PROGRAMACION WEB
Bienvenidos!!!

Julio Monetti
julio.monetti@itu.uncu.edu.ar

PROGRAMACION WEB

Horario

Comisión A.
Presencial: lunes 16:15 a 18:30
Virtual: Martes 17:50 a 19:10

Comisión B.
Presencial: lunes 18:30 a 20:30
Virtual: Martes 17:50 a 19:10

Aula Virtual: https://aulas.itu.uncu.edu.ar/itu/login/index.php

PROGRAMACION WEB

- HTML
- CGI
- CSS
- VANILLA JAVASCRIPT
- NODE
- REACT
- SPRING

PROGRAMACION WEB

Cómo Aprobar la Materia ?

- Trabajo en clase
- 2 parciales (Presentación del portafolio de trabajos)
- Presentación grupal de un tema de interés
- EGI (Proyecto Final)

Internet y la WWW

- ✓ **Internet**
- ✓ **WWW. Sitios Web y páginas Web.**
- ✓ **Otros objetos en la Web.**
- ✓ **¿Cómo funciona?**

Internet y la WWW

La WEB

La **Web** (abreviatura de "World Wide Web") es un sistema de distribución de información que permite acceder a documentos y recursos conectados entre sí mediante **hipervínculos**. Estos recursos están disponibles a través de **Internet**, pero la Web es solo una parte de Internet (que también incluye correo electrónico, FTP, etc.).

Internet y la WWW

Componentes Principales de la WEB

- **Cliente (navegador web)**
Es el programa que usamos para ver sitios web, como Chrome, Firefox o Safari. Envía peticiones y muestra las respuestas.

- **Servidor web**
Es una computadora que almacena páginas web y las entrega al cliente cuando se solicitan. Ej: Apache, Nginx.

- **Protocolo HTTP/HTTPS**
Es el lenguaje que usan el navegador y el servidor para comunicarse. HTTP (HyperText Transfer Protocol) define cómo se solicitan y entregan los recursos web.

- **URL (Uniform Resource Locator)**
Es la dirección de un recurso en la web. Ejemplo: https://www.ejemplo.com/index.html.

Línea de Tiempo Programación WEB

Década de 1990: Los comienzos

- **1990** – Se crea la **World Wide Web** (Tim Berners-Lee).
- **1991** – Aparece el primer sitio web. Se usa **HTML 1.0**.
- **1993** – Se lanza **Mosaic**, el primer navegador gráfico.
- **1994** – Nace **Netscape Navigator**.
- **1995**
  - Se lanza **JavaScript** (por Netscape).
  - Aparece **PHP**, creado por Rasmus Lerdorf. (como un conjunto de CGI en C).
  - Se introduce **HTML 2.0**.
- **1996** – Microsoft lanza **ASP (Active Server Pages)**.
- **1997** – Se lanza **HTML 4.0** y **CSS 1**.
- **1999** – Aparece **XML**, y **AJAX** empieza a utilizarse (aunque se populariza más tarde).

Línea de Tiempo Programación WEB

Década de 2000: Web dinámica y 2.0

- **2000** – Microsoft lanza **.NET**.
- **2002** – Nace **ASP.NET**.
- **2003** – Se lanza **WordPress**.
- **2004** – Surgen los conceptos de **Web 2.0** (interactividad, redes sociales).
- **2005**
  - Google populariza **AJAX** con Gmail y Google Maps.
  - Se crea **jQuery**.
- **2006** – Primeras versiones de **Amazon Web Services (AWS)**.
- **2008** – Nace **Google Chrome**.
- **2009** – Se lanza **Node.js**, permitiendo JavaScript del lado del servidor.

Línea de Tiempo Programación WEB

Década de 2010: SPA y frameworks modernos

- **2010** – Se empieza a usar **HTML5** y **CSS3**.
- **2010** – Lanzamiento de **AngularJS** (Google).
- **2011** – Facebook lanza **React**.
- **2014** – Se lanza **Vue.js**.
- **2015** – Se publica **ECMAScript 6 (ES6)** con mejoras clave en JavaScript.
- **2016** – Surge **Angular 2+**, una reescritura total de AngularJS.
- **2018** – Se populariza **TypeScript** junto a Angular y React.
- **2019** – JAMstack gana fuerza (JavaScript + APIs + Markup).

Línea de Tiempo Programación WEB

2020s: Web moderna, serverless y AI

- **2020** – Crece el uso de **serverless** (Firebase, Vercel, Netlify).
- **2020** – Se populariza **Next.js.** React SSR (Server-Side Rendering) y SSG (Static Site Generation) .
- **2021** – Aparecen frameworks como **SvelteKit** y **Remix**.
- **2022** – Boom de **WebAssembly (WASM)** para ejecutar código C, Rust, etc. en navegador.
- **2023** – Crecen herramientas como **Bun**, **Vite**, y se acelera el desarrollo por **IA**.
- **2024–2025** – Se integra IA generativa en herramientas de desarrollo web (copilots, asistentes). Web se vuelve más **modular**, **optimizada** y **descentralizada**.

INSTITUTO TECNOLÓGICO UNIVERSITARIO

HTML

HTML

- ✓ **HTML (Hypertext Markup Language)**
- ✓ **HTML = Hypertext + MarkUp**
- ✓ ***Hypertext***

  Es texto ordinario al que se le incorporan funcionalidades adicionales como:
  - ✓ **Formato,**
  - ✓ **Imágenes,**
  - ✓ **Multimedia**
  - ✓ **Y enlaces a otros documento.**
- ✓ ***MarkUp***

  Es el proceso de tomar el texto ordinario e incorporarle símbolos adicionales. Cada uno de estos símbolos identifica a un comando que le indica al navegador como mostrar ese texto.

Historia del HTML

Tim Berners-Lee
Director del W3C

- ✓ **En 1986 la organización internacional de Estándares publica el SGML (Standard Generalized Markup Language)**
- ✓ **En 1990 Tim Berners-Lee crea la WWW y el HTML con base en un subconjunto del SGML.**
- ✓ **En 1993 se crea el HTML 2.0 (o HTML+)**
- ✓ En 1993 nace CGI como una forma de permitir que los servidores web ejecutaran programas externos (scripts) para generar contenido dinámico en las páginas web.
- ✓ **En 1995 el W3C (World Wide Web Consortium) define el HTML 3.0**
- ✓ **El HTML 3.2 abandona las sugerencias del HTML 3.0 y adopta elementos desarrollados por Netscape. (Incorpora Tablas, Applets, Texto alrededor de imágenes)**
- ✓ **En 1997 se define el estándar HTML 4.0**
- ✓ **En 1999 aparece el estándar HTML 4.01**
- ✓ **En 2014 se publica oficialmente el HTML5.** Gran avance para la web moderna. Nuevas etiquetas semánticas: <header>, <footer>, <article>, etc. Multimedia: <audio>, <video>, <canvas>. APIs: geolocalización, almacenamiento local, arrastrar y soltar. Mejora formularios y accesibilidad.

