5.6 Clase en vivo - Profesores

Status	Done

Unidad 5: Funciones Constructoras y Almacenamiento

Objetivos de clase

- Repaso de Objetos y clases.
- Revisión microdesafío.
- JSON y Storage
- Stringify y parse

Repaso Unidad 5

Duración: 5 minuros

Consigna: Repasar lo visto en la Unidad 5

Ejemplo en vivo: ¿Qué es un objeto?

Duración: 5 minutos

Consigna: Se sugiere compartir pantalla y explicar lo que es un objeto con el siguiente ejemplo:

Un objeto es una colección de datos relacionados como una entidad. Se componen de un *listado* de pares clave-valor, es decir, contienen *propiedades* y *valores* agrupados.

```
const persona1 = { nombre: "Homero",

edad: 39,
```

```
calle: "Av. Siempreviva 742"}

persona1["nombre"] = "Marge"

persona1.edad = 36
```

Actividad: Creación y Manipulación de Objetos

Duración: 15 minutos

Instrucciones: Mostrar cómo se crea un objeto y cómo se pueden añadir o modificar propiedades utilizando tanto la notación de punto como de corchetes.

Solución esperada:

```
let persona = { nombre: "Homero", edad: 39, direccion: "Av.
Siempreviva 742" };
persona.edad = 40;
persona.hobby = "bolos";
```

Ejemplo en vivo: Función constructora

Duración: 10 minutos

Consigna: Se sugiere compartir pantalla y explicar lo que es el uso de this, new y clases con el siguiente ejemplo:

Uso de this y new

La palabra clave this (“este”) refiere al elemento actual en el que se está escribiendo el código. Cuando se emplea un función constructora para crear un objeto

(con la palabra clave new), this está enlazado al nuevo objeto instanciado.

This es muy útil para asegurar que se emplean las propiedades del objeto actual.

```
function Persona(*literal*) {

*this*.nombre = *literal*.nombre;

*this*.edad   = *literal*.edad;

*this*.calle  = *literal*.calle;

}

const persona1 = **new** Persona({ nombre: "Homero", edad: 39,

calle: "Av.Siempreviva 742" });
```

Clases

Las clases de JavaScript, introducidas en ES6, proveen una sintaxis mucho más clara y simple para crear objetos personalizados.

Son una equivalencia al empleo de función constructora y permite definir distintos tipos de métodos.

```
class Persona{

constructor(*nombre*, *edad*, *calle*) {

*this*.nombre = *nombre*;

*this*.edad   = *edad*;

*this*.calle  = *calle*;

}
```

```
}

const persona1 = **new** Persona("Homero", 39,

"Av. Siempreviva 742");
```

Actividad: Uso del Constructor y `this`

Duración: 15 minutos

Instrucciones: Explicar el concepto de función constructora y cómo `this` se utiliza para referenciar al objeto que se está creando.

Solución esperada:

```
function Auto(marca, modelo, año) {
  this.marca = marca;
  this.modelo = modelo;
  this.año = año;
}
let miAuto = new Auto("Toyota", "Corolla", 2021);
```

Actividad: Clases y Métodos

Duración: 15 minutos

Instrucciones: Demostrar la declaración de clases en ES6 y cómo agregar métodos a la clase.

Solución esperada:

```
class CuentaBancaria {
  constructor(titular, saldo) {
    this.titular = titular;
```

```
    this.saldo = saldo;
  }
  depositar(cantidad) {
    this.saldo += cantidad;
  }
}
let cuenta = new CuentaBancaria("Homero", 100);
cuenta.depositar(50);
```

Ejemplo en vivo: Storage

Duración: 15 minutos

Consigna: Se sugiere presentar un ejemplo práctico en vivo sobre localStorage y sessionStorage. Con el siguiente ejemplo:

LocalStorage: SetItem

Los datos almacenados en localStorage (variable global preexistente) se almacenan en el navegador de forma indefinida (o hasta que se borren los datos de navegación del browser):

La información persiste reinicios de navegador y hasta del sistema operativo.

```
// Método ->  localStorage.setItem(clave, valor)

// clave = nombre para identificar el elemento

// valor = valor/contenido del elemento

localStorage.setItem('bienvenida', '¡Hola Coder!');

localStorage.setItem('esValido', true);

localStorage.setItem('unNumero', 20);
```

- La información almacenada en el Storage se guarda en la forma de clave-valor.

