**JavaScript**

JavaScript è un linguaggio di scripting sviluppato per dare interattività alle pagine HTML, può essere inserito direttamente nelle pagine Web. *Javascript è considerato un linguaggio di scripting di tipo interpretato e client-side.*

JavaScript è :

* ***interpretato*** e non compilato
* ***object-based*** ma non ***class-based***, esiste il concetto di oggetto ma non di classe
* ***debolmente tipizzato*** (weakly typed), non è necessario definire il tipo di una variabile

Il codice JavaScript viene eseguito da un interprete contenuto all’interno del browser

Nasce per dare ***dinamicità*** alle pagine Web, consente di:

* Accedere e modificare elementi della pagina HTML
* Reagire ad eventi generati dall’interazione fra utente e pagina
* Validare i dati inseriti dall’utente
* determinare il browser utilizzato e la dimensione della finestra in cui viene mostrata la pagina, lavorare con i browser cookie, ecc…

**Sintassi del linguaggio**

La sintassi di JavaScript è modellata su quella del C con alcune varianti significative.

* È un linguaggio ***case-sensitive***
* Le istruzioni sono terminate da ‘;’ ma il terminatore può essere omesso se si va a capo
* Sono ammessi sia commenti multilinea (delimitati da /\* e \*/) che monolinea (iniziano con //)
* Gli identificatori possono contenere lettere, cifre e i caratteri ***‘\_’*** e ***‘$’*** , ma non possono iniziare con una cifra
* La sintassi JavaScript definisce due tipi di valori: valori fissi e valori variabili:
* I valori fissi sono chiamati ***letterali***
* I valori variabili sono chiamati ***variabili***

**Posizionamento di codice JavaScript -** [**https://www.w3schools.com/js/js\_whereto.asp**](https://www.w3schools.com/js/js_whereto.asp)

In HTML, il codice JavaScript deve essere inserito tra ***<script>*** e ***</script>*** tag.

<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>

Un JavaScript ***function*** è un blocco di codice JavaScript, che può essere eseguito quando lo "chiamato".

Ad esempio, una funzione può essere chiamata *quando si verifica un evento* , come quando l'utente fa clic su un pulsante.

Gli script possono essere inseriti nella ***<body>***, o nella ***<head>*** sezione di una pagina HTML, o in entrambi.

Il posizionamento degli script nella parte inferiore dell'elemento <body> migliora la velocità di visualizzazione, poiché l'interpretazione degli script rallenta la visualizzazione.

Gli script possono essere inseriti anche in file esterni:

function myFunction() {  
 document.getElementById("demo").innerHTML = "Paragraph changed.";  
}

Per utilizzare uno script esterno, inserisci il nome del file di script nell'attributo ***src*** (origine) di un ***<script>*** tag: <script src="myScript.js"></script>

Gli script esterni non possono contenere ***<script>*** tag.

**Output -** [**https://www.w3schools.com/js/js\_output.asp**](https://www.w3schools.com/js/js_output.asp)

JavaScript può "visualizzare" i dati in diversi modi:

* Scrivere in un elemento HTML, usando ***innerHTML***
* Scrivere nell'output HTML utilizzando ***document.write()***
* Scrivere in una casella di avviso, utilizzando ***window.alert()***
* Scrivere nella console del browser, utilizzando ***console.log()***

***innerHTML:***

Per accedere a un elemento HTML, JavaScript può utilizzare il metodo ***document.getElementById(id)***.

L'attributo ***id*** definisce l'elemento HTML. La ***innerHTML*** proprietà definisce il contenuto HTML:

<p id="demo"></p>  
<script>  
 document.getElementById("demo").innerHTML = 5 + 6;  
</script> <https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_dom>

***document.write ():***

A scopo di test, è conveniente usare ***document.write():***

<script>  
 document.write(5 + 6);  
</script> <https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write>

L'uso di document.write () dopo il caricamento di un documento HTML, **cancellerà tutto il codice HTML esistente** :

<button type="button" onclick="document.write(5 + 6)">Try it</button>

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write_over>

***window.alert ():***

È possibile utilizzare una casella di avviso per visualizzare i dati:

<script>  
 window.alert(5 + 6);  
</script> <https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert>

***console.log ():***

Per scopi di ***debug***, è possibile utilizzare il console.log() metodo per visualizzare i dati.

