TALLER CONCEPTOS BÁSICOS Y PROYECTO PRÁCTICO DE UNA CALCULADORA:

Funciones en JavaScript

Las funciones son uno de los conceptos más importantes de JavaScript. En esta sección, exploraremos las funciones en JavaScript, los parámetros y argumentos de una función, cómo utilizar funciones anónimas y algunos ejercicios resueltos para la práctica de funciones en JavaScript.

Explicación de las funciones en JavaScript

En JavaScript, las funciones son bloques de código reutilizables que se utilizan para realizar una tarea específica. Las funciones pueden aceptar entradas (parámetros), realizar operaciones en esas entradas y devolver un resultado. Las funciones se utilizan para simplificar el código y hacerlo más legible y mantenible.

Descripción de los parámetros y argumentos de una función

Los parámetros son los valores que se pasan a una función cuando se la llama. Los argumentos son los valores reales que se pasan a la función cuando se la llama. Los parámetros se utilizan para aceptar entradas en una función y los argumentos son los valores que se proporcionan para esos parámetros.

Descripción de cómo utilizar funciones anónimas en JavaScript

Una función anónima es una función que no tiene nombre y se define en línea. Las funciones anónimas se utilizan a menudo como argumentos para otras funciones o para crear funciones de una sola vez que no necesitan ser llamadas más de una vez. Las funciones anónimas son útiles cuando se necesita una función temporal o cuando se desea encapsular un bloque de código.

Ejercicios para la práctica de funciones en JavaScript

En esta sección, presentaremos algunos ejercicios resueltos para ayudarte a practicar tus habilidades en funciones de JavaScript. Los ejercicios incluyen la definición de funciones, la creación de parámetros y argumentos y la utilización de funciones anónimas. Estos ejercicios te ayudarán a mejorar tus habilidades de programación en JavaScript y a comprender mejor cómo funcionan las funciones en este lenguaje de programación.

1. Crea una función que tome un array de números como parámetro y devuelva el número más grande del array:

```
function encontrarNumeroMayor(array) {
  let mayor = array[0];
  for (let i = 1; i < array.length; i++) {
    if (array[i] > mayor) {
      mayor = array[i];
    }
  }
  return mayor;
}
console.log(encontrarNumeroMayor([2, 7, 3, 9, 4])); // Output: 9
```

2. Crea una función que tome un array de strings como parámetro y devuelva un nuevo array con todos los strings en mayúsculas:

```
function convertirAMayusculas(array) {
  let nuevoArray = [];
  for (let i = 0; i < array.length; i++) {
    nuevoArray.push(array[i].toUpperCase());
  }
  return nuevoArray;
}
console.log(convertirAMayusculas(["hola", "mundo"])); // Output: ["HOLA", "MUNDO"]
```

3. Crea una función que tome dos números como parámetros y devuelva el resultado de elevar el primer número a la potencia del segundo número:

```
function elevarAExponente(base, exponente) {
  return Math.pow(base, exponente);
}
console.log(elevarAExponente(2, 3)); // Output: 8
```

4. Crea una función que tome un array de números como parámetro y devuelva la suma de todos los números del array:

```
function sumarNumeros(array) {
  let suma = 0;
  for (let i = 0; i < array.length; i++) {
    suma += array[i];
  }
  return suma;
}
console.log(sumarNumeros([1, 2, 3, 4])); // Output: 10
```

5. Crea una función que tome un objeto como parámetro y devuelva un array con todas las propiedades del objeto:

```
function obtenerPropiedades(objeto) {
  return Object.keys(objeto);
}
console.log(obtenerPropiedades({a: 1, b: 2, c: 3})); // Output: ["a", "b", "c"]
```

Objetos y Arrays en JavaScript

Explicación de los objetos y arrays en JavaScript

En JavaScript, los objetos y arrays son dos tipos de estructuras de datos que permiten almacenar y manipular información de manera eficiente. Los objetos son estructuras que pueden contener cualquier tipo de datos, como cadenas, números, booleanos e incluso otras estructuras de datos. Por otro lado, los arrays son listas ordenadas de elementos del mismo tipo de datos, como números o cadenas.

