

JavaScript (1)



Marina Ribaudó, marina.ribaudó@unige.it

Programmazione lato client

2

JavaScript is THE scripting language of the Web

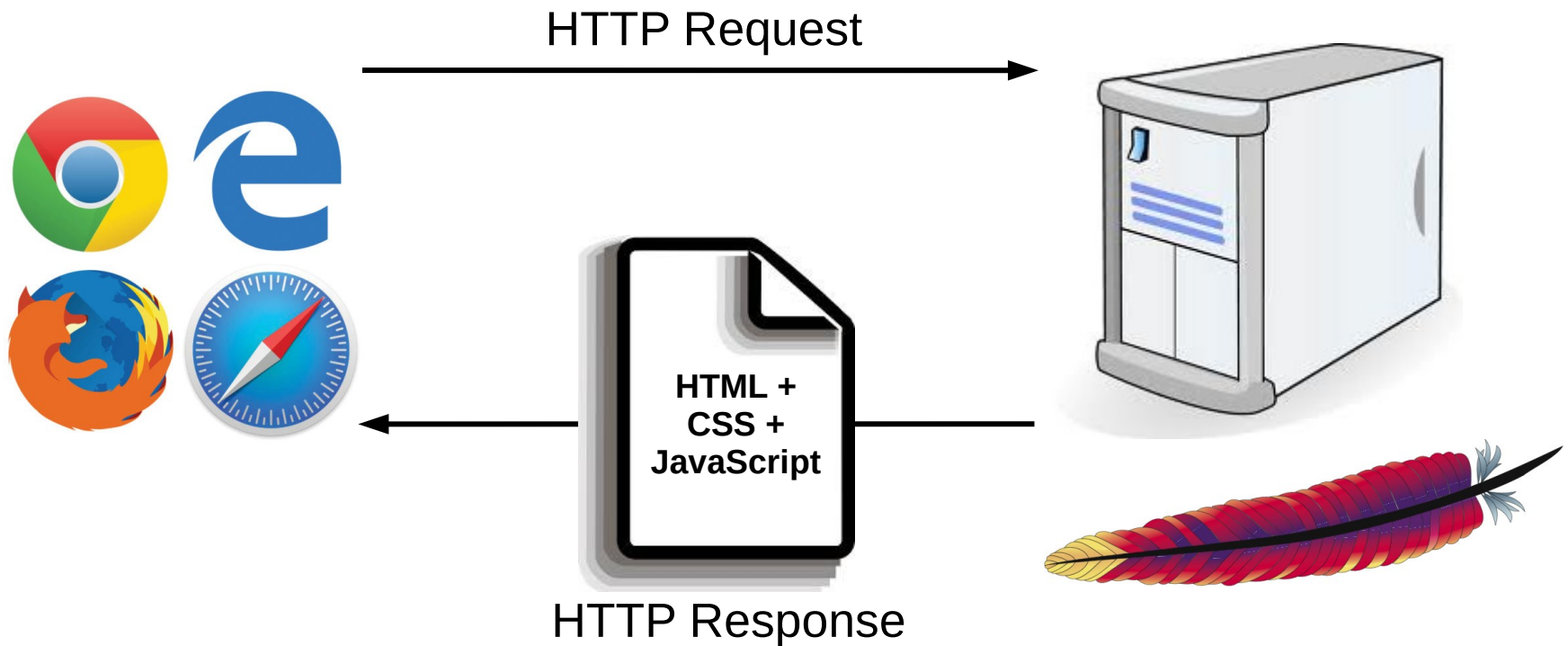


ECMAScript is the official name of the JavaScript standard
(European Computer Manufacture's Association)

