Fundamentos de JavaScript: Explicaciones Narrativas

1. Tipos de Datos en JavaScript

- Números: Valores enteros o decimales
- Textos: Contenedores que almacenan cadenas de caracteres
- Booleanos: Un interruptor que solo tiene dos posiciones
- Undefined: Un contenedor vacío sin propósito definido
- Null: Un espacio intencionalmente dejado en blanco
- Symbol: Un identificador único e irrepetible
- BigInt: Un contenedor para números extremadamente grandes

2. Creación de Objeto

```
let persona = {

   nombre: "María",

   edad: 28,

   profesión: "Desarrolladora"

};
```

3. Alcances (Scope) de Variables

- Scope global: Cualquiera puede utilizarlo
- Scope de función: Acceso restringido a la función
- Scope de bloque: Acceso a un bloque de código específico

4. Undefined vs Null

Undefined

- Indica que una variable ha sido declarada pero no tiene valor
- Es un tipo de dato primitivo

Null

- Representa la ausencia intencional de valor
- Debe ser asignado explícitamente

5. DOM (Document Object Model)

- Una representación jerárquica de documentos HTML/XML
- Permite manipular estructura, estilo y contenido web
- Funciona como una interfaz de programación para documentos
- Tiene una estructura jerárquica con relaciones padre-hijo
-

6. Métodos de Selección DOM

- // Selección por ID
- let elemento1 = document.getElementById('miElemento');

- // Selector CSS
- let elemento2 = document.querySelector('.miClase');
- let primerParrafo = document.querySelector('p');

- // Selector múltiple
- let todosParrafos = document.querySelectorAll('p');

7. Creación de Elementos DOM

- Se crea un nuevo bloque (elemento)
- Se le da color y forma etc (propiedades)
- Lo integras en tu construcción principal

```
// Crear elemento

let nuevoDiv = document.createElement('div');

// Configurar

nuevoDiv.textContent = 'Elemento Nuevo';

nuevoDiv.className = 'clase-ejemplo';

// Añadir al DOM

document.body.appendChild(nuevoDiv);
```

8. Operador this

- Referirse al contexto actual de ejecución
- Acceder a propiedades del objeto actual
- Variar según el contexto de llamada

```
let objeto = {

    nombre: "Objeto Ejemplo",

    metodo: function() {

        console.log(this.nombre);

    }

};
```

9. Promises

Representa operaciones asíncronas:

```
let Promesa = new Promise((resolve, reject) => {
    if (/* operación exitosa */) {
        resolve("Operación completada");
    } else {
        reject("Error en operación");
    }
});
```

10. Fetch

Para realizar solicitudes HTTPS:

```
fetch('https://api.ejemplo.com/datos')
    .then(respuesta => respuesta.json())
    .then(datos => console.log(datos))
    .catch(error => console.error(error));
```

11. Async/Await

Sintaxis para manejar promesas:

```
async function obtenerDatos() {
    try {
        let respuesta = await fetch('https://api.ejemplo.com/datos');
        let datos = await respuesta.json();
        console.log(datos);
    } catch (error) {
        console.error(error);
    }
```

Analogía::

- Tomas un turno en la fila
- Esperas pacientemente
- Mientras esperas, puedes hacer otras cosas

12. Callbacks

Utilizado para incluir una función como parámetro

```
function operacion(a, b, callback) {

    let resultado = a + b;

    callback(resultado);

}

operacion(5, 3, function(res) {

    console.log(res); // 8

});
```

13. Closures

Permite a una función "recordar" y acceder al entorno donde fue creada,incluso después de que esa función haya salido de su contexto original.

```
function crearContador() {

    let contador = 0;

    return function() {

        return ++contador;

    };
```

14. Cookies

```
// Crear cookie
document.cookie = "username=Juan; expires=Thu, 18 Dec 2023 12:00:00 UTC; path=/";

// Leer cookies
let cookies = document.cookie;
```

Como una tarjeta de identificación:

- Guarda información sobre un evento
- Caduca después de un tiempo

15. Variables: var, let, const

var

- Scope de función
- Puede ser re-declarada

let

- Scope de bloque
- No puede re-declararse en mismo scope

const

- Scope de bloque
- No puede re-asignarse
- No puede re-declararse

Referencias

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7th ed.). O'Reilly Media.

Simpson, K. (2015). *You Don't Know JS* (6-Volume Set). O'Reilly Media.

.