BOOTCAMP DESARROLLO WEB FULL STAC

TALENTOTECH FULLSTACK BÁSICO

LABORATORIO MODULO No. 2

ESTILOS CON CSS3

ACTIVIDAD GRUPO No. 6

POR

ANDRÉS LOTE

JORGE MORALES

STIVEN AGUIRRE

AMPARO FORTALECHÉ T.

MONICA P MENDEZ

DOCENTES:

JULIAN F. LATORRE

ROYMER ROMERO

BOGOTÁ

2024

**INTRODUCCIÓN**

Este laboratorio completo tiene como objetivo evaluar y reforzar los conocimientos adquiridos en el módulo "Estilos Con CSS3". Las actividades y preguntas están diseñadas para ayudarles a aplicar lo aprendido en su proyecto final del curso de desarrollo Fullstack

**OBJETIVOS DE LA ACTIVIDAD**

Este laboratorio nos reta a pensar críticamente y es un desafío como aprendices a aplicar nuestras habilidades. Aunque es un curso de nivel básico, este laboratorio nos desafiará a pensar críticamente y aplicar nuestras habilidades como seres capaces de crear proyectos y dar soluciones individuales y específicas dependiendo las necesidades de cada cliente. Es un comienzo de una carrera profesional como desarrolladores competentes y con seguridad ética.

OBJETIVOS GENERALES

* Implementar estilos CSS básicos y avanzados en un sitio web
* Comprender y aplicar el modelo de caja en el diseño web
* Crear layouts responsivos utilizando Flexbox y Grid
* Implementar diseños adaptables con Media Queries
* Añadir interactividad visual con transiciones y animaciones CSS
* Crear un mini-proyectos como: portafolio de servicios o una landing page, Feedback que nos permitan interactuar con otros compañeros o en foros y la retroalimentación para aprender y mejorar nuestros proyectos.
* El aprendizaje de herramientas como CodePen para experimentar con nuestros códigos y ver resultados en tiempo real.

OBJETIVOS ESPECÍFICOS

* Implementar estilos CSS básicos y avanzados.

Básicos: Aprender a dominar selectores, propiedades y valores. Practica estilos de texto, colores, fondos y márgenes.

Avanzados: Explora propiedades como box-shadow, border-radius, y opacity. Experimenta con gradientes y sombras para enriquecer tus diseños.

* Comprender y aplicar el modelo de caja: Cada elemento es un rectángulo que incluye márgenes, bordes, padding y el contenido, calcular el tamaño total de un elemento utilizando box-sizing: border box.
* Crear layouts responsivos utilizando Flexbox y Grid.
* Flexbox: Se utiliza para diseños unidimensionales por sus atributos: flex-direction, justify-content, y align-items.
* Grid: Se utiliza para diseños bidimensionales. Experimenta con grid-template-columns, grid-template-rows y grid-area.
* Implementar diseños adaptables con Media Queries.

Media Queries: Usamos @media para aplicar estilos específicos según el tamaño de pantalla por ejemplo en la creación de un diseño que se adapte a móviles, tabletas y desktops.

* Añadir interactividad visual con transiciones y animaciones CSS

Transiciones: Usa transition para suavizar cambios de propiedades como cambiar colores, tamaños y posiciones.

Animaciones: a través de: @keyframes para crear animaciones personalizadas y combinaciones como transform y opacity para efectos más dinámico.

**PARTE 1:**

**1. FUNDAMENTOS DE CSS**

1. Explique la diferencia entre CSS interno, en línea y externo. ¿Cuál es el más recomendado y por qué

El CSS Interno: se escribe dentro de la etiqueta <style> en la sección <head> del documento HTML. Se utiliza para aplicar estilos a un documento específico, sus ventajas son:

Facilita la gestión de estilos para una sola página.

Permite un control más preciso sobre la apariencia de esa página.

CSS En Línea: Se aplica directamente a los elementos HTML utilizando el atributo style. Se usa para dar estilos específicos a un solo elemento.

Ventajas: Cambios rápidos y específicos sin afectar otros elementos.

Desventajas: Dificulta la reutilización de estilos. Puede resultar en un código desordenado y menos legible.

