

Quiz Fundamentos de programación

Estudiantes:

Kevin Bermudez Esquivel.
Bryan David Beita Mohs.



Profesor: Francisco Jose Jimenez Bonilla.

Índice

- Portada
- Funciones fecha / hora
- Función flecha
- Funciones de control de flujo
- Funciones para manipulación de DOM
- Conclusión

Funciones fecha / hora.

Este objeto almacena la fecha, la hora, y brinda métodos para administrarlas.

Por ejemplo, podemos usarlo para almacenar horas de creación o modificación, medir tiempo, o simplemente mostrar en pantalla la fecha actual.

Un objeto de la clase Date se puede crear de dos maneras distintas. Por un lado podemos crear el objeto con el día y hora actuales y por otro podemos crearlo con un día y hora distintos a los actuales. Esto depende de los parámetros que pasemos al construir los objetos.



Para crear un objeto con fecha y hora actuales colocamos los paréntesis vacíos al llamar al constructor de la clase Date.

```
let miFecha = new Date()
```

Para crear un objeto con un día y hora distintos de los actuales tenemos que indicar entre paréntesis el momento con que inicializar el objeto.

Hay varias maneras de expresar un día y hora válidas, por eso podemos construir una fecha guiándonos por varios esquemas. Estos son dos de ellos, suficientes para crear todo tipo de fechas y horas.

```
let miFecha = new Date(año,mes,dia,hora,minutos,segundos)  
let miFecha2 = new Date(año,mes,dia)
```

Usos

Visualización de Fechas y Horas: Puedes utilizarla para mostrar la fecha y hora actual en tu sitio web o aplicación.

Conversión de Formatos de Fecha: Puedes usar `Date` para convertir fechas de un formato a otro, lo que es útil cuando necesitas mostrar fechas de manera diferente o interactuar con sistemas que utilizan formatos de fecha específicos.

Validación de Fechas: Es útil para validar fechas ingresadas por los usuarios y asegurarse de que sean válidas y cumplan con ciertos criterios.

Trabajo con Zonas Horarias: La función `Date` te permite trabajar con zonas horarias y realizar conversiones entre ellas.

Algunas ventajas de esta función son:

Formatos Personalizables: Puedes personalizar cómo se muestran las fechas y horas, lo que es importante para adaptar la presentación de la información a las preferencias de los usuarios o a requisitos específicos.

Compatibilidad Multiplataforma: La función `Date` es una característica estándar de JavaScript y es compatible en todos los navegadores web modernos, lo que la convierte en una opción confiable para el desarrollo web.

Manipulación de Fechas: Puedes realizar cálculos y manipulaciones complejas con fechas y horas, como sumar o restar días, semanas, meses, años, horas y minutos.

Formatos Personalizables: Puedes personalizar cómo se muestran las fechas y horas, lo que es importante para adaptar la presentación de la información a las preferencias de los usuarios o a requisitos específicos.

Tipos

setDate()

Actualiza el día del mes.

getDay()

Devuelve el día de la semana. Por ejemplo, si la fecha corresponde con el martes te devolvería 2.

getTime()

Devuelve los milisegundos transcurridos entre el día 1 de enero de 1970 y la fecha correspondiente al objeto al que se le pasa el mensaje.

**Mostrar HORA
en Javascript**

6:42:19


JS

Función flecha

Las funciones flecha (llamadas también “funciones de flecha gorda”) son indudablemente una de las características más populares de ES6 (EcmaScript 6). Ellas introducen una nueva forma de escribir funciones concisas.

Las funciones flecha permiten definir de manera compacta una función convencional. Si la función tiene solamente una sentencia que devuelve un valor, el uso de funciones flecha nos permite eliminar las llaves y la palabra `return`. Incluso utilizando parámetros también podemos ver mucho más reducido el código.

¡Es mucho más corto! Podemos omitir las llaves y la sentencia `return` debido a los retornos implícitos.



Algunos tipos y ejemplos de estas funciones

Muchos parámetros.

Los paréntesis son obligatorios para estas funciones:

```
(x, y) => 42
```

Objetos literales.

Si estás devolviendo un objeto literal, debe ir entre paréntesis. Esto obliga al intérprete a evaluar lo que hay dentro de los paréntesis, y se devuelve el objeto literal.

```
x =>({ y: x })
```

“Cuerpo del bloque”.

Si tu función está en un bloque, debes usar también la sentencia return explícita:

```
var sumarValores = (x, y) => {  
  return x + y  
}
```

Un solo parámetro.