<https://ecma-international.org/technical-committees/tc39/>

JavaScript

3



JavaScript

4

variabili
espressioni
istruzioni
funzioni
oggetti e array

core language
ECMAScript



client-side
DOM + BOM

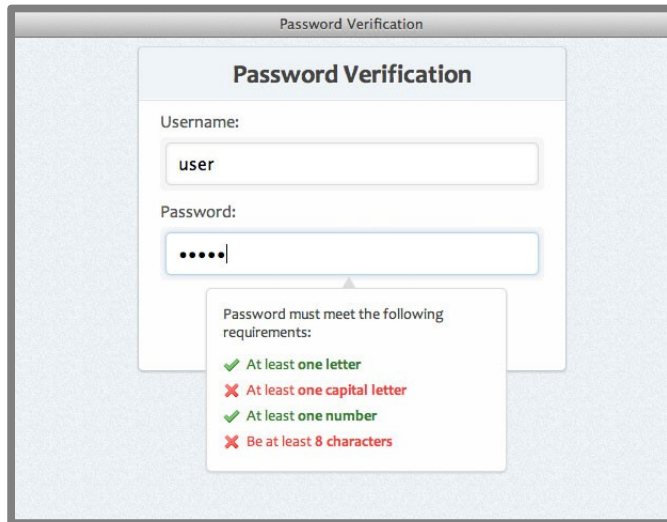
server-side

Per esempio Node.js

Usi di JavaScript

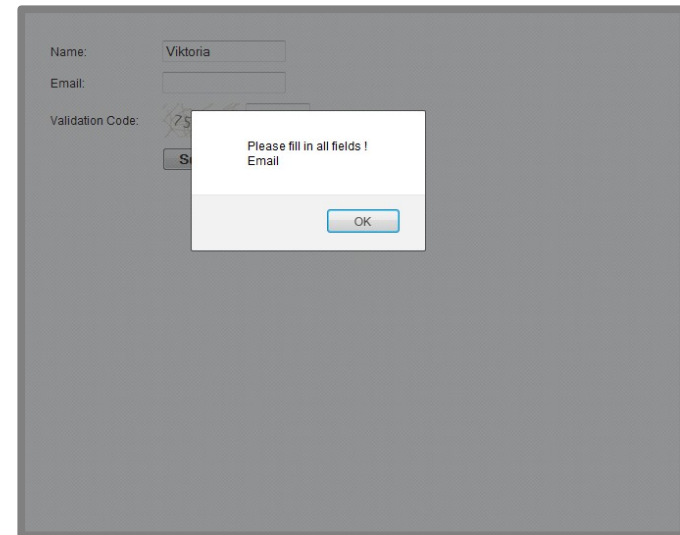
5

- Validazione input
- Apertura nuove finestre



A screenshot of a web form titled "Password Verification". It contains two input fields: "Username:" with the value "user" and "Password:" with masked characters ".....". Below the password field, a message box states: "Password must meet the following requirements:" followed by four items: "✓ At least one letter", "✗ At least one capital letter", "✓ At least one number", and "✗ Be at least 8 characters".

NB: oggi usiamo HTML5

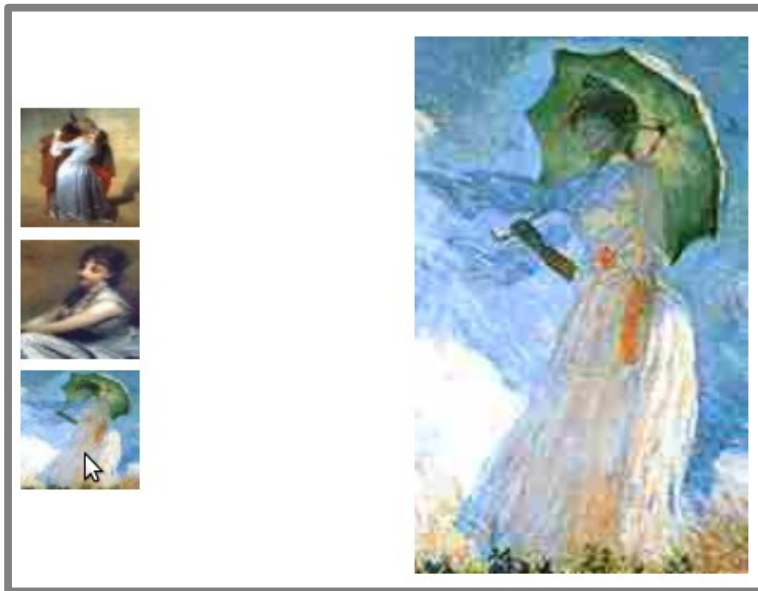


A screenshot of a registration form with fields for "Name:" (filled with "Viktoria"), "Email:", and "Validation Code:". A modal dialog box is displayed in the foreground with the text "Please fill in all fields ! Email" and an "OK" button.

Usi di JavaScript

6

- Rollover, Ajax



Spelling suggestions

Beging typing any word:

- temp
- temper
- tempera
- temperability
- temperable
- temperament
- temperamental
- temperamentality
- temperamentally
- temperaments

NB: oggi usiamo i CSS

Dove si scrivono gli script?