Descripción de cómo acceder a los elementos de un objeto o array

Para acceder a los elementos de un objeto en JavaScript, se utiliza la sintaxis de puntos o corchetes. Por ejemplo, si tenemos un objeto llamado «persona» con una propiedad «nombre», podemos acceder a ella de la siguiente manera: persona.nombre. Si la propiedad es un objeto anidado, se puede acceder utilizando la sintaxis de puntos en cascada: persona.direccion.calle.

Para acceder a los elementos de un array en JavaScript, se utiliza la sintaxis de corchetes y el índice del elemento que se desea acceder. Por ejemplo, si tenemos un array llamado «numeros» con cuatro elementos, podemos acceder al segundo elemento de la siguiente manera: numeros[1].

Descripción de cómo agregar o eliminar elementos de un objeto o array

Para agregar una propiedad a un objeto en JavaScript, se utiliza la sintaxis de puntos o corchetes. Por ejemplo, si queremos agregar una propiedad «edad» al objeto «persona», podemos hacerlo de la siguiente manera: persona.edad = 30. Si la propiedad es un objeto anidado, se puede agregar utilizando la sintaxis de puntos en cascada: persona.direccion.codigoPostal = 12345.

Para agregar un elemento a un array en JavaScript, se utiliza el método push(). Por ejemplo, si queremos agregar el número 5 al array «numeros», podemos hacerlo de la siguiente manera: numeros.push(5).

Para eliminar una propiedad de un objeto en JavaScript, se utiliza el operador delete. Por ejemplo, si queremos eliminar la propiedad «edad» del objeto «persona», podemos hacerlo de la siguiente manera: delete persona.edad.

Para eliminar un elemento de un array en JavaScript, se utiliza el método splice(). Por ejemplo, si queremos eliminar el segundo elemento del array «numeros», podemos hacerlo de la siguiente manera: numeros.splice(1, 1).

Ejercicios para la práctica de objetos y arrays en JavaScript

1. Crear un objeto «persona» con las propiedades «nombre» y «edad» y mostrarlas en la consola.

```
let persona = {
  nombre: "Juan",
  edad: 25
};

console.log(persona.nombre);
console.log(persona.edad);
```

2. Crear un array «numeros» con los números del 1 al 5 y mostrarlos en la consola.

```
let numeros = [1, 2, 3, 4, 5];

for (let i = 0; i < numeros.length; i++) {
  console.log(numeros[i]);
}
```

3. Agregar una propiedad «direccion» al objeto «persona» con las propiedades «calle» y «numero» y mostrarlas en la consola.

```
persona.direccion = {
  calle: "Calle 123",
  numero: 456
};

console.log(persona.direccion.calle);
console.log(persona.direccion.numero);
```

Eventos en JavaScript

Explicación de los eventos en JavaScript

Los eventos en JavaScript son acciones que ocurren en un documento, como hacer clic en un botón, mover el mouse, presionar una tecla, entre otros. Estos eventos pueden ser detectados por JavaScript, lo que permite que los desarrolladores web realicen acciones en respuesta a estos eventos. En otras palabras, los eventos en JavaScript permiten que los usuarios interactúen con los documentos web de manera dinámica.

Descripción de cómo utilizar eventos en JavaScript

Para utilizar eventos en JavaScript, se debe primero seleccionar el elemento HTML al que se desea agregar un evento. Luego, se debe especificar el tipo de evento que se desea detectar, como hacer clic en un botón. Una vez que se ha detectado el evento, se puede realizar una acción en respuesta a ese evento, como mostrar un mensaje o cambiar el contenido de la página.

Descripción de cómo crear y llamar a una función de evento

Para crear una función de evento en JavaScript, se debe primero definir la función que se desea llamar cuando ocurra el evento. Luego, se debe asignar esa función al evento utilizando el método adecuado, como addEventListener(). Este método permite especificar el tipo de evento y la función que se llamará cuando se produzca ese evento.

Para llamar a una función de evento en JavaScript, simplemente se debe realizar la acción que dispara el evento, como hacer clic en un botón. En ese momento, la función de evento que se ha asignado al evento se llamará automáticamente.

