**📑 Apuntes de contenidos por video**

**1. Intro. cómo ejecutar código**

Aprendemos cómo ejecutar código JavaScript en distintos entornos: desde la consola del navegador hasta el uso de editores como VSCode con Live Server.

**2. Variables JavaScript**

Exploramos cómo declarar variables con `var`, `let` y `const`, sus diferencias y en qué casos es recomendable usar cada una.

**3. Intro HTML y CSS**

Repaso básico de HTML para estructurar una página web, y CSS para darle estilo visual, colores, fuentes y organización.

**4. Funciones**

Se enseña a crear funciones para encapsular bloques de código reutilizables, incluyendo funciones tradicionales y funciones flecha.

**5. Unir HTML y JavaScript**

Vinculamos nuestros archivos HTML con JavaScript para permitir la interacción dinámica entre el usuario y la interfaz web.

**6. Botones**

Manejamos la interacción con botones: cómo detectar clics y asociarles funciones que se ejecuten al activarlos.

**7. If, objetos y fechas**

Utilizamos condicionales (`if/else`) para tomar decisiones, aprendemos a crear objetos con propiedades, y usamos el objeto `Date`.

**8. addEventListener vs onclick**

Comparación entre dos formas de manejar eventos. `addEventListener` permite una mejor separación del código y múltiples escuchas por evento.

**9. Inicio del proyecto de gestión económica**

Comenzamos a crear una aplicación para gestionar ingresos y egresos, aplicando lo aprendido hasta ahora en un proyecto real.

**10. Creando tabla en HTML**

Estructuramos una tabla en HTML para visualizar los datos registrados de manera ordenada y clara en filas y columnas.

**11. Name vs ID**

Diferencias clave: `id` sirve para identificar elementos de forma única, y `name` se utiliza especialmente en formularios para agrupar y enviar datos.

**12. preventDefault**

Método que se usa para evitar comportamientos predeterminados de elementos HTML, como recargar la página al enviar un formulario.

**13. FormData**

Utilizamos `FormData` para capturar y manipular fácilmente los datos enviados por un formulario en el navegador.

**14. Agregar datos a tabla desde un formulario**

Conectamos el formulario a la tabla: capturamos datos ingresados por el usuario y los mostramos dinámicamente como nuevas filas.

**15. Local Storage (base de datos del navegador)**

Introducción al `localStorage`, una forma de guardar datos en el navegador de manera persistente, incluso si el usuario cierra la página.

**16. Local Storage: almacenar un objeto en JSON**

Aprendemos a guardar objetos en `localStorage` convirtiéndolos a formato JSON con `JSON.stringify()`, y a recuperarlos como objetos válidos con `JSON.parse()`.

**17. Guardando datos de nuestra app en Local Storage**

Integramos el almacenamiento local con nuestra app para guardar ingresos y egresos. Así, los datos persisten aunque se cierre o recargue la página.

**18. Guardando el historial de transacciones en Local Storage (array de objetos)**

Guardamos un array de objetos representando cada transacción. Esto permite mantener un historial estructurado y fácilmente accesible desde el `localStorage`.

**19. Renderizar nuestra tabla desde el Local Storage al iniciar la app**

Automatizamos la carga de datos al iniciar la aplicación: la tabla se rellena dinámicamente con las transacciones guardadas previamente en `localStorage`.

**20. Limpiar un formulario con JavaScript - cómo resetear formulario**

Usamos el método `.reset()` para limpiar todos los campos de un formulario después de enviarlo, mejorando la experiencia del usuario y evitando duplicados.

**21. Los data atrributes en HTML y JS - guardar info en HTML**

Aprendimos a usar los data-\* attributes para almacenar información personalizada en los elementos HTML. Esto permite vincular datos específicos directamente en la interfaz, accesibles fácilmente desde JavaScript mediante element.dataset.

**22. querySelector y los data attributes, combinación implacable**

Combinamos querySelector con los data-attributes para seleccionar elementos de forma precisa y dinámica. Esta técnica facilita acceder a información embebida en HTML y manipular el DOM de manera eficiente e intuitiva.

**23. Eliminar fila de una tabla mediante un botón**

Aprendimos a eliminar dinámicamente una fila de una tabla HTML al hacer clic en un botón. Se usa addEventListener para detectar el clic y luego se accede al nodo padre (parentElement) para eliminar la fila correspondiente con remove().

