

Chuleta: Operadores de Asignación

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

El operador de **asignación** (=) asigna un valor a una variable.

% (Módulo de la división)

** (Exponente) 10**2

X==Y ¿Es X = Y?

X===Y ¿Es X estrictamente igual a Y?

X!=Y ¿Es X desigual a Y?

X!==Y ¿Es estrictamente desigual?

&& → Y

|| → o

X+=Y → x=x+y

X-=Y → x=x-y

X*=Y → x=x*y

X/=Y → x=x/y

+ Operador de cadena

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;    // resultado "John Doe"
```

```
txt1 = "What a very ";
txt1 += "nice day";        //What a very nice day
```

Si se agrega un número y una cadena, el resultado será una cadena!

X++ → x=x+1
x-- → x=x-1

Incrementando:

```
var x = 5;
x++;
var z = x;
```

Decrementando:

```
var x = 5;
x--;
var z = x;
```

Imprimir en el documento

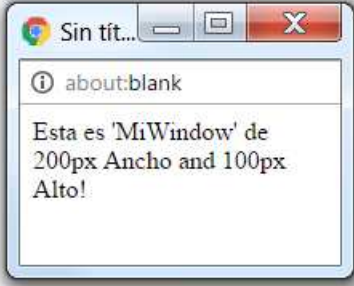
```
<!DOCTYPE html>
<html>
<body>

<p>Haz clic en el boton para abrir y escribir un texto.</p>

<button onclick="myFunction()">Aquí</button>

<script>
function myFunction() {
    document.open();
    document.write("<h1>Qué tal?</h1>");
    document.close();
}
</script>

