Material Imprimible

Curso de JavaScript

Módulo 5

Método window prompt()

Definición y uso

El método prompt() muestra un cuadro de diálogo que solicita al visitante la entrada.

A menudo se utiliza un cuadro de aviso si desea que el usuario introduzca un valor antes de entrar en una página.

Nota: Cuando aparece un cuadro de aviso, el usuario tendrá que hacer clic en "Aceptar" o "Cancelar" para continuar después de introducir un valor de entrada. No utilice en exceso este método, ya que impide que el usuario acceda a otras partes de la página hasta que se cierre el cuadro.

El método prompt() devuelve el valor de entrada si el usuario hace clic en "Aceptar". Si el usuario hace clic en "cancelar", el método devuelve null.

Sintaxis

prompt(text, defaultText)

Técnicas

Valor de retorno: Una cadena. Si el usuario hace clic en "Aceptar", se devuelve el valor de entrada. Si el usuario hace clic en "cancelar", se devuelve null. Si el usuario hace clic en Aceptar sin escribir ningún texto, se devuelve una cadena vacía.

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

```
<p>Click the button to demonstrate the prompt box.</p>

<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var person = prompt("Please enter your name", "Harry Potter");
  if (person != null) {
    document.getElementById("demo").innerHTML =
            "Hello " + person + "! How are you today?";
  }
}
</script>
</body></html>

 <!DOCTYPE html>
<html>
<body>

<p>Click the button to display a dialog box which will ask for your favorite drink.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var text;
  var favDrink = prompt("What's your favorite cocktail drink?", "Daiquiri");
  switch(favDrink) {
            case "Martini":
```

```
        text = "Excellent choice. Martini is good for your soul.";
        break;
        case "Daiquiri":
        text = "Daiquiri is my favorite too!";
        break;
        case "Cosmopolitan":
        text = "Really? Are you sure the Cosmopolitan is your favorite?";
        break;
        default:
        text = "I have never heard of that one..";
 }
 document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

Método Window alert()

Definición y uso

El método alert() muestra un cuadro de alerta con un mensaje especificado y un botón Aceptar.

A menudo se utiliza un cuadro de alerta si desea asegurarse de que la información llegue al usuario.

Nota: El cuadro de alerta quita el foco de la ventana actual y obliga al explorador a leer el mensaje. No utilice en exceso este método, ya que impide que el usuario acceda a otras partes de la página hasta que se cierre el cuadro.

Sintaxis

alert(*message*)

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

<p>Click the button to display an alert box.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("Hello! I am an alert box!");
}
</script>

</body>
</html>
```

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

<p>Click the button to demonstrate line-breaks in an alert box.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("Hello\nHow are you?");
}
</script>
```

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

<p>Click the button to alert the hostname of the current URL.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert(location.hostname);
}
</script>

</body>
</html>
```

Método confirm() de ventana

Definición y uso

El método confirm() muestra un cuadro de diálogo con un mensaje especificado, junto con un botón Aceptar y Cancelar.

A menudo se utiliza un cuadro de confirmación si desea que el usuario verifique o acepte algo.

Nota: El cuadro de confirmación quita el foco de la ventana actual y obliga al navegador a leer el mensaje. No utilice en exceso este método, ya que impide que el usuario acceda a otras partes de la página hasta que se cierre el cuadro.

El método confirm() devuelve true si el usuario clic en "Aceptar", y false en caso contrario.

Sintaxis

confirm(message)

Técnicas

Valor de retorno: Un valor booleano, que indica si se hizo clic en "Aceptar" o "Cancelar" en el cuadro de diálogo:

true - el usuario a hecho clic en "Aceptar"

false - el usuario ratón en "Cancelar" (o el botón "x" (cerrar) en la esquina superior derecha que está disponible en todos los navegadores principales, excepto Firefox)

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

<p>Click the button to display a confirm box.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var txt;
  var r = confirm("Press a button!");
  if (r == true) {
        txt = "You pressed OK!";
  } else {
        txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML = txt;
}
```

```
</script>

</body>
</html>
```

Entrada y salida de datos con formularios

```
<!DOCTYPE html>
<html>
<head>
	<title>Entrada y salida de datos...</title>
</head>
<body>
	<form action="">
	<input type="text" id="cuadro" >
	<input type="submit" value="Aceptar" onclick="var valor=
document.getElementById('cuadro').value;
	alert(valor);
	">
	</form>
</body>
</html>
```

Longitud de la cadena

Para encontrar la longitud de una cadena, utilice la propiedad integrada:length

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

<h2>JavaScript String Properties</h2>

<p>The length property returns the length of a string:</p>
```

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

<script>
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
document.getElementById("demo").innerHTML = txt.length;
</script>

