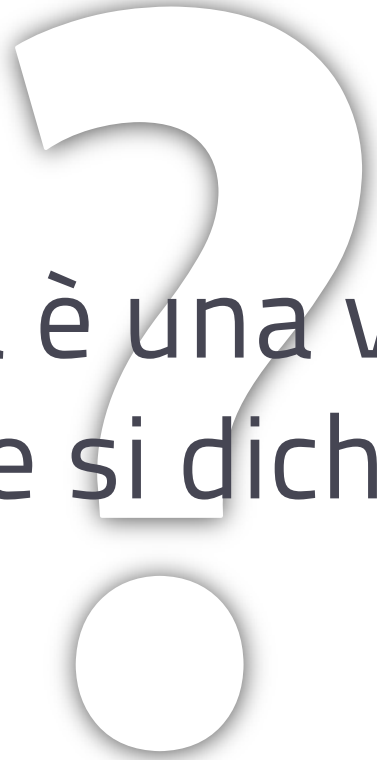
LEZIONE 5



Che cosa è una variabile e come si dichiara ?

DEFINIRE E/O INIZIALIZZARE UNA VARIABILE

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

Che cosa è una funzione e come la definisco?

DICHIARARE E DEFINIRE UNA FUNZIONE

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

funzione con nome

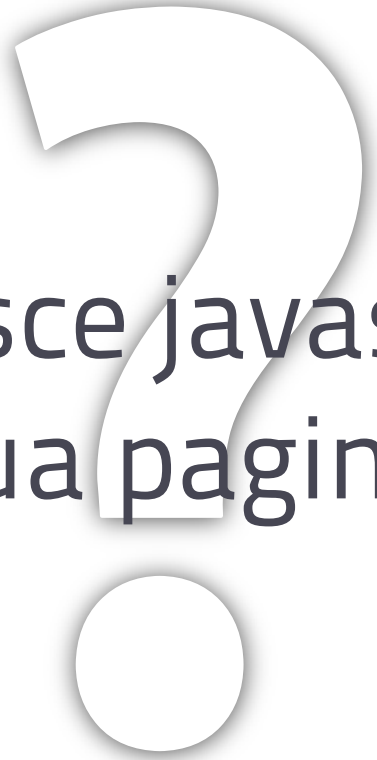
DICHIARARE E DEFINIRE UNA FUNZIONE

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

funzione anonima

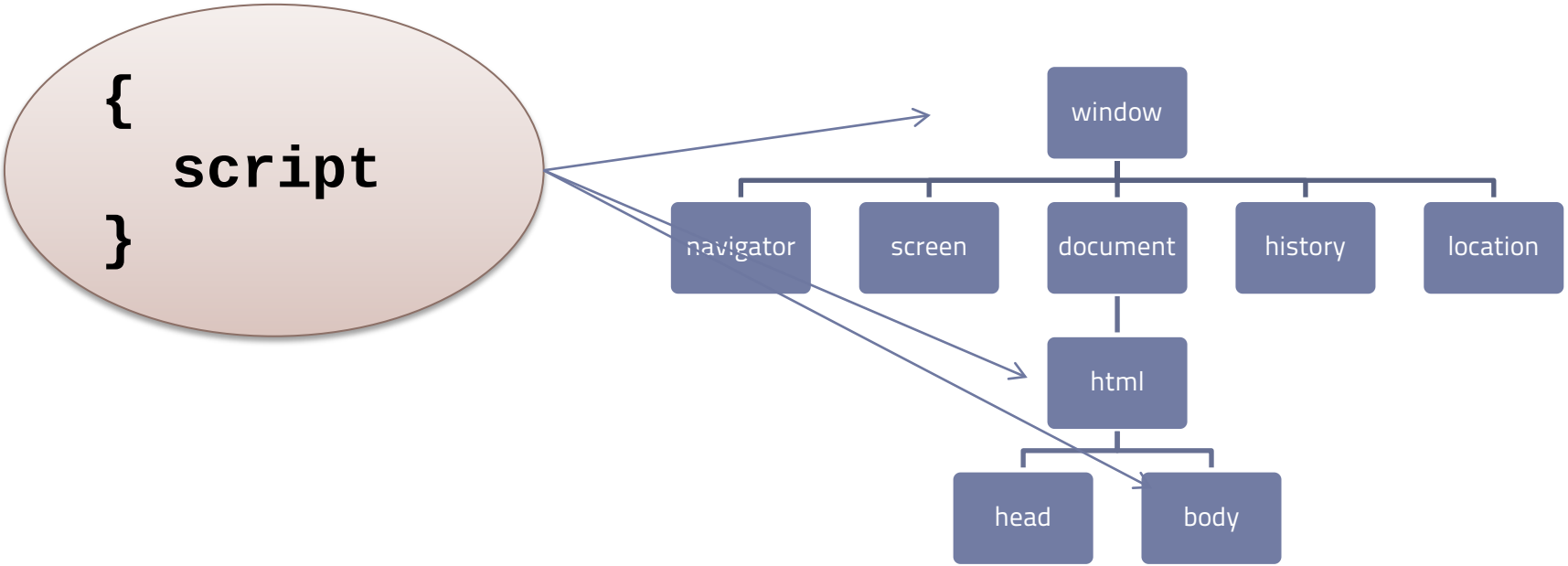
Che regole deve rispettare il nome di una funzione o di una variabile?

1. Iniziare il nome con una lettera (A-Z o a-z) l'underscore (_) o il segno del dollaro ($).
2. Continuare con un numero qualsiasi di lettere, numeri, "_" O "$".
3. Javascript è case sensitive.



Come agisce javascript sulla tua pagina?

JAVASCRIPT AGISCE SUL DOM



Perché javascript possa agire sugli oggetti il documento deve essere completamente caricato !

Come procedo? E perché?

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

A cosa serve il ";"?

