

JavaScript (2)



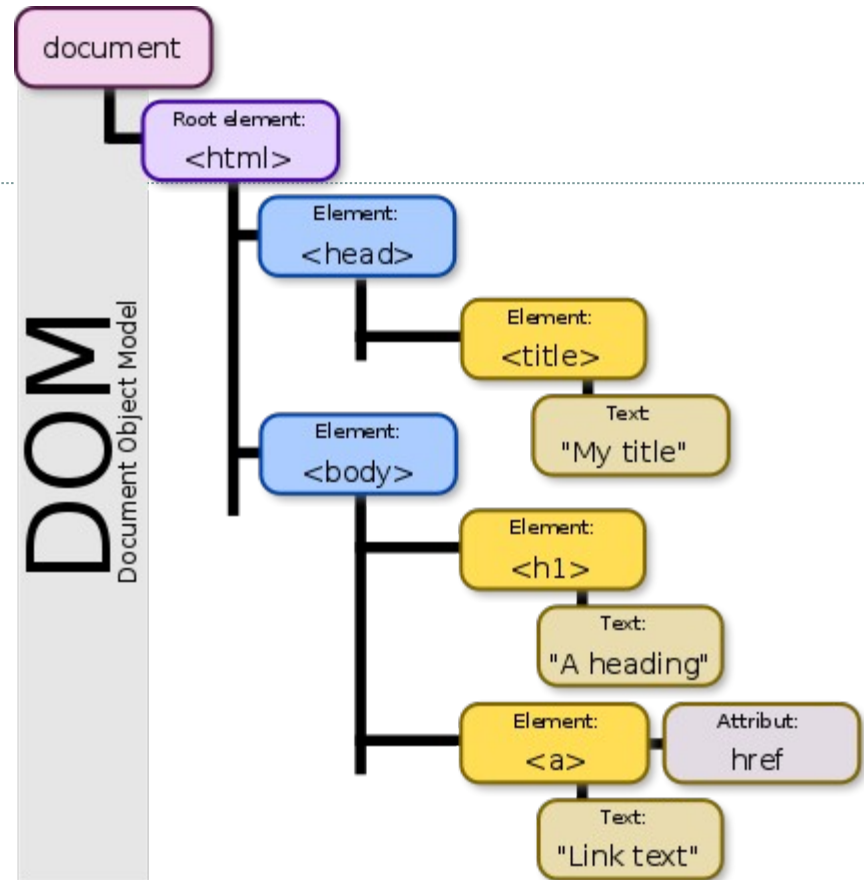
Marina Ribaudò, marina.ribaudò@unige.it

Programmazione lato client

2



Document Object Model



Document Object Model

4

- Standard ufficiale del W3C
- Permette di **rappresentare documenti strutturati** (per esempio HTML e XML)
- **Interfaccia di programmazione** che permette di accedere agli elementi del documento

Document Object Model

5

- Un **documento HTML** visualizzato in una finestra del browser diventa un **oggetto Document**, accessibile attraverso la variabile globale **document**
- Un oggetto Document ha **proprietà, metodi, eventi**
- Ogni **elemento HTML** all'interno della pagina è accessibile in modi diversi

Accesso agli Elementi del DOM

6

- Attraverso una **proprietà** dell'oggetto Document
Esempio: `document.referrer`, `document.cookie`
- Attraverso un **metodo** dell'oggetto Document che permette di selezionare ID, Name, Type, Class, CCS selectors
Esempio: `let el = document.getElementById(id)`
- Come **nodo dell'albero** derivabile dal documento

Albero del DOM

7

- Al DOM può essere associata una **rappresentazione ad albero** i cui **nodi** possono essere
 - il documento stesso, come radice
 - un elemento
 - un attributo
 - un testo
 - un commento
- La struttura dell'albero riflette l'organizzazione degli elementi HTML nella pagina