</body>
</html>
```

Carácter de escape

Dado que las cadenas deben escribirse entre comillas, JavaScript malinterpretará esta cadena: var x = "We are the so-called "Vikings" from the north."; La cuerda será cortada a "Somos los llamados".

La solución para evitar este problema, es utilizar el **carácter de escape de barra diagonal invertida**.

El carácter de escape de barra diagonal invertida () convierte caracteres especiales en caracteres de cadena:\

Code	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

La secuencia inserta una comilla doble en una cadena:\"

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

<h2>JavaScript Strings</h2>

<p>The escape sequence \" inserts a double quote in a string.</p>
```

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

<script>
var x = "We are the so-called \"Vikings\" from the north.";
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

La secuencia inserta una comilla simple en una cadena:\'

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

<h2>JavaScript Strings</h2>

<p>The escape sequence \' inserts a single quote in a string.</p>

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

<script>
var x = 'It\'s alright.';
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

La secuencia inserta una barra diagonal invertida en una cadena:\\

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

```
<h2>JavaScript Strings</h2>

<p>The escape sequence \\ inserts a backslash in a string.</p>

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

<script>
var x = "The character \\ is called backslash.";
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

tras seis secuencias de escape son válidas en JavaScript:

Code	Result
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

Romper largas líneas de código

Para una mejor legibilidad, a los programadores a menudo les gusta evitar las líneas de código de más de 80 caracteres.

Si una instrucción JavaScript no cabe en una línea, el mejor lugar para romperla es después de un operador:

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

```
<body>

<h2>JavaScript Statements</h2>

<p>
The best place to break a code line is after an operator or a comma.
</p>

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

<script>
document.getElementById("demo").innerHTML =
"Hello Dolly!";
</script>

</body>
</html>
```

También puede dividir una línea de código **dentro de una cadena** de texto con una sola barra diagonal invertida:

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

<h2>JavaScript Strings</h2>

<p>
You can break a code line within a text string with a backslash.
</p>

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

```
<script>
document.getElementById("demo").innerHTML = "Hello \
Dolly!";
</script>

</body>
</html>
```

Una forma más segura de romper una cadena es usar la adición de cadenas:

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

<h2>JavaScript Strings</h2>

<p>The safest way to break a code line in a string is using string addition.</p>

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

<script>
document.getElementById("demo").innerHTML = "Hello " +
"Dolly!";
</script>

</body>
</html>
```

No se puede dividir una línea de código con una barra diagonal invertida

Las cadenas pueden ser objetos

Normalmente, las cadenas de JavaScript son valores primitivos, creados a partir de literales:

```
var firstName = "John";
```

Pero las cadenas también se pueden definir como objetos con la palabra clave:new

```
var firstName = new String("John");
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>

<script>
var x = "John";             // x is a string
var y = new String("John");  // y is an object

document.getElementById("demo").innerHTML =
typeof x + "<br>" + typeof y;
</script>

</body>
</html>
```

Encontrar una cadena en una cadena

El método devuelve el índice de (la posición de) la aparición de un texto especificado en una cadena:indexOf()first

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

<h2>JavaScript String Methods</h2>

<p>The indexOf() method returns the position of the first occurrence of a specified text:</p>

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

<script>
```

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
document.getElementById("demo").innerHTML = pos;
</script>

</body>
</html>
```

JavaScript cuenta las posiciones desde cero.

0 es la primera posición en una cadena, 1 es la segunda, 2 es la tercera ...

El método devuelve el índice de la **última** aparición de un texto especificado en una cadena:lastIndexOf()

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

<h2>JavaScript String Methods</h2>

<p>The lastIndexOf() method returns the position of the last occurrence of a specified text:</p>

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

<script>
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
document.getElementById("demo").innerHTML = pos;
</script>

</body>
</html>
```

Ambos y devolver -1 si no se encuentra el texto.indexOf()lastIndexOf()

```
<!DOCTYPE html>
```

```
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>Both indexOf(), and lastIndexOf() return -1 if the text is not found:</p>

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

<script>
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("John");
document.getElementById("demo").innerHTML = pos;
</script>

</body>
</html>
```

Extracción de piezas de cadena

Hay 3 métodos para extraer una parte de una cadena:

- slice(*start, end*)
- substring(*start, end*)
- substr(*start, length*)

El método slice()

slice() extrae una parte de una cadena y devuelve la parte extraída en una nueva cadena.

El método toma 2 parámetros: la posición inicial y la posición final (final no incluido).

En este ejemplo se corta una parte de una cadena de la posición 7 a la posición 12 (13-1):

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

```
<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

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

<script>
var str = "Apple, Banana, Kiwi";
var res = str.slice(7,13);
document.getElementById("demo").innerHTML = res;
</script>