CSS Externo: Se escribe en un archivo separado con extensión *.css,* que luego se vincula al documento HTML mediante la etiqueta <link> en la sección <head>.Permite aplicar estilos a múltiples páginas de un sitio web.

Ventajas: Mejora la organización del código. Facilita la reutilización de estilos en diferentes páginas. Reduce el tamaño del HTML, lo que puede mejorar el rendimiento.

**RECOMENDADO**

EL CSS EXTERNO ES EL MÁS RECOMENDADO, especialmente para proyectos más grandes o aquellos que requieren un mantenimiento a largo plazo. En resumen, para un desarrollo web efectivo y eficiente, el uso de CSS externo es la mejor práctica por las siguientes razones:

* Reutilización: Puedes aplicar el mismo archivo CSS a múltiples páginas, lo que ahorra tiempo y esfuerzo.
* Mantenibilidad: Cambios en el archivo CSS se reflejan en todas las páginas que lo usan, facilitando actualizaciones y modificaciones.
* Separación de Concerns: Mantiene el contenido (HTML) y la presentación (CSS) separados, lo que mejora la legibilidad y la estructura del proyecto.

2. Cree un archivo CSS externo y vincúlelo a su página HTML del proyecto. Muestre

el código HTML relevante.

**1.2 TIPOS DE SELECTORES CSS**

**SELECTOR DE TIPO**

Descripción: Selecciona todos los elementos de un tipo específico.

Ejemplo: p selecciona todos los elementos <p> (párrafos).

Selector de clase:

Descripción: Selecciona todos los elementos que tienen una clase específica.

Ejemplo: .mi-clase selecciona todos los elementos con class="mi-clase".

Selector de ID:

Descripción: Selecciona un elemento único que tiene un ID específico.

Ejemplo: #mi-id selecciona el elemento con id="mi-id".

Selector de atributo:

Descripción: Selecciona elementos en función de un atributo específico y su valor.

Ejemplo: input[type="text"] selecciona todos los elementos <input> que tienen el atributo type con el valor text.

Selector descendiente:

Descripción: Selecciona elementos que son descendientes de un elemento específico.

Ejemplo: div p selecciona todos los elementos <p> que están dentro de un <div>.

2. Ejemplos de Aplicación de Estilos

A continuación, se muestran ejemplos de cómo aplicar estilos utilizando los tipos de selectores mencionados en tu proyecto:

1.2 Selectores y Propiedades CSS

1. Enumere y explique brevemente 5 tipos diferentes de selectores CSS.

2. En su proyecto, aplique estilos utilizando:

• Un selector de clase

• Un selector de ID

• Un selector de atributo

• Un selector descendiente

• Una pseudoclase

### Ejemplos de Aplicación de Estilos

A continuación, se muestran ejemplos de cómo aplicar estilos utilizando los tipos de selectores mencionados en tu proyecto:

#### CSS

Copiar código

/\* Selector de clase \*/

.mi-clase {

color: blue;

font-size: 16px;

}

/\* Selector de ID \*/

#mi-id {

background-color: yellow;

padding: 10px;

}

/\* Selector de atributo \*/

input[type="text"] {

border: 1px solid #ccc;

padding: 5px;

}

/\* Selector descendiente \*/

div p {

font-style: italic;

margin-bottom: 10px;

}

/\* Pseudoclase \*/

a:hover {

color: red;

text-decoration: underline;

}

HTML

En este ejemplo, cada selector aplica diferentes estilos a los elementos correspondientes, ilustrando cómo se pueden utilizar en un proyecto real.

html

Copiar código

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="styles.css">

<title>Ejemplo de Selectores CSS</title>

</head>

<body>

<div>

<p class="mi-clase">Este es un párrafo con clase.</p>

<p>Este es otro párrafo.</p>

<a href="#">Enlace de ejemplo</a>

</div>

<div id="mi-id">

Este div tiene un ID único.

</div>

<input type="text" placeholder="Texto aquí">

</body>

</html>

PARTE 2

LAYOUT Y DISEÑO

2.1 Modelo de Caja (Box Model)

El modelo de caja (Box Model) de CSS es fundamental para comprender cómo se visualizan los elementos en una página web, comprender el modelo de caja es decisivo para controlar el espacio y la disposición de los elementos dentro de la misma página. Manipular adecuadamente el padding, el borde y el margen te permitirá crear diseños más efectivos y estéticamente agradables. Cada elemento HTML se representa como una caja rectangular que consta de varios componentes:

1. Contenido

Es la parte central de la caja, donde se muestra el texto, imágenes u otros elementos. El tamaño del contenido se define mediante las propiedades width y height.

