Ejercicios Parte 3: Modernizando el Stack

Guía Extendida y Detallada

1º DAM - Desarrollo Web en Entorno Cliente

Comenzar →

Mapa del Recorrido

Esta guía está expandida para asegurar que comprendes cada herramienta nueva antes de usarla.

Bloque 1: JavaScript

- DOM: Manipulación HTML
- Eventos: Interacción usuario
- ES6: JavaScript Moderno

Bloque 2: SASS

- Variables y Nesting
- Mixins (Funciones CSS)
- Arquitectura Modular

Bloque 3: Automatización (Gulp)

Aquí aprendemos a crear un entorno de trabajo profesional.

- Node.js y NPM
- Tasks automatizados
- Compilación en tiempo real

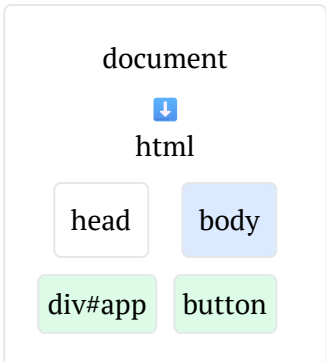
Bloque 1: JavaScript & DOM

Entendiendo la interactividad

21. El DOM: Teoría

Document Object Model

El HTML es texto plano. El navegador lo lee y lo convierte en una estructura de árbol en memoria: **El DOM**.



JavaScript vive "fuera" de este árbol, pero tiene herramientas para **seleccionarlos** y **modificarlos**.

21. Herramientas de JS

Antes de empezar, necesitas conocer tus herramientas:

1. Selección (`getElementById`)

Es como la "mano" que agarra el elemento.

```
const miCaja = document.getElementById('caja');
```

2. Escucha (`addEventListener`)

Es el "oído" que espera a que pase algo.

```
boton.addEventListener('click', () => {
   // Código que se ejecuta al clickar
});
```

3. Modificación (`.style` / `.innerText`)

Es la acción que realizas.

```
// Cambiar CSS directo
elemento.style.backgroundColor = 'red';

// Cambiar Texto
elemento.innerText = 'Nuevo Texto';
```

Nota importante: Cuando usas `.style` en JS, estás aplicando estilos "en línea" (inline styles), que tienen mucha prioridad CSS.

Ejercicio 21: Instrucciones Detalladas

Tienes los archivos `index.html` (estructura) y `script.js` (vacío).

Pasos a seguir:

1. Declara constantes para el `div` (la caja) y los `button` (botones). Usa `document.getElementById('ID_DEL_HTML')`.
2. Al botón `btnColor`, añádele un evento `click`.
   - Dentro de la función, haz que `caja.style.backgroundColor` sea igual a un color (ej. `'purple'`).
3. Al botón `btnTexto`, añádele un evento `click`.
   - Dentro, cambia `caja.innerText` a `'Hola JS'`.

Bonus: Usa `document.createElement('li')` y `lista.appendChild()` para el botón de "Agregar".

22. Clases Estáticas vs Dinámicas

En el ejercicio anterior cambiamos el estilo "a la fuerza" (`.style`). Pero, ¿y si queremos cambiar 20 propiedades a la vez?

Mala práctica:

```
div.style.backgroundColor = 'red';
div.style.color = 'white';
div.style.fontSize = '20px';
div.style.transform = 'scale(1.1)';
// ... interminable
```

Buena práctica: Definir una **clase CSS** y activarla con JS.

```
div.classList.add('activo');
```

22. Entendiendo `classList`

`classList` es una lista de las clases que tiene un elemento HTML.

Métodos Principales

Método	Descripción	Ejemplo
`.add('clase')`	Añade la clase (si no existe).	Activar un modal.
`.remove('clase')`	La elimina por completo.	Cerrar un modal.

Ejemplo Visual

Estado 1 (Sin clase): `<div class="menu">` *Estilo:* `left: -100%` (Oculto)

⬇️ `classList.add('open')`

Estado 2 (Con clase): `<div class="menu open">` *Estilo:* `.menu.open { left: 0; }` (Visible)

> **Ventaja:** Mantienes la lógica visual en CSS y la lógica de estado en JS.

Ejercicio 22: Instrucciones

Objetivo: Crear un panel lateral deslizable.

1. **Analiza el CSS**: Verás una clase `.menu` que está fuera de pantalla, y una `.menu.hidden` (o similar) que controla su posición. *Nota: En el ejercicio, he puesto la lógica para que `.hidden` lo oculte, así que empieza visible o viceversa. Revisa el CSS.*
2. **JS**: Selecciona el botón y el menú.
3. **Evento**: Al hacer click en el botón, ejecuta `menu.classList.toggle('hidden')`.

Prueba: Click -> Aparece. Click -> Desaparece.

Bloque 2: SASS

Syntactically Awesome Style Sheets

23. Por qué usar Pre-procesadores

CSS puro es genial, pero se vuelve difícil de mantener en proyectos grandes.