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_console>

**Istruzioni -** [**https://www.w3schools.com/js/js\_statements.asp**](https://www.w3schools.com/js/js_statements.asp)

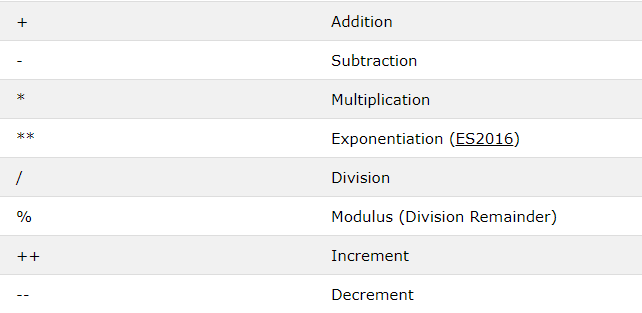
Un ***programma JavaScript*** è un elenco di ***istruzioni*** di programmazione .

*In HTML, i programmi JavaScript vengono eseguiti dal browser web*.

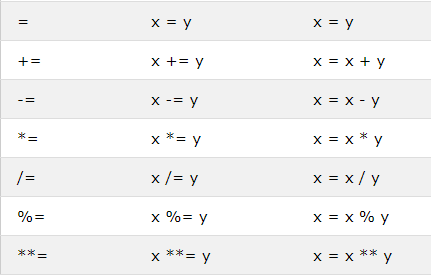
Le istruzioni JavaScript sono composte da: Valori, operatori, espressioni, parole chiave e commenti.

**Operatori -** [**https://www.w3schools.com/js/js\_operators.asp**](https://www.w3schools.com/js/js_operators.asp)

Operatori aritmetici sono usati per eseguire aritmetica sui numeri:



Operatori di assegnazione assegnano valori alle variabili JavaScript:



**Variabili -** [**https://www.w3schools.com/js/js\_datatypes.asp**](https://www.w3schools.com/js/js_datatypes.asp)

Le variabili vengono dichiarate usando la parola chiave ***var :***  ***var*** *nomevariabile;*

***Non hanno un tipo***, tecnicamente ha il valore di ***undefined***

È prevista la possibilità di inizializzare una variabile contestualmente alla dichiarazione: *var f = 15.8*

Esiste lo ***scope globale*** e quello ***locale*** (ovvero dentro una funzione) ma, a differenza di Java, non esiste lo scope di blocco.

Se inserisci un numero tra virgolette, il resto dei numeri verrà trattato come stringhe e concatenato.

var x = "5" + 2 + 3; -> 523 <https://www.w3schools.com/js/tryit.asp?filename=tryjs_variables_add_string_number>

var x = 2 + 3 + "5"; -> 55 <https://www.w3schools.com/js/tryit.asp?filename=tryjs_variables_add_number_string>

**Oggetto -** [**https://www.w3schools.com/js/js\_objects.asp**](https://www.w3schools.com/js/js_objects.asp)

Tutti i valori JavaScript, eccetto quelli ***primitivi***, sono oggetti.

Un **valore primitivo** è un valore che non ha proprietà o metodi. Un **tipo di dati primitivi** sono dati che hanno un valore primitivo. JavaScript definisce 5 tipi di tipi di dati primitivi:

* **string**
* **number**
* **boolean**
* **null**
* **undefined**

I valori primitivi sono immutabili (non possono essere modificati).

Le ***variabili*** JavaScript possono contenere singoli valori: var person = "John Doe";

Anche gli oggetti sono variabili, ma possono contenere molti valori, che sono scritti come coppie **nome:valori:** var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

Un oggetto JavaScript è una raccolta di **valori denominati**

***Proprietà dell'oggetto -*** [***https://www.w3schools.com/js/js\_object\_properties.asp***](https://www.w3schools.com/js/js_object_properties.asp)

I valori nominati, negli oggetti JavaScript, sono chiamati ***proprietà***, sono i valori associati a un oggetto. Un oggetto JavaScript è una raccolta di proprietà non ordinate. Le proprietà possono in genere essere modificate, aggiunte e cancellate, ma alcune sono di sola lettura.

È possibile accedere alle proprietà dell'oggetto in due modi:

*objectName.propertyName oppure objectName["propertyName"]*

Es person.lastName; person["lastName"];

Gli ***oggetti*** JavaScript sono scritti con parentesi ***graffe{},*** le ***proprietà*** dell'oggetto sono scritte come ***nome:valori***, separate da ***virgole***.

var person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
}; <https://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_object>

L' istruzione ***for...in*** (***Loop***) scorre le proprietà di un oggetto. In ***loop*** attraverso le proprietà di un oggetto:

var person = {fname:"John", lname:"Doe", age:25};   
for (x in person) {  
  txt += person[x];  
}

È possibile ***aggiungere nuove proprietà*** a un oggetto esistente semplicemente assegnandogli un valore. person.nationality = "English";

La parola chiave ***delete*** elimina una proprietà da un oggetto, cancella sia il valore della proprietà sia la proprietà stessa:

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
delete person.age;   // or delete person["age"];

In JavaScript, tutti gli ***attributi*** possono essere letti, ma solo l'attributo ***value*** può essere modificato.