</body>
</html>
```

Imprimir en una ventana nueva	<pre> <!DOCTYPE html> <html> <body> <p>Haz clic en el botón y se abrirá una nueva ventana "MiWindow" with some text.</p> <button onclick="myFunction()">Try it</button> <script> function myFunction() { var myWindow = window.open("", "MiWindow", "width=200,height=100"); myWindow.document.write("<p>Esta es 'MiWindow' de 200px Ancho and 100px Alto!</p>"); } </script> </body> </html> </pre> 
Salto de línea	<pre> <!DOCTYPE html> <html> <body> <p>write() no añade línea después del párrafo</p> <pre> <script> document.write("Hello World!"); document.write("Have a nice day!"); </script> </pre> <p>writeln() añade línea después del párrafo:</p> <pre> <script> document.writeln("Hello World!"); document.writeln("Have a nice day!"); </script> </pre> </body> </html> </pre>

Chuleta: Arrays	
Declarando un array. "Método Literal"	<pre>var cars = ["Saab", "Volvo", "BMW"]; var cars = ["Saab", "Volvo", "BMW"];</pre>
Operador NEW "Método el objeto Array"	<pre>var cars = new Array("Saab", "Volvo", "BMW");</pre>
- Modifico el primer valor	<pre>cars[0] = "Opel";</pre>
- Acceder a un Elemento	<pre>var cars = ["Saab", "Volvo", "BMW"]; document.getElementById("demo").innerHTML = cars[0];</pre>
- Acceder a todos los elementos.	<pre>var cars = ["Saab", "Volvo", "BMW"]; document.getElementById("demo").innerHTML = cars;</pre>
Matrices como propiedades de objetos	<pre>var person = {firstName:"John", lastName:"Doe", age:46}; person.firstName → John</pre>
	<pre>var person = []; person["firstName"] = "John"; person["lastName"] = "Doe"; person["age"] = 46; var x = person.length; // person.length → 0 var y = person[0]; // person[0] → undefined</pre>
Los elementos de una matriz pueden ser otras cosas	<pre>myArray[0] = Date.now; → Propiedad de un objeto myArray[1] = myFunction; → Una función myArray[2] = myCars; → Otro Array</pre>
Propiedades del Array	<pre>var x = cars.length; // length → Propiedad retorna "Nº de Elementos" var y = cars.sort(); // sort → Propiedad "ordena el array" var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.length; // length fruits → 4</pre>
Bucle "for"	<pre>var fruits, text, fLen, i; fruits = ["Banana", "Orange", "Apple", "Mango"]; fLen = fruits.length; text = ""; for (i = 0; i < fLen; i++) { text += "" + fruits[i] + ""; }</pre>
Añadir un elemento	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.push("Lemon"); // añadimos (Lemon) a fruits var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits[fruits.length] = "Lemon"; var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits[6] = "Lemon"; // ojo, con los agujeros</pre>
-Uso	<pre>- Cuando los índices son números, usar: ¡Mejor opción! var points = []; var points = [40, 100, 1, 5, 25, 10]; - Cuando los índices son cadenas de texto, usar: → Da problemas de elementos "undefined" var points = new Array(); var points = new Array(40, 100, 1, 5, 25, 10);</pre>
Operador "typeof", para saber que es un array	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; typeof fruits; // retorna → Object</pre>
	<pre>Array.isArray(fruits); // retorna → true</pre>
Operador "instanceof" para saber is es Array, y es creado por Constructor.	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits instanceof Array // retorna → true</pre>

Métodos:	
toString() Convierte el Array a una cadena separada por comas.	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits.toString();</pre> <p>→ Banana,Orange,Apple,Mango</p>
join() Convierte el Array a una cadena pero puedo especificar el separador	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits.join(" * ");</pre> <p>→ Banana*Orange*Apple*Mango</p>
pop() Elimina el último elemento de una matriz	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; var x = fruits.pop(); // El valor de x es "Mango" fruits.pop(); // Elimina el ultimo element de la matriz.</pre>
push() Añade un elemento al final	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.push("Kiwi"); // Añade al final ("Kiwi") a fruits var x = fruits.push("Kiwi"); // Devuelve longitud del array 5</pre>
shift() Elimina el primer elemento y desplaza el resto a la izquierda, a un índice menor	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.shift(); // Elimina el primer elemento "Banana" de fruits</pre>