</body>
</html>
```

Recuerde: JavaScript cuenta las posiciones desde cero. La primera posición es 0.

Si un parámetro es negativo, la posición se cuenta desde el final de la cadena.

En este ejemplo se corta una parte de una cadena de la posición -12 a la posición -6:

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

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

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

<script>
```

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(-12,-6);
document.getElementById("demo").innerHTML = res;
</script>

</body>
</html>
```

Si omite el segundo parámetro, el método cortará el resto de la cadena

o, contando desde el final

El método substring()

substring() es similar a .slice()

La diferencia es que no puede aceptar índices negativos.substring()

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

<h2>JavaScript String Methods</h2>

<p>The substring() method extract a part of a string and returns the extracted parts in a new
string:</p>

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

<script>
var str = "Apple, Banana, Kiwi";
var res = str.substring(7,13);
document.getElementById("demo").innerHTML = res;
</script>

</body>
```

```
</html>
```

El método substr()

substr() es similar a .slice()

La diferencia es que el segundo parámetro especifica la **longitud** de la pieza extraída.

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

<h2>JavaScript String Methods</h2>

<p>The substr() method extract a part of a string
and returns the extracted parts in a new string:</p>

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

<script>
var str = "Apple, Banana, Kiwi";
var res = str.substr(7,6);
document.getElementById("demo").innerHTML = res;
</script>

</body>
</html>
```

Sustitución de contenido de cadena

El método reemplaza un valor especificado por otro valor en una cadena:replace()

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

```
<body>

<h2>JavaScript String Methods</h2>

<p>Replace "Microsoft" with "W3Schools" in the paragraph below:</p>

<button onclick="myFunction()">Try it</button>

<p id="demo">Please visit Microsoft!</p>

<script>
function myFunction() {
  var str = document.getElementById("demo").innerHTML;
  var txt = str.replace("Microsoft","Google");
  document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

De forma predeterminada, el método distingue mayúsculas de minúsculas. Escribir MICROSOFT (con mayúsculas) no funcionará:replace() Para reemplazar mayúsculas y minúsculas, utilice una **expresión regular** con una marca (no sensible):/i

Extraer caracteres de cadena

Existen 3 métodos para extraer caracteres de cadena:

- charAt(*position*)
- charCodeAt(*position*)
- Acceso a la propiedad [ ]

El método charAt()

El método devuelve el carácter en un índice especificado (posición) en una cadena:charAt()

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

<h2>JavaScript String Methods</h2>

<p>The charAt() method returns the character at a given position in a string:</p>

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

<script>
var str = "HELLO WORLD";
document.getElementById("demo").innerHTML = str.charAt(0);
</script>

</body>
</html>
```

Convertir una cadena en una matriz

Una cadena se puede convertir en una matriz con el método:split()

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

<p>Click "Try it" to display the first array element, after a string split.</p>

<button onclick="myFunction()">Try it</button>
```

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

<script>
function myFunction() {
  var str = "a,b,c,d,e,f";
  var arr = str.split(",");
  document.getElementById("demo").innerHTML = arr[0];
}
</script>

</body>
</html>
```

Si se omite el separador, la matriz devuelta contendrá toda la cadena en el índice [0].

Si el separador es "", la matriz devuelta será una matriz de caracteres individuales:

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

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

<script>
var str = "Hello";
var arr = str.split("");
var text = "";
var i;
for (i = 0; i < arr.length; i++) {
  text += arr[i] + "<br>"
}
document.getElementById("demo").innerHTML = text;
</script>
```

```
</body>
</html>
```

Métodos de número de JavaScript

Los métodos numéricos le ayudan a trabajar con números.

Métodos y propiedades numéricas

Los valores primitivos (como 3.14 o 2014), no pueden tener propiedades y métodos (porque no son objetos).

Pero con JavaScript, los métodos y propiedades también están disponibles para los valores primitivos, porque JavaScript trata los valores primitivos como objetos al ejecutar métodos y propiedades.

El método toString()

El método devuelve un número como una cadena. toString()

Todos los métodos numéricos se pueden utilizar en cualquier tipo de número (literales, variables o expresiones)

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

<h2>JavaScript Number Methods</h2>

<p>The toString() method converts a number to a string.</p>

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

<script>
var x = 123;
document.getElementById("demo").innerHTML =
```

```
 x.toString() + "<br>" +
  (123).toString() + "<br>" +
  (100 + 23).toString();
</script>

</body>
</html>
```

El método toExponential()

toExponential() devuelve una cadena, con un número redondeado y escrito mediante notación exponencial.

Un parámetro define el número de caracteres detrás del punto decimal:

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

<h2>JavaScript Number Methods</h2>

<p>The toExponential() method returns a string, with the number rounded and written using exponential notation.</p>

<p>An optional parameter defines the number of digits behind the decimal point.</p>

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

<script>
var x = 9.656;
document.getElementById("demo").innerHTML =
  x.toExponential() + "<br>" +
  x.toExponential(2) + "<br>" +
  x.toExponential(4) + "<br>" +
```

```
 x.toExponential(6);
</script>

</body>
</html>
```

El parámetro es opcional. Si no lo especifica, JavaScript no redondeará el número.