Más información en: http://www.w3.org/

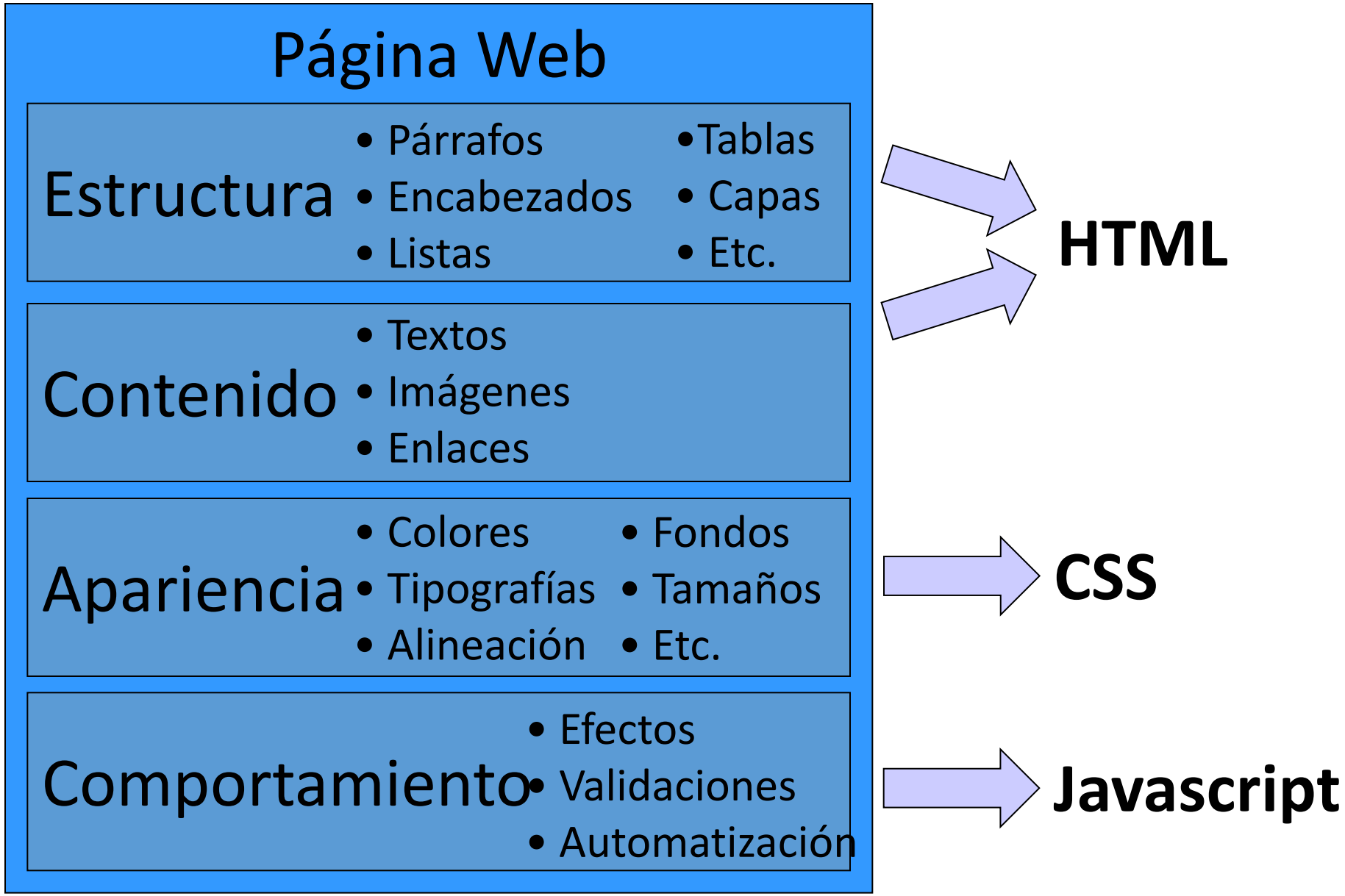
El W3C

- ✓ El W3C (World Wide Web Consortiun) es un consorcio internacional donde Organizaciones miembro, Personal Full-time y el público en general trabajan para desarrollar Estándares Web.
- ✓ La misión del W3C es la de maximizar el potencial de la WWW desarrollando protocolos y guías que aseguren el crecimiento futuro de la Web.
- ✓ Algunas Organizaciones miembro del W3C

Adobe	Ericsson	Nokia
Apple	Google, inc.	Opera Software
AT&T	HP	Sun Microsystems
Cisco	IBM Corporation	Telefónica de España
Citigroup	Microsoft	Yahoo, inc.
Deutsche Telekom	Mozilla Foundation	VeriSign

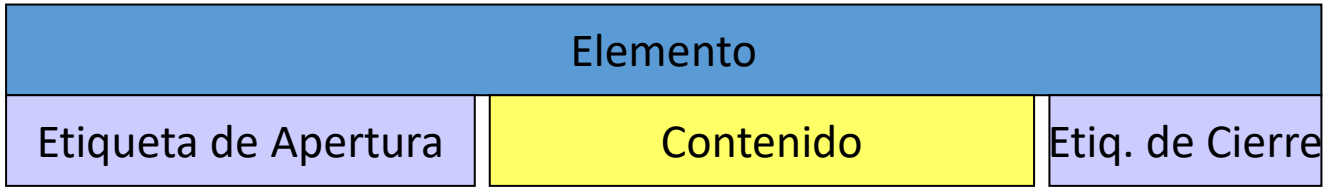
…Y decenas de universidades de todo el mundo

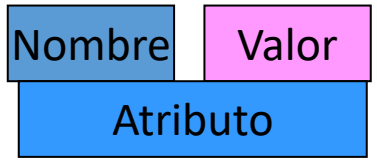
HTML – CSS – Javascript



Elementos HTML

- ✓ **Los elementos son los componentes fundamentales del HTML**
- ✓ **Cuentan con 2 propiedades básicas:**
  - ✓ **Atributos**
  - ✓ **Contenido**
- ✓ **En general se conforman con una Etiqueta de Apertura y otra Cierre.**
- ✓ **Los atributos se colocan dentro la etiqueta de apertura, y el contenido se coloca entre la etiqueta de apertura y la de cierre.**





Atributos

- ✓ **Los atributos de un elemento son pares de nombres y valores separados por un ‘=‘ que se encuentran dentro de la etiqueta de apertura de algún elemento. Los valores deben estar entre comillas.**

  Ejemplos:

```
<span id=‘iddeesteelemento‘   style='color:red;‘   title=‘Curso de HTML'>
        Curso de HTML
</span>

<a href=“http://www.uncu.edu.ar” class=“milink” target=“_blank”>
        Universidad Nacional de Cuyo
</a>
```

Tipos de Elementos

- ✓ **Algunos tipos de Elementos**
  - ✓ ***Estructurales***
    - ✓ **Describen el propósito del texto y no denotan ningún formato específico.**
    - ✓ **Por ejemplo:**

      <h1>Curso HTML</h1>
  - ✓ ***De Presentacion***
    - ✓ **Describen la apariencia del texto, independientemente de su función.**
    - ✓ **Por ejemplo:**

