert('Questa mi sembra una buona scelta!');
        if (document.getElementById('kerry').checked) {
            alert('Sei davvero sicuro della tua scelta?');
            document.getElementById('bush').checked = true;
        }
    }
    //-->
    </script>

<h1> Vota per il presidente degli Stati Uniti. </h1>

    <input type="radio" name="vote" value="bush" id="bush" onclick="voto();"/>
    <input type="radio" name="vote" value="kerry" id="kerry"
    onclick="voto();"/>
```

Esempio: controllo dell'input di un form

```
function check() {
    if (document.getElementById('cf').value==""){
        alert('Inserisci il tuo codice fiscale, grazie.');
        document.getElementById('cf').focus(); return false;
        …oppure…
        document.forms['id_form']['cf'].focus();
    }
    var elem = document.getElementById('cf').value;
    var pos =elem.search(/^[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]$/);
    if (pos != 0){
        alert('Codice Fiscale non inserito correttamente; riprova.');
        document.getElementById('cf').focus();
        document.getElementById('cf').select();
        return false;
    } else return true;
  }
…

<input type="text" id="cf" name="cf" onchange="check();" />
```

Pagine dinamiche con JavaScript

- Il linguaggio JavaScript permette di posizionare e dimensionare gli oggetti contenuti nel documento HTML
- Può quindi essere usato per creare dinamicamente il documento in fase di caricamento (onload)
- La modifica dinamica lato client delle pagine web deve avvenire attraverso script non intrusivo

Cosa si può rendere dinamico?

- Sulla base delle:
  - caratteristiche del browser
  - dimensioni della pagina
  - dell'input dell'utente
  - degli spostamenti del mouse e degli eventi in genere

- JavaScript è in grado di modificare
  - la posizione e la dimensione degli elementi
  - le caratteristiche di stile (colore, font disponibili, etc.)
  - il contenuto e la sua struttura
- Oppure può dare dei messaggi di aiuto e/o avviso

Esempio

- Questo <a style = "color:green"
  onmouseover="this.style.color = 'orange';
  this.style.font = 'normal 20pt Verdana'; "
  onmouseout="this.style.color = 'green';
  this.style.font = 'normal 12pt Times';">
  link.
  cambia colore e font quando vi è sopra il mouse.

Nota: questo stesso comportamento si può realizzare utilizzando semplicemente i CSS. In questi casi è in generale **scorretto** utilizzare Javascript!

Accesso agli elementi

```
<form id="form_colore">
   <p id="testo"> Testo che cambia colore</p>
   <input type="button" value="Cambia colore" onclick="colore();" />
</form>
```

- Browser attuali
  - elem = document.getElementById("testo").style
- Esempio di codice per browser molto datati che si può ancora trovare
  - elem = document.form_colore.testo; oppure
  - elem = eval (“document.” + “testo”);

L’oggetto Navigator

- Indica quale browser sta utilizzando l’utente.
  - appName indica il nome del browser
  - appVersion indica la versione
- Esempio:
  - alert(“Il browser usato è: ”+navigator.appName + “\n”
    + navigator.appVersion + “\n”);

- Sapere quale browser si sta usando è utile perché in alcuni casi bisogna predisporre codice diverso per i diversi browser
  - *code forking*: da non utilizzare, ma diventa inevitabile se gli oggetti non sono definiti in modo comune nel DOM

Differenze tra i diversi browser - 1

- Accesso agli elementi
- Ereditarietà degli attributi di carattere
  - Ex. lo stile specificato per il body non viene ereditato dalle tabelle
  - Netscape 4.X disegna i font più piccoli di circa un pixel
- Fogli di stile predefiniti di browser diversi sono diversi

```
<script language="javascript">
    var link='<link rel="stylesheet" type="text/css" href="';
    var css;
    if (ie4)  { css = link + 'ie4.css">'};
    if (ie5)  { css = link + 'ie5.css">'};
    if (ns4)  { css = link + 'ns4.css">'};
    document.write(link+"\n");
</script>
<noscript> <link rel="stylesheet" type="text/css" href="gen.css">
</noscript>
```

Differenze tra i diversi browser - 2

- Offset differenti per il margine sinistro e superiore
  - body {margin-left: 0px; margin-top:0px }
- Differenti risoluzioni
  - In particolare tra MacOS e Windows

- Bug differenti
  - Problema dell'arrotondamento non omogeneo del numero dei pixel in caso di misure relative (Mozilla)

La strada giusta

- Individuare di quale browser si tratta vuol dire richiedere un continuo aggiornamento dello script
- Uno script che testa il supporto al DOM non richiede aggiornamento