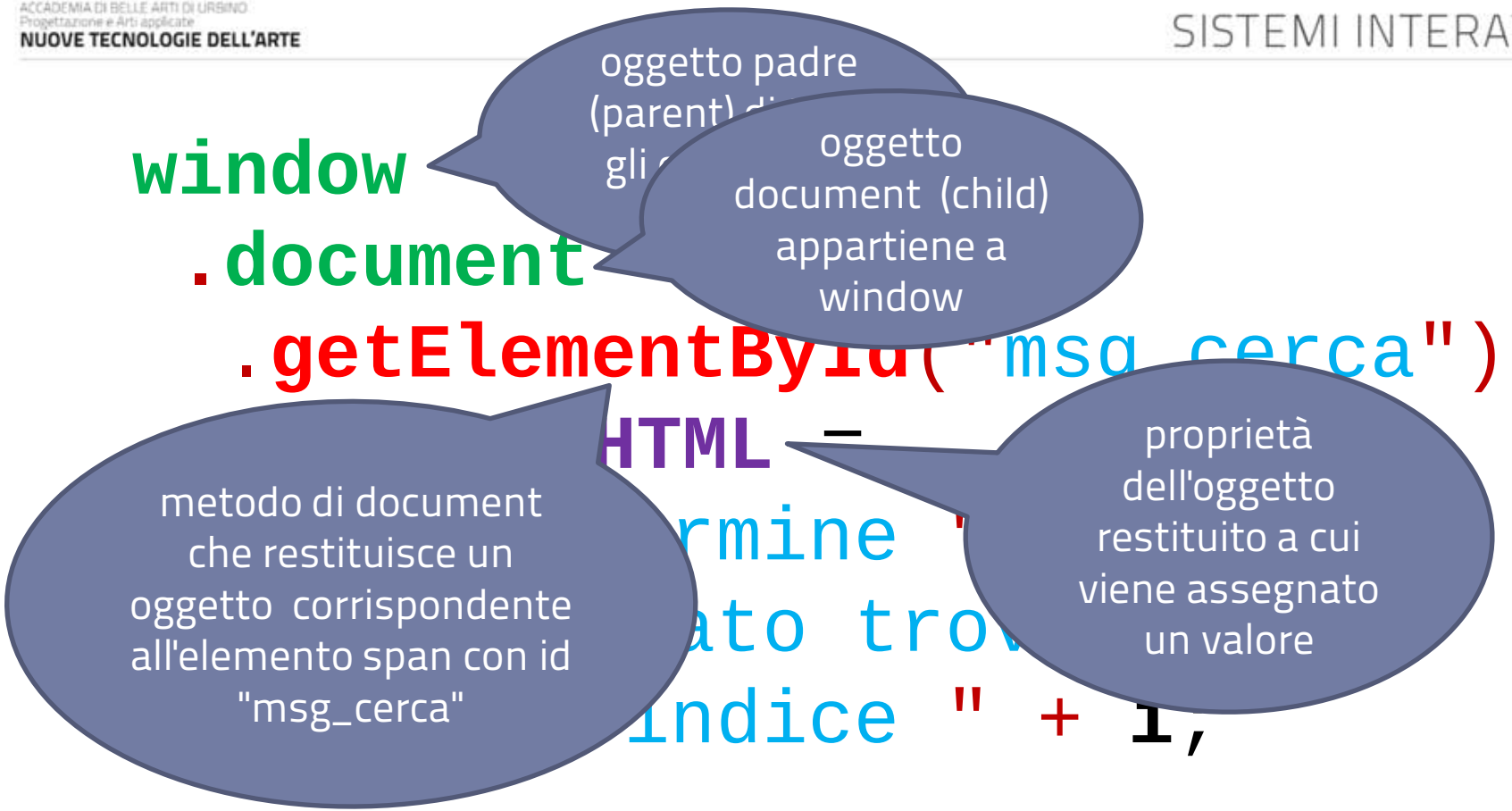
```
var nome;
var oggi;
```

- Il punto e virgole è il segno che separa le istruzioni fra loro.
- Le istruzioni NON sono separate dalla fine riga.

A cosa serve il "."?

```
if (str.length < 2)
```

- Il punto è l'operatore di appartenenza indica che la proprietà o il metodo che si trova alla sua destra appartiene all'oggetto che si trova alla sua sinistra.
- Gli operatori punto possono essere usati in catena.

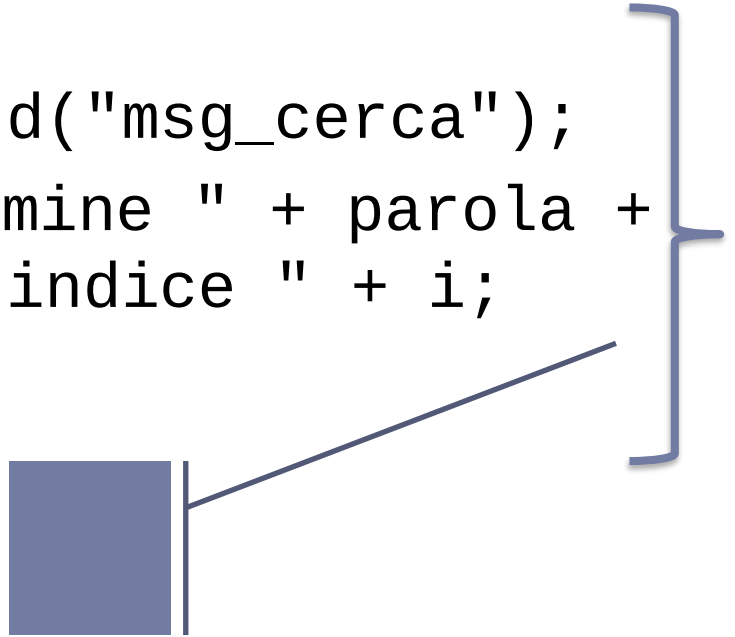

window
.document
.getElementById("msg_cerca")
HTML =
rmine "
ato trov
ndice " + i;
oggetto padre
(parent) di
gli
oggetto
document (child)
appartiene a
window
metodo di document
che restituisce un
oggetto corrispondente
all'elemento span con id
"msg_cerca"
proprietà
dell'oggetto
restituito a cui
viene assegnato
un valore

A cosa servono le parentesi graffe ?

{ . . . }

- Una coppia di { } definisce un blocco di istruzioni che vengo eseguite insieme, prima di proseguire con lesecuzione del programma.

```
if (mesi[i] == parola)
{
  var campo =
     document.getElementById("msg_cerca");
  campo.innerHTML = "Il termine " + parola +
     " è stato trovato all'indice " + i;
  return;
}
```