7

- Gli script **sono caricati ed eseguiti dal browser** durante la lettura del codice HTML oppure in risposta al verificarsi di un evento
- **Attenzione:** uno script non può fare riferimento a elementi della pagina (link, immagini, form) che non siano ancora stati definiti

Esempio

```
<script>  
    statement 1;  
    statement 2;  
    statement 3;  
  
    ...  
</script>
```

Dove si scrivono gli script?

8

External



Si scrive un file di funzioni JavaScript che viene incluso nel documento HTML nella parte di <head> o di <body> (spesso al fondo prima di </body>)

```
<script src="myfile.js"></script>
```

- Il codice non è “annegato” all'interno della pagina HTML e la manutenzione è più semplice
- Si può sfruttare il caching del browser

Dove si scrivono gli script?

9

Internal

Si inserisce codice JavaScript (istruzioni e funzioni) nella parte di `<head>` o di `<body>` all'interno dei tag `<script> ... </script>`

Inline

Si inserisce codice JavaScript direttamente nei tag HTML (nel caso di programmazione ad eventi, ma oggi questa scelta è **deprecata**)

Quando vengono eseguiti?

10

- **Dipende...**
 - Quando il **browser carica una pagina**
 - se trova **sequenze di istruzioni** le manda in esecuzione
 - se trova **definizioni di funzioni** ne fa il parsing per controllarne la correttezza sintattica
 - Terminato il caricamento della pagina, quando il **browser “sente” un evento**
 - se trova una funzione/gestore di evento collegati a quell'evento, li esegue

JavaScript: linguaggio interpretato

11

*JavaScript is a lightweight **interpreted programming language**. The web browser receives the JavaScript code in its original text form and runs the script from that.*

*From a technical standpoint, most modern JavaScript interpreters actually use a technique called **just-in-time compiling to improve performance**; the JavaScript source code gets compiled into a faster, binary format while the script is being used, so that it can be run as quickly as possible. However, JavaScript is **still considered an interpreted language, since the compilation is handled at run time, rather than ahead of time**.*

ECMAScript (core language)

12

Definisce la sintassi, i tipi di dato, le istruzioni, le parole riservate, gli operatori, gli oggetti predefiniti, ...

JavaScript: linguaggio dinamico

13

Solo gli **errori sintattici** vengono riconosciuti staticamente prima dell'esecuzione

Gli altri errori vengono riconosciuti a runtime durante l'esecuzione, e non sempre...

I browser mettono a disposizione strumenti per capire cosa sta succedendo...

JavaScript: console JavaScript

14

The image shows a Moodle course page for 'SVILUPPO DI APPLICAZIONI WEB - 65704'. The page includes a navigation bar with 'Course', 'Settings', 'Participants', and 'More'. A section titled 'Benvenuti nel corso di SAW, edizione 2023/24' contains a list of course details: 'Orario delle lezioni: LUN 14:00-16:00, MER 14:00-14:00, aula 505, Via Dodecaneso 35', 'Date degli esami: da definire', and 'Vai al programma del corso sul sito UniGe'. A 'Modalità d'esame' button is also visible. Overlaid on the right is a browser's developer console with the 'Console' tab selected (circled in red). The console shows a message: 'Starting Moodle session timeout warning.' with a link to 'first.js:88'.

SVILUPPO DI APPLICAZIONI WEB - 65704

Course Settings Participants More ▾

✓ **Benvenuti nel corso di SAW, edizione 2023/24** [Collapse all](#)

- **Orario delle lezioni:** LUN 14:00-16:00, MER 14:00-14:00, aula 505, Via Dodecaneso 35
- **Date degli esami:** da definire
- **Vai al programma del corso sul sito UniGe**

Modalità d'esame

Starting Moodle session timeout warning. [first.js:88](#)

JavaScript: struttura lessicale

15

Le istruzioni sono separate tra loro dal carattere ;
(obbligatorio solo per più istruzioni sulla stessa riga, ma suggerito)

Le parentesi graffe {...} permettono di definire i blocchi di istruzioni

Commenti

es. *// questo è un commento*

es. */* questo è un commento */*

es. *<!--questo è un commento*

JavaScript: struttura lessicale

16

Gli identificatori possono iniziare con `_` `$` o con una **lettera** e non possono iniziare con un numero