***Metodi dell'oggetto -*** [***https://www.w3schools.com/js/js\_object\_methods.asp***](https://www.w3schools.com/js/js_object_methods.asp)

I metodi sono **azioni** che possono essere eseguite sugli oggetti. Le proprietà dell'oggetto possono essere sia valori primitivi, altri oggetti e funzioni.

Un **metodo object** è una proprietà dell'oggetto che contiene una **definizione di funzione**.

var person = {  
  firstName: "John",  
  lastName : "Doe",  
  fullName : function() {  
     return **this**.firstName + " " + **this**.lastName;  
   }  
}; <https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_method>

Gli oggetti JavaScript sono contenitori per valori denominati, chiamati proprietà e metodi.

In una definizione di funzione, ***this*** riferisce al "proprietario" della funzione.

Nell'esempio sopra, ***this*** è l' **oggetto persona** che "possiede" la funzione **fullName**, **this.firstName** indica la proprietà **firstName** di **questo oggetto** .

Si accede a un metodo oggetto con la seguente sintassi: *objectName.methodName()*

Se si accede alla **proprietà** fullName , senza (), verrà restituita la **definizione** della **funzione**: name = person.fullName; <https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_function>

**Metodi incorporati:**

var message = "Hello world!";  
var x = message.toUpperCase();

Il valore di x, dopo l'esecuzione del codice di cui sopra sarà: HELLO WORLD!

**Aggiunta di un metodo a un oggetto:**

person.name = function () {  
  return this.firstName + " " + this.lastName;  
};

***-Creazione di un oggetto JavaScript-***

Esistono diversi modi per creare nuovi oggetti:

* Definisci e crea un singolo oggetto, usando un ***oggetto letterale***.
* Definire e creare un singolo oggetto, con la parola chiave ***new***.
* Definire un ***costruttore*** di oggetti e quindi creare oggetti del tipo costruito.

**Oggetto letterale**

Usando un oggetto letterale, definisci e crei un oggetto in un'istruzione. Un oggetto letterale è un elenco di coppie ***nomi: valori*** (come età: 50) all'interno di parentesi ***graffe {}.***

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

**Parola chiave** ***new***

var person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";

**Costruttori oggetti**

function Person(first, last, age, eye) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eye;  
} <https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_constructor>

Il modo per creare un "tipo di oggetto" è utilizzare una **funzione di costruzione di oggetti** .

Nell'esempio sopra, ***function Person()*** è una funzione di costruzione di un oggetto.

Gli oggetti dello stesso tipo vengono creati chiamando la funzione di costruzione con la parola chiave ***new***:

var myFather = new Person("John", "Doe", 50, "blue");  
var myMother = new Person("Sally", "Rally", 48, "green");

**Aggiunta di una proprietà/metodo a un costruttore**

Per aggiungere una nuova proprietà/metodo a un costruttore, è necessario aggiungerlo alla funzione di costruzione:

function Person(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
  this.nationality = "English"; this.name = function() { return this.firstName + " " + this.lastName; };  
}

**Gli oggetti JavaScript sono mutabili**

Gli oggetti sono mutabili: sono indirizzati per ***riferimento***, ***non*** per ***valore***.

var x = person;  // This will not create a copy of person.

Qualsiasi modifica a x cambierà anche la persona, perché x e persona sono lo stesso oggetto.

**Scope -** [**https://www.w3schools.com/js/js\_scope.asp**](https://www.w3schools.com/js/js_scope.asp)

JavaScript ha uno scope di funzioni: ogni funzione crea un nuovo scope.

Lo scope determina l'accessibilità (visibilità) di queste variabili.

Le variabili dichiarate all'interno di una funzione JavaScript, diventano **LOCALI** per le funzioni, sono accessibili solo dalla funzione.

// code here can NOT use carName  
function myFunction() {  
  var carName = "Volvo";  
  // code here CAN use carName  
}

Poiché le variabili locali sono riconosciute solo all'interno delle loro funzioni, è possibile utilizzare variabili con lo stesso nome in diverse funzioni.

Una variabile dichiarata all'esterno di una funzione diventa **GLOBALE**, tutti gli script e le funzioni su una pagina web possono accedervi.

var carName = "Volvo";  
// code here can use carName  
function myFunction() {  
  // code here can also use carName   
}

Se assegni un valore a una variabile che non è stata dichiarata, diventerà automaticamente una variabile **GLOBAL** .