Problemas de CSS puro:

1. **Repetición**: Escribes el mismo color hexadecimal `#3498db` 50 veces. Si quieres cambiarlo, tienes que buscar y reemplazar.
2. **Selectores Largos**: `.nav .menu li a span { ... }`.
3. **Sin Lógica**: No puedes hacer operaciones matemáticas sencillas ni funciones.

Solución SASS:

Escribes en un lenguaje enriquecido (SCSS) y un programa lo "traduce" (compila) a CSS normal para el navegador.

23. Variables y Nesting

Variables ($)

Almacenan valores reutilizables.

```scss
$brand-color: #ff5722;
$spacing: 16px;

button {
    background: $brand-color;
    margin: $spacing;
}
```

Nesting (Anidamiento)

Escribe CSS dentro de CSS.

```scss
.card {
    padding: 20px;

    h2 { // Esto equivale a .card h2
        font-size: 20px;
```

Ejercicio 23: Instrucciones

Requisito previo: Instala la extensión "**Live Sass Compiler**" en VSCode. Si no, no funcionará.

1. En la barra inferior de VSCode, pulsa "**Watch Sass**".
2. Abre `styles.scss`.
3. Crea variables para colores y fuentes arriba del todo.
4. Reescribe el CSS usando **nesting**.
   - Mete los estilos del `h1` dentro del `.container`.
   - Mete el estilo `:hover` del botón dentro de `.btn` usando `&:hover`.
5. Guarda y verifica que se ha creado un `styles.css`.

24. Mixins: Funciones de Estilo

A veces tienes trozos de código CSS que se repiten con pequeños cambios.

Ejemplo: Tres cajas de alerta (Info, Éxito, Error). Todas tienen el mismo borde, padding y fuente, pero distinto color de fondo.

Definición (`@mixin`)

```
@mixin alerta($color) {
    border: 1px solid $color;
    background-color: lighten($color, 40%); // Función de color nativa de SASS
    color: $color;
    padding: 10px;
}
```

Uso (`@include`)

```
.alert-error {
    @include alerta(red);
}
```

Ejercicio 24: Instrucciones

Vamos a crear un sistema de botones.

1. Abre `styles.scss`.
2. Crea un mixin llamado `btn-structure`. Debe tener `padding`, `border-radius` y `font-weight`.
3. Crea otro mixin llamado `btn-theme($color)` que reciba un color.
   - Debe poner el `background-color`.
   - Debe poner el color de texto (blanco o negro).
   - **Truco**: Usa `darken($color, 10%)` en el `&:hover`.
4. Usa estos mixins para crear las clases `.btn-primary` y `.btn-secondary`.

```
// Ejemplo de uso final

.btn-primary { @include btn-structure; @include btn-theme($blue); }

.btn-danger { @include btn-structure; @include btn-theme($red); }
```

25. Organización y Partials

Hasta ahora escribíamos todo en un solo archivo. En proyectos reales, el CSS puede tener miles de líneas.

La regla del guion bajo (`_`)

Si nombras un archivo `_header.scss`, SASS sabe que **NO** debe compilarlo a un CSS independiente. Sabe que es una pieza de un puzle.

Estructura Típica

- `_variables.scss` : Configuración global.
- `_reset.scss` : Normalización.
- `_layout.scss` : Estructura principal.
- `styles.scss` : El archivo maestro que importa todo.

Ejercicio 25: Instrucciones

Objetivo: Modularizar tu CSS.

1. Crea/Edita los archivos parciales:
   - `_variables.scss`: Define `$primary-color`.
   - `_base.scss`: Define estilos para `body` (fuente, margen 0).
2. Abre `main.scss`. ¡Este es el único que no lleva guion bajo!
3. Usa `@import 'variables';` y `@import 'base';`.
   - Nota: No hace falta escribir el guion bajo ni la extensión al importar.
4. Compila solo `main.scss`.
5. Comprueba que `main.css` tiene TODO el código junto.

Bloque 3: Automatización

Gulp, NPM y el Build Process

26. El Ecosistema Node.js

Para usar herramientas modernas profesionalmente, necesitamos salir del navegador y usar la terminal.

NPM (Node Package Manager)

Es la "tienda de apps" de los desarrolladores.

- `npm init`: Crea un `package.json` (la lista de la compra de tu proyecto).
- `npm install nombre-paquete`: Descarga una herramienta.

package.json

Es vital. Dice qué librerías necesita tu proyecto para funcionar. Si le pasas tu código a otro, solo con este archivo puede reinstalar todo.