Ejercicios resueltos para la práctica de eventos en JavaScript

1. Cambiar el color de fondo al hacer clic en un botón:

```
<!DOCTYPE html>
<html>
<head>
	<title>Ejemplo de eventos en JavaScript</title>
</head>
<body>
	<button id="miBoton">Haz clic aquí</button>
	<script>
		var boton = document.getElementById("miBoton");
		boton.addEventListener("click", function() {
			document.body.style.backgroundColor = "yellow";
		});
	</script>
</body>
</html>
```

2. Mostrar un mensaje de alerta al pasar el mouse por encima de un elemento:

```
<!DOCTYPE html>
<html>
<head>
	<title>Ejemplo de eventos en JavaScript</title>
</head>
<body>
	<div id="miDiv">Pasa el mouse por aquí</div>
	<script>
		var div = document.getElementById("miDiv");
```

```
                div.addEventListener("mouseover", function() {
                        alert("¡Hola! Estás pasando el mouse por encima de mí.");
                });
        </script>
</body>
</html>
```

3. Cambiar el contenido de un elemento al presionar una tecla:

```
<!DOCTYPE html>
<html>
<head>
        <title>Ejemplo de eventos en JavaScript</title>
</head>
<body>
        <p id="miParrafo">Presiona una tecla para cambiar el contenido de este párrafo.</p>
        <script>
                var parrafo = document.getElementById("miParrafo");
                document.addEventListener("keydown", function(event) {
                        parrafo.innerHTML = "Presionaste la tecla " + event.key;
                });
        </script>
</body>
</html>
```

Proyecto práctico: Creación de una calculadora con JavaScript

Descripción del proyecto práctico

En este proyecto práctico, utilizaremos los conceptos de JavaScript que hemos aprendido en este artículo para crear una calculadora básica. La calculadora tendrá la capacidad de realizar operaciones matemáticas simples, como sumar, restar, multiplicar y dividir.

Para este proyecto, necesitaremos utilizar los elementos fundamentales de JavaScript, como las funciones, los objetos y los arrays. También utilizaremos los eventos de JavaScript para hacer que la calculadora sea interactiva y fácil de usar.

Explicación de cómo utilizar los conceptos de JavaScript aprendidos en el artículo para crear una calculadora

Primero, debemos crear la interfaz de usuario de la calculadora utilizando HTML y CSS. A continuación, podemos utilizar JavaScript para hacer que la calculadora funcione.

Para hacer esto, crearemos funciones para cada operación matemática que deseemos incluir en la calculadora. Por ejemplo, para la suma, podemos crear una función que tome dos números como argumentos y devuelva la suma de estos números.

Luego, podemos utilizar eventos de JavaScript para detectar cuándo el usuario hace clic en los botones de la calculadora. Cuando el usuario hace clic en un botón, podemos llamar a la función correspondiente y mostrar el resultado en la pantalla de la calculadora.

Para hacer que la calculadora sea más interactiva, también podemos agregar eventos para detectar cuando el usuario presiona la tecla «Enter» en el teclado, lo que permitirá al usuario utilizar la calculadora sin necesidad de hacer clic en los botones con el mouse.

En resumen, crear una calculadora con JavaScript es una excelente manera de poner en práctica los conceptos básicos que hemos aprendido en este artículo. Al utilizar funciones, objetos, arrays y eventos, podemos crear una calculadora interactiva y funcional que será útil para cualquier persona que necesite realizar operaciones matemáticas simples en su trabajo o en su vida diaria.

Ejercicios para la práctica de la creación de una calculadora con JavaScript

¡Excelente! Ya aprendiste los conceptos básicos, funciones, objetos, arrays y eventos en JavaScript. Ahora, vamos a poner en práctica todo lo que hemos aprendido con un proyecto práctico de creación de una calculadora utilizando JavaScript.