Con estas funciones, los paréntesis son opcionales:

```
x => 42 || (x) => 42
```

Desventajas

Más difíciles de depurar.

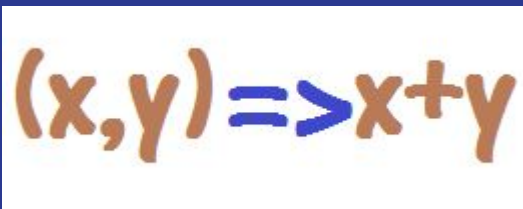
Cuando obtengas un error, no serás capaz de rastrear el nombre de la función o el número de línea exacto donde ocurrió.

Sin autorreferencia.

Si tu función necesita tener autorreferencia en algún punto (por ejemplo, recursión, controlador de evento que necesita desvincularse), no funcionará.

Principal beneficio

En las expresiones de función clásicas, la palabra reservada `this` está vinculada a diferentes valores en función del contexto en el que se llama. Sin embargo, con las funciones flecha, `this` está vinculada léxicamente. Esto significa que usa `this` desde el código que contiene la función flecha.


$$(x,y) \Rightarrow x+y$$

Un ejemplo de esta función

Función setTimeout:


En el ejemplo de ES5, es obligatorio `.bind(this)` para ayudar a pasar el contexto de `this` a la función. De lo contrario, `this` sería indefinido por defecto.

// ES5

```
var objeto = {  
  id: 42,  
  contador: function contador() {  
    setTimeout(function() {  
      console.log(this.id);  
    }.bind(this), 1000);  
  }  
};
```

// ES6

```
var objeto = {  
  id: 42,  
  contador: function contador() {  
    setTimeout(() => {  
      console.log(this.id);  
    }, 1000);  
  }  
};
```



```
1  setTimeout(() => console.log('timeout log'), 0);  
2  console.log('plain log');
```

Funciones de control de flujo

Las funciones de control de flujo en programación se refieren a las estructuras y sentencias que permiten dirigir el flujo de ejecución de un programa.

Estas funciones permiten tomar decisiones y ejecutar ciertos bloques de código en función de condiciones o criterios específicos.

Las funciones de control de flujo son esenciales para la lógica y la estructura de un programa, ya que determinan el orden en el que se ejecutan las instrucciones.



Tipos

Estructuras Condicionales:

Permiten ejecutar un bloque de código si se cumple una condición, como la sentencia if...else que ejecuta un bloque si una condición es verdadera y otro si es falsa.

Bucles:

Permiten repetir un bloque de código mientras se cumple una condición, como los bucles for, while, y do...while, que ejecutan el código repetidamente hasta que se cumple una condición de salida.

Importancia

Automatización:

Las funciones de control de flujo son fundamentales en la automatización de tareas. Por ejemplo, un script puede tomar decisiones y realizar acciones repetitivas en lugar de depender de la intervención manual.

Repetición de Tareas:

Los bucles (como for, while, do...while) son esenciales para realizar tareas repetitivas o iterar sobre colecciones de datos. Esto ahorra tiempo y esfuerzo al permitir que el programa realice la misma tarea varias veces sin tener que escribir el mismo código una y otra vez.

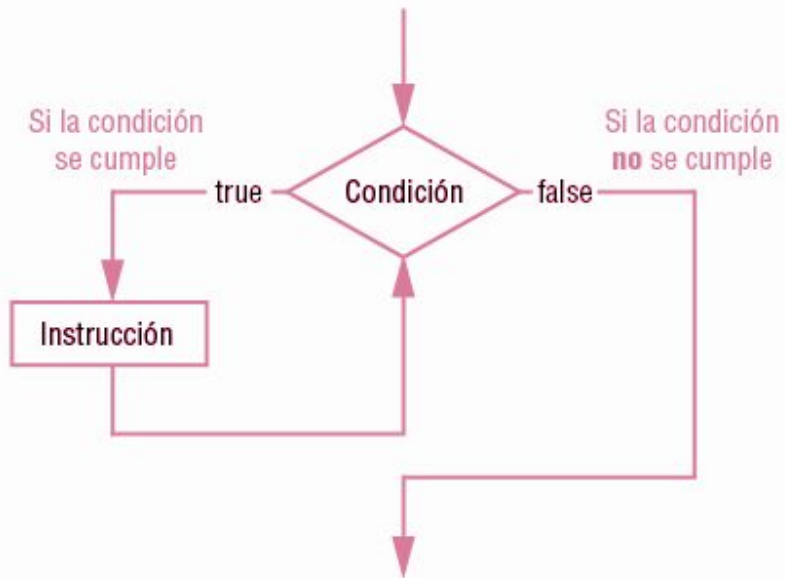
Ejemplos

Ejemplo de Bucle (for):

javascript

Copy code

```
for (let i = 1; i <= 5; i++) {  
    console.log(i);  
}
```



Ejemplo de Bucle (while):

javascript

Copy code


```
let numero = 2;  
  
while (numero <= 10) {  
    console.log(numero);  
    numero += 2;  
}
```