Uso delle parentesi tonde: Quale è la differenza tra

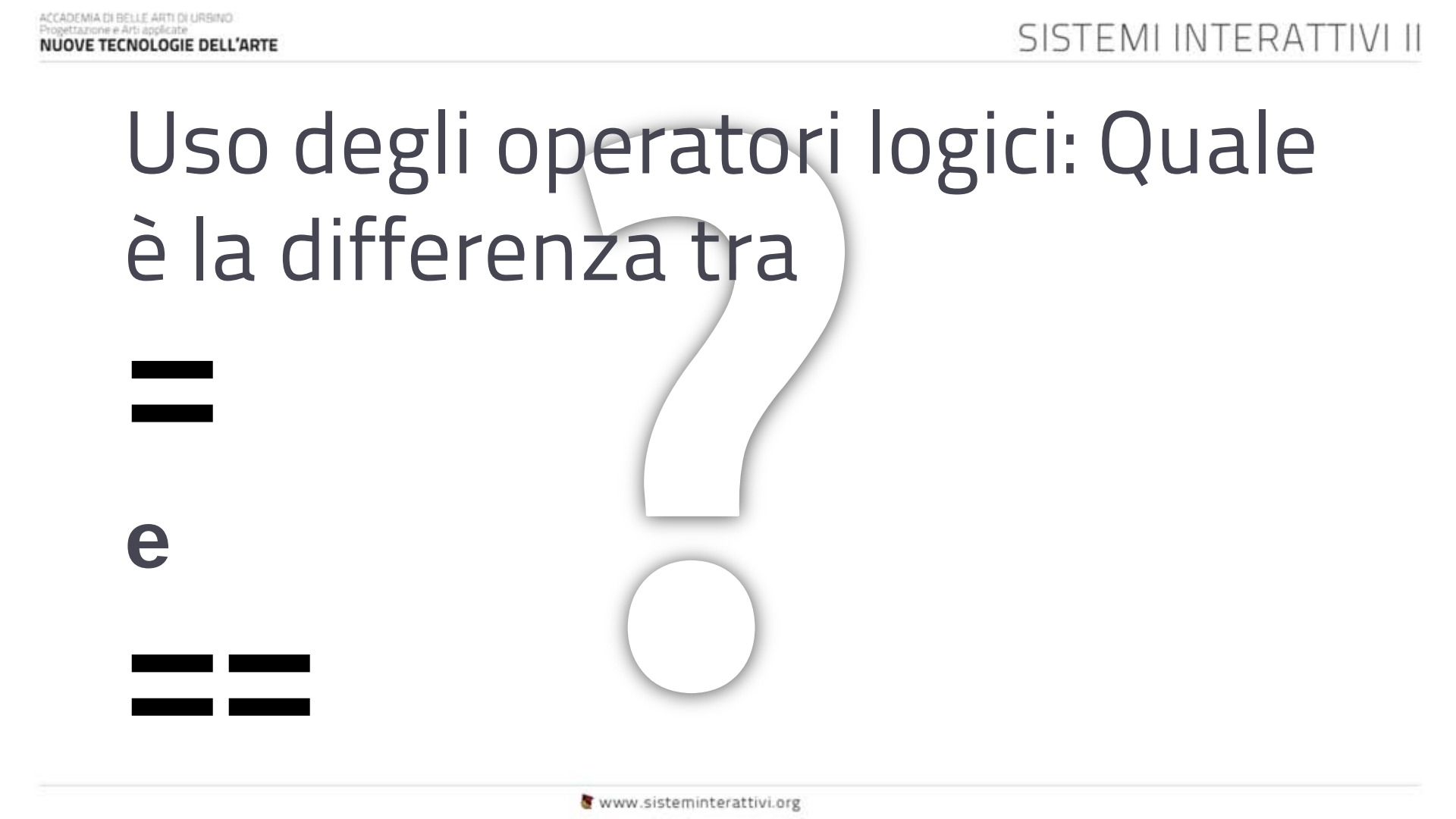
```
somma;
```

e

```
somma(5, 8);
```

- **`somma`** (che è il nome che ho dato alla funzione) è la **variabile** che contiene le **istruzioni** di cui la funzione è composta. Il valore rappresentato da **`somma`** è la funzione stessa, il **blocco di script** di cui è costituita.
- se faccio seguire a **`somma`** le parentesi tonde, tra le quali inserirò la lista dei parametri se previsti, la funzione viene eseguite e il valore di **`somma(…)`** sarà il valore restituito dell'istruzione **`return`** se presente altrimenti sarà **undefined**.

Uso degli operatori logici: Quale è la differenza tra

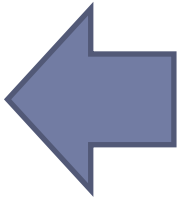


=

e

==

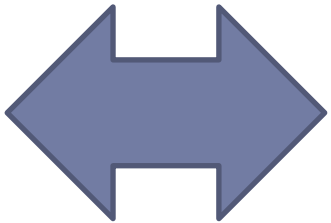
- = è l'operatore di assegnazione. Assegna il valore che si trova alla sua destra alla variabile o alla proprietà che si trova alla sua sinistra:



```
a = 10;
```

- il risultato di questa espressione è 10;

- == è un operatore di confronto. Confronta i due valori (costanti, variabili od espressioni) che unisce. Il risultato dell'espressione è true se i due valori sono uguali altriementi false.



a == b;

Per quali valori di **a** e di **b** questa espressione dà come risultato true ?

a > b || b == 5

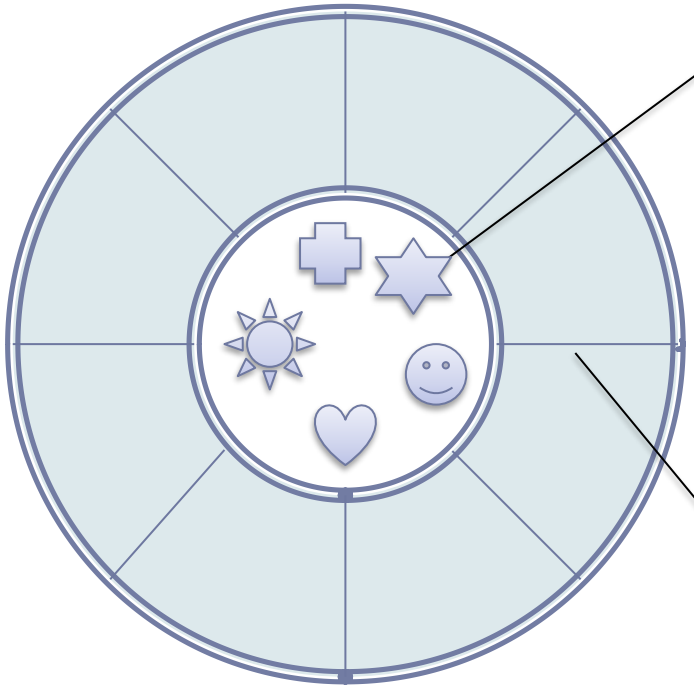
e questa ?

a > b && b == 5

OGGETTI

- l'oggetto è un esemplare (in inglese: ***instance***, comunemente anche se impropriamente tradotto con istanza) di una classe.
- Ogni istanza è separata dalle altre, ma condivide le sue caratteristiche generali con gli altri oggetti della stessa classe.

