Laboratorio 3 — Selectores, Box Model y Positioning

> Instituto Tecnológico de Costa Rica
> Escuela de Computación
> IC8057 - Introducción al Desarrollo de Páginas Web
> Laboratorio 3

Introducción

El diseño y maquetación de páginas web modernas requiere un dominio sólido de las hojas de estilo en cascada (CSS). Más allá de aplicar estilos básicos, es fundamental comprender la especificidad de los selectores, el uso correcto de pseudo-clases, así como el impacto del modelo de caja y el posicionamiento de elementos en la composición visual. Este laboratorio busca que los estudiantes apliquen de manera práctica conceptos de **selectores avanzados**, **especificidad**, **box model** y **positioning**, para reforzar las bases de un desarrollo web profesional y accesible.

Objetivo general

Aplicar técnicas avanzadas de **CSS3** —incluyendo selectores de tipo, clase, id, atributos, combinadores, pseudo-clases, así como el modelo de caja y el posicionamiento— con el fin de desarrollar páginas web con mayor precisión, consistencia visual y control sobre la presentación de los elementos.

Objetivos específicos

1. Aplicar selectores de tipo, clase e id.
2. Usar selectores combinadores (descendiente, hijo directo, adyacente, hermanos).
3. Emplear pseudo-clases de estado (ej. `:hover`, `:focus`, `:active`) y pseudo-clases estructurales (ej. `:first-child`, `:last-child`, `:nth-child`, `:not`, `:has`).
4. Demostrar especificidad: al menos un uso de `!important` para sobreescribir una clase y un estilo en línea que conviva con una clase (discute la decisión).
5. Dominar **box model**: `box-sizing: border-box`, márgenes, padding, colapso de márgenes y overflow-x controlado.
6. Implementar **positioning**: `position: relative` y `absolute` en un componente.

Parte 1 – Aplicación del CSS

Aplicarás CSS3 para refinar el sitio del evento del Lab 2, usando selectores (tipo, clase, id y atributos), combinadores, pseudo-clases, control del modelo de caja (márgenes, padding, overflow) y posicionamiento `relative/absolute`, destacando decisiones de especificidad (incluido un caso con `!important` y un estilo en línea).

Requisitos

1. **Crear una rama a partir del laboratorio 2 para trabajar el laboratorio**
    - Nombre de la rama: `lab3-selectores-boxmodel`
    - Al finalizar crear un *pull request* hacia `main` (**no hacer merge**)

- Crear un nuevo deploy en Netlify a partir del branch creado (se deben mantener ambos deploys)

2. **Se permite la modificación del documento HTML para:**

   - Agregar contenedores ligeros para organización visual (por ejemplo, `.container`, `.grid`, `.row`, `.card`).
   - Incorporar clases en elementos existentes para aplicar estilos (`class="btn"`, `class="card"`).
   - Asignar IDs a secciones principales para navegación y selectores (`#agenda`, `#expositores`).
   - Añadir atributos de datos para selectores de atributo.
   - Insertar elementos decorativos no semánticos necesarios para positioning (por ejemplo, un `<span class="badge">` dentro de `.card`).
   - Reordenar mínimamente elementos dentro de la misma sección si es indispensable para aplicar combinadores (hijo directo, adyacente, hermanos).

3. **Qué NO se debe modificar:**

   - No cambiar el significado semántico (no reemplazar `section` por `div` si pierde sentido; no convertir `h2` en `h3` solo para estilo).
   - No eliminar contenido ni accesibilidad existente (atributos `alt`, `aria-*`, texto).
   - No introducir JavaScript (este laboratorio es de CSS).
   - No romper la jerarquía de encabezados (mantener `h1 > h2 > h3`).

4. **Crear la carpeta `styles/` con cuatro archivos y enlazarlos en el `<head>` en este orden:**

   - `styles/base.css`: reset ligero, variables, tipografía y box-sizing global.
   - `styles/layout.css`: reglas de layout y combinadores (no responsive en este lab).
   - `styles/components.css`: componentes reutilizables (`.btn`, `.card`, `.tag`, `.badge`).
   - `styles/overrides.css`: sobrescrituras, casos de `!important` y excepciones.