unshift() Añade un elemento al principio	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.unshift("Lemon"); // Añadir el elemento "Lemon" a fruits</pre>
Cambio de elementos	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits[0] = "Kiwi"; // Cambia el primer elemento de fruits a "Kiwi"</pre> <pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits[fruits.length] = "Kiwi"; // Añado "Kiwi" a fruit</pre>
Borrado de elementos delete	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; delete fruits[0]; // Cambio el primer element de fruits to undefined //!Ojo! como deja huecos es conveniente usar el método pop() ó shift() para no dejar huecos</pre>
splice() Para añadir varios elementos al mismo tiempo	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.splice(2, 0, "Lemon", "Kiwi");</pre> <p>El primer parámetro (2) posición en la que se añaden nuevos elementos. El segundo parámetro (0) el número de elementos debe ser eliminado. El resto de los parámetros ("Lemon", "Kiwi") los nuevos elementos que se añaden.</p>

splice() para eliminar elementos	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.splice(0, 1); // Borra el primer element de fruits</pre> <p>El primer parámetro (0) define la posición en la que se van a añadir elementos</p> <p>El segundo parámetro (1) define el número de elementos debe ser eliminado.</p> <p>El resto de los parámetros se omiten. No se agregarán nuevos elementos.</p>
concat() para unir dos arrays	<pre>var myGirls = ["Cecilie", "Lone"]; var myBoys = ["Emil", "Tobias", "Linus"]; var myChildren = myGirls.concat(myBoys); //Concatena myGirls y myBoys</pre> <pre>var arr1 = ["Cecilie", "Lone"]; var arr2 = ["Emil", "Tobias", "Linus"]; var arr3 = ["Robin", "Morgan"]; var myChildren = arr1.concat(arr2, arr3); //Concatena arr1, arr2 y arr3</pre>
slice() rebanar un trozo de matriz de uno o varios elementos	<pre>var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"]; var citrus = fruits.slice(1); //Crea una nueva matriz a partir del elemento con índice 1</pre> <pre>var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"]; var citrus = fruits.slice(3); //Crea una nueva matriz a partir del elemento con índice 3</pre> <pre>var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"]; var citrus = fruits.slice(1, 3); //Crea una nueva matriz a partir del elemento con índice 1 hasta el 3</pre>

valueOf() convierte un array en una cadena	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits;</pre>
toString() convierte un array en una cadena	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits.valueOf(); var fruits = ["Banana", "Orange", "Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits.toString();</pre>

Ordenación:	
sort() ordena la matriz por orden alfabético	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.sort(); // Ordena los elementos de fruits</pre>
reverse() invierte los elementos de una matriz	<pre>var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.sort(); // Ordena los elementos de fruits fruits.reverse(); // Ordena en orden inverso los elementos</pre>
Si necesito ordenar números, ojo 22 es mas grande que 100	<pre>var points = [40, 100, 1, 5, 25, 10]; points.sort(function(a, b){return a - b}); // Orden ascendente var points = [40, 100, 1, 5, 25, 10]; points.sort(function(a, b){return b - a}); // Orden deccendente</pre> <p><u>La función de comparación:</u></p> <pre>function(a, b){return a-b} devuelve un valor, positivo, negativo o cero.</pre> <p><u>Ejemplo:</u></p> <ul style="list-style-type: none"> - Al comparar 40 y 100, el método sort () llama a la función de comparación (40.100). - La función calcula 40-100, y devuelve -60 (un valor negativo). - La función de clasificación clasificará 40 como un valor inferior a 100.
Ordenación en orden aleatorio	<pre>var points = [40, 100, 1, 5, 25, 10]; points.sort(function(a, b){return 0.5 - Math.random()});</pre>
Encontrar el valor más alto de una matriz	<pre>var points = [40, 100, 1, 5, 25, 10]; points.sort(function(a, b){return b - a}); // Ahora points[0] contiene el más alto valor</pre>
Encontrar el valor más pequeño	<pre>var points = [40, 100, 1, 5, 25, 10]; points.sort(function(a, b){return a - b}); // Ahora points[0] contiene el valor más bajo.</pre>

Ejemplo 1: Mostrar el Array en la página web

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

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

</body>
</html>
```