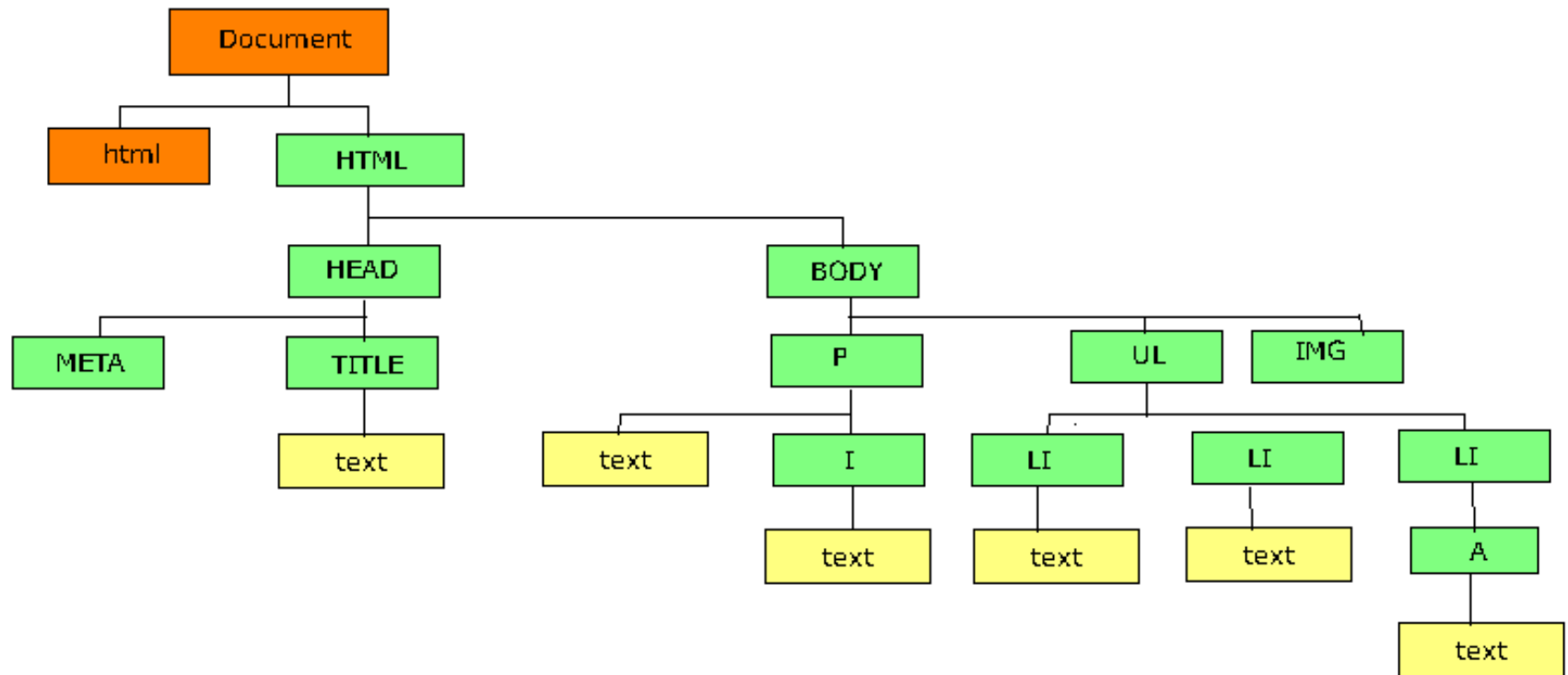
Esempio: pagina HTML

8

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Albero DOM</title>
</head>
<body>
  <p>The DOM tree of this<i>page</i></p>
  <ul id="mylist">
    <li>item 1</li>
    <li>item 2</li>
    <li><a id="mylink" href="http://www.mysite.com">Item 3</a></li>
  </ul>
  