```
if (!document.getElementById){
   window.location = http://www.sito.it/altra_pagina.html”;
}
```

Libreria Modernizr

- Il supporto ad HTML5 e CSS3 non è ancora completo da parte di tutti i browser, inoltre c'è il problema dei browser non aggiornati (IE supporta HTML5 a partire dalla versione 9)
- Modernizr è una libreria opensource che aiuta a testare le funzionalità e non il tipo di brower utilizzato

```
if (Modernizr.video) {
      //video supportato
} else { //video non supportato }

function isTagVideoSupported() {
      return !!document.createElement("video").canPlayType;
}
```

Spedire una mail con JavaScript

```
function Email() {
        var email = document.getElementById('email').value;
        var oggetto = document.getElementById('oggetto').value;
        var testo = document.getElementById('testo').value;
        var pos = email.search(/^([\w\-\+\.]+)@([\w\-\+\.]+).([\w\-\+\.]+)$/);
        if (pos != 0) {
            alert('Inserire un indirizzo Email valido!');
            document.getElementById('email').value = '';
            document.getElementById('email').focus();
        }
        else if (testo == '') {
            alert('Il campo \"Messaggio\" è obbligatorio!');
              document.getElementById('testo').focus()
            }
            else {
                location.href = 'mailto:' + email + '?Subject=' + oggetto +
                        '&Body=' + testo;
            }
    }
```

Ordinare un array

```
<button onclick="myFunction()">Ordina in modo ascendente</button>

<p id="demo"></p>



<script>
  var points = [40, 100, 1, 5, 25, 10];
  document.getElementById("demo").innerHTML = points;

  function myFunction() {
        points.sort(function(a, b){return a-b});
        document.getElementById("demo").innerHTML = points;
  }
</script>
```

Attenzione all'uso di JavaScript - 1

```
<script language="text/javascript">
    <!--
    var aiuti = ["Inserisci il nome in questo modo:\n \t nome cognome",
                 "L'email deve avere questa forma: \n \t login@dominio",
                 "La login deve avere almeno 6 caratteri",
                 "La password deve contenere almeno 6 caratteri e deve
    contenere almeno un valore numerico",
                 "Qui ci sono i messaggi di aiuto per compilare la form.
    Metti il mouse sopra un campo per ottenere l'aiuto"];

    function messages(n_messaggio){
        document.getElementById("aiuti").value=aiuti[n_messaggio];
    }
    //-->
</script>
```

Attenzione all'uso di JavaScript - 2

```
<form id="f_email" action="#" >
    <fieldset>
    <label for="nome"> Nome:</label> <input type="text" id="nome"
    onmouseover="messages(0)" onmouseout="messages(4)" />
    <label for="email">Email:</label> <input type="text" id="email"
    onmouseover="messages(1)" onmouseout="messages(4)" />
    <label for="login">Login:</label> <input type="text" id="login"
    onmouseover="messages(2)" onmouseout="messages(4)" />
    <label for="passwd">Password:</label> <input type="text" id="passwd"
    onmouseover="messages(3)" onmouseout="messages(4)" />
    <label for="aiuti">Istruzioni:</label>
    <textarea id="aiuti" rows="4" cols="50" ></textarea>
    </fieldset>
</form>
```

Un esempio più complesso - 1

- Si vuole popolare questa tabella con le informazioni sul browser

```
<table id="table">
        <tr><td>appName</td><td></td></tr>
        <tr><td>appVersion</td><td></td></tr>
        <tr><td>userAgent</td><td></td></tr>
        <tr><td>platform</td><td></td></tr>
        <tr><td>onLine</td><td></td></tr>
        <tr><td>geolocation</td><td></td></tr>
        <tr><td>javaEnabled()</td><td></td></tr>
        <tr><td>cookieEnabled</td><td></td></tr>
</table>
```

Un esempio più complesso - 2

```
function giveInformations(){
        var table = document.getElementById("table");
        var ix;
        var _n = window.navigator; // oggetto navigator
        // array delle info del browser
        var b_infos = [_n.appName, _n.appVersion, _n.userAgent,
                        _n.platform, _n.onLine, _n.geolocation,
                        _n.javaEnabled(), _n.cookieEnabled];
        // tr della tabella
        var trs = table.getElementsByTagName("tr");
        for(ix = 0; ix < trs.length; ix++)  {
                // popola ogni seconda cella
                var td_2 = trs[ix].getElementsByTagName("td")[1];
                td_2.textContent = b_infos[ix];
        }
}
```

I cookie

- I cookie sono piccoli file di testo memorizzati sul computer dell'utente, e scambiati tra client e server, che contengono informazioni salvate dai siti web
- Sono usati per memorizzare in modo permanente delle informazioni univoche rispetto ad un utente, in modo da poterlo riconoscere e/o poterle riusare
  - problemi di privacy
- Ogni cookie contiene dei parametri, tra cui:
  - nome: un nome identificativo per il cookie
  - valore: il valore da memorizzare
  - scadenza (*expiration date*): è opzionale, stabilisce la data di scadenza del cookie, cioè la data dopo la quale questi vengono eliminati dal disco rigido dell'utente

Creazione e distruzione di un cookie