      Curso HTML
    - ✓ **Los elementos de presentación se encuentran obsoletos desde la aparición del CSS.**
  - ✓ ***De HiperTexto***
    - ✓ **Relaciona una parte del documento a otros documentos.**
    - ✓ **Por ejemplo:**

      Universidad del Cuyo

Estructura de un Documento HTML

- ✓ **<HTML>… </HTML>**
  - ✓ **Delimita el Documento HTML**
- ✓ **<HEAD> … </HEAD>**
  - ✓ **Delimita el encabezado del Documento HTML**
  - ✓ **En general incluye los metadatos del documento y Scripts.**
- ✓ **<BODY> … </BODY>**
  - ✓ **Delimita el Cuerpo del Documento HTML.**
  - ✓ **Es donde se incluyen los contenidos visibles del documento.**

- ✓ **Ejemplo**

```
<html>
   <head>
                    Aquí se incluyen los distintos elementos del encabezado
   </head>
   <body>
                    Aquí se incluyen los distintos elementos contenedores o scripts
   </body>
</html>
```

Elementos del HEAD

Alguno de los elementos factibles de incluir en el HEAD son:

- ✓ **<TITLE> … </TITLE>**
  - ✓ **Define el título del documento HTML**
- ✓ **<SCRIPT> … </SCRIPT>**
  - ✓ **Se utiliza para incluir programas al documento. En general se tratan de Javascripts.**
- ✓ **<STYLE> … </STYLE>**
  - ✓ **Especifica un estilo CSS para ser utilizado en el documento.**
- ✓ **<META> … </META>**
  - ✓ **Permite especificar información de interés como: autor, fecha de publicación, descripción, palabras claves, etc.**

Elementos del HEAD

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8"> <!-- Codificación de caracteres -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Escalado en móviles -->
  <meta name="description" content="Página de ejemplo sobre etiquetas meta en HTML5."> <!-- Descripción para buscadores -->
  <meta name="keywords" content="HTML5, meta, etiquetas, ejemplo"> <!-- Palabras clave (menos usado hoy) -->
  <meta name="author" content="Julio Monetti"> <!-- Autor del documento -->
  <meta http-equiv="refresh" content="10;url=https://ejemplo.com"> <!-- Redirección después de 10 segundos -->
   <meta http-equiv="X-UA-Compatible" content="IE=edge"> <!-- Compatibilidad con IE -->
  <title>Ejemplo de Meta</title>
</head>
<body>
  <h1>Hola Mundo</h1>
</body>
</html>
```

Contenedores (Block-Level Elements)

- ✓ **Párrafos**
  - ✓ **Es el contenedor mas común.**
  - ✓ **Su sintaxis es: <P> … </P>**
- ✓ **Encabezados**
  - ✓ **Indican una jerarquía de secciones dentro del documento**
  - ✓ **Su sintáxis: <h1>…</h1>, <h2>…</h2>, <h3>…</h3>,.. <h6>…</h6>,**
- ✓ **Listas**
  - ✓ **Listas de Definiciones (consistente de pares de términos y definiciones)**
    - ✓ **<dl>…</dl> Crea la lista**
    - ✓ **<dt>…</dt>** Crea un nuevo término
    - ✓ **<dd>…</dd>** Crea una nueva definición
  - ✓ **Lista Ordenada Enumerada**
    - ✓ ** … Crea una nueva lista**
    - ✓ ** … Crea un nuevo ítem en la lista**
  - ✓ **Lista Ordenada No Enumerada**
    - ✓ ** … Crea una nueva lista**
    - ✓ ** … Crea un nuevo ítem en la lista**
- ✓ **Capas**
  - ✓ **Permiten agrupar y diagramar contenidos en los documentos.**
  - ✓ **Su sintaxis es: <DIV> … </DIV>**

Contenedores (HTML5)

Contenedor	Para qué se usa
<header>	Cabecera del sitio o sección
<nav>	Menú de navegación
<main>	Contenido principal del documento
<section>	Sección temática del contenido
<article>	Contenido independiente (como una noticia o post)
<aside>	Información secundaria (como una barra lateral)
<footer>	Pie de página

Práctica a1
Contenedores (HTML5)

Utilice las etiquetas

- header
- nav
- main
- section
- article
- aside
- footer

Para generar la siguiente página

Mi Sitio Web (colocar esto en un header)

Bienvenido a mi página de ejemplo

- Inicio
- Acerca
- Contacto

Sección principal (usar la etiqueta section)

Artículo 1

Este es el contenido del primer articulo (usar la etiqueta article).

Artículo 2

Este es el contenido del segundo artículo

Información adicional

Noticias, enlaces, publicidad o cualquier contenido secundario.

Tablas

- ✓ **Qué son y para qué sirven.**
- ✓ **Atributos de las Tablas**
- ✓ **Atributos de las Celdas y Filas**
- ✓ **Prioridades en los formatos**
- ✓ **Tablas anidadas**
- ✓ **Tablas Irregulares (Atributos colspan y rowspan)**
- ✓ **Anchos (Pixels Vs. Porcentajes)**

Contenedores (Tablas)

- ✓ **<table> … </table>** — **Crea la tabla**
- ✓ **<tr> … </tr>** — **Crea una nueva fila**
- ✓ **<td> … </td>** — **Crea una nueva celda dentro de la fila**

Por ejemplo: Creación de una tabla de 2 x 2

```
<table>
   <tr>
          <td> … </td>
          <td> … </td>
   </tr>
   <tr>
          <td> … </td>
          <td> … </td>
   </tr>
</table>
```

Contenedores (Tablas)

```
<table border="1" style="width: 100%;">
 <colgroup>
  <col style="width: 150px;"> <!-- pixeles -->
  <col style="width: 60%;">  <!-- porcentaje -->
  <col style="width: 40%;">  <!-- porcentaje -->
 </colgroup>
 <tr>
  <th>Fijo 150px</th>
  <th>60%</th>
  <th>40%</th>
 </tr>
 <tr>
  <td>Dato 1</td>
  <td>Dato 2</td>
  <td>Dato 3</td>
 </tr>
</table>
```

Si se combinan píxeles y porcentajes, el navegador hará un cálculo proporcional para el ancho total.

Con colgroup se definen todos los anchos en un solo lugar y no se necesita repetirlos en cada celda.

Práctica a2
tablas

Cree una página web que contenga una tabla con el siguiente layout

<table>
<tr><th>Nombre</th><th>Edad</th><th colspan="2">Ciudades</th><th>Profesión</th></tr>
<tr><td>Ana</td><td>28</td><td colspan="2">Buenos Aires</td><td rowspan="3">Ingeniero</td></tr>
<tr><td>Lucas</td><td rowspan="2">35</td><td>Córdoba</td><td>Mendoza</td></tr>
<tr><td>Martina</td><td>Rosario</td><td>Córdoba</td></tr>
</table>

Hipervínculos

- ✓ ¿Qué es un hipervínculo?
- ✓ Ubicación y rutas de documentos.
  - ✓ Rutas Absolutas
  - ✓ Rutas Relativas
  - ✓ Rutas relativas a la raíz del sitio
- ✓Vínculos a otras páginas. Etiqueta <A>.
- ✓Uso del atributo target (Destino).
- ✓ Anclaje de nombre. Atributo name.
- ✓Comportamientos del Navegador ante distintos tipos de archivos enlazados.

Hipervínculos

Un **hipervínculo** es un enlace que permite **navegar entre páginas web**, ya sea dentro del mismo sitio o hacia otros sitios externos. Es una de las características clave que hizo posible la "web" como una red de documentos interconectados.

Sintaxis Básica