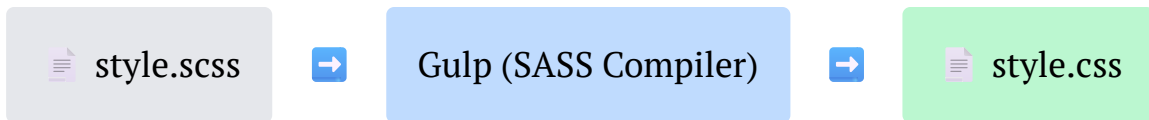
Ejercicio 26: Instrucciones

Objetivo: Instalar Gulp.

1. Abre una terminal en la carpeta `Ejercicio_26`.
2. Ejecuta `npm install`.
   - Esto leerá el `package.json` que he preparado y bajará `gulp` y `sass`.
   - Verás aparecer una carpeta `node_modules` (nunca se toca, es gigante).
3. Crea un archivo llamado `gulpfile.js`.
4. Escribe una "Tarea Hola Mundo" (ver código en diapositiva anterior o README).
5. Ejecuta `npx gulp` en la terminal para ver el saludo.

27. Entendiendo Gulp Pipes

Gulp funciona como una tubería de agua. Los archivos entran por un lado (`src`), pasan por filtros/transformaciones, y salen por el otro (`dest`).



Ejercicio 27: Instrucciones

Configuraremos la tarea más común del frontend.

1. En `gulpfile.js`, importa las funciones:

```
const { src, dest, watch } = require('gulp');
const sass = require('gulp-sass')(require('sass'));
```

2. Crea la función `compilarCSS()`:
   - `src`: 'src/scss/style.scss'
   - `pipe`: sass()
   - `dest`: 'dist/css'
3. Exporta la tarea.
4. **PLUS**: Añade `watch`.
   - `watch('src/**/*.scss', compilarCSS)`
   - Esto hace que Gulp se quede "escuchando" cambios y compile solo.

¿Por qué 'dist'?
Es convención llamar 'src' (source/fuente) a lo que tú escribes y 'dist' (distribution/público) a lo que se sube al servidor (código optimizado y compilado).

28. JavaScript Moderno (ES6)

Desde 2015, JS ha mejorado mucho. Debes conocer la sintaxis actual.

Arrow Functions

```
// Función Flecha
const sumar = (a, b) => a + b;
```

Más concisa, especialmente para callbacks.

Métodos de Array

Olvídate de los bucles `for` para transformar datos.

- `.map()` : Transforma cada elemento.
- `.filter()` : Selecciona elementos.

Ejercicio 28: Instrucciones

Tienes un array de usuarios en `script.js`.

1. Usa `.filter()` para crear una nueva lista llamada `admins` que solo contenga los usuarios con `rol: 'admin'`.
2. Usa `.map()` para generar un array de strings HTML:

   ```
   const listaHTML = admins.map(u => `<li>${u.nombre}</li>`
   ```

   (Nota el uso de comillas invertidas `` ` `` para template strings).
3. Inyecta ese array en el HTML usando `.join('')` y `innerHTML`.

29. Componente Completo: Tabs

Este ejercicio une **todo** lo aprendido.

La Estructura

```
<div class="tabs">
  <button class="tab-btn active" data-target="uno">Uno</button>
  <button class="tab-btn" data-target="dos">Dos</button>
</div>
<div class="content active" id="uno">...</div>
<div class="content" id="dos">...</div>
```

Concepto `data-attributes` : `data-target="uno"` es una forma de guardar información extra en el HTML para que JS la lea (`dos.dataset.target`).

Ejercicio 29: El Paso a Paso

Fase 1: SASS

1. Estila `.tab-btn`. Usa `&:hover` y `&.active` (para cuando esté pulsado).
2. Estila `.content`. Por defecto `display: none`.
3. Estila `.content.active` con `display: block`.

Fase 2: JS

1. Selecciona todos los botones (`querySelectorAll`).
2. Añade click event a cada uno (usando `forEach`).
3. **Lógica del Click**:
   - Quitar `.active` de todos los botones y contenidos.
   - Poner `.active` al botón pulsado (`this` o `e.target`).
   - Leer el `dataset.target` del botón pulsado.
   - Buscar el contenido con ese ID y ponerle `.active`.

30. Proyecto Final Automatizado

Bienvenidos al flujo de trabajo real. Aquí no hay "instrucciones" paso a paso, sino un entorno que debes gestionar.

Estructura del Proyecto

- **src/**: Tu zona de trabajo. Nadie más ve esto.
- **dist/**: La web final. Se genera sola.
- **gulpfile.js**: Ya preconfigurado para compilar SASS, mover JS y HTML.

Tu Misión

1. Ejecuta `npm install` y luego `gulp`.
2. El terminal se quedará "esperando" (Watch Mode).
3. Edita `src/scss/main.scss`. Cambia colores, fuentes. ¡Mira cómo `dist` cambia!
4. Crea un menú responsive en JS dentro de `src/js`.
5. Comprueba que todo funciona en el navegador abriendo `dist/index.html`.

¡Felicidades! Has completado el curso de CSS Avanzado y Herramientas Modernas.