5. **Uso de selectores**: Se debe incluir el uso de selectores en el el css

   - **Selectores de tipo**: Aplicar estilos a etiquetas HTML como `header`, `nav`, `section`, `img`, `p`, `h1`, `h2`.
   - **Selectores de clase**: Usar clases reutilizables como `.btn`, `.card`, `.tag`, `.badge`.
   - **Selectores de ID**: Estilizar secciones principales con IDs como `#agenda`, `#expositores`.
   - **Selectores de atributo (mínimo 3 distintos)**
     - `a[target="_blank"]`
     - `img[alt]`
     - `input[type="email"]`
     - `a[href^="https://"]`
     - `a[href$=".pdf"]`
   - **Selectores combinadores**
     - Descendiente: `.card p`
     - Hijo directo: `header > nav`
     - Adyacente: `nav a + a`
     - Hermanos: `.tag ~ .tag`

6. **Pseudo-clases (estado y estructurales)** : Se debe incluir el uso de psudo clases en el el css

   - **Pseudo-clases de estado (mínimo 3):**

     - `:hover` : cuando el usuario pasa el mouse.
     - `:focus-visible` : cuando el elemento recibe foco accesible (teclado/tab).
     - `:active` : cuando el elemento está siendo presionado.

     Ejemplo:

     ```
     .btn:hover {
       transform: translateY(-2px);
     }
     .btn:focus-visible {
       outline: 3px solid var(--c-accent);
     }
     .btn:active {
       transform: translateY(0);
     }
     ```

   - **Pseudo-clases estructurales (mínimo 3):**

     - `:first-child` : selecciona el primer elemento de un grupo.
     - `:last-child` : selecciona el último.
     - `:nth-child()` : selecciona elementos en base a un patrón (ej. pares, impares).
     - `:not()` : excluye elementos que coinciden con un selector.

     Ejemplo:

     ```
     .listado li:first-child {
       font-weight: bold;
     }
     .listado li:last-child {
       opacity: 0.8;
     }
     .listado li:nth-child(2n) {
       background: #0b1220;
     }
     .listado li:not(.activo) {
       border-left: 2px solid transparent;
     }
     ```

7. **Especificidad: `!important` + estilo en línea**

   - En `overrides.css`, sobrescribe una clase ejem `.badge` dentro de `.card` usando `!important`.
   - En el HTML, agrega un estilo en línea que conviva con una clase (deben verse ambos estilos).

   Ejemplo (HTML):

```
<h2 class="titulo-seccion" style="margin-bottom:24px;">Expositores</h2>
```

8. **Box model: márgenes, padding y colapso de márgenes**
   - Define elementos que elementos usen `box-sizing: border-box`.
   - Asegúrate de que los contenedores tengan márgenes y paddings adecuados.
   - Aplica el uso correcto del **colapso de márgenes** (ejemplo: un `h2` con `margin-bottom` seguido de una `.card` con `margin-top`).
9. **Overflow en un componente**
   - Dentro de una componente, coloca un texto largo que provoque **desbordamiento**.
   - Usa la propiedad `overflow` para controlarlo (`hidden`, `scroll` o `auto`).
10. **Uso de Display: Flex y Grid**
    - Utiliza `display: flex` en algún contenedor (por ejemplo, el `nav` o un grupo de botones). Aplica propiedades como:
      - `justify-content` (ej. `space-between`, `center`).
      - `align-items` (ej. `center`, `flex-start`).
      - `gap` para el espaciado entre elementos.
    - Utiliza display: grid en otra sección (por ejemplo, un listado de .card).
      - Define columnas con grid-template-columns y controla el espacio con gap.
      - Usa repeat() y minmax() para crear una cuadrícula adaptable.
11. **Uso de Position: Relative y Absolute**: Debes demostrar el uso de `position: relative` y `position: absolute` en al menos un caso dentro de tu sitio.
    - Elemento contenedor (`relative`)
    - Elemento posicionado (`absolute`)
12. **`README.md`**: Debes incluir un archivo `README.md` en la raíz del proyecto.