Aquí te presento algunos ejercicios resueltos para la práctica de la creación de una calculadora con JavaScript:

Ejercicio 1: Crear una función para realizar una suma entre dos números ingresados por el usuario.

```
function suma() {
  let num1 = parseInt(prompt("Ingrese el primer número: "));
  let num2 = parseInt(prompt("Ingrese el segundo número: "));
  let resultado = num1 + num2;
  alert("El resultado de la suma es: " + resultado);
}
```

Ejercicio 2: Crear una función para realizar una resta entre dos números ingresados por el usuario.

```
function resta() {
  let num1 = parseInt(prompt("Ingrese el primer número: "));
  let num2 = parseInt(prompt("Ingrese el segundo número: "));
  let resultado = num1 - num2;
  alert("El resultado de la resta es: " + resultado);
}
```

Ejercicio 3: Crear una función para realizar una multiplicación entre dos números ingresados por el usuario.

```
function multiplicacion() {
  let num1 = parseInt(prompt("Ingrese el primer número: "));
  let num2 = parseInt(prompt("Ingrese el segundo número: "));
  let resultado = num1 * num2;
  alert("El resultado de la multiplicación es: " + resultado);
}
```

Ejercicio 4: Crear una función para realizar una división entre dos números ingresados por el usuario.

```
function division() {
  let num1 = parseInt(prompt("Ingrese el primer número: "));
  let num2 = parseInt(prompt("Ingrese el segundo número: "));
  let resultado = num1 / num2;
  alert("El resultado de la división es: " + resultado);
}
```

Ejercicio 5: Crear una función para calcular el porcentaje de un número ingresado por el usuario.

```
function porcentaje() {
  let num1 = parseInt(prompt("Ingrese el número: "));
  let porcentaje = parseInt(prompt("Ingrese el porcentaje a calcular: "));
  let resultado = (porcentaje / 100) * num1;
  alert("El " + porcentaje + "% de " + num1 + " es: " + resultado);
}
```

Con estos ejemplos, ya puedes empezar a crear tu propia calculadora utilizando JavaScript. ¡Ponte en práctica!

Conclusiones

Resumen de los conceptos aprendidos en el artículo

En este taller, hemos aprendido los conceptos fundamentales de JavaScript, incluyendo variables, operadores, condicionales, ciclos, funciones, objetos, arrays y eventos. Además, hemos practicado estos conceptos a través de ejercicios y un proyecto práctico para crear una calculadora en JavaScript.

Descripción de los beneficios de aprender JavaScript

Aprender JavaScript tiene numerosos beneficios. Primero, es uno de los lenguajes de programación más populares y utilizados en todo el mundo, lo que significa que hay una gran cantidad de recursos y herramientas disponibles para aprender y utilizar. Además, JavaScript es un lenguaje versátil que se puede utilizar tanto en el lado del cliente como en el del servidor, lo que lo convierte en una habilidad valiosa para cualquier desarrollador web.

Otro beneficio de aprender JavaScript es que es un lenguaje en constante evolución, lo que significa que siempre hay nuevas características y funcionalidades que aprender y utilizar. Además, JavaScript es una habilidad altamente demandada en el mercado laboral, lo que puede llevar a oportunidades de trabajo y salarios más altos.

Validación de Formulario Con JavaScript

En el pasado, una vez que alguien había introducido toda su información y pulsado el botón de enviar, la validación de los formularios tenía lugar en el servidor.
Si faltaba algún dato o era inexacto, el servidor tenía que enviar toda la solicitud hacia atrás junto con un mensaje en el que se indicaba al destinatario que modificara primero el formulario antes de volver a enviarlo.
Era un proceso interminable que suponía una gran carga para el servidor.
Hoy en día, JavaScript ofrece diversos métodos para validar los datos de los formularios en el navegador antes de enviarlos al servidor la mejor manera es utilizar expresiones regulares, pero en este articulo veras una validación Básica de Formularios, para que puedas tener idea de como lo aplicarías en un proyecto simple.