```
// imposta il cookie con nome = valore per la durata di giorni
function setCookie(nome, valore, giorni) {
  var oggi = new Date();
  var scadenza= new Date();
  scadenza.setTime(oggi.getTime() + 24 * giorni * 3600000);
  document.cookie = nome + "=" + escape(valore) +"; expires=" +
                                  scadenza.toGMTString();
}

// rimuove un cookie
function delCookie(nome) {
  setCookie(nome, "");
}
```

Accesso ad un cookie

```
// restituisce il valore del cookie nome
function getCookie(nome) {
  // genera un array di coppie "Nome = Valore" separate da ';'
  var asCookies = document.cookie.split("; ");
  var  stringa="";
  // ciclo su tutti i cookies
  for (var i = 0; i < asCookies.length; i++){
    // leggo singolo cookie "Nome = Valore"
    var info = asCookies[i].split("=");
    if (nome == info[0]) {
      stringa = unescape(info[1]);
    }
  }
  return stringa;}          //stringa="" se il cookie non esiste
```

Utilizzo dei cookie per riconoscere un utente

Utilizzo dei cookie per riconoscere un utente

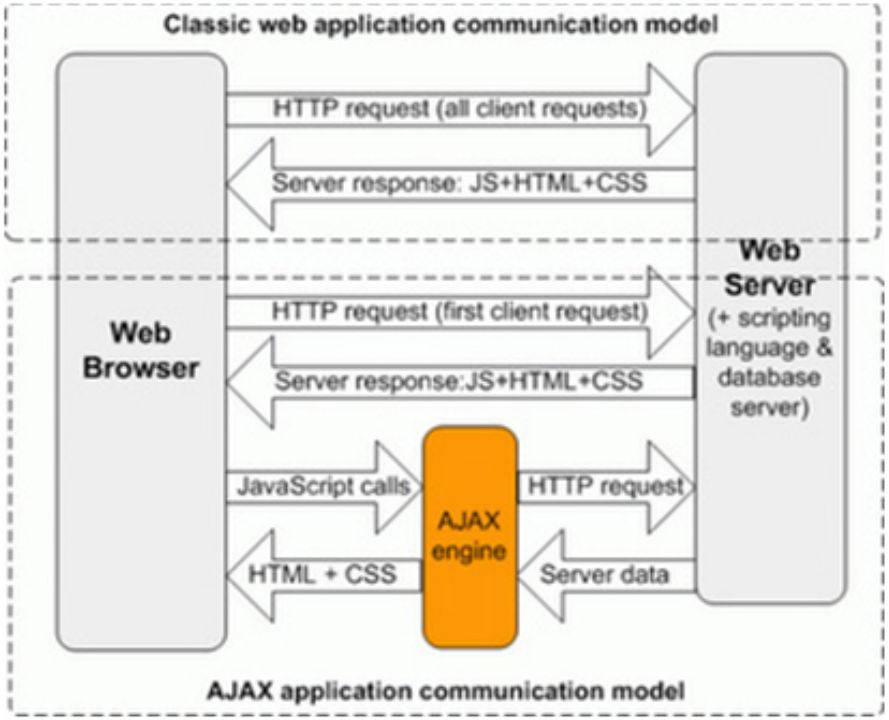
```
function legginome(){
        var nome=prompt("Inserisci il nome");
        setCookie("utente",nome,2);
        leggiutente();
    }
 function leggiutente(){
        var utente=getCookie("utente");
        if (utente!="")  stringa_path="Bentornato " + utente + "...";
        document.getElementById("path").innerHTML =stringa_path;
    }

<body onload="leggiutente();">
…
<a href="javascript:legginome();">Crea un cookie con il tuo utente</a>
```

AJAX

- AJAX è l'acronimo di *Asynchronous JavaScript and XML* e realizza uno scambio asincrono di dati attraverso cui:
  - si può aggiornare una pagina senza doverla ricaricare
  - possiamo inviare dati al server in background
  - invio / ricezione dati dal server avviene dopo la lettura della pagina

- Come funziona AJAX?



AJAX – creazione oggetto XMLHttp

```
function getXMLHttp(){
  var xmlHttp;
  try{
    xmlHttp = new XMLHttpRequest();
  }catch(e){ //Internet Explorer usa un oggetto ActiveX
    try{
      xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
    }catch(e){
      try{
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      }catch(e){
        alert("Messaggio di errore per i vecchi browsers");return false;
      }
    }
  }
  return xmlHttp;}
```

AJAX - esempio