Funciones para manipulación de DOM

Son herramientas esenciales para interactuar dinámicamente con los elementos de una página web. El DOM es una representación en forma de árbol de la estructura de una página web, y mediante el uso de funciones del DOM, puedes realizar acciones como cambiar el contenido, el estilo y la estructura de la página en respuesta a eventos o acciones del usuario.

Algunas de las funciones clave para la manipulación del DOM incluyen `getElementById`, `querySelector`, `appendChild`, `removeChild`, `addEventListener`, y muchas más.

Estas funciones permiten a los desarrolladores web crear experiencias interactivas y dinámicas al tiempo que acceden y modifican elementos HTML y CSS en tiempo real. La manipulación del DOM es fundamental para el desarrollo web moderno y es la base de la mayoría de las aplicaciones web interactivas que vemos en línea.



Algunas características

Acceso a Elementos HTML:

Estas funciones permiten acceder a elementos HTML en una página web utilizando selectores, identificadores o rutas, lo que facilita la interacción con los elementos de la página.

Navegación en el DOM:

Las funciones de navegación en el DOM te permiten moverte de un elemento a otro, lo que es útil para buscar elementos relacionados o anidados.

Estilos y Clases:

Puedes modificar los estilos CSS de los elementos y agregar o quitar clases CSS para cambiar la apariencia y el comportamiento de los elementos en la página.

Manejo de Eventos:

Puedes utilizar funciones para adjuntar y gestionar eventos, como clics de ratón, pulsaciones de teclas y cambios en el estado de los elementos. Esto permite crear interacciones de usuario, como botones y formularios que responden a las acciones del usuario.

Mayores ventajas

Personalización de la Experiencia del Usuario:

Puedes personalizar la experiencia del usuario mostrando o ocultando elementos, cambiando estilos y contenidos en función de las preferencias y acciones del usuario.

Actualización de Contenido en Vivo:

Puedes actualizar el contenido de una página web en tiempo real sin necesidad de recargar la página completa, lo que mejora la eficiencia y la velocidad de la aplicación.

Interactividad en Tiempo Real:

Las funciones del DOM permiten que una página web responda de manera dinámica a las acciones del usuario, lo que crea una experiencia de usuario más interactiva y atractiva.

Validación de Formularios:

Las funciones del DOM son útiles para validar datos de entrada en formularios, como asegurarse de que los campos se completen correctamente antes de enviar la información.

Ejemplo

Acceso a Elementos HTML:

Para acceder a un elemento HTML por su identificador (ID) y cambiar su contenido:

javascript

Copy code

// Acceder al elemento con el ID "miElemento"

```
let elemento = document.getElementById("miElemento");
```

```
//Validate text box
if(document.getElementById("Message").value.length > 0)
{
    valid = false;
    document.getElementById("errMsgForm").innerHTML = "Empty!";
    document.getElementById("errMsgForm").style.display= "inline";
}
else{
    document.getElementById("errMsgForm").innerHTML = "";
    document.getElementById("errMsgForm").style.display = "none";
}
```

Conclusión

Este documento proporciona una comprensión de algunos de los conceptos clave en la programación JavaScript y el desarrollo web. Se han abordado varios temas esenciales, desde el manejo de fechas y horas hasta la creación de funciones utilizando funciones date, flecha, control de flujo y la manipulación del DOM en páginas web.

Podemos obtener un entendimiento base de cada función destacada anteriormente y algunos de los usos que podemos darle en el desarrollo web. Algunas de estas funciones han sido vistas en clases, sin embargo acá podemos destacar un poco ciertos conceptos y aplicaciones que podemos ejercer con ellas.

Este documento repasa las bases para una programación efectiva y la creación de aplicaciones web interactivas. Hemos indagado en conocimientos específicos que permitirán desarrollar aplicaciones web dinámicas, automatizar tareas y tomar decisiones basadas en condiciones específicas. Estos conceptos son esenciales para el desarrollo web y proporcionan una base sólida para futuros proyectos en el campo de la programación y el desarrollo web.