

## PROCESO DIRECCIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

### FORMATO GUÍA DE APRENDIZAJE

*Estimad@ aprendiz, la identificación de la presente guía está vacía para que usted pueda diligenciarla. Si tiene alguna duda pregúntele a su instructor líder.*

#### 1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- Denominación del Programa de Formación: Desarrollo de Aplicaciones en JavaScript
- Código del Programa de Formación:
- Nombre del Proyecto:
- Fase del Proyecto: Aplicación de fundamentos avanzados de JavaScript.
- Actividad de Proyecto: Desarrollo de una aplicación en JavaScript que integre el uso de variables, funciones, objetos y manejo de errores.
- Competencia: Utilizar estructuras y conceptos avanzados de JavaScript para resolver problemas.
- Duración de la Guía: 6 Horas

#### 2. PRESENTACIÓN

Esta guía te permitirá aplicar conocimientos avanzados de JavaScript. Aprenderás a definir variables, crear y usar funciones, trabajar con objetos, y manejar errores usando try...catch. Al finalizar, serás capaz de construir una aplicación que realice cálculos, gestione datos a través de objetos y controle errores de manera efectiva.

#### 3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

##### 3.1 ACTIVIDAD DE REFLEXIÓN INICIAL

Reflexiona sobre los siguientes puntos y responde en un documento:

- ¿Por qué crees que el manejo de errores es importante en el desarrollo de aplicaciones?
- ¿De qué manera el uso de funciones y objetos puede mejorar la organización de tu código?
- ¿Cómo te ayuda el control de variables y tipos en la reducción de errores?

### 3.2 ACTIVIDAD DE APROPIACIÓN

#### Paso 1: Definir Variables y Funciones Básicas

##### 1. Declaración de Variables

- Crea variables para almacenar datos de un producto: nombre, precio, cantidad y descuento.
- Asegúrate de usar const para datos constantes y let para datos variables.

Ejemplo:

```
const nombre = "Laptop";  
let precio = 1000;  
let cantidad = 2;  
let descuento = 0.1
```

##### 2. Función de Cálculo

- Define una función calcularTotal(precio, cantidad, descuento) que calcule el precio total después de aplicar el descuento.

Ejemplo de función:

```
function calcularTotal(precio, cantidad, descuento) {  
  return precio * cantidad * (1 - descuento);  
}
```

#### Paso 2: Crear un Objeto para Gestionar el Producto

##### 1. Definir el Objeto Producto

- Crea un objeto producto con las propiedades nombre, precio, cantidad, y descuento.
- Agrega un método calcularTotal dentro del objeto que utilice las propiedades del mismo objeto.

Ejemplo de objeto:

```
const producto = {  
  nombre: "Laptop",  
  precio: 1000,  
  cantidad: 2,  
  descuento: 0.1,  
  calcularTotal: function() {  
    return this.precio * this.cantidad * (1 - this.descuento);  
  }  
};  
console.log(`Total: ${producto.calcularTotal()}`);
```

### Paso 3: Agregar Manejo de Errores con try...catch

#### 1. Agregar Validación en el Método

- Actualiza el método calcularTotal para que incluya una verificación de valores (asegúrate de que precio, cantidad, y descuento sean números válidos).
- Utiliza try...catch para manejar posibles errores en el cálculo.

#### Ejemplo de manejo de errores:

```
const producto = {
  nombre: "Laptop",
  precio: 1000,
  cantidad: 2,
  descuento: 0.1,
  calcularTotal: function() {
    try {
      if (typeof this.precio !== 'number' || typeof this.cantidad !== 'number' || typeof this.descuento !== 'number') {
        throw new Error("Valores no válidos. Todos deben ser números.");
      }
      return this.precio * this.cantidad * (1 - this.descuento);
    } catch (error) {
      console.error(`Error: ${error.message}`);
      return 0;
    }
  }
};
```

#### 2. Prueba del Manejo de Errores

- Intenta llamar a producto.calcularTotal() después de asignar un valor no numérico a precio o cantidad para verificar que el manejo de errores funcione correctamente.

#### Ejemplo de prueba:

```
producto.precio = "mil"; // Valor incorrecto
console.log(`Total con error: ${producto.calcularTotal()}`);
```

### Paso 4: Ejercicio Práctico Integrador

#### 1. Ampliar el Objeto Producto

- Agrega un método detalles que devuelva una cadena describiendo el producto, incluyendo nombre, cantidad, precio y descuento.

#### Ejemplo de método detalles:

```
detalles: function() {
  return `Producto: ${this.nombre}, Precio: ${this.precio}, Cantidad: ${this.cantidad}, Descuento: ${this.descuento * 100}%`;
}
```

#### 2. Crear Función para Validar Datos

- Crea una función validarProducto(producto) que reciba un objeto y verifique que todas sus propiedades sean del tipo adecuado, arrojando un error si alguna no cumple.

#### Ejemplo de función validarProducto:

```
function validarProducto(producto) {
  try {
    if (typeof producto.precio !== 'number' || typeof producto.cantidad !== 'number' || typeof producto.descuento !== 'number') {
      throw new Error("Datos del producto no válidos.");
    }
    console.log("Producto válido.");
  } catch (error) {
    console.error(error.message);
  }
}
validarProducto(producto);
```

### 3.3. ACTIVIDAD DE TRANSFERENCIA DE CONOCIMIENTO

- **Consulta:** Investiga sobre la diferencia entre throw y return en el manejo de errores en JavaScript.
- **Evidencia:** Escribe un documento que explique la diferencia entre ambas y proporciona ejemplos en JavaScript.

Unifica las evidencias en una sólo archivo o carpeta, comprímelo y adjúntalo en el enlace que les comparta la instructora para tal fin.

**¡Recuerda! No importa que algunas de las actividades sean grupales, las evidencias se montan de forma individual, de tal forma que todos los integrantes de los equipos tengan su trabajo subido en plataforma.**

Si tienes dudas o inquietudes pregúntale a tu instructora.

### 3.4. ACTIVIDAD DE EVALUACIÓN

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
<b>Evidencias de Conocimiento:</b>	Implementación de variables y funciones en JavaScript.	Técnica: Evaluación Colaborativa Instrumento: Lista de chequeo
<b>Evidencias de Desempeño:</b>	Creación y manipulación de objetos con métodos y propiedades.	Técnica: Observación Instrumento: Lista de chequeo
<b>Evidencias de Producto:</b>	Uso de try...catch para el manejo de errores.	Técnica: Evaluación Colaborativa Instrumento: funcionalidad

#### 4. GLOSARIO DE TÉRMINOS:

- **Variable:** Espacio en la memoria para almacenar datos que pueden cambiar durante la ejecución.
- **Función:** Bloque de código que realiza una tarea específica.
- **Objeto:** Colección de propiedades y métodos que representan una entidad.
- **try...catch:** Estructura de control para manejar errores y excepciones.
- **Manejo de errores:** Técnicas para prevenir fallos en la ejecución del código y mejorar su estabilidad.

#### 5. REFERENTES BIBLIOGRÁFICOS

- [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Control\\_flow\\_and\\_error\\_handling](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Control_flow_and_error_handling)
- <https://eloquentjavascript.net/>

#### 6. CONTROL DEL DOCUMENTO

Nombre Autor (es)	Cargo	Dependencia		Fecha
Vanesa Medina Cuadrado	Instructora	ADSO	12 de noviembre de 2024	

#### 7. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

Nombre Autor (es)	Cargo	Dependencia		Fecha
	Instructor	ADSO	actualización	