Estructura de nuestro formulario con HTML

Para crear una validación de formulario con JavaScript, primero creamos la estructura de nuestro proyecto en el archivo index.html con un formulario sencillo y también haciendo referencia a nuestro archivo de JS y CSS

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Validaciones en JavaScript</title>
```

```
</head>

<body>

   <div class="container">
      <h2>BIENVENIDO</h2>
      <form id="formulario">
         <input type="text" id="usuario" placeholder="Nombre de Usuario">
         <input type="password" id="password" placeholder="Contraseña">
         <button type="submit" class="boton">Iniciar Sesión</button>
      </form>
   </div>

   <script src="app.js"></script>
</body>
</html>
```

Estilos con CSS para el formulario

Luego te dejo aquí unos estilos sencillos para que tenga un diseño agradable nuestro formulario para eso creamos un archivo con el nombre de style.css

```
body {

   background-image: linear-gradient(to right, rgb(20, 32, 48), rgb(13, 24, 38));

   font-family: Verdana, Geneva, Tahoma, sans-serif;

}

h1 {

   color: #6c5ce7;

   text-align: center;

   text-transform: uppercase;

   font-size: 60px;

}

.container {

   width: 400px;

   margin: 50px auto 0;

   padding: 15px;
```

```
    border-radius: 8px;
    background-color: #fff;
}
.container h2 {
    text-align: center;
    text-transform: uppercase;
    color: rgb(24, 32, 47);
}
form {
    display: flex;
    flex-direction: column;
    padding: 20px;
    gap: 15px;
}
input {
    padding: 10px;
    border: 1px solid rgba(102, 131, 140, 0.139);
    border-radius: 5px;
}
input:focus {
    outline: none;
    border-bottom: 2px solid #6c5ce7;
}
.boton {
    border: none;
```

```
    background-color: #6c5ce7;
    padding: 10px;
    color: #fff;
    cursor: pointer;
    border-radius: 5px;
    text-transform: uppercase;
}
.alerta {
    padding: 5px;
    border-radius: 5px;
    text-align: center;
    text-transform: uppercase;
    color: #fff;
}
.error {
    background-color: rgb(255, 0, 93);
}
.exito {
    background-color: rgb(27, 122, 46);
}
```

Validación en JavaScript

Y por último creamos la lógica de nuestro proyecto y para eso utilizamos JavaScript, entonces procedemos a crear nuestro archivo con el nombre de app.js

```
const formulario = document.querySelector('#formulario')

//evento
formulario.addEventListener("submit", validarFormulario);
```

```
//funcion validar el formulario al dar submit
function validarFormulario(e) {
    e.preventDefault();

    //ver los valores de los inputs
    const usuario = document.querySelector("#usuario").value
    const password = document.querySelector("#password").value

     // Lógica para validacion en JavaScript
    if (usuario == '' || password == "") {
        mostrarAlerta('Complete los campos', 'error');
    } else {
        if (password == '12345' && usuario == 'digitalnest') {
            mostrarAlerta('cargando...', 'exito');
            setTimeout(() => {
                window.location.href = "./paginaInicio.html"
            }, 2000);
            return
        }
        mostrarAlerta('Datos incorrectos', 'error');
    }
}

//Mostrar la alerta cuando agregamos datos incorrectos o vacíos
function mostrarAlerta(mensaje, tipo) {
    const alerta = document.createElement('div');
    alerta.className = `alerta ${tipo}`
    alerta.textContent = mensaje