OGGETTO

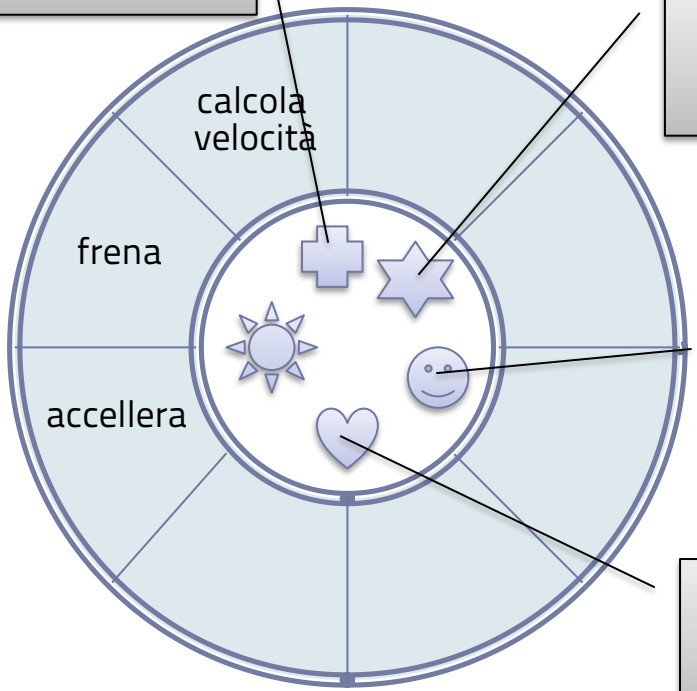


stati
(variabili)

comportamenti o
metodi
(funzioni)

BICICLETTA

diametro ruota



colore

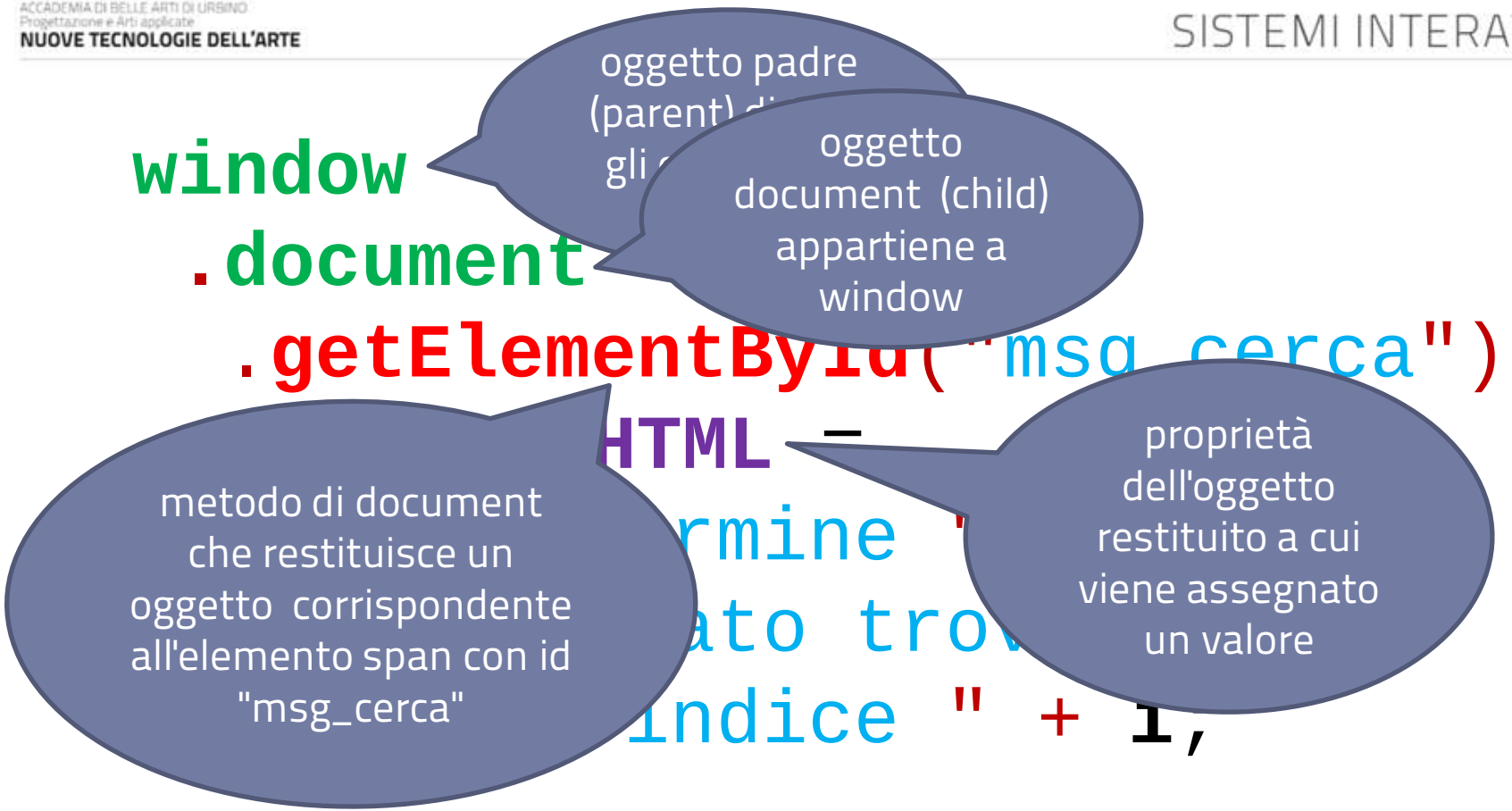
pedalate / minuto

rapporto

Modello mountain-bike

- Telaio alluminio
- Moltiplica anteriore a 3 rapporti
- Cambio a 8 rapporti
- Freni a disco
- ecc...