LocalStorage: GetItem

Podemos acceder a la información almacenada en localStorage utilizando getItem. Las claves y valores de Storage se guardan en formato de cadena de caracteres (DOMString).

```
let mensaje = localStorage.getItem('bienvenida');

let bandera = localStorage.getItem('esValido');

let numero  = localStorage.getItem('unNumero');

console.log(mensaje); // '¡Hola Coder!'

console.log(bandera); // 'true'

console.log(numero);  // '20'
```

SessionStorage: SetItem

La información almacenada en sessionStorage (variable global preexistente) se almacena en el navegador hasta que el usuario cierra la ventana.

Solo existe dentro de la pestaña actual del navegador. Otra pestaña con la misma página tendrá otro sessionStorage distinto, pero se comparte entre iframes en la pestaña (asumiendo que tengan el mismo origen).

```
// Método ->  sessionStorage.setItem(clave, valor)

// clave = nombre del elemento

// valor = Contenido del elemento
```

```
sessionStorage.setItem('seleccionados', [1,2,3]);

sessionStorage.setItem('esValido', false);

sessionStorage.setItem('email', 'info@email.com');
```

SessionStorage: getItem

Podemos acceder a la información almacenada en sessionStorage utilizando getItem. Las claves y valores de Storage se guardan siempre en formato de cadena de caracteres.

```
let lista   =  sessionStorage.getItem('seleccionados').split(",'

let bandera = (sessionStorage.getItem('esValido') == 'true');

let email   =  sessionStorage.getItem('email');

console.log(typeof lista);   //object ["1","2","3"];

console.log(typeof bandera); //boolean;

console.log(typeof email);   //string;
```

Recorrer el storage

El bucle a emplear es for con el método key:

```
//Ciclo para recorrer las claves almacenadas en el objeto localS

for (let i = 0; i < localStorage.length; i++) {

let clave = localStorage.key(i);
```

```
console.log("Clave: " + clave);

console.log("Valor: " + localStorage.getItem(clave));

}
```

Eliminar datos del Storage

Podemos eliminar la información almacenada en sessionStorage o localStorage usando el método removeItem o clear:

```
localStorage.setItem('bienvenida', '¡Hola Code!');

sessionStorage.setItem('esValido', true);

localStorage.removeItem('bienvenida');

sessionStorage.removeItem('esValido');

localStorage.clear()    //elimina toda la información

sessionStorage.clear() //elimina toda la información
```

Ejemplo en vivo

Duración: 10 minutos

Consigna: Imaginemos una función que se ejecuta al momento en el cual un usuario ingresa a una página. La misma debe saludar al usuario por su nombre, o solicitarle el nombre la primera vez que ingresa y conservar el mismo para saludarlo en sus futuras visitas.

Veamos cómo almacenar y recuperar esta información en LocalStorage, utilizando sus diferentes métodos.

Paso a Paso:

1. **Crear la Función de Saludo:**

```
javascript
Copiar código
function saludarUsuario() {
    // Verifica si ya existe un nombre guardado en LocalSt
orage
    const nombreGuardado = localStorage.getItem('nombreUsu
ario');

    if (nombreGuardado) {
        // Si hay un nombre guardado, se saluda al usuario
        alert(`¡Hola de nuevo, ${nombreGuardado}!`);
    } else {
        // Si no hay un nombre guardado, se solicita el no
mbre al usuario
        const nombre = prompt('Bienvenido, ¿cuál es tu nom
bre?');

        // Guarda el nombre en LocalStorage para futuras v
isitas
        if (nombre) {
            localStorage.setItem('nombreUsuario', nombre);
            alert(`¡Hola, ${nombre}! Tu nombre ha sido gua
rdado.`);
        } else {
            alert('No se ingresó ningún nombre.');
        }
    }
}
```

2. **Invocar la Función al Cargar la Página:**

```
javascript
Copiar código
document.addEventListener('DOMContentLoaded', saludarUsuar
io);
```

Ejemplo en vivo: JSON - Conversión de objetos

Duración: 5 minutos

Consigna: Se sugiere compartir pantalla y explicar el uso de conversiones de/hacia JSON utilizando los método **stringify** y **parse,** con el siguiente ejemplo:

Stringify

Con JSON.stringify podemos transformar un objeto JavaScript a un string en formato JSON.

```
const producto1 = { id: 2, producto: "Arroz" };

const enJSON    = JSON.stringify(producto1);

console.log(enJSON); // {"id":2,"producto":"Arroz"}

console.log(typeof producto1); // object

console.log(typeof enJSON);    // string

localStorage.setItem("producto1", enJSON);

// Se guarda {"id":2,"producto":"Arroz"}
```

Parse

Con JSON.parse podemos transformar string en formato JSON a objeto JavaScript.

```
const enJSON    = '{"id":2,"producto":"Arroz"}';

const producto1 = JSON.parse(enJSON);

console.log(typeof enJSON);     // string

console.log(typeof producto1);  // object

console.log(producto1.producto); // Arroz

const producto2 = JSON.parse(localStorage.getItem("producto1"))

console.log(producto2.id);  // 2
```