Questo esempio di codice dichiarerà una variabile globale ***carName***, anche se il valore è assegnato all'interno di una funzione.

myFunction();  
// code here can use carName   
function myFunction() {  
  carName = "Volvo";  
}

Con JavaScript, l'ambito globale è l'ambiente JavaScript completo.

In HTML, l'ambito globale è l'oggetto finestra. Tutte le variabili globali appartengono all'oggetto della finestra.

var carName = "Volvo";  
// code here can use window.carName

document.getElementById("demo").innerHTML = "I can display " + ***window***.carName;

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_scope_window>

La durata di una variabile JavaScript inizia quando viene dichiarata.

Le variabili locali vengono cancellate quando la funzione è completata.

In un browser Web, le variabili globali vengono eliminate quando si chiude la finestra del browser (o scheda), ma rimangono disponibili per le nuove pagine caricate nella stessa finestra.

***Tipo di operatore***

È possibile utilizzare l' ***typeof*** operatore JavaScript per trovare il tipo di una variabile JavaScript, restituisce il tipo di una variabile o un'espressione:

typeof ""             // Returns "string"  
typeof "John"         // Returns "string" typeof 314            // Returns "number"  
typeof 3.14           // Returns "number"  
typeof (3)            // Returns "number"

Qualsiasi variabile può essere svuotata, impostando il valore su ***undefined***. Sarà anche il tipo ***undefined***.

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_undefined_2>

***Null***

In JavaScript ***null*** è "niente". Dovrebbe essere qualcosa che non esiste. Sfortunatamente, in JavaScript, il tipo di dati ***null*** è un oggetto.

Puoi considerarlo un bug in JavaScript che ***typeof null*** è un oggetto. Dovrebbe essere null.

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_null>

**Funzioni -** [**https://www.w3schools.com/js/js\_functions.asp**](https://www.w3schools.com/js/js_functions.asp)

Una funzione JavaScript è un blocco di codice progettato per eseguire una determinata attività.

function myFunction(p1, p2) {  
  return p1 \* p2;    // The function returns the product of p1 and p2  
} <https://www.w3schools.com/js/tryit.asp?filename=tryjs_functions>

Una funzione JavaScript è definita con la ***function*** parola chiave, seguita da un ***nome***, seguito da parentesi ***()*** .

I nomi delle funzioni possono contenere lettere, cifre, trattini bassi e simboli del dollaro (stesse regole delle variabili).

function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}

Gli **argomenti di** funzione sono i **valori** ricevuti dalla funzione quando viene richiamata.

All'interno della funzione, gli argomenti (i parametri) si comportano come variabili locali.

L'accesso a una funzione senza () restituirà la definizione della funzione anziché il risultato della funzione:

function toCelsius(fahrenheit) {  
  return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius;

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_farenheit_to_celsius_2>

Le funzioni possono essere utilizzate nello stesso modo in cui si usano le variabili, in tutti i tipi di formule, assegnazioni e calcoli.

var text = "The temperature is " + toCelsius(77) + " Celsius";

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_function_variable>

Si può distinguere fra tipi valore e tipi riferimento

* Numeri e booleani sono tipi valore
* Array e Oggetti sono tipi riferimento

Per le stringhe abbiamo una situazione incerta, pur essendo un tipo primitivo si comportano come un tipo riferimento.

**Stringhe -** [**https://www.w3schools.com/js/js\_strings.asp**](https://www.w3schools.com/js/js_strings.asp)

Per trovare la lunghezza di una stringa, usa la ***length*** proprietà built-in :

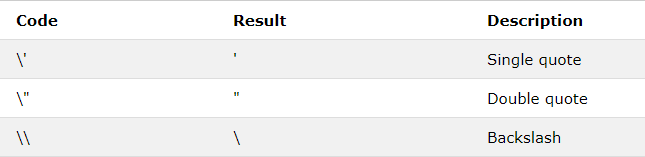
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length; <https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_length>

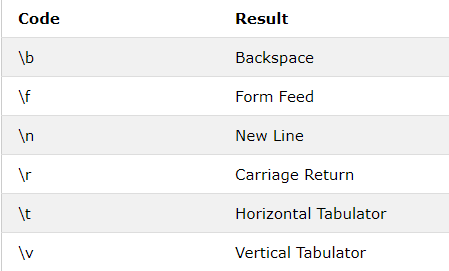
Poiché le stringhe devono essere scritte tra virgolette, è utilizzare il carattere di ***escape backslash***.