    formulario.appendChild(alerta);
    setTimeout(() => {
        alerta.remove();
    }, 2000)
}
```

La validación de formularios en JavaScript es esencial para evitar el envío de datos incorrectos y asegurar la calidad de los datos recibidos. Con la verificación en tiempo real, es posible aplicar reglas de validación específicas a cada campo del formulario y mostrar mensajes de error claros y personalizados en caso de que los datos introducidos no cumplan con las condiciones requeridas como te muestro en el ejemplo prebio. La validación de correo electrónico y contraseñas también es importante para garantizar la seguridad de los datos del usuario. Al utilizar JavaScript y HTML juntos, es posible crear formularios eficientes y de fácil uso para los visitantes de un sitio web.

Vamos a hacer una pequeña Pausa

Evaluemos nuestro conocimiento hasta el momento...

Realizar test JavaScript, HTML y CSS

Quiz JavaScript Code

#1. ¿Qué empresa fue la que desarrolló JavaScript?
a) Microsoft
b) Google
c) Netscape
d) Apple

#2. ¿Para qué sirve el operador typeof en JavaScript?
a) Para comparar valores
b) Para asignar valores
c) Para verificar el tipo de una variable
d) Para concatenar cadenas

#3. ¿Qué es el operador ===?
a) Operador de asignación
b) Operador de igualdad estricta
c) Operador de suma
d) Operador de comparación

#4. ¿Qué es el Event Loop?
a) Un bucle que maneja eventos asíncronos
b) Una función en JavaScript
c) Un tipo de bucle for
d) Un método de ordenamiento

#5. ¿Qué es JavaScript?
a) Un lenguaje de programación de alto nivel
b) Un sistema operativo
c) Un navegador web
d) Un editor de texto

#6. ¿Cómo funcionan los "closures" en JavaScript?
a) Son funciones que devuelven un valor booleano
b) Son funciones que pueden acceder a variables de un ámbito superior
c) Son funciones que solo pueden ser llamadas una vez
d) Son funciones que no devuelven ningún valor

#7. ¿A qué se refiere la palabra "this" en JavaScript?
a) Al valor de retorno de una función
b) Al ámbito global
c) Al objeto actual en el contexto de ejecución
d) A una palabra reservada en JavaScript

#8. ¿Qué es el Hoisting en JavaScript?
a) Un error en la sintaxis de JavaScript
b) Un tipo de bucle
c) El comportamiento de mover las declaraciones de variables y funciones al inicio de su ámbito
d) Una técnica de optimización de código

#9. ¿Qué significa NULL en JavaScript?
a) Representa un valor nulo o vacío
b) Es un tipo de dato numérico
c) Indica un error en el código
d) Es una palabra reservada en JavaScript

#10. ¿Cuáles son los tipos de datos primitivos en JavaScript?
a) String, Number, Boolean, Object
b) Null, Undefined, Array, Function
c) Symbol, BigInt, Date, Array
d) String, Number, Boolean, Undefined

¡Claro! A continuación, te proporciono algunas preguntas de opción múltiple relacionadas con HTML y CSS:

Quiz HTML Code

#11. ¿Qué etiqueta HTML se utiliza para crear un enlace?
a) `<link>`
b) `<a>`
c) `<nav>`
d) `<button>`

#12. ¿Qué atributo se utiliza para especificar el idioma de una página web?
a) `lang`
b) `language`
c) `locale`
d) `idioma`

#13. ¿Qué etiqueta HTML se utiliza para crear una lista no ordenada?
a) `<ol>`
b) `<ul>`
c) `<li>`
d) `<dl>`

#14. ¿Qué atributo se utiliza para especificar el destino de un enlace?
a) `target`
b) `href`
c) `action`
d) `method`

#15. ¿Qué etiqueta HTML se utiliza para crear un formulario?
a) `<form>`
b) `<input>`
c) `<textarea>`
d) `<label>`

Quiz CSS Code

#16. ¿Qué propiedad CSS se utiliza para cambiar el color de texto?
a) `text-color`
b) `color`
c) `font-color`
d) `text-colorize`

#17. ¿Qué valor se utiliza para establecer un fondo transparente en CSS?
a) `transparent`
b) `clear`
c) `none`
d) `invisible`

#18. ¿Qué propiedad CSS se utiliza para definir el tamaño de una caja?
a) `size`
b) `dimension`
c) `box-size`
d) `width` y `height`

#19. ¿Qué valor se utiliza para centrar un elemento horizontalmente en CSS?
a) `center`
b) `horizontal`
c) `middle`
d) `auto`

#20. ¿Qué propiedad CSS se utiliza para definir el espaciado entre letras?
a) `letter-spacing`
b) `word-spacing`
c) `char-spacing`
d) `text-spacing`

Claro, aquí te dejo algunas preguntas adicionales sobre CSS:

#21. ¿Qué propiedad CSS se utiliza para definir el margen exterior de un elemento?
a) `padding`
b) `border`
c) `margin`
d) `space`

#22. ¿Qué valor se utiliza para establecer un borde redondeado en CSS?
a) `round`
b) `border-radius`
c) `radius`
d) `curve`

#23. ¿Qué propiedad CSS se utiliza para definir la fuente de un elemento?
a) `text-font`
b) `font-family`
c) `typeface`
d) `font-type`

#24. ¿Qué valor se utiliza para establecer una caja con sombra en CSS?
a) `shadow`
b) `box-shadow`
c) `drop-shadow`
d) `shadow-box`

#25. ¿Qué propiedad CSS se utiliza para definir el ancho y alto de una imagen?
a) `size`
b) `dimension`

c) `box-size`
d) `width` y `height`

#26. ¿Qué valor se utiliza para establecer un fondo degradado en CSS?
a) `gradient`
b) `background-gradient`
c) `linear-gradient`
d) `degrade`

#27. ¿Qué propiedad CSS se utiliza para definir el comportamiento de un elemento cuando se pasa el cursor sobre él?
a) `hover`
b) `onmouseover`
c) `cursor`
d) `pointer`

#28. ¿Qué valor se utiliza para establecer un fondo repetido en CSS?
a) `repeat`
b) `background-repeat`
c) `tile`
d) `repeat-all`

#29. ¿Qué propiedad CSS se utiliza para definir la posición de un elemento en la página?
a) `position`
b) `location`
c) `place`
d) `site`

#30. ¿Qué valor se utiliza para establecer un fondo fijo en CSS?
a) `fixed`
b) `background-fixed`
c) `sticky`
d) `lock`

lista de algunos métodos JavaScript comunes para mostrar la salida, junto con una breve descripción de su funcionalidad:

1. `document.write()`: Este método escribe el texto o código especificado en el documento HTML actual. Se puede usar para insertar contenido dinámicamente, pero debe usarse con precaución, ya que puede sobrescribir todo el sitio si se llama después de que la página haya terminado de cargar.