**24. Método útil: manipular elementos del localStorage mediante un identificador**

Vimos cómo acceder, modificar o eliminar un elemento específico del localStorage utilizando un identificador único (como un id). Esto nos permite mantener una estructura organizada y realizar operaciones puntuales sin afectar el resto de los datos almacenados.

**25. Eliminar un elemento de un array en el localStorage**

Aprendimos a eliminar un elemento específico de un array almacenado en localStorage. Se obtiene el array con JSON.parse, se filtra el elemento por su id u otra propiedad, y se vuelve a guardar con JSON.stringify.

**26. Validar desde HTML e introducción a la validación desde JavaScript**

Exploramos las validaciones básicas integradas en HTML (required, type, pattern) y cómo mejorar la experiencia del usuario complementando con validaciones personalizadas en JavaScript usando condicionales y mensajes de error dinámicos.

**27. Agregando selectores a nuestra aplicación de gastos**

Incorporamos select y option a nuestro formulario para permitir que el usuario elija categorías u otras opciones. Se manipuló el valor seleccionado desde JavaScript con .value y se utilizó para categorizar o filtrar los datos.

**28. Maquetando nuestra aplicación sin usar CSS: aprendiendo la librería MaterializeCSS**

Introducción a la librería MaterializeCSS para estructurar visualmente nuestra aplicación sin escribir CSS desde cero. Usamos clases predefinidas para botones, formularios y layouts, facilitando una interfaz limpia y responsiva.

**29. Cómo hacer nuestro diseño responsivo sin CSS gracias a los md, sm, lg, etc.**

Aprendimos a utilizar el sistema de grid responsivo de MaterializeCSS (y frameworks similares) mediante clases como col s12 m6 l4, que permiten adaptar el diseño automáticamente según el tamaño del dispositivo, sin necesidad de escribir CSS personalizado.

**30. Agregando opciones a los selects desde JS - interpolación de strings**

Vimos cómo crear dinámicamente opciones (<option>) dentro de un select usando JavaScript. Utilizamos interpolación de strings (template literals con backticks) para insertar valores directamente en el HTML desde arrays u objetos.

**31. Debuggear en la consola de Chrome - resolver errores de nuestro código**

Exploramos herramientas del navegador (F12 → Consola) para identificar errores, usar console.log estratégicamente, y hacer pausas con debugger para seguir el flujo del código paso a paso. Esto mejora la capacidad para detectar y corregir bugs.

**32. Git / GitHub / GitHub Pages**

Introducción al control de versiones con Git y al uso de plataformas como GitHub para almacenar código en la nube, colaborar y hacer seguimiento de cambios. Aprendimos a usar GitHub Pages para publicar sitios estáticos desde un repositorio.

**33. ¿Cuándo usar var, let y const? Diferencias fundamentales**

Aprendimos las diferencias clave entre var, let y const: var tiene alcance de función y se hoistea, mientras que let y const tienen alcance de bloque. const no permite reasignación. Se recomienda evitar var en código moderno.

**34. Programación Orientada a Objetos (POO)**

Introducción a los conceptos básicos de POO en JavaScript: clases, objetos, constructores, métodos y encapsulamiento. Entendimos cómo modelar entidades del mundo real usando estructuras reutilizables de código.

**35. Una API no es lo mismo que una API web - ¿Qué es HTTP? ¿Qué son GET, POST, PUT, DELETE? - Consola F12**

Aprendimos qué es una API (interfaz de programación de aplicaciones) y su diferencia con una API web. Exploramos el protocolo HTTP y los métodos comunes (GET, POST, PUT, DELETE). Usamos la pestaña “Red” en la consola para observar las peticiones.

**36. ¿Qué es una API REST y cómo funciona? Las bases para crear una API RESTful**

Descubrimos el concepto de REST (Transferencia Representacional de Estado) como estilo arquitectónico. Aprendimos cómo funcionan las rutas, recursos y métodos HTTP en una API RESTful para acceder y manipular datos de manera organizada.

**37. Instalación de Node.js para crear APIs en JavaScript y uso de NPM**

Instalamos Node.js en nuestro entorno local para ejecutar JavaScript del lado del servidor. Aprendimos a usar npm (Node Package Manager) para gestionar paquetes y dependencias necesarios para crear nuestras APIs.

**38. ¿Qué es Node.js y cómo funciona? ¿Para qué sirve?**

Conocimos Node.js como entorno de ejecución para JavaScript fuera del navegador. Entendimos su funcionamiento basado en eventos y su utilidad para construir servidores, herramientas de línea de comandos y aplicaciones web en tiempo real.