Il **\** carattere escape backslash ( ) trasforma i caratteri speciali in caratteri stringa:

var x = "We are the so-called \"Vikings\" from the north.";

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_quotes2>





Puoi anche suddividere una riga di codice **all'interno di una stringa di testo** con una singola barra rovesciata:

document.getElementById("demo").innerHTML = "Hello \  
Dolly!";

document.getElementById("demo").innerHTML = "Hello " +   
"Dolly!";

Normalmente, le stringhe di JavaScript sono valori primitivi, creati da letterali: var firstName = "John";

Ma le stringhe possono anche essere definite come oggetti con la parola chiave new: var firstName = new String("John");

Quando si utilizza l'operatore ***==***, le stringhe uguali sono uguali:

var x = "John";               
var y = new String("John");  
// (x == y) is true because x and y have equal values

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object1>

Quando si utilizza l'operatore ***===***, le stringhe uguali non sono uguali, poiché l'operatore ***===*** si aspetta l'uguaglianza sia nel tipo che nel valore.

Notare la differenza tra (x==y)e (x===y). Il confronto tra due oggetti JavaScript verrà **sempre** restituito false.

**Metodi di stringa -** [**https://www.w3schools.com/js/js\_string\_methods.asp**](https://www.w3schools.com/js/js_string_methods.asp)

La proprietà ***length*** restituisce la lunghezza di una stringa:

var sln = txt.length;

Il metodo ***indexOf()*** restituisce l'indice dell'occorrenza ***first*** di un testo specificato in una stringa (Case Sensitive):

var pos = str.indexOf("locate");

Il metodo ***lastIndexOf()*** restituisce l'indice dell'***ultima*** occorrenza di un testo specificato in una stringa:

var pos = str.lastIndexOf("locate");

Entrambi ***indexOf()*** e ***lastIndexOf()*** restituiscono -1 se il testo non viene trovato.

Entrambi i metodi accettano un secondo parametro come posizione iniziale per la ricerca:

var str = "Please locate where 'locate' occurs!";  
var pos = str.indexOf("locate", 15);

Il metodo ***search()*** ricerca una stringa per un valore specificato e restituisce la posizione della corrispondenza:

var pos = str.search("locate");

I due metodi **NON** sono uguali. Queste sono le differenze:

* Il metodo ***search()*** non può accettare un secondo argomento di posizione iniziale.
* Il metodo ***indexOf()*** non può assumere potenti valori di ricerca (espressioni regolari).

Se si omette il secondo parametro, ***substring()*** verrà troncato il resto della stringa:

var str = "Apple, Banana ,Kiwi";  
var res = str.substring(7, 13); 🡪 Banana

Il metodo ***replace()*** sostituisce un valore specificato con un altro valore in una stringa:

str = "Please visit Microsoft!";  
var n = str.replace("Microsoft", "W3Schools");

Il replace()metodo non modifica la stringa su cui è chiamato. Restituisce una nuova stringa.

Per sostituire la distinzione tra maiuscole e minuscole, utilizzare ***un'espressione regolare*** con una bandiera ***/i*** (non sensibile):

str = "Please visit Microsoft!";  
var n = str.replace(/MICROSOFT/i, "W3Schools");

Per sostituire tutte le corrispondenze, utilizza ***un'espressione regolare*** con una bandiera ***/g*** (corrispondenza globale):

str = "Please visit Microsoft and Microsoft!";  
var n = str.replace(/Microsoft/g, "W3Schools");

Una stringa viene convertita in maiuscolo con ***toUpperCase():*** var text2 = text1.toUpperCase();

Una stringa viene convertita in minuscolo con ***toLowerCase():*** var text2 = text1.toLowerCase();

Il metodo ***trim()*** rimuove gli *spazi bianchi* da entrambi i lati di una stringa: str.trim()

Il metodo ***charAt()*** restituisce il carattere in un indice (posizione) specificato in una stringa:

var str = "HELLO WORLD";  
str.charAt(0);            // returns H

**Numeri -** [**https://www.w3schools.com/js/js\_numbers.asp**](https://www.w3schools.com/js/js_numbers.asp)

var x = 3.14;    // A number with decimals  
var y = 3;       // A number without decimals

var x = 123e5;    // 12300000  
var y = 123e-5;   // 0.00123

JavaScript utilizza l'operatore + sia per l'aggiunta che per la concatenazione. I numeri sono sommati. Le stringhe sono concatenate.

var x = 10;  
var y = 20;  
var z = x + y;           // z will be 30 (a number)

var x = "10";  
var y = "20";  
var z = x + y;           // z will be 1020 (a string)

Se aggiungi un numero e una stringa, il risultato sarà una concatenazione di stringhe:

var x = 10;  
var y = "20";  
var z = x + y;           // z will be 1020 (a string)

Se aggiungi una stringa e un numero, il risultato sarà una concatenazione di stringhe:

var x = "10";  
var y = 20;  
var z = x + y;           // z will be 1020 (a string)

L'interprete JavaScript funziona da sinistra a destra. Il primo 10 + 20 viene aggiunto perché x e y sono entrambi numeri. Quindi 30 + "30" è concatenato perché z è una stringa.