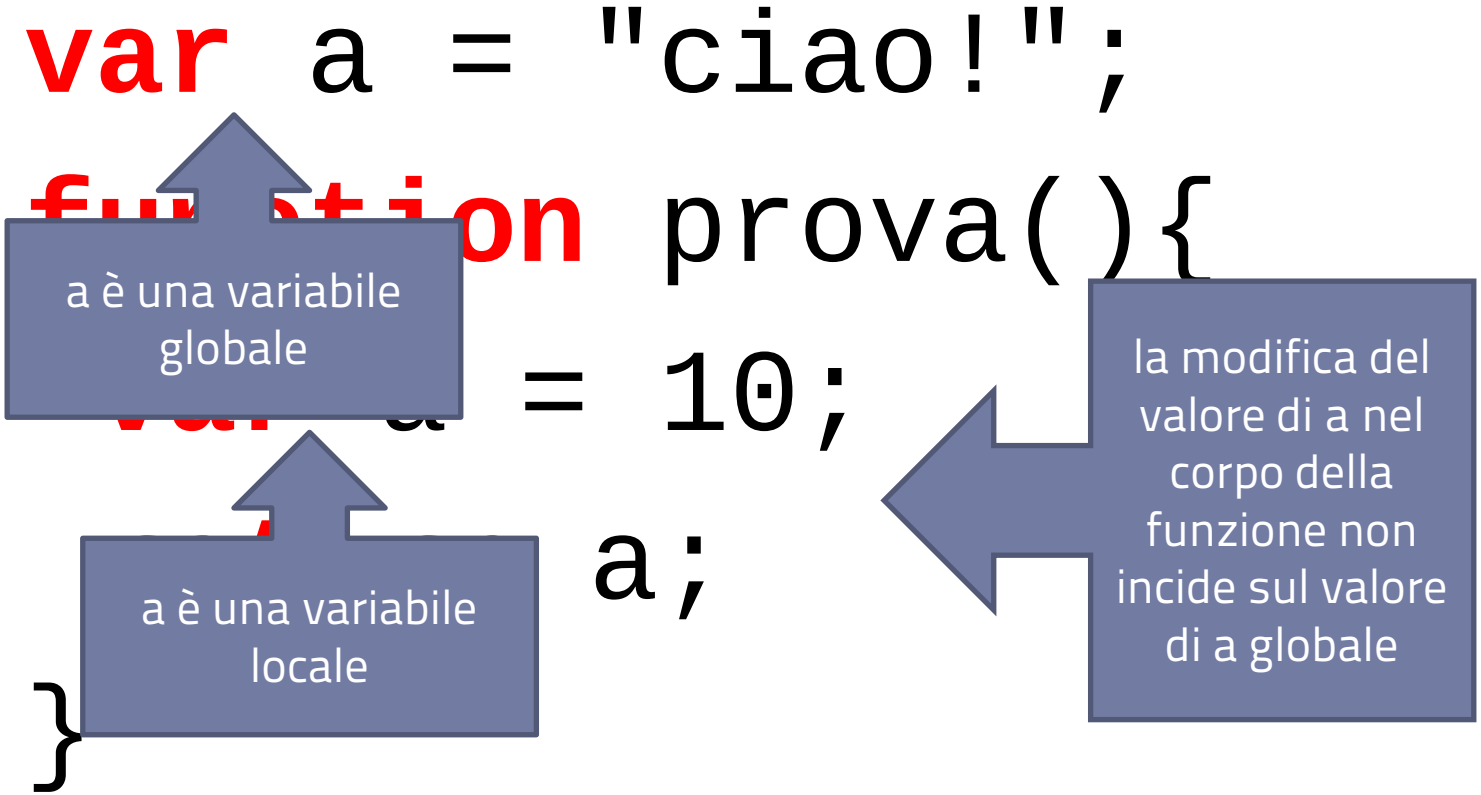
- Alle proprietà e ai metodi di un oggetto si accede tramite l'operatore punto.
- Gli operatori punto possono essere usati in catena.

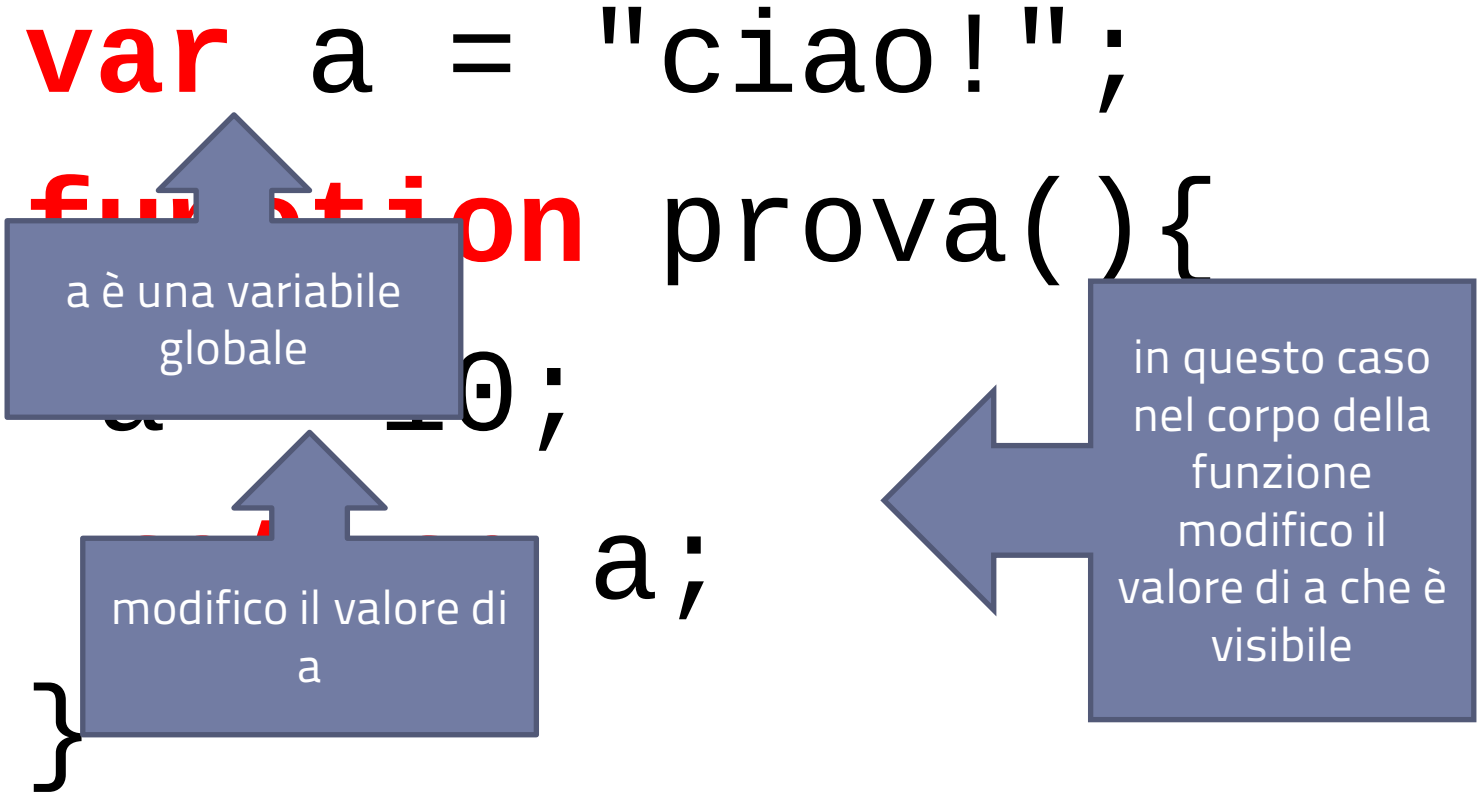

window
.document
.getElementById("msg_cerca")
HTML =
rmine "
ato trov
ndice " + i;
oggetto padre
(parent) di
gli
oggetto
document (child)
appartiene a
window
metodo di document
che restituisce un
oggetto corrispondente
all'elemento span con id
"msg_cerca"
proprietà
dell'oggetto
restituito a cui
viene assegnato
un valore

OGGETTI E SCOPO

LO SCOPO

- Lo scopo è l'ambito in cui una variabile o una funzione è visibile.
- Una variabile o una funzione dichiarate a livello dello script principale (cioè non nel corpo di una funzione) sono globali.
- Una variabile o una funzione dichiarate ne corpo di una funzione sono locali.


var a = "ciao!";
function prova(){
= 10;
a;
}
a è una variabile globale
a è una variabile locale
la modifica del valore di a nel corpo della funzione non incide sul valore di a globale


var a = "ciao!";
function prova(){
a = 10;
a;
}
a è una variabile globale
modifico il valore di a
in questo caso nel corpo della funzione modifico il valore di a che è visibile

SCOPO E OGGETTI

- All'interno dello stesso scopo non posso avere due nomi uguali. Se dichiaro all'interno di uno scopo dichiaro un nome già esistente la nuova dichiarazione viene ignorata.
- Uno scopo viene definito anche spazio dei nomi, namespace in inglese.