Non si possono usare le parole riservate

JavaScript è **case-sensitive** `var a != var A`

JavaScript: tipi di dato

17

JavaScript è un linguaggio **tipato dinamicamente**, la stessa variabile può essere usata per memorizzare dati di tipo diverso

```
i = "Hello world";
```

```
i = 1;
```

```
i = true;
```

JavaScript: tipi di dato

18

Tipo	Descrizione	Esempi
<code>undefined</code>	Valore non inizializzato	
<code>boolean</code>		<code>true</code> , <code>false</code>
<code>number</code>	64bit, floating point	Infinity (es. <code>5/0</code>), NaN (es. <code>0/0</code>), <code>Number.PI</code>
<code>string</code>	Sequenza di caratteri	<code>"Hello world"</code>
<code>symbol</code>	Chiave univoca per una proprietà	<code>var key = Symbol("k")</code>
<code>object</code>	Oggetto oppure <code>null</code>	

JavaScript: array

19

- In molti linguaggi gli array sono collezioni di dati omogenei accessibili mediante indice
- In JavaScript (come in PHP) gli elementi degli array possono essere di tipo diverso

```
var empty = [ ];
```

```
var numbers = [ 'zero', 'uno', 'due', 'tre', 'quattro' ];
```

```
var misc = [ 'ciao', 10, true, [ 'M', 'F' ], NaN ];
```

JavaScript: oggetti

20

Gli oggetti hanno **proprietà** definite come associazioni di **chiavi** e **valori**

```
var person = {  
  firstname: "Alice",  
  lastname  : "Smith",  
  email     : "alice@email.com"  
};
```

```
var empty_object = {};
```

JavaScript: oggetti

21

Si può accedere alle proprietà con

- **dot notation** “.”

```
a = object_name.property_name;
```

- **mediante chiave**

```
a = object_name['property_name'];
```

Se il nome della proprietà non è un identificatore valido, per es. contiene uno spazio oppure altri caratteri speciali, è indispensabile la seconda notazione:

```
myObject.first name           // non valido  
myObject['first name'] = 'John'; // ok
```

JavaScript: oggetti

22

Si possono aggiungere/rimuovere proprietà dinamicamente

Se si accede ad una proprietà non definita, il valore di ritorno è `undefined`

JavaScript Object Notation

23

Meglio nota come **JSON**, indica un formato testo usato per lo **scambio dei dati**

La **sintassi** è molto **simile** a quella degli **oggetti** espressi nella notazione Object Literal

I nomi delle **proprietà** devono sempre essere scritti tra **doppi apici**

... ne riparlermo...

JavaScript: classi

24

```
class Person {  
  constructor(name) { this.name = name }  
  
  sayHello() { return "Ciao, " + this.name + "!" }  
}  
  
let p = new Person("Nicolò");  
p.sayHello(); // restituisce "Ciao, Nicolò!"
```

Le classi si definiscono con **class**, hanno un metodo chiamato **constructor** che viene eseguito una sola volta quando si crea una istanza, per inizializzare le sue proprietà

I metodi seguono le regole delle funzioni

JavaScript: Map

25

- Tipo di dato built-in che consente di memorizzare coppie **chiave-valore**
- Mantiene l'ordine delle coppie chiave-valore in base all'ordine di inserimento
- Le chiavi sono univoche
- Esistono metodi per leggere, scrivere, rimuovere valori, data la chiave
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

Oggetti built-in: Date

26

Offre proprietà e metodi per lavorare con tempo e date

```
var oggi = new Date();  
  
var a = oggi.getDate();  
var b = oggi.getDay();  
var c = oggi.getFullYear();  
var d = oggi.getHours();  
  
var e = oggi.getTime();  
restituisce i millisecondi trascorsi  
dalla mezzanotte  
del 1/1/1970
```

Oggetti built-in: Math

27

Offre proprietà e metodi per le operazioni matematiche

```
var a = Math.min(x, y);  
var b = Math.sqrt(x);  
var c = Math.abs(x);  
var d = Math.random();
```

```
var  $\pi$  = Math.PI;  
var e = Math.E;
```