2. Padding (Relleno)

El espacio entre el contenido y el borde de la caja. El padding crea un área de separación, empujando el contenido hacia adentro.

Propiedades: Se puede especificar para todos los lados (top, right, bottom, left) utilizando propiedades como padding, padding-top, padding-right, padding-bottom, y padding-left.

3. Borde (Border)

La línea que rodea el padding (si lo hay) y el contenido. El borde puede tener diferentes estilos, grosores y colores.

Propiedades: Se puede personalizar usando border, border-width, border-style, y border-color. También se pueden especificar bordes por lado.

4. Margin (Margen)

El espacio que rodea la caja, separándola de otros elementos en la página. A diferencia del padding, el margen crea espacio fuera de la caja.

Propiedades: Al igual que el padding, se puede especificar para cada lado usando margin, margin-top, margin-right, margin-bottom, y margin-left.

5. Modelo de Caja Alternativo (box-sizing)

El comportamiento de cómo se calcula el tamaño de la caja. Por defecto, el tamaño total de un elemento se calcula como:

Width/Height Total=Width/Height+Padding+Border+Margin

Sin embargo, al usar box-sizing: border-box, el tamaño total incluye el padding y el borde, lo que puede facilitar el diseño de layouts.

Visualización del Modelo de Caja

+-------------------------------+

| Margin |

| +---------------------------+ |

| | Border | |

| | +-----------------------+ | |

| | | Padding | | |

| | | +-------------------+ | | |

| | | | Contenido | | | |

| | | +-------------------+ | | |

| | +-----------------------+ | |

| +---------------------------+ |

+-------------------------------+

2. En su proyecto, ajuste el modelo de caja de un elemento para que tenga:

• Padding de 20px en todos los lados

• Un borde sólido de 2px

• Un margen de 10px arriba y abajo, 15px a los lados

Proporcione el código CSS utilizado.

2.2 Layout con Flexbox

1. Cree una barra de navegación horizontal utilizando Flexbox. Debe contener al

menos 4 elementos y estar centrada en la página.

2. Implemente un diseño de "tarjetas" para mostrar productos o servicios en su

proyecto utilizando Flexbox. Las tarjetas deben reorganizarse automáticamente

según el ancho de la pantalla.

2.3 Layout con Grid

1. Diseñe el layout principal de su página de inicio utilizando CSS Grid. Debe incluir

header, nav, main content, sidebar y footer.

2. Cree una galería de imágenes responsive utilizando Grid, que muestre 4 columnas

en pantallas grandes y 2 en móviles.

PARTE 3: ESTILIZACIÓN Y EFECTOS VISUALES

3.1 ESTILIZACIÓN DE TEXTOS Y FUENTES

1. Integre una fuente personalizada de Google Fonts en su proyecto. Aplíquela a los encabezados.

2. Estilice un párrafo de texto para mejorar su legibilidad, considerando:

• Tamaño de fuente

• Altura de línea

• Espaciado entre letras

• Alineación

1. Integrar una Fuente Personalizada de Google Fonts

Paso 1: Elegir la fuente

Ve a Google Fonts y elige una fuente que te guste. Por ejemplo, seleccionemos "Roboto".

**Paso 2: Obtener el enlace**

* Haz clic en "Select this style" y copia el enlace en la parte superior que se verá algo así:

html

Copiar código

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

**Paso 3: Integrar en tu proyecto**

* Añade el enlace en la sección <head> de tu archivo HTML.

html

Copiar código

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="styles.css">

<title>Estilización de Textos</title>

</head>

<body>

<h1>Encabezado Principal</h1>

<p>Este es un párrafo de ejemplo para estilizar.</p>

</body>

</html>

**Paso 4: Aplicar la fuente a los encabezados**

* En tu archivo CSS, aplica la fuente a los encabezados (h1, h2, etc.).

css

Copiar código

h1, h2, h3 {

font-family: 'Roboto', sans-serif;