    En este archivo solo debes indicar **dónde aplicaste cada tipo de elemento solicitado** en el laboratorio, por ejemplo:

    - **Selectores de tipo**: aplicados en `header`, `nav`, `section`, `img`.
    - **Selectores de clase**: `.btn` en botones, `.card` en secciones de expositores.
    - **Selectores de ID**: `#agenda` y `#expositores` en secciones principales.
    - **Selectores de atributo**: `a[target="_blank"]` en enlaces externos, `img[alt]` en imágenes.
    - **Combinadores**: `nav a + a` en menú, `.card p` en tarjetas.
    - **Pseudo-clases de estado**: `:hover` en `.btn`, `:focus-visible` en enlaces.
    - **Pseudo-clases estructurales**: `:first-child` y `:nth-child()` en listas.
    - **Especificidad (`!important` e inline style)** → ejemplo en `.badge` y en un `h2`.
    - **Box model**: aplicado en `.card` y secciones con márgenes/padding.
    - **Overflow**: en un párrafo con texto largo dentro de `.card`.
    - **Flexbox**: en `nav` para alinear enlaces.

- **Grid**: en `.cards-container` para organizar tarjetas.
- **Position relative/absolute**: en `.banner .etiqueta` para destacar un texto.

Rubrica de evaluación

La evaluación del laboratorio se realizará bajo los siguientes criterios:

Rúbrica de evaluación — Laboratorio 3

Rúbrica de evaluación — Laboratorio 3

Criterio	1 - Insuficiente	3 - Aceptable	5 - Excelente
Estructura del proyecto (rama y organización de archivos)	No crea rama ni respeta estructura de carpetas.	Crea rama y organiza parcialmente, con errores en archivos o enlaces.	Rama creada correctamente y archivos CSS separados en `base`, `layout`, `components`, `overrides`.
Modificación del HTML (según reglas)	Cambia semántica, elimina accesibilidad o introduce JS.	Modifica mínimamente el HTML pero con algunos errores.	Ajusta HTML solo lo necesario, conserva semántica y accesibilidad.
Uso de selectores (tipo, clase, id, atributo, combinadores)	Usa muy pocos o con errores.	Aplica selectores básicos, faltan atributos o combinadores.	Aplica correctamente todos los tipos requeridos (≥3 atributos y los 4 combinadores).
Pseudo-clases (estado y estructurales)	No se aplican o están mal implementadas.	Se aplican parcialmente (solo estado o solo estructurales).	Implementa correctamente ≥3 de estado y ≥3 estructurales.
Especificidad (`!important` + inline style)	No demuestra especificidad.	Aplica un caso pero no queda claro.	Usa correctamente un `!important` y un inline style, identificados en README.
Box model y colapso de márgenes	No aplica box-sizing o hay errores en márgenes/paddings.	Aplica box-sizing pero sin mostrar colapso de márgenes.	Aplica `border-box`, márgenes/paddings correctos y demuestra colapso.
Overflow	No implementa control de overflow.	Implementa overflow pero no es claro.	Demuestra control de overflow en un texto u otro elemento dentro de `.card`.
Flexbox	No implementa flexbox.	Implementa flexbox pero sin propiedades relevantes.	Usa flexbox con `justify-content`, `align-items` y `gap` correctamente.
Grid	No implementa grid.	Implementa grid básico con errores.	Usa grid con `repeat()` o `minmax()` y un layout claro.

Criterio	1 - Insuficiente	3 - Aceptable	5 - Excelente
Position (relative + absolute)	No aplica relative/absolute.	Aplica relative/absolute pero mal ubicado.	Implementa relative/absolute correctamente sin afectar accesibilidad.
README.md (documentación breve)	No entrega README o está vacío.	Entrega README pero incompleto (faltan varios puntos).	README incluye dónde se aplicó cada tipo de elemento solicitado (lista clara).

Entrega

Cada estudiante deberá entregar un archivo comprimido (`.zip`) con el proyecto completo, siguiendo estas instrucciones:

1. Una carpeta con el **repositorio completo** (todos los archivos del proyecto).
2. Un archivo denominado `enlaces.txt` que contenga:
    - **URL del pull request público en GitHub**
    - **URL del sitio desplegado** (Netlify)

Fecha límite: Domingo 24 de agosto a las 11:55 p.m.

Forma de entrega: El archivo `.zip` debe subirse a la plataforma **TEC Digital**, en el espacio correspondiente al curso.

Portafolio del curso: Este laboratorio forma parte del **portafolio individual del curso**.