```
<a href="URL">Texto del enlace</a>
```

Enlace a otra página externa

```
<a href="https://www.google.com">Ir a Google</a>
```

Enlace a otra página del mismo sitio

```
<a href="contacto.html">Ir a la página de contacto</a>
```

Hipervínculos

Enlace a una parte específica de la misma página (anclas)

```
<h2 id="seccion1">Sección 1</h2>

<a href="#seccion1">Ir a la Sección 1</a>
```

Enlace para enviar un correo

```
<a href="mailto:ejemplo@correo.com">Enviar correo</a>
```

Enlace que abre en una nueva pestaña

```
<a href="https://www.wikipedia.org" target="_blank">Wikipedia</a>
```

Hipervínculos

Atributos comunes

Atributo	**Descripción**
href	Dirección a la que apunta el enlace
target	Define dónde abrir el enlace (_blank, _self, top_ etc.)
title	Texto que aparece al pasar el mouse sobre el enlace
download	Indica que el recurso se descargará (si es posible)

Ver página ejercicio_hipervinculos.html!!!

Práctica a3
Hipervínculos

Ejercicio:

Crea una página con enlaces:
- Uno que lleve a Google
- Otro que lleve a una segunda página del sitio, pero que se abra en una segunda pestaña
- Otro que abra un correo a tu dirección
- Agregar un enlace que lleve a una sección inferior de la misma página usando un id.
- Agregar un enlace que abra un archivo .pdf y que tenga el atributo download.

Imágenes

- ✓ **Elemento <IMG>**
- ✓ **¿Qué imágenes se pueden usar?**
- ✓ **Ventajas y desventajas de cada formato.**
- ✓ **Imágenes e Hipervínculos**

Imágenes

Formato	Transparencia	Animación	Compresión	Ideal para
JPEG	❌	❌	Con pérdida	Fotos
PNG	✅	❌	Sin pérdida	Logos, gráficos
GIF	✅ (limitada)	✅	Con pérdida	Animaciones simples
SVG	✅	✅	Vectorial	Iconos, logos escalables
WebP	✅	✅	Muy eficiente	Imágenes modernas en la web

Formularios

- ✓ **¿Para qué sirven?**
- ✓ **Elementos para Formularios**
  - ✓ **Campos de Texto**
  - ✓ **Casillas de Verificación**
  - ✓ **Botones de opción**
  - ✓ **Menús**
  - ✓ **Botones**
  - ✓ **Campos ocultos**
  - ✓ **Campos de carga de archivos**
- ✓ **¿Cómo se envía la información?**
- ✓ **¿Se pueden validar los Campos?**

Ver pagina formulario.html!!!

Formularios

- ✓ **Elemento <FORM>**

  Atributos: method, action

- ✓ **Elemento <INPUT>**

  Atributo: type (text, checkbox, radio, button, hidden)

- ✓ **Elemento <SELECT>**
- ✓ **Elemento <TEXTAREA>**

Formularios

Atributo	Función	Ejemplo
required	Obliga a completar el campo.	<input type="text" required>
type	Valida formato (email, url, number, date...).	<input type="email">
pattern	Valida con una expresión regular.	<input type="text" pattern="[A-Za-z]{3,}">
min / max	Rango para números o fechas.	<input type="number" min="1" max="10">
minlength / maxlength	Longitud mínima y máxima de texto.	<input type="text" minlength="3" maxlength="10">
step	Intervalo de incremento para números o fechas.	<input type="number" step="5">
multiple	Permite varios valores (emails, archivos).	<input type="email" multiple>
placeholder	Texto de ayuda. *(No es validación, pero mejora la usabilidad)*	<input placeholder="Escribe tu nombre">

Más Elementos

- ✓**Otros Elementos**
  - ✓ **Nueva linea
 y Línea Horizontal <hr>**
  - ✓**Encabezados <h1> a <h6>**
  - ✓**Párrafos <p>**
  - ✓**Enfasis y formato   <b> <i> <u>**
  - ✓**Etiquetas semánticas: <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>**
  - ✓**Comentarios. <!- - xxxxx - ->**

Elementos embebidos

En HTML, **los elementos embebidos** son aquellos que permiten **incrustar contenido externo** o interactivo dentro de la página web: imágenes, videos, audio, documentos, aplicaciones, etc.
Son muy útiles para enriquecer el contenido y no limitarse solo a texto y formato.

- Permiten **integrar** recursos que no forman parte directamente del HTML, sino que provienen de **fuentes externas** o de archivos dentro del mismo proyecto.
- Se usan para mostrar **imágenes, videos, audio, mapas, documentos PDF, aplicaciones interactivas, etc.**
- Normalmente utilizan **etiquetas específicas** que indican al navegador cómo renderizar el recurso.

Elementos embebidos

Elemento	Descripción	Ejemplo básico
<img>	Inserta imágenes.	<img src="foto.jpg" alt="Descripción">
<audio>	Reproduce audio.	<audio src="cancion.mp3" controls></audio>
<video>	Muestra video.	<video src="video.mp4" controls></video>
<iframe>	Incrusta otra página o aplicación.	<iframe src="https://www.wikipedia.org" width="400" height="300"></iframe>
<embed>	Inserta contenido interactivo como PDFs o multimedia.	<embed src="documento.pdf" type="application/pdf" width="500" height="400">
<object>	Similar a <embed>, más versátil.	<object data="animacion.swf" type="application/x-shockwave-flash"></object>
<canvas>	Área para gráficos dibujados con JavaScript.	<canvas id="miCanvas" width="300" height="150"></canvas>
<svg>	Dibuja gráficos vectoriales.	<svg width="100" height="100"><circle cx="50" cy="50" r="40" fill="red" /></svg>

Estilos CSS

- ✓ **Antes de la aparición de los estilos, la presentación se manejaba directamente dentro de los elementos HTML por medio de atributos. Por ejemplo:**

```
<h2 align="center">
        <font color="blue" size="3" face="Times New Roman, serif">
                <i>Seminario de HTML ITU</i>
        </font>
</h2>
```

- ✓ **CSS permite separar el contenido de un documento de su presentación.**

En el documento HTML:

```
<h2>Seminario de HTML ITU</h2>
```

En la hoja de estilos se define el formato de los encabezados h2:

```
h2 {
        text-align: center;
        color: blue;
        font: italic large "Times New Roman", serif;
}
```

Ventajas del uso de CSS

- ✓ **Estandarizar la presentación de un sitio web completo. Haciendola fácil de mantener.**
- ✓ **Diferentes usuarios pueden contar con diferentes estilos acordes a sus necesidades. Ejemplos:**
  - ✓ **Estilos para personas con dificultades visuales,**
  - ✓ **Estilos para dispositivos móviles,**
  - ✓ **Estilos para dispositivos monocromos,**
  - ✓ **Estilos para impresión,**
  - ✓ **Etc.**
- ✓ **Los documentos HTML se reducen en tamaño y complejidad.**

Qué puedo configurar con CSS ?

- ✓**Propiedades de fuentes**
- ✓**Propiedades de color y fondo**
- ✓**Propiedades de texto**
  - ✓**espaciado de palabras**
  - ✓**alineación**
- ✓**Propiedades de caja**
  - ✓**Margen**
  - ✓**Borde**
  - ✓**Relleno**
- ✓**Estilos de listas**

Estilos CSS

Una hoja de estilos consiste en un conjunto de reglas.

Cada regla esta formada por:

- ✓ **El Selector (nombre del estilo)**
- ✓ **La Declaración (define el estilo)**
  - ✓ **Propiedad**
  - ✓ **Valor**

¿Qué podemos hacer con los estilos?

- ✓ **Redefinir estilos de Etiquetas HTML.**
- ✓ **Crear Estilos Personalizados para uso genérico (Clases)**
- ✓ **Crear Estilos para un elemento HTML específico (por Id)**

```
h2 {
    text-align: center;
    color: blue;
    font: italic large "Times New Roman", serif;
}
```

```
.textoresaltado {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 12px;
    font-style: normal;
    font-weight: bold;          /* Esto es un comentario */
    color: #000000;
}
```

```
#logo {
    background-image: url("/img/logo.gif");
    background-position:center;
    background-repeat:no-repeat;
    height: 50px; width: 150px;
    position: absolute; left: 0px; top: 0px;
}
```

Como incluir estilos CSS

- ✓ ***Inline Styles***

  Utilizando el atributo "style" se define el estilo de un elemento HTML en forma individual.

- ✓ ***Embedded Style***

  Se define la regla CSS dentro de un documento HTML. Se puede aplicar a cualquier elemento de ese documento.

- ✓ ***Hojas de Estilos externas***

  Un archivo CSS independiente que se encuentra referenciado en cada uno de los documentos HTML que desean utilizarlo.

```
<h2 style="color: blue; background: green;">
    Curso HTML ITU
</h2>
```

```
<head>
    <style type="text/css">
        h2 {
            font-style: italic;
            font-weight: bold;
            color: blue;
        }
    </style>
</head>
<body>
    <h2>Curso HTML ITU</h2>
</body>
```

```
<head>
<link rel=stylesheet type="text/css" href="estilos.css">
</head>
```

Práctica a4

Crear una página WEB con tus datos. La misma debe contener como mínimo.

- Una tabla.
- Una imagen.
- Un div con un estilo específico
- Una referencia a una segunda página.
- Un formulario con al menos 5 campos de diferente tipo.
- Una lista de elementos.
- 5 estilos con al menos 4 atributos cada uno. (de clase e identificador). En una hoja de estilos externa.

Javascript

- ✓ Creado por Brendan Eich para Netscape. Aparece en Netscape 2.0B3 en 1995.
- ✓ Es un lenguaje de programación interpretado con base en la sintaxis del lenguaje C.
- ✓ Está basado en objetos y guiado por eventos.
- ✓ Se ejecuta en el navegador, y en el servidor
- ✓ No tiene nada que ver con Java
- ✓ Las funciones Javascript se incluyen en los documentos HTML interactuando con el DOM (Document Object Model) de la página para realizar tareas que HTML no puede realizar.

Brendan Eich
Creador del Javascript

DOM:

Es una interface (independiente del lenguaje) que permite a los scripts acceder dinámicamente y actualizar el contenido, la estructura y el estilo de los documentos.

Capacidades de Javascript

- ✓ **Algunas de las cosas que se pueden hacer con Javascript:**
  - ✓ **Abrir nuevas ventanas controlando su tamaño, look & feel, controles, etc.**
  - ✓ **Incorporar validaciones a los formularios.**
  - ✓ **Cambios de imágenes al colocar el mouse sobre algún objeto de la página web.**
  - ✓ **Generar respuestas ante distintos eventos**
  - ✓ **Efectos visuales en la página.**
  - ✓ **Crear, Eliminar o cambiar atributos de elementos de una página HTML en forma dinámica.**
  - ✓ **Crear o Leer Cookies**
  - ✓ **Detectar la configuración del Browser.**

Dónde incluir el Javascript

- ✓ En general se utiliza el elemento *script*
- ✓ Las funciones Javascript pueden estar en archivos independientes. Por ejemplo:

```
<script language="JavaScript" src ="archivo.js"> </script>
```

- ✓ También se pueden incluir las instrucciones dentro del elemento Script. Por ejemplo:

```
<script language="JavaScript" type="text/JavaScript">
    function AbroVentana (URL,nombre,features) { //Esto es un comentario
            window.open(URL,nombre,features);
    }
</script>
```

- ✓ El código Javascript también se puede incluir directamente en un evento asociado a algún elemento del documento. Por ejemplo:

```
<input type="button" onclick="alert('Gracias por su click');return false;" value="Click">
```

Ejemplo Javascript

```
<!DOCTYPE html>
<html>
<body>
    <h1>Hola JavaScript</h1>
    <script>
        console.log("¡Hola desde JavaScript!");
        alert("Bienvenido al curso de JavaScript");
    </script>
</body>
</html>
```

Ejemplo Javascript

```
let nombre = "Juan";
const PI = 3.1416;
let edad = 25;
console.log("Hola " + nombre + ", tienes " + edad + " años.");
```

Ejemplo Javascript. Estructuras de Control

```
let nota = 8;
if (nota >= 6) {
    console.log("Aprobado");
} else {
    console.log("Desaprobado");
}
```

Ejemplo Javascript. Estructuras de Control

```
switch (fruit) {
  case 'manzana':
    console.log('Rojo');
    break;
  case 'banana':
    console.log('Amarillo');
    break;
  default:
    console.log('Color desconocido');
}
```

Ejemplo Javascript. Estructuras de Control

```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}

for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

```
let j = 0;
do {
  console.log(j);
  j++;
} while (j < 5);
```

Ejemplo Javascript. Arrays

```
let frutas = ["Manzana", "Banana", "Pera"];
frutas.push("Mango");
console.log(frutas);
```

Ejemplo Javascript. Arrays

Diferentes formas de creación:

```
// Declaración simple
let numeros = [1, 2, 3, 4, 5];

// Array con longitud fija (valores vacíos)
let vacios = new Array(5);  // [ <5 ítems vacíos> ]

// Array lleno con un valor
let relleno = Array(5).fill(0); // [0,0,0,0,0]

// Array a partir de otro objeto iterable
let texto = "Hola";
let letras = Array.from(texto); // ["H", "o", "l", "a"]

// Array generado dinámicamente
let cuadrados = Array.from({length: 5}, (_, i) => i * i); // [0,1,4,9,16]
```

Ejemplo Javascript. Arrays

Búsqueda

```
let arr = [10, 20, 30, 40, 50];

console.log(  arr.includes(20)  ); // true
console.log(  arr.indexOf(30)  );  // 2
console.log(  arr.find(x => x > 25)  ); // 30
console.log(  arr.findIndex(x => x > 25)  ); // 2
```

Ejemplo Javascript. Arrays

Transformaciones con map, filter y reduce.

```
let nums = [1, 2, 3, 4, 5];
// Map: transformar
let dobles = nums.map(n => n * 2); // [2,4,6,8,10]
// Filter: filtrar
let pares = nums.filter(n => n % 2 === 0); // [2,4]
// Reduce: acumular
let suma = nums.reduce((acc, n) => acc + n, 0); // 15
// Combinados
let resultado = nums
    .filter(n => n % 2 !== 0) // impares [1,3,5]
    .map(n => n ** 2)        // cuadrados [1,9,25]
    .reduce((a, b) => a + b); // 35
```

Ver ordenar_agrupar.html!!!