JavaScript proverà a convertire stringhe in numeri in tutte le operazioni numeriche. Questo funzionerà:

var x = "100";  
var y = "10";  
var z = x / y;       // z will be 10

var x = "100";  
var y = "10";  
var z = x \* y;       // z will be 1000

var x = "100";  
var y = "10";  
var z = x - y;       // z will be 90

***NaN - Not a Number***

NaN è una parola riservata JavaScript che indica che un numero non è un numero legale.

Cercando di fare aritmetica con una stringa non numerica comporterà NaN(Non un numero):

var x = 100 / "Apple";  // x will be NaN (Not a Number)

Puoi utilizzare la funzione JavaScript globale ***isNaN()*** per scoprire se un valore è un numero:

var x = 100 / "Apple";  
isNaN(x);               // returns true because x is Not a Number

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_isnan_true>

typeof NaN;            // returns "number"

***Infinito***

***Infinity(o -Infinity)*** è il valore che JavaScript restituirà se si calcola un numero al di fuori del numero più grande possibile.

var myNumber = 2;  
while (myNumber != Infinity) {   // Execute until Infinity  
  myNumber = myNumber \* myNumber;  
} <https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_infinity>

La divisione per 0 (zero) genera anche ***Infinity***:

var x =  2 / 0;       // x will be Infinity  
var y = -2 / 0;       // y will be -Infinity

typeof Infinity;     // returns "number"

Normalmente i numeri JavaScript sono valori primitivi creati da letterali: var x = 123;

Ma i numeri possono anche essere definiti come oggetti con la parola chiave ***new***: var y = new Number(123);

**Metodi di numero -** [**https://www.w3schools.com/js/js\_number\_methods.asp**](https://www.w3schools.com/js/js_number_methods.asp)

Il metodo ***toString()***restituisce un numero come stringa.

Tutti i metodi numerici possono essere utilizzati su qualsiasi tipo di numeri (letterali, variabili o espressioni):

var x = 123;  
x.toString();            // returns 123 from variable x  
(123).toString();        // returns 123 from literal 123  
(100 + 23).toString();   // returns 123 from expression 100 + 23

Il metodo ***toFixed()*** restituisce una stringa, con il numero scritto con un numero specificato di decimali:

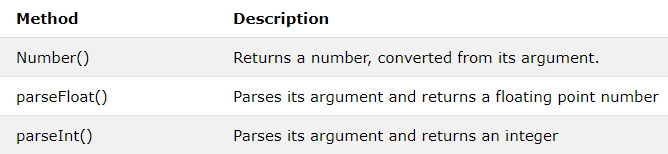
var x = 9.656;  
x.toFixed(0);           // returns 10  
x.toFixed(2);           // returns 9.66  
x.toFixed(4);           // returns 9.6560  
x.toFixed(6);           // returns 9.656000

Il metodo ***toPrecision()*** restituisce una stringa, con un numero scritto con una lunghezza specificata:

var x = 9.656;  
x.toPrecision();        // returns 9.656  
x.toPrecision(2);       // returns 9.7  
x.toPrecision(4);       // returns 9.656  
x.toPrecision(6);       // returns 9.65600

***Metodi JavaScript globali***

I metodi globali di JavaScript possono essere utilizzati su tutti i tipi di dati JavaScript. Questi sono i metodi più rilevanti quando si lavora con i numeri:



***Number()*** può essere usato per convertire le variabili JavaScript in numeri:

Number(true);          // returns 1  
Number(false);         // returns 0  
Number("10");          // returns 10  
Number("10.33");       // returns 10.33  
Number("10,33");       // returns NaN  
Number("10 33");       // returns NaN   
Number("John");        // returns NaN

Number(new Date("2017-09-30"));    // returns 1506729600000

***parseInt()*** analizza una stringa e restituisce un numero intero. Gli spazi sono ammessi. Viene restituito solo il primo numero:

parseInt("10");         // returns 10  
parseInt("10.33");      // returns 10  
parseInt("10 20 30");   // returns 10  
parseInt("10 years");   // returns 10  
parseInt("years 10");   // returns NaN