</body>
</html>
```


Esempio: albero dei nodi

9



Accesso al DOM

10

- JavaScript can change
 - all the HTML elements in the page
 - all the HTML attributes in the page
 - all the CSS styles in the page
 - all the events in the page

Nota: il contenuto HTML di una pagina è statico, il contenuto del DOM è dinamico

Oggetto Document: proprietà

11

- Permettono di accedere alle **caratteristiche del documento**, per esempio colore dello sfondo, colore del testo, ... e di modificarle via JavaScript
 - Prova a digitare *document* nella console JS
- Le proprietà degli elementi HTML non si possono ricordare a memoria

Oggetto Document: metodi

12

Si possono usare gli **identificatori** per recuperare elementi con il metodo

`document.getElementById(id)`

Questo metodo restituisce un **referimento all'elemento** che possiede l'identificatore specificato come parametro

```

```

```
let elem = document.getElementById("img1");
```

Oggetto Document: metodi

13

document.getElementsByName(name)

Restituisce tutti gli elementi con attributo name specificato come parametro

document.getElementsByTagName(tagname)

document.getElementsByClassName(classname)

document.querySelectorAll(CSSselector)

Es. let elems = document.querySelectorAll("ul.menu")

id vs name

14

Prima della specifica HTML 4.0 che ha introdotto l'attributo **id**, gli script potevano accedere agli elementi solo attraverso l'attributo **name**

Oggi è facile vedere usati **entrambi** gli attributi, **id** e **name**, inizializzati allo stesso valore, usati per la programmazione sul **client** (**id**) e per l'invio dei dati al **server** (**name**)

Eventi

15

- Gli elementi di una pagina possono “sentire” diversi eventi