Ejemplo Javascript. Arrays

Ordenar y agrupar

Ejemplo Javascript. Arrays

Ordenar y agrupar

```
let arr1 = [1, 2];
let arr2 = [3, 4];
let combinado = [...arr1, ...arr2]; // [1,2,3,4]
// Concatenar
let arr3 = arr1.concat(arr2); // [1,2,3,4]
// Aplanar
let anidado = [1, [2, 3], [4, [5, 6]]];
console.log(anidado.flat(2)); // [1,2,3,4,5,6].  El parámetro (2) representa niveles de aplanado
// Transformar y aplanar a la vez
let frases = ["hola mundo", "adios sol"];
let palabras = frases.flatMap(f => f.split(" "));   // ["hola","mundo","adios","sol"]
```

Ejemplo Javascript. Arrays

Desestructuración

```
let [primero, segundo, ...resto] = [10, 20, 30, 40];
console.log(primero); // 10
console.log(resto);   // [30,40]

// Intercambiar valores
let a = 1, b = 2;
[a, b] = [b, a];
console.log(a, b); // 2, 1
```

Ejemplo Javascript. Arrays

Set y Map con Arreglos

```
// Eliminar duplicados
let repetidos = [1,2,2,3,3,4];
let unicos = [...new Set(repetidos)]; // [1,2,3,4]
```

Set es una colección de valores únicos

```
// Contar frecuencia
let palabras2 = ["sol", "luna", "sol", "estrella", "luna"];
let conteo = palabras2.reduce((map, p) => {
  map[p] = (map[p] || 0) + 1;
  return map;
}, {});        //empieza con un arreglo vacío
console.log(conteo); // { sol:2, luna:2, estrella:1 }
```

Ejemplo Javascript. Arrays

Iteradores

```
let numeros3 = [10, 20, 30];

// for...of
for (let n of numeros3) console.log(n);

// entries() → índice y valor
for (let [i, n] of numeros3.entries()) console.log(i, n);

// keys() y values()
console.log([...numeros3.keys()]);   // [0,1,2]
console.log([...numeros3.values()]); // [10,20,30]
```

Práctica a4_3. Arreglos en Javascript

Realice los siguientes ejercicios en javascript

const numeros = [3, 6, 9, 12, 15, 18];
Genera un nuevo arreglo que contenga los cubos de los números pares.

```
const empleados = [
  { nombre: "Ana", departamento: "Ventas" },
  { nombre: "Luis", departamento: "IT" },
  { nombre: "Marta", departamento: "Ventas" },
  { nombre: "Pedro", departamento: "IT" },
  { nombre: "Sofía", departamento: "Marketing" }
];
```

Agrupa los empleados por departamento en un objeto

Práctica a4_3. Arreglos en Javascript

Resolución

```
let cubosPares = numeros
  .filter(n => n % 2 === 0)
  .map(n => n * n * n);
```

```
let agrupados = empleados.reduce((acc, emp) => {
  // si no existe aún la clave del departamento, la creo como
arreglo vacío
  if (!acc[emp.departamento]) {
    acc[emp.departamento] = [];
  }
  // agrego el empleado al depto correspondiente
  acc[emp.departamento].push(emp);
  return acc;
}, {});
```

Ver el_mas_repetido.html

Práctica a4_4. Arreglos en Javascript

Buscar el número más repetido de un arreglo.

Ver a4_41.html

Práctica a4_41. Arreglos en Javascript

```
const estudiantes = [
  { nombre: "Ana", nota: 8 },
  { nombre: "Luis", nota: 5 },
  { nombre: "Carla", nota: 9 },
  { nombre: "Juan", nota: 6 },
  { nombre: "Pedro", nota: 4 }
];
```

- Obtener un nuevo arreglo solo con los nombres de los estudiantes que aprobaron (nota ≥ 6).
- Ordenar el arreglo original por nota de mayor a menor.
- Calcular el promedio general de las notas.

Práctica a4_42. Matrices en Javascript

Dada una matriz
Entrada:
[[1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]]

Obtener

- La **suma de cada fila**.
- La **suma de cada columna**.

- Salida:
- Suma de filas: [6, 15, 24]
- Suma de columnas: [12, 15, 18]

Práctica a4_43. Matrices en Javascript

Dada una **matriz cuadrada** (n x n), escribir una función que obtenga:

La **diagonal principal**.
La **diagonal secundaria**.
La **suma de ambas diagonales**.

Práctica a4_44. Matrices en Javascript

Aplanar una matriz tridimensional (pasarla a un arreglo), y ordenarlo.

Ejemplo Javascript. Objetos

En JavaScript, un objeto es una colección de pares **clave-valor**.

Se usa para modelar entidades del mundo real (persona, coche, empresa, etc.).

Ejemplo Javascript. Objetos

```
let persona = {
    nombre: "Carlos",
    edad: 30,
    saludar: function() {
        console.log("Hola, soy " + this.nombre);
    }
};
```

Ejemplo Javascript. Objetos

```
const calculadora = {
  sumar: function(a, b) {
    return a + b;
  },
  restar(a, b) {
    return a - b;
  }
};

console.log(calculadora.sumar(5, 3)); // 8
console.log(calculadora.restar(10, 4)); // 6
```

Ejemplo Javascript. Objetos

Objetos Anidados

```
const empresa = {
  nombre: "TechCorp",
  direccion: {
    calle: “San Martín",
    numero: 123
  }
};

console.log(empresa.direccion.calle);
```

Ejemplo Javascript. Objetos

Iterar sobre objetos

```
let persona = { nombre: "Ana", edad: 28, ciudad: "Madrid" };

for (let clave in persona) {
  console.log(`${clave}: ${persona[clave]}`);
}

// Obtener claves y valores
console.log(Object.keys(persona));   // ["nombre", "edad", "ciudad"]
console.log(Object.values(persona)); // ["Ana", 28, "Madrid"]
console.log(Object.entries(persona));// [["nombre","Ana"],["edad",28],["ciudad","Madrid"]]
```

Ejemplo Javascript. Objetos

Práctica a4_441

- Crea un objeto Libro con propiedades titulo, autor, anio. Agrega un método descripcion() que muestre la información.
- Modela un objeto cuentaBancaria con propiedades saldo, titular. Métodos: depositar(monto) y retirar(monto).
- Crea un objeto Rectangulo con ancho y alto. Métodos: area() y perimetro().