Oggetti built-in: String

28

Offre proprietà e metodi per manipolare testo

```
var a = "Hello world";  
var a = 'Hello world';  
  
var b = a.length;  
var c = a.charAt(n);  
var d = a.indexOf(substr);  
var f = a.replace(espr1, espr2);  
  
// restituisce una sottostringa  
var e = a.slice(start, end);  
  
// converte una stringa in un array  
var g = a.split("separator");
```

JavaScript: variabili

29

In passato non era obbligatorio dichiarare le variabili, ma questa pratica oggi è deprecata: se il codice è in **strict mode** (`"use strict"`) genera un errore

A partire da ECMAScript 2015 questa direttiva è sottintesa in molti nuovi costrutti

JavaScript: variabili

30

Per dichiarare una variabile si usano **var**, **let**, o **const**

- **var** declares a **global** or **local** variable, optionally initializing it to a value
- **let** declares a **block scope local** variable, optionally initializing it to a value
- **const** declares a **read-only** named constant

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements>

JavaScript: funzioni

31

```
function fName(param1, param2) {  
    // some JavaScript code to be executed  
    return result;  
}  
  
function square(x) { return x ** 2; }  
var a = square(4);
```

Le variabili dichiarate all'interno della funzione sono **locali**

Le variabili che vengono usate (ma non dichiarate) all'interno di una funzione sono **globali**

JavaScript: eventi

32

Examples of events

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

JavaScript: eventi

33

load
unload

change
reset
submit

blur
focus

click

mousedown
mouseup
mouseover
mouseout

keydown
keypress
keyup

<https://developer.mozilla.org/en-US/docs/Web/Events>

JavaScript: eventi

34

- Ad ogni **evento** è associato un **elemento HTML** corrispondente (inizia con **on** e poi segue il **nome dell'evento**)
- Se nel file HTML esiste del codice JavaScript associato a questo elemento HTML, il codice viene eseguito quando si verifica l'evento

Esempio (oggi deprecato)

```

```

JavaScript: input/output

35

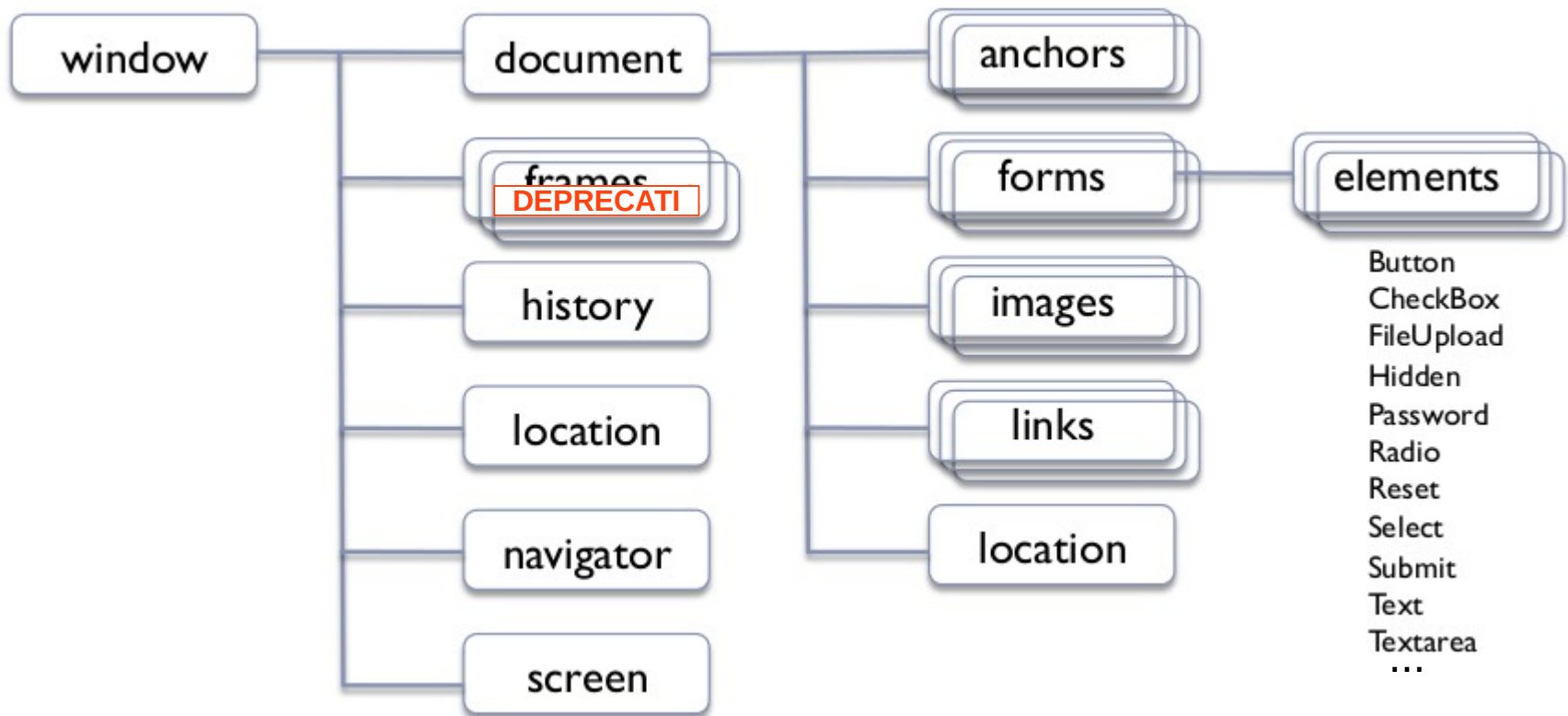
Come nel C, JavaScript **non ha propri costrutti di input/output**; mentre il C si affida alle librerie I/O standard, un interprete JavaScript si basa su un **programma ospite** in cui è integrato

