

# Progettazione Web



Alessandra Degan Di Dieco

Analyst @icubedsrl

[alessandra.degan@icubed.it](mailto:alessandra.degan@icubed.it)



# Javascript

Javascript è un linguaggio di **scripting**

- Non è necessaria compilazione
- È semplice testo facilmente leggibile

Javascript è un linguaggio **interpretato\***

- È eseguito direttamente dal browser
- Ha funzioni limitate = maggiore sicurezza

Javascript interagisce con gli **elementi HTML** della pagina

- Buone capacità di interattività e modellazione

\* Le ultime iterazioni dei browser fanno una compilazione "al volo"



# Inserimento del codice nella pagina

Come codice nell'HTML

- `<script><!-- .... --></script>`
- Semplice codice “**occasionale**”

Referenziando un file esterno

- `<script src="javascript.js"></script>`
- **Libreria di funzioni** di uso ricorrente nel progetto

Attenzione all'**ordine di esecuzione** degli script

- Il codice inserito nel tag `<head>` viene eseguito prima degli altri

# Commenti, testi, apici

Il codice può essere **commentato**:

- `/* commento di più righe */`
- `// commento riga singola`

I testi possono essere racchiusi tra **apici singole** ( ' ') o **doppie** ( " ")

- `alert('javascript!'); alert("javascript!");`

I **caratteri speciali** devono essere anteposti da \

- `alert("javascript \= \'scripting\'");`

# Var, Let o Const?

Var, let e Cost permettono di dichiarare una variabile.

**Var** : definisce la variabile sia a livello temporaneo che a livello globale in maniera dipendente alla sua dichiarazione

**Let** : definisce la variabile in maniera coerente al suo contesto di funzionalità.

**Const** : definisce una variabile il cui valore non può essere cambiato.

L'uso di var è deprecato. Si utilizza soprattutto let!

# Funzioni

**Blocco di codice** eseguibile

- ```
function nomeFunzione(parametri) {  
    //istruzioni  
}
```

Chiamata **diretta**

- `nomeFunzione(parametri)`

Chiamata da **link**

- `<a href="javascript:nomeFunzione(parametri)"..`

Le funzioni sono **Case Sensitive**

Attenzione al posizionamento del codice

# Tipi di base

| Tipo      | Descrizione                         |
|-----------|-------------------------------------|
| Object    | Tipo base di tutti gli oggetti      |
| Date      | Rappresenta una data                |
| Boolean   | Rappresenta un booleano             |
| Number    | Rappresenta un numero               |
| String    | Rappresenta una stringa             |
| Array     | Lista di oggetti                    |
| Undefined | Rappresenta un valore non esistente |

# Eventi

| Eventi      | Descrizione                                        |
|-------------|----------------------------------------------------|
| onchange    | Al cambiamento di un elemento HTML                 |
| onclick     | L'utente clicca su un elemento HTML                |
| onmouseover | L'utente passa sopra un elemento HTML con il mouse |
| onmouseout  | L'utente allontana il mouse da un elemento HTML    |
| onkeydown   | L'utente clicca un tasto della tastiera            |
| onload      | Il browser finisce di caricare la pagina           |



# Principali funzioni e proprietà

**typeof:** Verifica del tipo di una variabile

**window:** accesso alla pagina

- open: popup
- opener: accesso alla pagina “chiamante”
- history: storico di navigazione
- location: spostamenti di pagina
- navigator: informazioni del browser dell'utente

**document:** accesso al DOM della pagina

- getElementById
- getElementByName

**alert:** messaggi di avviso

# Javascript – Principali selettori di elementi nell'HTML

Cercare elementi per **Id**:

- `document.getElementById('idElemento');`

Cercare elementi per **classe**:

- `document.getElementsByClassName('nomeClasse')`

Cercare elementi per nome **tag**:

- `document.getElementsByTagName('nomeTag');`

Cercare elementi tramite **selettori CSS (Selectors API)**:

- `document.querySelectorAll("ul li");` → seleziona tutti gli elementi `<li>` in elemento `<ul>`
- `document.querySelector("ul");` → seleziona il primo `<ul>` che trova

# Espressioni condizionali

```
If (variabile == valore){  
    //istruzioni  
}
```

== uguale

!= diverso

> maggiore di

>= maggiore uguale di

< minore di

<= minore uguale di

Operatori logici: &&, ||, !(negazione)

Else

**Operatore ternario:** <espr1> ? <espr2> :<espr3>

# Caratteri speciali per le stringhe

`\n` : nuova riga

`\t` : tab orizzontale

`\b` : backspace

`\r` : ritorno a capo

`\\` : commento

`\'` : apice singolo

`\"` : apice doppio

# Web Storage

Un'applicazione richiede la persistenza di alcuni dati:

- Parametri di configurazione
- Preferenze dell'utente
- Informazioni di accesso ...

Questi possono essere:

1. Mantenuti nel server
2. Salvati come cookie



Local Storage o Session Storage

# Gestione dello Storage

| Metodo              | Descrizione                                                  |
|---------------------|--------------------------------------------------------------|
| Length              | Numero totale di elementi memorizzati                        |
| Key(index)          | Ottenere una chiave in base alla posizione nell'indice       |
| getItem(key)        | Ottenere un valore memorizzato data la chiave corrispondente |
| setItem(key, value) | Salvare un nuovo element con la corrispettiva chiave         |
| removeItem(key)     | Rimuovere element tramite la chiave                          |
| Clear()             | Rimuovere tutti gli elementi precedentemente memorizzati     |

# © 2019 iCubed Srl

La diffusione di questo materiale per scopi differenti da quelli per cui se ne è venuti in possesso è vietata.

iCubed s.r.l. • Piazza Durante, 8 – 20131, Milano  
• Phone: +39 02 57501057 • P.IVA 07284390965