El método toFixed()

toFixed() devuelve una cadena, con el número escrito con un número especificado de decimales:

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

<h2>JavaScript Number Methods</h2>

<p>The toFixed() method rounds a number to a given number of digits.</p>
<p>For working with money, toFixed(2) is perfect.</p>

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

<script>
var x = 9.656;
document.getElementById("demo").innerHTML =
 x.toFixed(0) + "<br>" +
 x.toFixed(2) + "<br>" +
 x.toFixed(4) + "<br>" +
 x.toFixed(6);
</script>

</body>
</html>
```

toFixed(2) es perfecto para trabajar con dinero.

El método toPrecision()

toPrecision() devuelve una cadena, con un número escrito con una longitud especificada:

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

<h2>JavaScript Number Methods</h2>

<p>The toPrecision() method returns a string, with a number written with a specified
length:</p>

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

<script>
var x = 9.656;
document.getElementById("demo").innerHTML =
  x.toPrecision() + "<br>" +
  x.toPrecision(2) + "<br>" +
  x.toPrecision(4) + "<br>" +
  x.toPrecision(6);
</script>

</body>
</html>
```

El método valueOf()

valueOf() devuelve un número como número.

```
<!DOCTYPE html>
```

```
<html>
<body>

<h2>JavaScript Number Methods</h2>

<p>The valueOf() method returns a number as a number:</p>

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

<script>
var x = 123;

document.getElementById("demo").innerHTML =
 x.valueOf() + "<br>" +
 (123).valueOf() + "<br>" +
 (100 + 23).valueOf();
</script>

</body>
</html>
```

Conversión de variables en números

Hay 3 métodos JavaScript que se pueden utilizar para convertir variables en números:

El método Number()

El método parseInt()

El método parseFloat()

Estos métodos no son métodos numéricos, sino métodos De JavaScript globales.

Métodos Globales de JavaScript

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

Estos son los métodos más relevantes, cuando se trabaja con números:

El método Number()

Number() se puede utilizar para convertir variables de JavaScript en números:

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

<h2>JavaScript Global Methods</h2>

<p>The Number() method converts variables to numbers:</p>

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

<script>
document.getElementById("demo").innerHTML =
 Number(true) + "<br>" +
 Number(false) + "<br>" +
 Number("10") + "<br>" +
 Number("  10") + "<br>" +
 Number("10  ") + "<br>" +
 Number(" 10  ") + "<br>" +
 Number("10.33") + "<br>" +
 Number("10,33") + "<br>" +
 Number("10 33") + "<br>" +
 Number("John");
</script>

</body>
</html>
```

Si el número no se puede convertir, (No es un número) se devuelve.NaN

El método Number() utilizado en las fechas

Number() también puede convertir una fecha en un número:

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

<h2>JavaScript Global Methods</h2>

<p>The Number() method can convert a date to a number:</p>

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

<script>
var x = new Date("2017-09-30");
document.getElementById("demo").innerHTML = Number(x);
</script>

</body>
</html>
```

El método anterior devuelve el número de milisegundos desde 1.1.1970. Number()

El método parseInt()

parseInt() analiza una cadena y devuelve un número entero. Se permiten espacios. Solo se devuelve el primer número

Si el número no se puede convertir, (No es un número) se devuelve.NaN

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

```
<h2>JavaScript Global Functions</h2>

<p>The global JavaScript function parseInt() converts strings to numbers:</p>

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

<script>
document.getElementById("demo").innerHTML =
 parseInt("10") + "<br>" +
 parseInt("10.33") + "<br>" +
 parseInt("10 6") + "<br>" +
 parseInt("10 years") + "<br>" +
 parseInt("years 10");
</script>

</body>
</html>
```

El método parseFloat()

parseFloat() analiza una cadena y devuelve un número. Se permiten espacios. Solo se devuelve el primer número

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

<h2>JavaScript Global Methods</h2>

<p>The parseFloat() method converts strings to numbers:</p>

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

```
<script>
document.getElementById("demo").innerHTML =
 parseFloat("10") + "<br>" +
 parseFloat("10.33") + "<br>" +
 parseFloat("10 6") + "<br>" +
 parseFloat("10 years") + "<br>" +
 parseFloat("years 10");
</script>

</body>
</html>
```

Si el número no se puede convertir, (No es un número) se devuelve.NaN

Fuentes: http://www.w3bai.com/es/jsref/met_win_confirm.html

https://www.w3bai.com/es/js/js_number_methods.html