Se integrato in un browser Web, JavaScript si collega alle applicazioni tramite interfacce chiamate **BOM** (Browser Object Model) e **DOM** (Document Object Model)

Anticipiamo l'oggetto **window** (BOM) che permette di vedere alcuni aspetti dell'input/output

JavaScript BOM+DOM

36



Oggetto Window

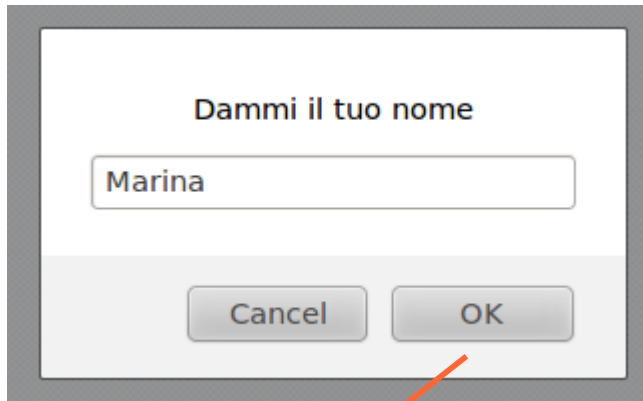
37

- **Rappresenta il browser**
- Permette di realizzare
 - operazioni di input/output
 - aprire nuove finestre (popup)
 - chiudere finestre
 - ...

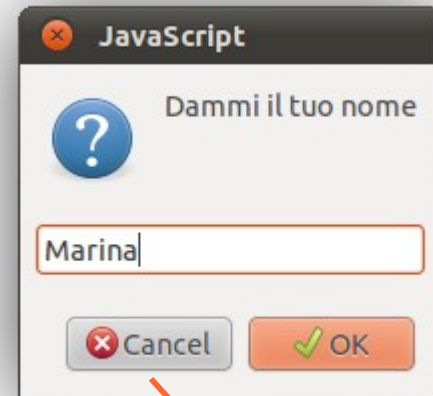
JavaScript: prompt

38

```
window.prompt("Dammi il tuo nome", "")
```



valore in input

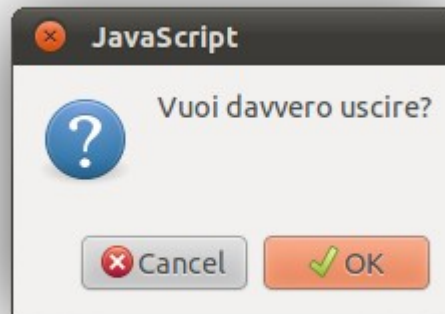


"" (null)

JavaScript: confirm

39

```
window.confirm("Vuoi davvero uscire?")
```



JavaScript: alert

40

`window.alert("Hai restituito ...")`



Altri oggetti del browser

41

- Navigator

descrive il **browser** che sta facendo la richiesta HTTP

- Location

descrive la **barra degli indirizzi** e può essere usato per operazioni di redirect verso nuove pagine

- Screen

descrive lo **schermo** dell'utente

Uso della console JS

42

