Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ciencias de la Computación e Informática
Ingeniería de Software

Laboratorio #3b

2024

Autor

David González Villanueva - C13388

Profesora

Rebeca Obando Vásquez

Laboratorio #3

Repositorio de Github: https://github.com/d4v7d/IS_2024b_c13388

1. ¿Cuáles son los data types que soporta javascript ?

JavaScript soporta los siguientes tipos de datos:

- String.
- Number.
- Boolean.
- Undefined.
- Null.
- Object.

2. ¿Cómo se puede crear un objeto en javascript? De un ejemplo

Un objeto en javascript se utiliza con el siguiente formato de llames, se coloca el nombre del archivo y entre llaves se colocan sus atributos separados por comas.

```
const persona = {
        nombre: "Juan",
        edad: 30,
};
```

3. ¿Cuáles son los alcances (scope) de las variables en javascript?

JavaScript tiene tres tipos de alcance de variables:

- Global Scope.
- Function Scope.
- Block Scope.

4. ¿Cuál es la diferencia entre undefined y null?

- **undefined:** Es el valor por defecto de las variables que han sido declaradas pero no

- **null**: Es un valor asignado manualmente que indica que la variable no tiene ningún valor..

5. ¿Qué es el DOM?

El DOM (Document Object Model) es una representación estructurada del documento HTML o XML en la memoria. Permite a los lenguajes de programación manipular el contenido, la estructura y el estilo de un documento.

6. Usando Javascript se puede acceder a diferentes elementos del DOM. ¿Qué hacen, que retorna y para qué funcionan las funciones getElement y querySelector? Cree un ejemplo

- **getElementById:** Selecciona un elemento por su ID y retorna el primer elemento con ese ID.
- **querySelector:** Selecciona el primer elemento que coincida con un selector CSS.

```
// Suponiendo que existe un elemento con id "miElemento" en el HTML
const elementoPorId = document.getElementById('miElemento');

// Seleccionar un elemento usando un selector CSS
const elementoPorSelector = document.querySelector('.miClase');

console.log(elementoPorId); // Retorna el elemento con id "miElemento"
console.log(elementoPorSelector); // Retorna el primer elemento con la clase "miClase"
```

7. Investigue cómo se pueden crear nuevos elementos en el DOM usando Javascript. De un ejemplo

Puedes usar document.createElement() para crear un nuevo elemento y appendChild() para añadirlo al DOM.

```
// Crear un nuevo párrafo
const nuevoParrafo = document.createElement('p');
nuevoParrafo.textContent = 'Este es un nuevo párrafo creado con JavaScript';

// Añadir el nuevo párrafo al cuerpo del documento
document.body.appendChild(nuevoParrafo);
```

8. ¿Cuál es el propósito del operador this?

El operador this hace referencia al objeto en el contexto actual en el que se está ejecutando el código. Dentro de una función, this se refiere al objeto que invoca la función, osea a si mismo.

9. ¿Qué es un promise en Javascript?De un ejemplo

Un Promise es un objeto que representa la eventual finalización (o falla) de una operación asíncrona y su resultado.

```javascript
const miPromesa = new Promise((resolve, reject) => {
    const exito = true;

    if (exito) {
      resolve("La operación fue exitosa");
    } else {
      reject("La operación falló");
    }
  });

  miPromesa
    .then((mensaje) => console.log(mensaje))
    .catch((error) => console.log(error));
```

10.¿Qué es Fetch en Javascript? De un ejemplo

Fetch es una API que proporciona una interfaz para realizar solicitudes HTTP. Devuelve un Promise.

```javascript
fetch('https://api.example.com/datos')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

En este ejemplo se intenta acceder a una base de datos, si lo logra recibe los datos del .json con el fetch, si no logra entrar a la base de datos, avanza a las opciones then y si no logra acceder desde las opciones then, se queda en el catch y muestra el error.

11. ¿Qué es Asynch/Await en Javascript ? De un ejemplo

Async/Await es una forma más fácil de trabajar con Promises. async convierte una función en una promesa y await detiene la ejecución de la función hasta que la promesa se resuelva.

```javascript
async function obtenerDatos() {
    try {
      const respuesta = await fetch('https://api.example.com/datos');
      const datos = await respuesta.json();
      console.log(datos);
    } catch (error) {
      console.error('Error:', error);
    }
  }

  obtenerDatos();
```

12.¿Qué es un Callback? De un ejemplo

Un Callback es una función que se pasa como argumento a otra función y se ejecuta después de que la primera función ha terminado.

```
function saludar(nombre, callback) {
    console.log('Hola ' + nombre);
    callback();
}

function despedirse() {
    console.log('Adiós!');
}

saludar('Juan', despedirse);
```

13. ¿Qué es Closure?

Un Closure es una función que recuerda el entorno en el que se creó. Esencialmente, permite que una función interna acceda a las variables de una función externa, incluso después de que la función externa haya terminado de ejecutarse.

14.¿Cómo se puede crear un cookie usando Javascript?

Para crear una cookie usando document.cookie y agregando los atributos que se desea en la cookie.
ej:
document.cookie = "usuario=David; expires=Fri, 11 Feb 2024 23:59:59 GMT; path=/";

15.¿Cuál es la diferencia entre var, let y const?

- **var:** Tiene alcance de función y puede ser reasignada y redeclarada.
- **let:** Tiene alcance de bloque, puede ser reasignada, pero no redeclarada en el mismo ámbito.
- **const**: Tiene alcance de bloque y no puede ser reasignada ni redeclarada.