2. `alert()`: Este método muestra una caja de diálogo con un mensaje y un botón OK. Se usa comúnmente para mostrar mensajes simples al usuario, pero debe usarse con moderación, ya que puede interrumpir el flujo de trabajo del usuario.
3. `console.log()`: Este método escribe el mensaje especificado en la consola de JavaScript. Se usa comúnmente con fines de depuración, ya que permite ver la salida de variables y expresiones en tiempo de ejecución.
4. `innerHTML`: Esta propiedad permite obtener o establecer el contenido HTML de un elemento. Se puede usar para insertar o modificar contenido HTML de forma dinámica, pero debe usarse con precaución, ya que puede exponer la página a ataques de inyección de código si el contenido no se sanea adecuadamente.

5. `textContent`: Esta propiedad permite obtener o establecer el contenido de texto de un elemento. Es una alternativa más segura a `innerHTML`, ya que no interpreta el contenido como HTML.
6. `document.createElement()`: Este método crea un nuevo elemento HTML. Se puede usar para crear elementos de forma dinámica y luego insertarlos en la página utilizando métodos como `appendChild()` y `insertBefore()`.
7. `document.createTextNode()`: Este método crea un nuevo nodo de texto. Se puede usar para crear contenido de texto de forma dinámica y luego insertarlo en la página utilizando métodos como `appendChild()` y `insertBefore()`.
8. `document.getElementById()`: Este método recupera el elemento con el ID especificado. Se puede usar para seleccionar un elemento específico y modificar su contenido o comportamiento.
9. `document.getElementsByClassName()`: Este método recupera todos los elementos con la clase especificada. Se puede usar para seleccionar múltiples elementos y modificar su contenido o comportamiento.
10. `document.querySelector()`: Este método recupera el primer elemento que coincide con el selector especificado. Se puede usar para seleccionar un elemento específico y modificar su contenido o comportamiento.
11. `document.querySelectorAll()`: Este método recupera todos los elementos que coinciden con el selector especificado. Se puede usar para seleccionar múltiples elementos y modificar su contenido o comportamiento.
12. `document.addEventListener()`: Este método adjunta un listener de eventos al documento o a un elemento. Se puede usar para manejar interacciones de usuario y otros eventos, y ejecutar código en respuesta.

Estos son solo algunos ejemplos de los muchos métodos disponibles en JavaScript para mostrar la salida y manipular el documento. La elección del método depende de las necesidades y objetivos específicos del proyecto, así como de las habilidades y experiencia del equipo de desarrollo.
Eventos en JavaScript son acciones que ocurren en el navegador web, como hacer clic en un botón, cargar una página web, o cambiar el valor de un campo de entrada. Estos eventos pueden activar código JavaScript para realizar una acción específica en respuesta a la acción del usuario o al comportamiento del navegador.

EVENTOS Y DOM

La relación entre eventos y el DOM (Document Object Model) en JavaScript es que los eventos ocurren en los elementos del DOM, y JavaScript permite responder a esos eventos. El DOM es una representación estructurada del documento HTML, y los eventos ocurren en los elementos del DOM cuando el usuario interactúa con la página web o cuando el navegador realiza una acción específica.

La definición de DOM es la representación de un documento HTML o XML en forma de un árbol de nodos, donde cada nodo representa un elemento, atributo o texto en el documento[1][2][4]. El DOM permite a JavaScript interactuar con los elementos de la página web, como cambiar el texto de un elemento, agregar o quitar elementos, y responder a eventos que ocurren en los elementos del DOM.

En resumen, los eventos en JavaScript son acciones que ocurren en el navegador web, y el DOM es la representación estructurada del documento HTML que permite a JavaScript interactuar con los elementos de la página web y responder a eventos que ocurren en los elementos del DOM.
Los eventos son acciones que ocurren cuando interactuamos con una web, por ejemplo, cuando hacemos click en un botón, escribimos en un input text... son la forma que tienen de interactuar el html y el javascript. Cuando un evento se lanza, el navegador notifica a la aplicación que algo ha

ocurrido y que hay que manejarlo. Los eventos se controlan con funciones de javascript, de forma que cuando dicho evento ocurre, el navegador alerta a nuestra función que está escuchando que debe ejecutarse.

¿Cómo manejamos los eventos?

Hay dos formas de añadir eventos. Una desde el html con atributos añadiendo por ejemplo un onClick() en el elemento que queremos que desencadene la acción y otra desde el javascript utilizando el método addEventListener(). No sé si hay una forma mejor u otra peor, pero es mejor separar el javascript del html y evitar el antipatrón de mezclar html y js.

Sin embargo, con este enfoque “tenemos un problema” cuando queremos añadir el mismo evento a varios elementos del DOM. De esta forma, tendríamos tantos eventos como elementos del DOM que desencadenan la acción. Además, los elementos que se cargan de forma dinámica no se verían afectados por los eventos. Para solucionar estos problemas surge la delegación de eventos que es de lo que vamos a hablar en este post.

La delegación de eventos consiste en escuchar los eventos en el elemento padre para capturarlo cuando ocurra en sus hijos.

Antes de entrar en profundidad en la delegación de eventos, vamos a ver cómo funcionan los eventos con un ejemplo sencillo, añadiendo un evento a un botón. Aquí en el codesandbox podéis verlo.