```
function AjaxRequest(){
    var xmlHttp = getXMLHttp();
    xmlHttp.onreadystatechange =
            function(){ //quando l'operazione è completata
                    if(xmlHttp.readyState ==4){
                      inserisciTesto(xmlHttp.responseText);
                    }
            }
   xmlHttp.open("GET", "ScriptPHP.php", true);
   xmlHttp.send(null);
}
function inserisciTesto(response){
   document.getElementById('IDdoveInserire').innerHTML = response;
}
```

Bibliografia

- Siti ufficiale (in inglese)
  - http://www.javascript.com/
  - http://modernizr.com/
  - http://modernizr.com/docs/
- Tutorial W3C
  - http://www.w3schools.com/js/default.asp
  - http://www.w3schools.com/htmldom/default.asp
  - http://www.w3schools.com/xml/ajax_intro.asp
- Tutorial in italiano (datati e non sempre corretti)
  - http://javascript.html.it/guide/leggi/25/guida-javascript-di-base/
  - http://javascript.html.it/guide/leggi/26/guida-javascript-per-esempi/

Ora tocca a voi!

WEB MARKETING HORROR

www.webmarketinggarden.it

Iniziare con sigle in flash così lunghe da far rimorire gli zombie

Non mettere nel sito una mappa

Mettere una pagina "in progress" al posto del sito

Scrivere troppo

Fare un sito senza sapere cosa la gente può volere da te

Non misurare il comportamento degli utenti con un sistema di web analytics

Sottovalutare i tempi di caricamento

Scrivere sciattamente senza buoni titoli

Creare un sito totalmente refrattario ai motori di ricerca

Chiedere agli utenti di riempire troppi form, con troppe voci, troppo presto

Impostare una campagna di email marketing senza una landing page

Non inserire né un telefono, né un indirizzo email per contattarti

Far partire filmati pesantissimi che impallano il computer dell'utente.

Esame: regole e scadenze

- Appelli (scritto): **5/2/2018** e **20/2/2018**
- Consegna progetto: **12/2/2018**
  - preiscrizione obbligatoria alla consegna il 5/2/2018
  - deadline: ore **12:00** (improrogabile)
  - consegna in ritardo: ore **17:00** dello stesso giorno (comporta -4 punti sulla valutazione del progetto)
- Regole per la valutazione e attribuzione del **voto finale**:
  - il voto dello scritto pesa 25% mentre il progetto vale 75%
  - in entrambi i casi la valutazione è nel range [0,…,32]
  - il voto finale è dato dalla media pesata dei due “parziali”
- Bonus Febbraio: chi supera entrambe le parti entro la fine di Febbraio riceverà +2 punti sul voto finale

Esame: regole e scadenze

- Scritto (25%):
  - domande a scelta multipla o con brevi risposte / testi da completare
  - le domande coprono tutto il programma (non saranno comunque eccessivamente mnemoniche)
  - brevi esercizi sulla parte XML Schema, DTD
  - potrebbero essere presenti esercizi (semplici) in cui è richiesto di completare una parte di codice XHTML/CSS seguendo le istruzioni date, o di correggere eventuali errori presenti

Esame: regole e scadenze

- Progetto (75%):
  - le scadenze e le regole per la valutazione sono quelle precedentemente riportate
  - la consegna entro le 12:00 va fatta su Moodle e - con la stessa scadenza - deve essere inserito il progetto **funzionante** sul vostro account sul server *tecweb* (NB: noi correggeremo il materiale inserito su *tecweb*!)
  - dopo le 12:00 non potrete più accedere a tecweb; per sfruttare le 5 ore di ritardo dovete scrivere subito a me, Matteo Ciman e a support@math.unipd.it per farvi riattivare gli account
  - la consegna su Moodle deve contenere (i) file zip con l'intero progetto (ii) il pdf con la relazione (dove deve essere indicato chiaramente nella prima pagina chi fa parte del gruppo e l'indirizzo web del progetto su *tecweb*)

Relazione

- La relazione deve essere sintetica ma completa
  - ci aspettiamo di trovare una documentazione precisa di ciascuna scelta fatta (sia progettuale che tecnologica)
  - deve chiarire eventuali limiti delle soluzioni proposte e – in tali casi – dovete mostrare di essere consapevoli degli effetti delle scelte fatte
  - dovete chiarire (anche qui in modo sintetico ma chiaro) come avete suddiviso il lavoro tra i vari componenti del gruppo e di chi sono stati i vari contributi
  - tranne casi particolari in cui lo valutiamo necessario, non è prevista alcuna discussione orale del progetto