}

### 2. Estilizar un Párrafo de Texto

Ahora, estiliza un párrafo para mejorar su legibilidad. Aquí tienes un ejemplo de cómo hacerlo:

css

Copiar código

p {

font-family: 'Roboto', sans-serif; /\* Fuente personalizada \*/

font-size: 16px; /\* Tamaño de fuente \*/

line-height: 1.6; /\* Altura de línea \*/

letter-spacing: 0.5px; /\* Espaciado entre letras \*/

text-align: justify; /\* Alineación \*/

margin: 20px 0; /\* Espacio vertical entre párrafos \*/

}

### Ejemplo Completo

Aquí tienes cómo se vería todo junto: Este enfoque mejora la legibilidad del texto y proporciona un diseño atractivo

#### HTML

html

Copiar código

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="styles.css">

<title>Estilización de Textos</title>

</head>

<body>

<h1>Encabezado Principal</h1>

<p>Este es un párrafo de ejemplo para estilizar. La legibilidad es importante para que los lectores puedan comprender fácilmente el contenido sin esfuerzo. Con un tamaño adecuado, altura de línea y espaciado, el texto se vuelve más accesible.</p>

</body>

</html>

#### CSS

css

Copiar código

h1, h2, h3 {

font-family: 'Roboto', sans-serif;

}

p {

font-family: 'Roboto', sans-serif;

font-size: 16px;

line-height: 1.6;

letter-spacing: 0.5px;

text-align: justify;

margin: 20px 0;

}

Muestre el CSS utilizado.

3.2 Colores y Fondos

1. Cree una paleta de colores para su proyecto utilizando el modelo de color HSL.

Justifique su elección.

2. Aplique un degradado de fondo a una sección de su sitio. El degradado debe cambiar

sutilmente al hacer hover.

3.3 Transiciones y Animaciones

1. Cree una transición suave para los enlaces de navegación al hacer hover.

2. Implemente una animación simple para un icono o logo en su página de inicio. La

animación debe repetirse infinitamente.

### 3.2 Colores y Fondos

#### 1. Crear una paleta de colores utilizando HSL

**Paleta de colores:**

* **Color Primario:** HSL(210, 100%, 50%) - Azul brillante
* **Color Secundario:** HSL(120, 60%, 40%) - Verde medio
* **Color de Fondo:** HSL(0, 0%, 95%) - Gris claro
* **Color de Texto:** HSL(0, 0%, 20%) - Negro suave
* **Color de Enlace:** HSL(210, 100%, 40%) - Azul oscuro

**Justificación:**

* **Azul Brillante (Color Primario)**: Evoca confianza y profesionalismo, ideal para sitios corporativos o de tecnología.
* **Verde Medio (Color Secundario)**: Aporta frescura y equilibrio, complementando bien al azul.
* **Gris Claro (Color de Fondo)**: Proporciona un contraste suave que no distrae del contenido, facilitando la lectura.
* **Negro Suave (Color de Texto)**: Mejora la legibilidad y es menos duro que el negro puro.
* **Azul Oscuro (Color de Enlace)**: Asegura que los enlaces sean claramente visibles y se relacionen con el color primario.

#### 2. Aplicar un degradado de fondo con hover

css

Copiar código

.section {

background: linear-gradient(135deg, hsl(210, 100%, 50%), hsl(120, 60%, 40%));

padding: 20px;

transition: background 0.5s ease; /\* Transición suave \*/

}

.section:hover {

background: linear-gradient(135deg, hsl(210, 80%, 50%), hsl(120, 40%, 40%)); /\* Cambia el degradado al hacer hover \*/

}

### 3.3 Transiciones y Animaciones

#### 1. Transición suave para los enlaces de navegación al hacer hover

css

Copiar código

nav a {

color: hsl(210, 100%, 40%);

text-decoration: none;

transition: color 0.3s ease; /\* Transición suave para el color \*/

}

nav a:hover {

color: hsl(120, 60%, 40%); /\* Cambia a color secundario al hacer hover \*/

}

#### 2. Animación simple para un icono o logo

css

Copiar código

.logo {

animation: bounce 2s infinite; /\* Aplicar animación de rebote infinitamente \*/

}

@keyframes bounce {

0%, 20%, 50%, 80%, 100% {

transform: translateY(0);

}

40% {

transform: translateY(-15px); /\* Sube 15px \*/

}

60% {

transform: translateY(-7px); /\* Sube 7px \*/

}

}

Con estos estilos CSS, tu proyecto contará con una paleta de colores armónica, un degradado atractivo que reacciona al hover, transiciones suaves en los enlaces y una animación divertida para el logo. Estas características mejorarán la experiencia del usuario y la estética de tu sitio.