***parseFloat()*** analizza una stringa e restituisce un numero. Gli spazi sono ammessi. Viene restituito solo il primo numero:

parseFloat("10");        // returns 10  
parseFloat("10.33");     // returns 10.33  
parseFloat("10 20 30");  // returns 10  
parseFloat("10 years");  // returns 10  
parseFloat("years 10");  // returns NaN

**Array -** [**https://www.w3schools.com/js/js\_arrays.asp**](https://www.w3schools.com/js/js_arrays.asp)

Gli array JavaScript sono scritti con parentesi quadre, separati da virgole.

var cars=["Saab","Volvo","BMW"];

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_array>

L'utilizzo di un array letterale è il modo più semplice per creare una matrice JavaScript.

Sintassi: var array\_name = [item1, item2, ...];

Si accede a un elemento dell'array facendo riferimento al **numero di indice** .

var cars = ["Saab", "Volvo", "BMW"];  
document.getElementById("demo").innerHTML = cars[0];

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_element_change>

Con JavaScript, è possibile accedere all'intero array facendo riferimento al nome dell'array:

var cars = ["Saab", "Volvo", "BMW"];  
document.getElementById("demo").innerHTML = cars;

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_full>

Gli array sono un tipo speciale di oggetti. L'operatore ***typeof*** in JavaScript restituisce "oggetto" per gli array.

Puoi avere oggetti in un array. Puoi avere funzioni in un array. Puoi avere più array in un array:

myArray[0] = Date.now;  
myArray[1] = myFunction;  
myArray[2] = myCars;

***Proprietà e metodi dell'array***

var x = cars.length;   // The length property returns the number of elements  
var y = cars.sort();   // The sort() method sorts arrays

La proprietà ***length*** di un array restituisce la lunghezza di un array (il numero di elementi dell'array).

var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.length;   // the length of fruits is 4

Il modo più semplice per aggiungere un nuovo elemento a un array è utilizzare il metodo ***push():***

var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.push("Lemon");    // adds a new element (Lemon) to fruits

typeof fruits;    // returns object

Array.isArray(fruits);   // returns true

Il metodo ***pop()*** rimuove l'ultimo elemento da un array:

var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.pop();              // Removes the last element ("Mango") from fruits

**Eventi -** [**https://www.w3schools.com/js/js\_events.asp**](https://www.w3schools.com/js/js_events.asp)

Gli eventi HTML sono "***cose***" che accadono agli ***elementi*** HTML.

Un evento HTML può essere qualcosa che fa il browser o qualcosa che un utente fa. Es:

* Una pagina Web HTML ha finito il caricamento
* È stato modificato un campo di input HTML
* È stato fatto clic su un pulsante HTML

JavaScript ti consente di eseguire il codice quando vengono rilevati eventi.

HTML consente agli attributi del gestore eventi, con ***codice JavaScript***, di essere aggiunti agli elementi HTML.

<element event=**"*some JavaScript*"**> o <element event=**’*some JavaScript*’**>

Nell'esempio seguente, un attributo ***onclick*** (con codice), viene aggiunto a un elemento ***<button>:***

<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>

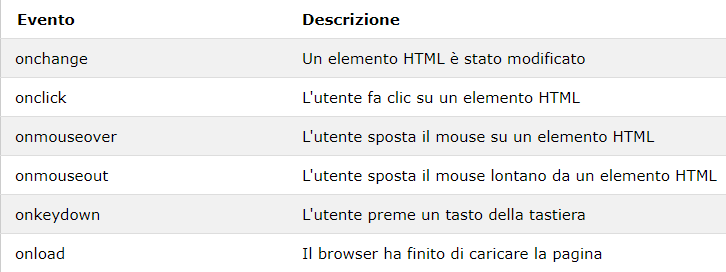
<https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick1>

Il codice cambia il contenuto del proprio elemento (usando ***this.innerHTML***):

<button onclick="this.innerHTML = Date()">The time is?</button>

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick>

Il codice JavaScript è spesso lungo diverse righe. È più comune vedere le funzioni di chiamata degli attributi degli eventi: <https://www.w3schools.com/js/tryit.asp?filename=tryjs_events1>\



**Onkeyup:**

L'evento onkeyup si verifica quando l'utente rilascia un tasto (sulla tastiera):

object.onkeyup = function(){*myScript*};

<input type="text" onkeyup="myFunction()">

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onkeyup>

**Onload:**

Esegui un JavaScript subito dopo aver caricato una pagina:

object.onload = function(){*myScript*};

<body onload="myFunction()">

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onload>

L***'onload*** è spesso usato all'interno dell'elemento <body> per eseguire uno script una volta che una pagina web ha completamente caricato tutto il contenuto (incluse immagini, file di script, file CSS, ecc.).