Ejemplo 2: Propiedades de Objetos

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Objects</h1>
<p>JavaScript uses names to access object properties.</p>
<p id="demo"></p>

<script>
var person = {firstName:"John", lastName:"Doe", age:46};
document.getElementById("demo").innerHTML = person["firstName"];
</script>

</body>
</html>
```

Ejemplo 3:

Recorrer el Array.

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Arrays</h1>

<p>Recorremos el Array con un for:</p>
<p id="demo"></p>

<script>
var fruits, text, fLen, i;

fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Ejemplo 4: Creación de un array y mostrar su primer elemento.

```
<!DOCTYPE html>
<html>
<body>

<p> Usar [] en vez de new Array y dejar este para objetos.</p>

<p id="demo"></p>

<script>
//var points = new Array(40, 100, 1, 5, 25, 10);
var points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points[0];
</script>

</body>
</html>
```

Ejemplo 5: Ordenar Alfabéticamente y Numéricamente

Ordenar alfabéticamente y numéricamente.

```
<button onclick="myFunction1()">Ordenar Alfabéticamente</button>
<button onclick="myFunction2()">Ordenar Numéricamente</button>

<p id="demo"></p>

<script>
var points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points;

function myFunction1() {
  points.sort();
  document.getElementById("demo").innerHTML = points;
}
function myFunction2() {
  points.sort(function(a, b){return a - b});
  document.getElementById("demo").innerHTML = points;
}
</script>
```

Chuleta:	Métodos Numéricos	Number
<p>JavaScript permite trabajar con métodos y propiedades para valores primitivos, porque JavaScript trata a los valores simples como objetos al ejecutar los métodos y propiedades.</p> <p>Los valores primitivos o simples son por ejemplo: 14, 12.5, ...</p> <p>En JavaScript, un número puede ser un valor simple (typeof = número) o un objeto (typeof = objeto).</p> <p>El método valueOf () se utiliza internamente en JavaScript para convertir objetos de número para valores primitivos.</p>		
Métodos		
toString () devuelve un número como una cadena.	<pre>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</pre>	
toExponential () devuelve una cadena, con un número redondeado y escrita usando la notación exponencial.	<pre>var x = 9.656; x.toExponential(2); // returns 9.66e+0 x.toExponential(4); // returns 9.6560e+0 x.toExponential(6); // returns 9.656000e+0</pre>	
toFixed () devuelve una cadena, con el número escrito con un número especificado de decimales:	<pre>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</pre>	
toPrecision () devuelve una cadena, con un número escrito con una longitud especificada:	<pre>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</pre>	
valueOf () devuelve un número como un número.	<pre>var x = 123; x.valueOf(); // returns 123 from variable x (123).valueOf(); // returns 123 from literal 123 (100 + 23).valueOf(); // returns 123 from expression 100 + 23</pre>	

Convertir Variables a Números.	Métodos globales, son aquellos que se pueden utilizar en todos los tipos de datos de JavaScript.
Number () puede ser usado para convertir las variables JavaScript a los números:	<pre>x = true; Number(x); // returns 1 x = false; Number(x); // returns 0 x = new Date(); Number(x); // returns 1404568027739 x = "10" Number(x); // returns 10 x = "10 20" Number(x); // returns NaN</pre> <p>Se utiliza en la fecha (), el método del Número () devuelve el número de milisegundos desde el 1.1.1970.</p>
parseInt () analiza una cadena y devuelve un número entero. Se permiten espacios. Sólo se devuelve el primer número.	<pre>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 (No es un Número)</pre>
parseFloat () analiza una cadena y devuelve un número. Se permiten espacios. Sólo se devuelve el primer número.	<pre>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</pre>

Propiedades	
Los métodos globales de JavaScript se pueden utilizar con todos los tipos de datos de JavaScript	
MAX_VALUE	<code>var x = Number.MAX_VALUE;</code>
toExponential () devuelve una cadena, con un número redondeado y escrita usando la notación exponencial.	<pre> var x = 9.656; x.toExponential(2); // returns 9.66e+0 x.toExponential(4); // returns 9.6560e+0 x.toExponential(6); // returns 9.656000e+0 </pre>
toFixed () devuelve una cadena, con el número escrito con un número especificado de decimales:	<pre> 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 </pre>
toPrecision () devuelve una cadena, con un número escrito con una longitud especificada:	<pre> 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 </pre>
valueOf () devuelve un número como un número.	<pre> var x = 123; x.valueOf(); // returns 123 from variable x (123).valueOf(); // returns 123 from literal 123 (100 + 23).valueOf(); // returns 123 from expression 100 + 23 </pre>

Convertir Variables a Números. Métodos globales, son aquellos que se pueden utilizar en todos los tipos de datos de JavaScript.	
Number () puede ser usado para convertir las variables JavaScript a los números:	<pre> x = true; Number(x); // returns 1 x = false; Number(x); // returns 0 x = new Date(); Number(x); // returns 1404568027739 x = "10"; Number(x); // returns 10 x = "10 20"; Number(x); // returns NaN </pre> <p>Se utiliza en la fecha (), el método del Número () devuelve el número de milisegundos desde el 1.1.1970.</p>