PARTE 4: RESPONSIVE DESIGN

4.1 MEDIA QUERIES

1. CONCEPTO DE "mobile-first"

El enfoque **"mobile-first"** en el diseño web implica diseñar primero para dispositivos móviles y luego adaptar el diseño a pantallas más grandes. Este enfoque se basa en la premisa de que es más fácil escalar un diseño simple y funcional para pantallas más grandes que intentar simplificar un diseño complejo. Al priorizar la experiencia del usuario en dispositivos móviles, se asegura que el sitio sea accesible y utilizable en la plataforma más comúnmente utilizada.

**2. Implementar Media Queries**

Para hacer que tu sitio sea completamente responsive, puedes implementar media queries. Aquí tienes un ejemplo con tres breakpoints diferentes:

css

Copiar código

/\* Estilos básicos (mobile-first) \*/

body {

font-size: 16px;

background-color: white;

}

/\* Breakpoint para tabletas (min-width: 768px) \*/

@media (min-width: 768px) {

body {

font-size: 18px;

background-color: #f0f0f0; /\* Cambio de color de fondo \*/

}

h1 {

font-size: 2.5em; /\* Aumentar el tamaño del encabezado \*/

}

}

/\* Breakpoint para computadoras de escritorio (min-width: 1024px) \*/

@media (min-width: 1024px) {

body {

font-size: 20px;

background-color: #e0e0e0; /\* Otro cambio de color de fondo \*/

}

h1 {

font-size: 3em; /\* Aumentar aún más el tamaño del encabezado \*/

}

.container {

max-width: 1200px; /\* Limitar el ancho máximo del contenedor \*/

margin: 0 auto; /\* Centrar el contenedor \*/

}

}

**Explicación de los cambios en cada breakpoint:**

1. **Mobile (por defecto)**: Se aplican estilos básicos adecuados para dispositivos móviles, asegurando una buena legibilidad y usabilidad.
2. **Tableta (min-width: 768px)**: Aumenta el tamaño de la fuente y cambia el color de fondo, adaptando el diseño para pantallas más grandes, lo que mejora la legibilidad y el aspecto visual.
3. **Computadora de escritorio (min-width: 1024px)**: Se realiza otro aumento en el tamaño de la fuente y se establece un ancho máximo para el contenedor, mejorando la disposición general del contenido en pantallas más amplias.

**4.2 Imágenes Responsivas**

**1. Usar object-fit**

Para asegurar que las imágenes mantengan su relación de aspecto sin importar el tamaño del contenedor, puedes utilizar la propiedad object-fit. Aquí tienes un ejemplo:

css

Copiar código

.image {

width: 100%; /\* Asegurarse de que la imagen ocupe todo el ancho del contenedor \*/

height: 300px; /\* Altura fija \*/

object-fit: cover; /\* Mantiene la relación de aspecto \*/

}

**2. Implementar "Art Direction" con <picture>**

Utiliza el elemento <picture> para mostrar diferentes imágenes según el tamaño de pantalla:

html

Copiar código

<picture>

<source media="(min-width: 1024px)" srcset="images/imagen-grande.jpg">

<source media="(min-width: 768px)" srcset="images/imagen-mediana.jpg">

<img src="images/imagen-pequena.jpg" alt="Descripción de la imagen">

</picture>

**Explicación:**

* **<source>**: Se utiliza para definir diferentes imágenes para diferentes tamaños de pantalla.
* **media**: Especifica la condición para mostrar cada imagen.
* **srcset**: Define la ruta de la imagen correspondiente a cada tamaño.
* **<img>**: Se proporciona como un recurso de respaldo (fallback) para navegadores que no soportan el elemento <picture>.

**Resultado**

Con estas implementaciones, tu sitio será completamente responsive, asegurando que el contenido y las imágenes se ajusten adecuadamente a diferentes dispositivos.