SCOPO E OGGETTI

- Le proprietà di un oggetto sono variabili dichiarate nello spazio dei nomi dell'oggetto
- I metodi di un oggetto sono funzioni dichiarate nello spazio dei nomi dell'oggetto
- Lo stesso nome può quindi rappresentare valori diversi a secondo dell'oggetto a cui appartiene.

```
var s = "Ciao a tutti!";
var a = [1,5,4,34,56,11];
var lenght = 100;
var p1 = s.length;
var p2 = a.length;
```

TIPI DI OGGETTO

- Istanze di classe
  - Tipi
  - Classi destinati a funzioni specifiche
  - Classi create da me
- Classi statiche
- Oggetti predefiniti legati al DOM

I TIPI COME OGGETTI

- In un linguaggio OOP i tipi sono classi e le variabili di un determinato tipo sono istanze di quella classe
- Per creare una nuova istanza di una classe si utilizza l'operatore **new** seguito dal constructor della classe. Questa operazione crea nella memoria del computer lo spazio necessario ad ospitare il nuovo oggetto.
- Il constructor è una funzione che lo stesso nome della classe di cui si vuole creare una nuova istanza:

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

CREAZIONE AUTOMATICA

- Per i tipi predefiniti la creazione dell'istanza è automatica:

creazione con constructor	equivale a
`str = new String();`	`str = "";`
`str = new String("Ciao!");`	`str = "Ciao!";`
`lista = new Array();`	`lista = [];`
`lista = new Array(2,3,4);`	`lista = [2,3,4];`
`flag = new Boolean(true);`	`flag = true;`
`oggetto = new Object();`	`oggetto = {};`

- Ma è obbligatoria. Solo dopo la creazione posso applicare i metodi della classe alla variabile che contiene il riferimento all'oggetto creato

PROPRIETÀ COMUNI

- Tutti le classi hanno in comune due proprietà:
  - **constructor**: contiene la funzione utilizzata quando si crea una nuova istanza della classe.
  - **prototype**: oggetto che contiene tutte le proprietà e i metodi che avrà la nuova istanza creata.

GLOBAL

PROPRIETÀ GLOBALI

Property	Description
Infinity	Un valore numerico che rappresenta l'infinito positivo e negativo
NaN	Il valore "Not-a-Number"
undefined	Indica che ha una variabile (o a una proprietà) non è stato assegnato alcun valore.

FUNZIONI GLOBALI

Function	Description
decodeURI(uri)	Decodifica un URI codificata con encodeURI
decodeURIComponent(uri)	Decodifica un URI codificata con decodeURIComponent
encodeURI(uri)	Codifica un URI (codifica i caratteri speciali eccetto / ? : @ & = + $ #)
encodeURIComponent(uri)	Codifica un URI (codifica i caratteri speciali compresi / ? : @ & = + $ #)
escape(str)	Questa funzione rende una stringa portatile, in modo che possa essere trasmessa attraverso qualsiasi rete a qualsiasi computer che supporti i caratteri ASCII.
eval(str)	Valuta una stringa e la esegue come se fosse il codice di script
isFinite()	Determina se un valore è un numero finito (e legale)
isNaN()	Determina se un valore è non è lo speciale valore NaN
Number()	Converte il valore di un oggetto in un numero
parseFloat()	Analizza una stringa e restituisce un numero in virgola mobile o NaN
parseInt()	Analizza una stringa e restituisce un intero o NaN
String()	Converte il valore di un oggetto in una stringa
unescape()	Decodifica una stringa codificata con escape.

STRING

CONSTRUCTOR

```
var str = "Ciao!";
var str = new String("Ciao!");
```

PROPRIETÀ

- Gli oggetti della classe String hanno una sola proprietà, la proprietà length che restituisce la lunghezza della stringa, cioè il numero di caratteri di cui è composta.

MANIPOLAZIONE

Method	Description
charAt(pos)	Restituisce il carattere alla posizione **pos**
charCodeAt(pos)	Restituisce il carattere (in formato Unicode) alla posizione **pos**
concat(s1, s2)	Concatena due stringhe (come s1 + s2)
fromCharCode(code)	Restituisce il carattere corrispondente al valore unicode **code**
indexOf(searchstring, start)	Restituisce la posizione della prima occorrenza della stringa **searchstring** in una stringa (-1 se non lo trova). Opzionalmente la ricerca può partire dalla posizione **start**
lastIndexOf(searchstring, start)	Restituisce la posizione dell'ultima occorrenza della stringa **searchstring** in una stringa (-1 se non lo trova). Opzionalmente la ricerca può partire dalla posizione **start** in vece che dall'ultimo carattere.
match(regexp)	Il metodo **match** cerca le corrispondenza e tra l'espressione regolare **regexp** e la stringa, e restituisce un array di corrispondenze. Se non viengono trovate corrispondenze viene restituito null.
replace(regexp/substr, newstring)	**Replace**() cerca una corrispondenza tra una stringa (o un'espressione regolare) e una stringa, e sostituisce la corrispondenze trovate con **newstring**
search(regexp)	Il metodo **search** cerca le corrispondenza e tra l'espressione regolare **regexp** e la stringa, e restituisce la posizione in cui è stata trovata oppure -1 se non viengono trovate corrispondenze.
slice(inizio, fine)	Estrae la parte di una stringa compresa tra **inizio** e **fine** e restituisce la parte estratta in una nuova stringa. In caso non sia passato un valore, **fine** sarà l'ultimo carattere della stringa.
split(char)	Converte la stringa in un **array** usando **char** come carattere di separazione.
substr(start, length)	Estrae **lentgth** caratteri dalla stringa, a partire da **start** e li restituisce in una nuova stringa. Se **length** non è specificati vengono restituiti i caratteri da **start** fino alla fine della stringa.
substring(from, to)	Estrae i caratteri delle stringa tra **from** e **to** non compreso. Se **to** è omesso fino alla fine della stringa.
toLowerCase()	Converte in minuscolo
toUpperCase()	Converte in maiuscolo

METODI PER CREARE TAG HTML

Metodo	Stringa che viene restituita
anchor(anchorname)	string
big()	<big>string</big>
blink()	<blink>string</blink>
bold()	string
fixed()	<tt>string</tt>
fontcolor("*#rrggbb*")	string
fontsize(dimensione)	string
italics()	string
link(url)	string
small()	string
strike()	<strike>string</strike>
sub()	_{string}
sup()	^{string}

ARRAY

CONSTRUCTOR

```
var a = [1,6,78,23];
var a = new Array(1,6,78,23);
```

PROPRIETÀ

- Gli oggetti della classe Array hanno una sola proprietà, la proprietà **length** che restituisce la lunghezza dell'array, cioè il numero di elementi di cui è composto.