Ejemplo Javascript. Funciones y Eventos

```
function saludar() {
  console.log('Hola, mundo!');
}

saludar();        //imprime “Hola, mundo!”
```

Ejemplo Javascript. Funciones y Eventos

También se puede definir funciones asignando una función anónima a una variable. Estas son conocidas como **funciones expresadas**.

```
const despedirse = function() {
  console.log('Adiós, mundo!');
};

despedirse();
```

Ejemplo Javascript. Funciones y Eventos

Parámetros por defecto

```
function multiplicar(a, b = 1) {
  return a * b;
}
console.log(multiplicar(5));  // Imprime 5
```

Ejemplo Javascript. Funciones y Eventos

Funciones de orden superior. JavaScript permite el uso de funciones como valores, lo que significa que pueden ser pasadas como argumentos a otras funciones, retornadas como valores desde otras funciones, y asignadas a variables.

```
function aplicarOperacion(a, b, operacion) {
  return operacion(a, b);
}
console.log(aplicarOperacion(4, 2, sumar));  // Imprime 6
```

Ver a4_442

Ejemplo Javascript. Funciones y Eventos

Práctica a4_442. Funciones de orden superior.

Dado un arreglo de objetos con {nombre, clave}, aplicar funciones de primer orden para capitalizar el nombre, y encriptar la clave.

Funciones y Eventos

En el desarrollo web moderno, interactuar con los usuarios a través de **eventos** es fundamental. JavaScript proporciona un poderoso modelo de eventos que permite a los desarrolladores responder a acciones del usuario como clics, movimientos del mouse, pulsaciones de teclas, entre otros.

Acompañando a este modelo, las funciones anónimas ofrecen una manera concisa y conveniente de manejar estos eventos sin necesidad de nombrar explícitamente las funciones

Funciones y Eventos

Fundamentos del Manejo de Eventos

Los eventos son acciones o sucesos que ocurren en el navegador que la página web puede responder, como un usuario clickeando un botón, llenando un formulario, o moviendo el mouse. Cada uno de estos eventos puede ser detectado y manejado en JavaScript a través de **event listeners** (escuchas de eventos).

Ejemplo Javascript. Funciones y Eventos

Funciones Anónimas

Una función anónima es una función sin un nombre identificador. Son útiles especialmente en contextos donde una función se utiliza temporalmente y no necesita ser referenciada en otro lugar. Son frecuentemente usadas en JavaScript para manejar eventos o realizar operaciones de callback.

```
boton.addEventListener('click', function() {
  console.log('Botón clickeado!');
});
```

Ver botones.html!!!

Ejemplo Javascript. Funciones y Eventos

Agregar un Event Listener:
Utilizando el método addEventListener, se puede registrar una función para ser llamada cada vez que un evento específico ocurre en un elemento.

```
const boton = document.getElementById('miBoton');
boton.addEventListener('click', function() {
  console.log('Botón clickeado!');
});
```

Ejemplo Javascript. Manejo del DOM

```
<p id="texto">Texto original</p>
<button onclick="cambiarTexto()">Cambiar texto</button>


<script>
function cambiarTexto() {
    document.getElementById("texto").innerText = "Texto cambiado";
}
</script>
```

Ejemplo Javascript. Manejo del DOM

Creación de elementos

```
<div id="lista"></div>


<script>
const lista = document.getElementById("lista");
// Crear un fragmento (mejor que ir agregando nodo por nodo al DOM directamente)
const fragment = document.createDocumentFragment();
for (let i = 1; i <= 1000; i++) {
  const item = document.createElement("p");
  item.textContent = `Elemento número ${i}`;
  fragment.appendChild(item);
}
// Insertamos todo de una vez (muy eficiente)
lista.appendChild(fragment);
</script>
```

Ejemplo Javascript. Manejo del DOM

Delegación de eventos + modificación de nodos

```
<ul id="tareas">
  <li>Estudiar JS <button class="eliminar">borrar</button></li>
  <li>Hacer ejercicio <button class="eliminar">borrar</button></li>
</ul>

<script>
const tareas = document.getElementById("tareas");

// Delegación: un solo listener para todos los botones
tareas.addEventListener("click", e => {
  if (e.target.classList.contains("eliminar")) {
    e.target.parentElement.remove(); // elimina el <li> completo
  }
});
</script>
```

Ejemplo Javascript. Manejo del DOM

Modificación masiva con classList y atributos

```
<div id="contenedor">
  <p data-role="admin">Usuario: Juan</p>
  <p data-role="user">Usuario: María</p>
</div>
<script>
const usuarios = document.querySelectorAll("#contenedor p");
usuarios.forEach(u => {// Cambiar atributos y clases dinámicamente
  if (u.dataset.role === "admin") {
    u.classList.add("destacado");
    u.setAttribute("title", "Administrador del sistema");
  } else {
    u.classList.add("normal");
    u.removeAttribute("title");
  }
});
</script>
```

Ver observer.html

Ejemplo Javascript. Manejo del DOM

Observer para cambios en el DOM (MutationObserver)

Ejemplo Javascript. Manejo del DOM

Edición en vivo estilo "Google Docs" (contentEditable)

```
<div id="editor" contenteditable="true" style="border:1px solid black; padding:10px;">
  Escribe aquí...
</div>

<script>
const editor = document.getElementById("editor");

editor.addEventListener("input", () => {
  console.log("Contenido actual:", editor.innerHTML);
});
</script>
```

Ver todolist.html

Ejemplo Javascript. Manejo del DOM

Práctica a4_443

- Crear una página que lleve cuenta de una Todo List como la de la figura.
- Al presionar el botón agregar se renderiza el nuevo ítem Todo y se actualiza el registro de cambios llevado por un observer.

Ver formavanzado.html

Ejemplo Javascript. Eventos y Formularios

```
<form id="formulario”>
   <input type="text" name="nombre" placeholder="Tu nombre">
   <button type="submit">Enviar</button>
</form>

<script>
document.getElementById("formulario").addEventListener("submit", function(e) {
   e.preventDefault();
   alert("Formulario procesado");
});
</script>
```

Ver a4_444.html

Ejemplo Javascript. Manejo de Formularios

Práctica a4_444

Crear una página con un formulario con las siguientes características:

- Un formulario de login con usuario y contraseña.
- Validar que ambos campos estén completos
- Si los datos no coinciden con un usuario válido, mostrar un error en color rojo y aumentar un contador de intentos.
- Bloquear el login tras 3 intentos fallidos.