```
const button = document.querySelector(".btn");

const changeTextButton = () => {
 button.classList.toggle("clicked");
};

button.addEventListener("click", changeTextButton);
```

¿Qué hemos hecho?

El método addEventListener es la forma más habitual que usamos para registrar eventos. Recibe dos parámetros: el tipo de evento que queremos “escuchar” y la acción (o función) que queremos ejecutar cuando el evento suceda.

¿Cómo funcionan los eventos?

Cuando clicamos en un botón, se desencadenan las siguientes fases:

- Fase de captura: el evento empieza en el window, document y en el root y luego entra dentro de cada uno de los hijos.
- Fase target: el evento se lanza sobre el elemento en el que se hace click.
- Fase bubble: finalmente el evento sube hacia los ancestros desde el target hasta llegar de nuevo al root, document y window.

En este ejemplo se puede ver cómo se propagan las diferentes fases de los eventos.

Desde el punto de vista del código, un evento no es más que un objeto donde se almacena información relacionada con el mismo.

Algunas de las propiedades que podemos encontrar en el objeto evento son las siguientes:

- event.type: es el tipo de evento que hemos lanzado, en el caso del ejemplo sería el click, pero podría ser cualquier otro.
- event.target: es el elemento en el que se ha disparado el evento.
- defaultPrevented: indica si el evento puede ser cancelado con event.preventDefault.
- event.Phase: indica qué fase del flujo de eventos se está procesando.
- event.currentTarget: es el elemento en el que declaramos el evento, es decir, en que hemos declarado el addEventListener.