METODI

Method	Description
concat(array2, array3, ..., arrayX)	Unisce uno o più array all'array a cui il metodo è applicato, e restituisce una copia degli array così uniti.
indexOf(elemento, start)	Cerca elemento in un array partendo da start (o dall'inizio se start è omesso) e ne restituisce la posizione. Se start è negativo indica la posizione reativa alla fine dell'array.
join(separatore)	Unisce gli elementi di un array in una stringa, e restituisce la stringa. Gli elementi sono separati da separatore. Il separatore di default è la virgola .
lastIndexOf(elemento, start)	Cerca l'ultima ricorrenza di elemento in un array partendo da start (o dall'iniziose start è omesso) e ne restituisce la posizione o -1 se elemento non viene trovato.
pop()	Rimuove l'ultimo elemento di un array, e restituisce l'elemento rimosso.
push(elemento)	Aggiunge elemento alla fine dell'array e restituische la nuova lunghezza.
reverse()	Inverte l'ordine degli elementi dell"array.
shift()	Rimuove il primo elemento di un array, e restituisce l'elemento rimosso.
slice(inizio, fine)	Estrae gli elementi a partire da inizio, fino a fine, non incluso e li restituisce in un nuovo array. L'array originale non viene modificato.
sort(sortfunct)	Ordina gli elementi di un array (alfabetico ascendente) o usa sortfunct per stabilire l'ordine
splice(indice, quanti, item1,....., itemX)	Rimuove quanti elementi dall'array a partire dalla posizine indice e inserisce gli elementi item1, ..., itemX (se forniti) a partire dalla posizine indice. Restituisce gli elementi rimossi.
toString()	Restituisce l'array convertito in stringa.
unshift(elemento)	Aggiunge elemento all'inizio dell'array e restituische la nuova lunghezza

SORT

```
var rubrica = [
                {nome:"Mario", cognome:"Rossi" },
                {nome:"Luigi", cognome:"Neri" },
                {nome:"Piero", cognome:"Verdi" },
                {nome:"Mario", cognome:"Bianchi" }
              ];
var sortCognome = function (a,b){
                              if (a.cognome > b.cognome){
                                return 1;
                              } else if (a.cognome == b.cognome){
                                return 0;
                              } else {
                                return 1;
                              }
                          };
rubrica.sort(sortCognome);
```

DATE

CONSTRUCTOR

```
var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d = new Date(year, month, day,
                 hours, minutes,
                 seconds,
                 milliseconds);
```

METODO STATICO

Date.parse(str)	Analizza una data in formato stringa e restituisce il numero di millisecondi dalla mezzanotte del 1 Gennaio 1970.

Metodi	Descrizione
getDate()	Restituisce il giorno del mese (1-31)
getDay()	Restituisce il giorno della settimana (0-6, 0 = domenica)
getFullYear()	Restituisce l'anno (quattro cifre)
getHours()	Restituisce l'ora (da 0-23)
getMilliseconds()	Restituisce i millisecondi (0-999)
getMinutes()	Restituisce i minuti (0-59)
getMonth()	Restituisce il mese (0-11)
getSeconds()	Restituisce i secondi (0-59)
getTime()	Restituisce il numero di millisecondi trascorsi dalla mezzanotte del 1 gennaio 1970
getTimezoneOffset()	Restituisce la differenza di tempo tra il GMT e l'ora locale, in pochi minuti
getUTCDate()	Restituisce il giorno del mese, in base all'ora universale (da 1-31)
getUTCDay()	Restituisce il giorno della settimana, in base all'ora universale (da0-6)
getUTCFullYear()	Restituisce l'anno, in base all'ora universale (quattro cifre)
getUTCHours()	Restituisce l'ora, in base all'ora universale (da 0-23)
getUTCMilliseconds()	Restituisce i millisecondi, in base all'ora universale (0-999)
getUTCMinutes()	Restituisce i minuti, in base all'ora universale (da 0-59)
getUTCMonth()	Restituisce il mese, in base all'ora universale (da 0-11)
getUTCSeconds()	Restituisce i secondi, in base all'ora universale (da 0-59)

Metodi	Descrizione
setDate()	Imposta il giorno del mese di un oggetto data
setFullYear()	Imposta l'anno (quattro cifre) di un oggetto data
setHours()	Imposta l'ora di un oggetto data
setMilliseconds()	Imposta i millisecondi di un oggetto data
setMinutes()	Impostare i minuti di un oggetto data
setMonth()	Imposta il mese di un oggetto data
setSeconds()	Imposta i secondi di un oggetto data
setTime()	Consente di impostare una data e un'ora aggiungendo o sottraendo un determinato numero di millisecondi per/da mezzanotte del primo gennaio 1970
setUTCDate()	Imposta il giorno del mese di un oggetto data, in base all'ora universale
setUTCFullYear()	Imposta l'anno di un oggetto data, in base all'ora universale (quattro cifre)
setUTCHours()	Imposta l'ora di un oggetto data, in base all'ora universale
setUTCMilliseconds()	Imposta i millisecondi di un oggetto data, in base all'ora universale
setUTCMinutes()	Impostare i minuti di un oggetto data, in base all'ora universale
setUTCMonth()	Imposta il mese di un oggetto data, in base all'ora universale
setUTCSeconds()	Impostare i secondi di un oggetto data, in base all'ora universale
setDate()	Imposta il giorno del mese di un oggetto data
setFullYear()	Imposta l'anno (quattro cifre) di un oggetto data
setHours()	Imposta l'ora di un oggetto data

Metodi	Descrizione
toDateString()	Converte la parte relativa alla data di un oggetto Date in una stringa leggibile
toISOString()	Restituisce la data come una stringa, utilizzando lo standard ISO
toJSON()	Restituisce la data come una stringa, formattato come una dataJSON
toLocaleDateString()	Restituisce la parte relativa alla data di un oggetto Date come una stringa, utilizzando le convenzioni di localizzazione
toLocaleTimeString()	Restituisce la parte di ora di un oggetto Date come una stringa, utilizzando le convenzioni di localizzazione
toLocaleString()	Converte un oggetto Date in una stringa, utilizzando le convenzioni di localizzazione
toString()	Converte un oggetto Date in una stringa
toTimeString()	Converte la parte ora di un oggetto Date in una stringa
toUTCString()	Converte un oggetto Date in una stringa, in base all'ora universale
UTC()	Restituisce il numero di millisecondi in una stringa data a partire dalla mezzanotte del 1 gennaio 1970, in base all'ora universale

NUMBER

CONSTRUCTOR

```
var n = 5;
var n = new Number(5);
var n = 10.6;
var n = new Number(10.6);
```

PROPRIETÀ STATICHE

Proprietà	Descrizione
MAX_VALUE	Restituisce il massimo numero consentito in JavaScript
MIN_VALUE	Restituisce il minimo numero consentito in JavaScript
NEGATIVE_INFINITY	Rappresenta l'infinito negativo.
POSITIVE_INFINITY	Rappresenta l'infinito positivo.