L'evento ***onload*** può essere utilizzato anche per gestire i *cookie*.

**Onmouseover/Onmouseout:**

L'evento ***onmouseover*** si verifica quando il puntatore del mouse viene spostato su un elemento o su uno dei suoi figli: object.onmouseover = function(){*myScript*};

L'evento ***onmouseout*** si verifica quando il puntatore del mouse viene spostato da un elemento o fuori da uno dei suoi figli: object.onmouseout = function(){*myScript*};

<img onmouseover="bigImg(this)" onmouseout="normalImg(this)" src="smiley.gif" alt="Smiley">

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onmouseover>

**onblur:**

Esegui un JavaScript quando un utente lascia un campo di input:

object.onblur = function(){*myScript*};

<input type="text" onblur="myFunction()">

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onblur>

L'evento ***onblur*** si verifica quando un oggetto perde lo stato attivo, spesso utilizzato con il codice di convalida del modulo. l'evento ***onblur*** è l'opposto dell'evento ***onfocus*** .

**onfocus:**

Esegui un JavaScript quando viene attivato un campo di input:

object.onfocus = function(){*myScript*};

<input type="text" onfocus="myFunction()">

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onfocus>

L'evento ***onfocus*** si verifica quando un elemento diventa attivo, viene spesso utilizzato con <input>, <select> e <a>.

**onclick:**

Esegui un JavaScript quando si fa clic su un pulsante:

object.onclick = function(){*myScript*};

<button onclick="myFunction()">Click me</button>

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onclick>

**onreset:**

Esegui un JavaScript quando viene resettato un modulo: object.onreset = function(){*myScript*};

<form onreset="myFunction()">  
  Enter name: <input type="text">  
  <input type="reset">  
</form> <https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onreset>

**onsubmit**:

Esegui un JavaScript quando viene inviato un modulo:

<form onsubmit="myFunction()">  
  Enter name: <input type="text">  
  <input type="submit">  
</form> <https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onsubmit>

**Metodi**

**setInterval:**

Chiama una funzione o valuta un'espressione a intervalli specificati (in millisecondi).

Il metodo ***setInterval*** () continuerà a chiamare la funzione finché non viene chiamato ***clearInterval*** () o la finestra viene chiusa.

Il valore ID restituito da ***setInterval*** () viene utilizzato come parametro per il metodo ***clearInterval*** ().

**Suggerimento:** 1000 ms = 1 secondo.

**Suggerimento:** per eseguire una funzione solo una volta, dopo un numero specificato di millisecondi, utilizzare il metodo ***setTimeout*** () .

setInterval(function(){ alert("Hello"); }, 3000);

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_win_setinterval>

**clearInterval:**

Il metodo ***clearInterval*** () cancella un timer impostato con il metodo ***setInterval*** () .

Il valore ID restituito da ***setInterval*** () viene utilizzato come parametro per il metodo ***clearInterval*** ().

*Nota*: per poter utilizzare il metodo ***clearInterval*** (), è necessario utilizzare una variabile durante la creazione del metodo dell'intervallo:

myVar = setInterval("*javascript function*",*milliseconds*);

Quindi sarai in grado di interrompere l'esecuzione chiamando il metodo ***clearInterval*** ().

clearInterval(myVar);

**blur:**

Il metodo ***blur*** () viene utilizzato per rimuovere lo stato attivo da un elemento.

**Suggerimento:** usa il metodo ***focus*** () per dare fuoco a un elemento.

Rimuovi lo stato attivo da un elemento <a>: document.getElementById("myAnchor").blur();

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_html_blur>

**focus:**

Il metodo ***focus*** () è usato per focalizzare un elemento (se può essere focalizzato).

**Suggerimento:** usa il metodo ***blur*** () per rimuovere lo stato attivo da un elemento.

Dà attenzione a un elemento <a>: document.getElementById("myAnchor").focus();

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_html_blur>

**click:**

Il metodo click () simula un clic del mouse su un elemento.

Questo metodo può essere utilizzato per eseguire un clic su un elemento come se l'utente avesse fatto clic manualmente su di esso.

document.getElementById("myCheck").click(); // Click on the checkbox

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_html_click>

**reset:**

Reimposta i valori di tutti gli elementi in un modulo (come se si stesse facendo clic sul pulsante Ripristina):

document.getElementById("myForm").reset();

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_form_reset>

**submit:**

Il metodo ***submit*** () invia il modulo (come per il pulsante Invia).

document.getElementById("myForm").submit();

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_form_submit>