Ejemplo en vivo: Almacenar y obtener array de objetos

Duración: 5 minutos

Consigna: Se sugiere compartir pantalla y explicar almacenar y obtener array de objetos, con el siguiente ejemplo:

Ejemplo aplicado: Almacenar array de objetos

```
const productos = [{ id: 1,  producto: "Arroz", precio: 125 },

{  id: 2,  producto: "Fideo", precio: 70 },

{  id: 3,  producto: "Pan"  , precio: 50},

{  id: 4,  producto: "Flan" , precio: 100}];
```

```
const guardarLocal = (*clave*, *valor*) => { localStorage.setIt

//Almacenar producto por producto

for (const producto of productos) {

guardarLocal(producto.id, JSON.stringify(producto));

}

// o almacenar array completo

guardarLocal("listaProductos", JSON.stringify(productos));
```

Ejemplo aplicado: Obtener array almacenado

```
class Producto {

constructor(*obj*) {

*this*.nombre  = *obj*.producto.toUpperCase();

*this*.precio  = parseFloat(*obj*.precio);

}

sumaIva() {

*this*.precio = *this*.precio * 1.21;

}

}
```

```
const almacenados = JSON.parse(localStorage.getItem("listaProduc

const productos = [];

for (const objeto of almacenados) //Iteramos con for...of para t

productos.push(**new** Producto(objeto));

for (const producto of productos) //Ahora tenemos objetos produc

producto.sumaIva();
```

Ejemplo en vivo: Recuperar estados previos

Duración:5 minutos

Consigna: Se sugiere compartir pantalla y explicar recuperar estados previos, con el siguiente ejemplo:

Ejemplo aplicado: Recuperar estados previos

```
let usuario;

let usuarioEnLS = JSON.stringify(localStorage.getItem(*‘usuario

// Si había algo almacenado, lo recupero. Si no le pido un ingre

if (usuarioEnLS) {

usuario = usuarioEnLS

} else {

usuario = prompt(*‘Ingrese su nombre de usuario’*)

}
```

```
**Ejemplo aplicado: Recuperar estados previos**

let carrito = []

let carritoEnLS = JSON.stringify(localStorage.getItem(*'carrito

// Inicializo mi app con *carrito* como array vacío, o con lo qu

if (carritoEnLS) {

carrito = carritoEnLS

}

*renderCarrito(*carrito*)* // Función que renderizaría el carri
```

Actividad: Completa el código

Duración: 15 minutos

Resolución:

```
// Completa el código de la clase 'Persona'
class Persona {
// Completa el constructor con los parámetros 'nombre' y 'edad'
constructor( nombre, edad) {
this.nombre = nombre;
this.edad = edad;
}
// Completa el método 'saludar' para que retorne un mensaje de s
saludar() {
return Hola, mi nombre es ${this.nombre} y tengo ${this.edad} añ
}
}
// Crea una instancia de la clase 'Persona' con el nombre 'Juan
```

```
let persona1 = new Persona('Pedro', 40);
// Llama al método 'saludar' en la instancia y muestra el mensaj
console.log(persona1.saludar()); // Debería mostrar "Hola, mi no
tengo 40 años."
```

Preclase - 30 minutos previos

Repaso de Conceptos Clave (10 minutos)

- **Objetos en JavaScript:** Revisión rápida de la creación de objetos, propiedades y métodos, y la sintaxis de acceso a los datos.
- **Funciones Constructoras y `new`:** Explicación breve sobre cómo crear objetos usando funciones constructoras y la instrucción `new`.
- **Clases en ES6:** Introducción a la sintaxis de clases, métodos y el uso de `constructor` para inicializar objetos.
- **Uso de `this`:** Repaso del contexto de `this` dentro de los métodos y funciones constructoras.

Espacio para Preguntas y Respuestas (15 minutos)

- Abrir el espacio para que los estudiantes hagan preguntas sobre objetos, clases, y uso de `this`.
- Resolver dudas con ejemplos en vivo si es necesario, como la diferencia práctica entre una función constructora y una clase ES6.
- Si no hay preguntas, sugerir temas comunes como la diferencia entre acceso a propiedades con punto y con corchetes, o el uso de `this` dentro de métodos.

Demostraciones Prácticas (Opcional, dependiendo de las preguntas)

- Realizar demostraciones en vivo si surgen preguntas que lo requieran, por ejemplo, crear un objeto utilizando una clase y una función constructora.
- Ejemplos posibles: Creación de un objeto persona con propiedades y métodos, o mostrar cómo se accede y modifica propiedades.