METODI

Method	Description
toExponential(x)	Reastituisce una stringa, che rappresenta il numero come notazione esponenziale dove x (opzionale) indica il numero dei decimali da usare
toFixed(x)	Converte il numero in una stringa, con x numero di decimali. Se x non viene specificato nessun decimale.
toPrecision(x)	Converte il numero in una stringa di lunghezza x. Se necessario vengono aggiunti punto decimale e 0.
toString(base)	Converte il numero in una stringa secondo la base specificata da base. La base di default è 10 (numero decimale). Se base vale 2 si ottiene la rappresentazione binaria del numero, se 16 quella esadecimale, ecc.

MATH

PROPRIETÀ STATICHE

Propietà	Descrizione
E	Restituisce il numero di Eulero (circa 2,718)
LN2	Restituisce il logaritmo naturale di 2 (circa 0,693)
LN10	Restituisce il logaritmo naturale di 10 (circa 2,302)
LOG2E	Restituisce il logaritmo in base 2 di E (circa 1,442)
LOG10E	Restituisce il logaritmo in base 10 di E (circa 0,434)
PI	Restituisce PI (circa 3.14)
SQRT1_2	Restituisce la radice quadrata di 1/2 (circa 0,707)
SQRT2	Restituisce la radice quadrata di 2 (circa 1,414)

METODI STATICI

Method	Description
abs(x)	Restituisce il valore assoluto di x
acos(x)	Restituisce l'arcocoseno di x, in radianti
asin(x)	Restituisce l'arcoseno di x, in radianti
atan(x)	Restituisce l'arcotangente di x come un valore numerico compreso tra-PI / 2 e PI 2 radianti
atan2(y,x)	Restituisce l'arcotangente del quoziente dei suoi argomenti
ceil(x)	Restituisce X arrotondato per eccesso al numero intero più vicino
cos(x)	Restituisce il coseno di x (x è in radianti)
exp(x)	Restituisce il valore di E elevato alla x
floor(x)	Restituisce X arrotondato per difetto al numero intero più vicino
log(x)	Restituisce il logaritmo naturale (base e) di x
max(x,y,z,...,n)	Restituisce il numero con il valore più alto
min(x,y,z,...,n)	Restituisce il numero con il valore più basso
pow(x,y)	Restituisce il valore di x alla potenza y
random()	Restituisce un numero casuale compreso tra 0 e 1
round(x)	Arrotonda x al numero intero più vicino
sin(x)	Restituisce il seno di x (x è in radianti)
sqrt(x)	Restituisce la radice quadrata di x
tan(x)	Restituisce la tangente di un angolo

REGEXP

CONSTRUCTOR

```
var re = new RegExp(pattern,
                    mod);
var re = /pattern/modificatori;
var re = /0-9/g;
```

MODIFICATORI

Modificatore	Descrizione
i	Eseguire case-insensitive di corrispondenza
g	Eseguire una partita globale (trovate tutte le partite, piuttosto che fermarsi dopo la prima partita)
m	Effettuare ricerche su righe multiple

PARENTESI QUADRE

Espressione	Descrizione
[abc]	Trova qualsiasi carattere tra le parentesi
[^abc]	Trova qualsiasi carattere non tra le parentesi
[0-9]	Trova qualsiasi cifra 0-9
[A-Z]	Trova un carattere tra A maiuscola e Z maiuscola
[a-z]	Trova un carattere tra a minuscola a z minuscola
[A-z]	Trova un carattere da maiuscolo a minuscolo A z
[adgk]	Trova qualsiasi carattere nell'elenco
[^adgk]	Trova un carattere non compreso nell'elenco
(red|blue|green)	Trova una delle alternative indicate

METACARATTERI

Metacarattere	Descrizione
..	Trova un singolo carattere, eccetto newline o terminatore di linea
\w	Trova un carattere alfanumerico
\W	Trova un carattere non alfanumerico
\d	Trova una cifra
\D	Trova un carattere non numerico
\s	Trova uno spazio bianco
\S	Trova un carattere non-spazio
\b	Trova un match ad inizio / fine di una parola
\B	Trovare non è una partita ad inizio / fine di una parola
\0	Trova un carattere NUL
\n	Trova un carattere di nuova riga
\f	Trova un carattere di avanzamento modulo
\r	Trova un carattere di ritorno
\t	Trova un carattere di tabulazione
\v	Trova un carattere di tabulazione verticale
\xdd	Trova il carattere specificato da un numero esadecimale dd
\uxxxx	Trova il carattere Unicode specificato da un numero esadecimale xxxx

QUANTIFICATORI

Quantificatore	Descrizione
n+	Corrisponde a qualsiasi stringa che contiene almeno un n
n*	Corrisponde a qualsiasi stringa che contiene zero o più occorrenze di n
n?	Corrisponde a qualsiasi stringa che contiene zero o una occorrenze di n
n {X}	Corrisponde a qualsiasi stringa che contiene una sequenza di *X n*
n {X, Y}	Corrisponde a qualsiasi stringa che contiene una sequenza di n da X a Y
n {X,}	Corrisponde a qualsiasi stringa che contiene una sequenza di almeno X *n*.
n$	Corrisponde a qualsiasi stringa con n alla fine.
^n	Corrisponde a qualsiasi stringa con n all'inizio.
?=n	Corrisponde a qualsiasi stringa che viene seguita dalla stringa specifica n
?!n	Corrisponde a qualsiasi stringa che non è seguita dalla stringa specifica n

PROPRIETÀ E METODI

Proprietà	Descrizione
global	Specifica se il modificatore "g" è impostato
ignoreCase	Specifica se il modificatore "i" è impostato
lastIndex	L'indice da cui iniziare la prossima ricerca
multiline	Specifica se il modificatore "m" è impostato
source	Il testo del pattern RegExp

Metodo	Descrizione
compile()	Compila un espressione regolare
exec ()	Cerca la prima occorrenza e la restituisce
test ()	Cerca la prima occorrenza . Restituisce vero o falso