parseInt () analiza una cadena y devuelve un número entero. Se permiten espacios. Sólo se devuelve el primer número.	<pre> 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 (No es un Número) </pre>
parseFloat () analiza una cadena y devuelve un número. Se permiten espacios. Sólo se devuelve el primer número.	<pre> 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 </pre>

Valores Constantes		
MAX_VALUE máximo valor en JS	<pre> <!DOCTYPE html> <html> <body> <p id="p1"></p> <p id="p2"></p> <p id="p3"></p> <p id="p4"></p> <p id="p5"></p> <p id="p6"></p> <script> var x = 6; document.getElementById("p1").innerHTML = Number.MAX_VALUE; document.getElementById("p2").innerHTML = Number.MIN_VALUE; document.getElementById("p3").innerHTML = Number.NEGATIVE_INFINITY; document.getElementById("p4").innerHTML = Number.POSITIVE_INFINITY; document.getElementById("p5").innerHTML = Number.NaN; document.getElementById("p6").innerHTML = x.NaN; </script> </body> </html> </pre>	1.7976931348623157e+308
MIN_VALUE mínimo valor en JS		5e-324
NEGATIVE_INFINITY infi. Negativo		-Infinity
overflow		Infinity
NaN → No es un Número		NaN
POSITIVE_INFINITY infi. Positivo		undefined

Chuleta:	Fechas	Date
<p>El objeto Date le permite trabajar con fechas (años, meses, días, horas, minutos, segundos y milisegundos) y las fechas en JavaScript puede ser escritas:</p> <ul style="list-style-type: none"> - Como una cadena: <code>Sun Oct 23 2016 21:48:38 GMT+0200</code> - Como un número: <code>1477252119175</code> → número de milisegundos desde el 1 de enero de 1970, 00:00:00. 		
El Objeto Date		
<p>Los Objetos Fecha, una vez creados son Estáticos, aunque la computadora siga corriendo en tiempo.</p> <p>Al conseguir una Fecha cogemos siempre la del Navegador.</p>	<p>El <u>objeto Date</u> nos permite trabajar con fechas. La fecha se compone de un año, un mes, un día, una hora, un minuto, un segundo y milisegundos. Los Objetos Fecha se crean con el constructor <code>new Date()</code>.</p> <p><u>Hay 4 formas de iniciar una fecha:</u></p> <pre> new Date() new Date(milliseconds) new Date(dateString) new Date(year, month, day, hours, minutes, seconds, milliseconds) </pre>	
<pre> <p id="demo"></p> <script> document.getElementById("demo").innerHTML = Date(); </script> </pre>		
<pre> <script> var d = new Date(); document.getElementById("demo").innerHTML = d; </script> </pre>		
<pre> <script> var d = new Date("October 13, 2014 11:13:00"); document.getElementById("demo").innerHTML = d; </script> </pre>		
<pre> <script> var d = new Date(86400000); document.getElementById("demo").innerHTML = d; </script> </pre>		
<pre> <script> var d = new Date(99,5,24,11,33,30,0); document.getElementById("demo").innerHTML = d; </script> </pre>		
<pre> <script> var d = new Date(99,5,24); document.getElementById("demo").innerHTML = d; </script> </pre>		

Creación de la Fecha.
<p>Cuando se crea un objeto Date, una serie de métodos nos permiten operar con él.</p> <p>Los métodos con las Fechas nos permiten obtener y establecer el año, mes, día, hora, minuto, segundo y milisegundo de objetos, utilizando la hora local o UTC (Universal, o GMT).</p>
<p>Al mostrar un objeto de fecha en HTML, se convierte automáticamente en una cadena, con el método <code>toString ()</code>.</p> <pre> <p id="demo"></p> <script> d = new Date(); document.getElementById("demo").innerHTML = d; // document.getElementById("demo").innerHTML = d.toString(); → Es lo mismo </script> </pre>
<p>El método <code>toUTCString ()</code> convierte una fecha en una cadena UTC (un estándar de visualización de la fecha).</p> <p>El método <code>toDateStrig ()</code> convierte una fecha a un formato más legible:</p> <pre> <script> var d = new Date(); document.getElementById("demo").innerHTML = d.toUTCString(); document.getElementById("demo").innerHTML = d.toDateStrig(); </script> </pre>

Formatos

Los métodos globales de JavaScript se pueden utilizar con todos los tipos de datos de JavaScript.

Al establecer una fecha, sin especificar la zona horaria, JavaScript utilizará la zona horaria del navegador.

En otras palabras: si se crea una fecha / hora GMT (Greenwich Mean Time), la fecha / hora se convertirá en CST (Central de Estados Unidos Hora de verano) si un usuario navega desde el centro de Estados Unidos.

Fechas de Entrada

Existen 4 tipos de formatos de entrada Fecha de JavaScript:

<u>Tipo</u>	<u>Ejemplo</u>
ISO Fecha	"03/25/2015" (La Norma Internacional)
Cita corta	"03.25.2015" o "25/03/2015"
Fecha larga	"Mar 25 de 2015" o "25 mar 2015"
Fecha completa	"Miércoles 25 marzo 2015"

Fechas de Salida

Por defecto son en formato de **cadena de texto**.

Wed Mar 25 2015 01:00:00 GMT+0100 (Hora estándar)

ISO 8601 es la norma internacional para la representación de fechas y horas.

En la mayoría de los navegadores, data ISO con meses o días **sin ceros a la izquierda** será **interpretado como fechas cortas**

La sintaxis de la norma ISO 8601 (**AAAA-MM-DD**) es también el formato de fecha JavaScript preferido:

