Actividad 16 Javascript

Crea un documento HTML (página web) donde exista un botón “Crear cuenta bancaria”. Cuando el usuario pulse sobre el botón debe:

a) Pedirse al usuario un nombre de titular, apellidos de titular y saldo de la cuenta.

b) Crear un nuevo objeto cuentaBancaria que se inicializará con los datos facilitados por el usuario.

c) Mostrar un mensaje informando de que se ha creado la nueva cuenta bancaria y de los datos asociados a la cuenta bancaria creada.

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo aprenderaprogramar.com</title>
<meta charset="utf-8">
<script type="text/javascript">
function CuentaBancaria () {
this.nombre = "María";
this.apellidos = "Pérez";
this.saldo = 500000;
this.mostrarDatos = function () {
      var msg = 'Los datos de la cuenta son Nombre: ' + this.nombre;
      msg = msg + '; Apellidos: ' + this.apellidos + '; Saldo: ' + this.saldo;
      alert(msg);
      }
}

function ejemploCreaObjetos() {
var cuenta1 = new CuentaBancaria();
msg = 'El saldo en la cuenta bancaria de ‘ + cuenta1.nombre + ’ es: ' + cuenta1.saldo + ' euros';
alert (msg);
cuenta1.mostrarDatos();
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplo funciones JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploCreaObjetos()"> Probar </div>
</body>
</html>
```

Objetos y clases en JavaScript

¿Qué es un objeto?

Un **objeto** en JavaScript es una **estructura que agrupa información relacionada** dentro de una sola entidad. Cada objeto contiene **propiedades** (datos) y, opcionalmente, **métodos** (funciones que actúan sobre esos datos).

Ejemplo de objeto simple:

```
const cuentaBancaria = {
```

```
  nombreTitular: "Ana",
  apellidosTitular: "García López",
  saldo: 500.0
};
```

➡ En este ejemplo:

- `cuentaBancaria` es el **objeto**.
- `nombreTitular`, `apellidosTitular` y `saldo` son sus **propiedades**.
- Cada propiedad tiene un **valor asociado**.

Podemos acceder a las propiedades usando el **punto (`.`)**:

```
console.log(cuentaBancaria.nombreTitular); // Muestra: Ana
console.log(cuentaBancaria.saldo);         // Muestra: 500
```

También se pueden **modificar**:

```
cuentaBancaria.saldo = 700;
```

🔹 ¿Por qué usar objetos?

Usar objetos permite **organizar los datos** de manera más clara y lógica. En lugar de tener varias variables separadas (por ejemplo: `nombre`, `apellidos`, `saldo`), las agrupamos en una sola unidad (`cuentaBancaria`).

Esto facilita:

- La **lectura y mantenimiento** del código.
- La **reutilización** de estructuras.
- La **creación de varios objetos similares** sin repetir código.

🔹 ¿Qué es una clase?

Una **clase** es un **molde** o **plantilla** para crear muchos objetos con la misma estructura y comportamiento.

Por ejemplo, si queremos crear muchas cuentas bancarias distintas, podemos definir una clase `CuentaBancaria` y luego generar objetos a partir de ella.

📘 Ejemplo:

```
class CuentaBancaria {
  constructor(nombreTitular, apellidosTitular, saldo) {
    this.nombreTitular = nombreTitular;
    this.apellidosTitular = apellidosTitular;
    this.saldo = saldo;
  }

  mostrarDatos() {
     return `Titular: ${this.nombreTitular} ${this.apellidosTitular} - Saldo: $
{this.saldo} €`;
```

```
  }
}
```

➡ **Explicación:**

- `class CuentaBancaria` define la **clase**.
- `constructor(...)` es una **función especial** que se ejecuta automáticamente al crear un nuevo objeto de esta clase.
- `this` se refiere al objeto que se está creando.
- `mostrarDatos()` es un **método** (una función dentro del objeto).

🔹 Crear objetos a partir de la clase

Una vez definida la clase, podemos crear diferentes cuentas:

```
const cuenta1 = new CuentaBancaria("Luis", "Pérez", 300);
const cuenta2 = new CuentaBancaria("María", "Gómez", 1000);

console.log(cuenta1.mostrarDatos());
console.log(cuenta2.mostrarDatos());
```

📤 **Salida:**

```
Titular: Luis Pérez - Saldo: 300 €
Titular: María Gómez - Saldo: 1000 €
```

Cada cuenta es un **objeto independiente** con sus propios datos, pero todos comparten la misma estructura y comportamiento definidos por la clase.

🔹 Ventajas de usar clases

- Permiten **crear múltiples objetos fácilmente**.
- Favorecen la **organización** y la **reutilización del código**.
- Hacen que el programa sea más **escalable** (más fácil de ampliar o modificar).
- Siguen el enfoque de la **Programación Orientada a Objetos (POO)**.

🧠 Resumen final

Concepto	Qué es	Ejemplo
Objeto	Agrupación de datos y comportamientos.	`{ nombre: "Ana", saldo: 500 }`
Clase	Plantilla para crear objetos.	`class CuentaBancaria { ... }`
Instancia	Objeto creado a partir de una clase.	`new CuentaBancaria("Ana", "García", 500)`