- Ad esempio, si può cambiare l’elemento visualizzato nella pagina con un effetto di **rollover**, usando i tag HTML **onmouseover** e **onmouseout**
- Lo stesso effetto si ottiene anche con i CSS!
- **Vedi** imageHTML.html, imageCSS.html

Eventi

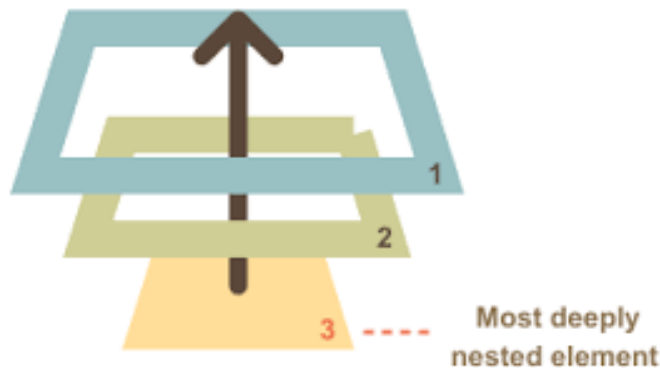
16

- Si possono associare dei **gestori di eventi** (funzioni) agli elementi di una pagina usando il metodo **addEventListener()**
- Ogni volta che si verifica l'evento sull'oggetto cui è stato associato un "listener", viene invocata la funzione corrispondente

Bubbling

17

- Se si verifica un evento su un elemento, viene eseguito il gestore associato all'elemento stesso (se esiste) e poi l'evento viene **propagato verso l'alto** (bubbling) fino a raggiungere la radice dell'albero



Bubbling

18

- Molto comodo per gestire eventi su elementi uguali tra di loro scrivendo il codice una sola volta
- Si possono includere questi elementi in uno stesso elemento padre, per esempio un `<div>`
- Si associa il gestore dell'evento al `<div>` e nel codice si può usare **`event.target`** che identifica sempre **l'elemento che ha causato l'evento**

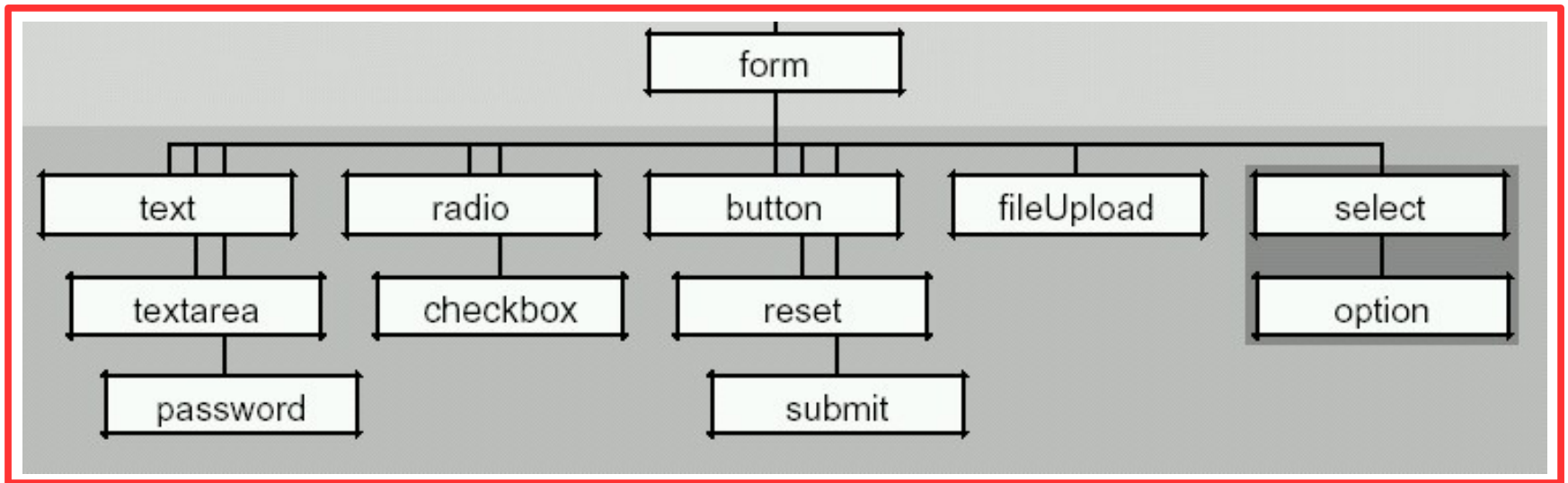
Bubbling

19

- Attenzione:
 - **event.target** è l'elemento “target” che ha causato l'evento e non cambia durante il processo di bubbling
 - **this** è l'elemento “current”, cioè quello che sta eseguendo il gestore di evento
 - Se ci sono più gestori per lo stesso evento, questi vengono eseguiti tutti mano a mano che si sale nella gerarchia del documento
- **Vedi** `imageEvent.html`

Oggetto Form

20



Oggetto Form

21

- È uno degli oggetti più importanti del DOM
- Durante la lettura di un file HTML, viene creato un array con tante celle quanti sono i form all'interno del file

Vedi: http://www.w3schools.com/jsref/dom_obj_form.asp

Oggetto Form

22