```
var d = new Date("2015-03-25");
```

Se puede escribir sin especificar el día (AAAA-MM): **Año y Mes**

```
var d = new Date("2015-03");
```

Se puede escribir sólo el año: **Año**

```
var d = new Date("2015");
```

Fecha completa más **horas, minutos y segundos**:

```
var d = new Date("2015-03-25T12:00:00");
```

Formato Corto: **"MM / DD / AAAA"**

```
var d = new Date("03/25/2015");
```

Formato Corto: **"AAAA / MM / DD"**

```
var d = new Date("2015/03/25");
```

Formato Largo: "MMM DD AAAA"

```
var d = new Date("Mar 25 2015");
```

```
var d = new Date("25 Mar 2015");
```

 Mes y día puede estar en cualquier orden.

```
var d = new Date("En 25 2015");
```

 Se puede poner abreviado

Cadena Completa:

```
var d = new Date("Wed Mar 25 2015 09:56:24 GMT+0100 (W. Europe Standard Time)");
```

Métodos para Obtener una parte de la fecha o para Establecer una parte de la fecha.

Métodos **get**

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

getTime () devuelve el número de milisegundos desde 1 de enero de 1970:

getFullYear () devuelve el año de una fecha como un número de cuatro dígitos:

getDay () devuelve el día de la semana como un número (0-6):

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getFullYear();

document.getElementById("demo").innerHTML = d.getDay();

var days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];
document.getElementById("demo").innerHTML = days[d.getDay()];
</script>
```

Métodos **set**

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

setFullYear () define un objeto de fecha a una fecha específica. En este ejemplo, al 14 de enero, 2020:

setDate () establece el día del mes (1-31):

El método **setDate ()** también se puede utilizar para **agregar días** a una fecha:

```
<script>
var d = new Date();
d.setFullYear(2020, 0, 14);
document.getElementById("demo").innerHTML = d;
</script>

<script>
var d = new Date();
d.setDate(20);
document.getElementById("demo").innerHTML = d;
</script>

<script>
var d = new Date();
d.setDate(d.getDate() + 50);
document.getElementById("demo").innerHTML = d;
</script>
```

Comparar Fechas:

El siguiente ejemplo compara la fecha de hoy con el 14 de enero de, 2100:

```
var today, someday, text;
today = new Date();
someday = new Date();
someday.setFullYear(2100, 0, 14);

if (someday > today) {
    text = "Today is before January 14, 2100.";
} else {
    text = "Today is after January 14, 2100.";
}
document.getElementById("demo").innerHTML = text;
```