```
var myform = document.getElementById("frm1")
```

Proprietà

```
myform.length  
myform.action  
myform.method  
myform.encoding  
myform.name  
myform.target  
myform.elements[ ]
```

Oggetto Form

23

myform.elements[] è a sua volta un array con tanti elementi quanti sono gli elementi del modulo

```
let usr = myform.elements[0];
```

```
let usr = document.getElementById("username");
```

Oggetto Form: eventi

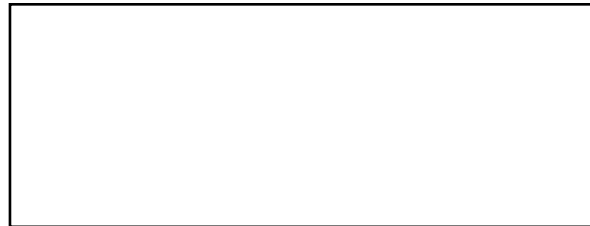
24

text	<input type="text"/>	} change input focus blur
password	<input type="password"/>	
file	<input type="text"/> <input type="button" value="Browse"/>	
checkbox	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	} click focus blur
radio	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	
button	<input type="button" value="Button"/>	

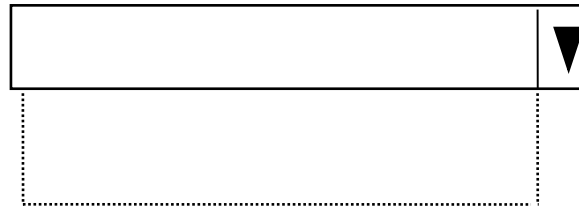
Oggetto Form: eventi

25

textarea

A rectangular box representing a text area input field.

select
(option)

A diagram of a select dropdown menu. It consists of a rectangular box with a downward-pointing triangle on its right side. Below the box is a dotted rectangular outline, indicating the expanded list of options.

**input
change
focus
blur**

hidden

campo nascosto, non sente nessun evento

reset/submit

Invia

**click, focus, blur,
submit, reset**

Oggetto Form: value

26

- Tutti gli elementi di un modulo hanno un valore (**value**)
 - Il valore fornito in input da un utente per un campo di testo
 - Il valore associato ai pulsanti di tipo radio, checkbox, o alle opzioni di un menu (option)
 - l'etichetta di un pulsante

Oggetto Form: value

27

campo di testo con id **txt1**

```
<input type="text" id="txt1" name="txt1"
onchange="alert(this.value);"/>
```

```
<script>
document.getElementById("txt1")
    .addEventListener("change", function() {
        alert(document.getElementById("txt1").value);
    });
</script>
```

Vedi textfield.html

Oggetto Form: checked

28

- I pulsanti radio e checkbox hanno anche la proprietà **checked**

```
var el = document.getElementById("rd1");  
el.checked  
el.value
```

Oggetto Form: option

29

- Il tag `<select>` possiede le proprietà **selectedIndex** e **options[]**

```
let el = document.getElementById("sel1");  
let i = el.selectedIndex;  
alert(el.options[i].value);
```

Oggetto Form: invio

30

- Al momento dell'invio di un modulo si possono fare i controlli mediante codice JavaScript associato agli eventi **submit** e **reset**

```
<form name="frm1" id="frm1"  
method="post" action="..."  
    onsubmit="return checkdata();"   
    onreset="return checkreset();">  
  
...  
...  
</form>
```

- Nota:** **molti controlli oggi si demandano a HTML5!**

Oggetto Form: invio

31

- Se le funzioni **checkdata()** e **checkreset()** restituiscono **false** e nel gestore di eventi si ha **return false**;
- Il modulo non viene inviato (submit), i dati nel modulo non vengono cancellati (reset)

Oggetto Form: invio

32

- Se per gestire il **submit** si associa una funzione usando il metodo **addEventListener()** si può disabilitare l'invio del form usando il metodo **preventDefault()**
- L'**evento viene cancellato** e questo corrisponde a disabilitare il comportamento di default del browser relativamente all'evento stesso
- **Vedi:** login.php

Espressioni regolari

33

- Permettono di descrivere **pattern testuali**
- In JavaScript esiste l'oggetto **RegExp** ma le espressioni regolari possono anche essere create mediante assegnazione

```
let patt=new RegExp(pattern,modifiers);  
  
let patt=/pattern/modifiers;
```

Vedi: http://www.w3schools.com/jsref/jsref_obj_regexp.asp

Espressioni regolari

34

`/[abc]/` “a”, “b”, “c”

`/^a/` tutte le stringhe che iniziano con “a”

`/a$/` tutte le stringhe che finiscono con “a”

`/[a-z]/`

`/[A-Z]/`

`/[0-9]/`

`/[a-zA-Z0-9]/`

`/[a-z]{n}/`

`/[0-9]{n,m}/`

`/[a,b,c]{n,}/`

Espressioni regolari: metodi

35

- `RegExpObject.exec(string)`
- `RegExpObject.test(string)`
- `String.search(/pattern/)`
- `String.match(/pattern/)`

Esempio da stackoverflow

```
function validateEmail(email) {  
    var emailpattern = /^[^<>()[]\.,;:\s@\"']+(\.[^<>()[]\.,;:\s@\"']+)*|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;  
    return emailpattern.test(email);  
}
```