Ejemplo Javascript. Programación Asincrónica

```
console.log("Iniciando...");
setTimeout(() => {
    console.log("Han pasado 2 segundos");
}, 2000);
```

Ejemplo Javascript. Programación Asincrónica

Callbacks

```
function descargarDatos(callback) {
  console.log("Descargando...");
  setTimeout(() => {
    callback("Datos recibidos");
  }, 1500);
}

descargarDatos((resultado) => {
  console.log(resultado);
});
```

Ejemplo Javascript. Manejo de Json

```
let persona = { nombre: "Ana", edad: 25 };
let json = JSON.stringify(persona);
console.log(json);

let obj = JSON.parse(json);
console.log(obj.nombre);
```

Ejemplo Javascript. Manejo de Json

Excluir una propiedad

```
const persona = {
  id: 1,
  nombre: "Ana",
  edad: 30,
  password: "secreta123"
};
// Excluir la propiedad "password"
const jsonSeguro = JSON.stringify(persona, (key, value) => {
  if (key === "password") return undefined;
  return value;
});
console.log(jsonSeguro);  //  {"id":1,"nombre":"Ana","edad":30}
```

Ejemplo Javascript. Manejo de Json

Transformar valores antes de serializarlos

```
const jsonEdad = JSON.stringify(persona, (key, value) => {
  if (key === "edad") return value + " años";
  return value;
});
console.log(jsonEdad);
//  {"id":1,"nombre":"Ana","edad":"30 años","password":"secreta123"}
```

Ejemplo Javascript. Manejo de Json

Transformación de datos

```
const json = '{"id":1,"nombre":"Luis","nacimiento":"1995-06-15"}';

const objeto = JSON.parse(json, (key, value) => {
  if (key === "nacimiento") return new Date();
  return value;
});

console.log(objeto.nacimiento);
```

Ejemplo Javascript. Manejo de Json

stringify con espaciado

```
const jsonBonito = JSON.stringify(persona, null, 2);
console.log(jsonBonito);
/*
{
  "id": 1,
  "nombre": "Ana",
  "edad": 30,
  "password": "secreta123"
}
*/
```

Ejemplo Javascript. Manejo de Json

Clonación de objetos

```
const original = { a: 1, b: { c: 2 } };
const copia = JSON.parse(JSON.stringify(original));

//const copia = original;

copia.b.c = 99;
console.log(original.b.c);
```

Ejemplo Javascript. Manejo de Json

Validación

```
function parseJSONSeguro(str) {
  try {
    return JSON.parse(str);
  } catch (error) {
    console.error("JSON inválido:", error.message);
    return null;
  }
}

console.log(parseJSONSeguro('{"a":1}')); // OK
console.log(parseJSONSeguro("{a:1}"));   // error controlado
```

Ver tareas_json.html!!

Ejemplo Javascript. Manejo de Json

Combinar Json con funciones de arrays

```
const empleadosJSON = `
[
  {"id":1,"nombre":"Ana","salario":3000},
  {"id":2,"nombre":"Luis","salario":2500},
  {"id":3,"nombre":"Marta","salario":4000}
]`;

const empleados = JSON.parse(empleadosJSON);

// Aumentar salarios en 10% y volver a JSON
const empleadosActualizados = empleados.map(e => ({
  ...e,
  salario: e.salario * 1.1
}));

console.log(JSON.stringify(empleadosActualizados, null, 2));
```

Ejemplo Javascript. Clases y objetos

Definición de una Clase:

```
class Persona {
  constructor(nombre, edad) {
    this.nombre = nombre;
    this.edad = edad;
  }
  presentarse() {
    console.log(`Soy ${this.nombre} y tengo ${this.edad} años.`);
  }
}
```

Ejemplo Javascript. Clases y objetos

Definición de una Clase:

```
let alice = new Persona('Alice', 30);
alice.presentarse();
```

Ver herencia.html!!!

Ejemplo Javascript. Clases y objetos

Herencia

```
class Empleado extends Persona {
  constructor(nombre, edad, puesto) {
    super(nombre, edad);  // Llama al constructor de la clase Persona
    this.puesto = puesto;
  }

  describirTrabajo() {
    console.log(`Trabajo como ${this.puesto}.`);
  }
}
```

Ejemplo Javascript. Clases y objetos

Métodos Estáticos

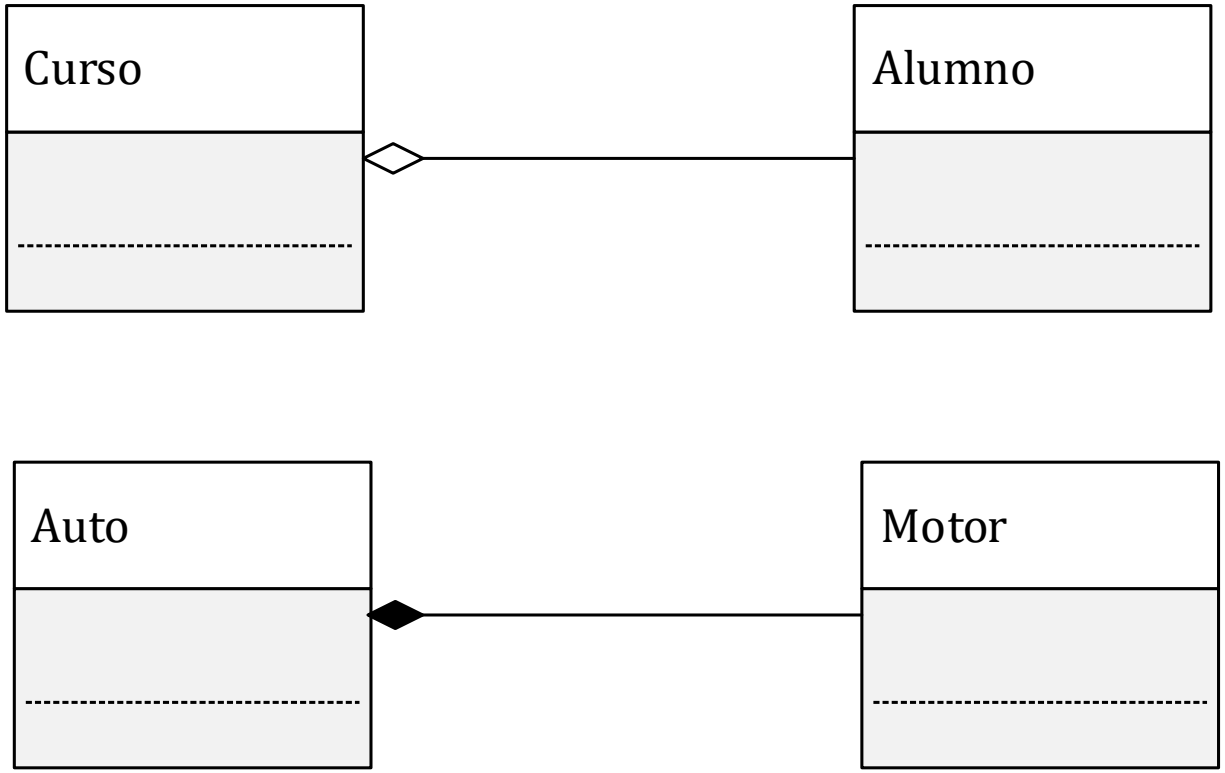
```
class Utilidades {
  static numeroAleatorio() {
    return Math.floor(Math.random() * 100);
  }
}

console.log(Utilidades.numeroAleatorio());  // Devuelve un número
aleatorio
```

Ver agregación.html y composición.html!!!

Ejemplo Javascript. Clases y objetos

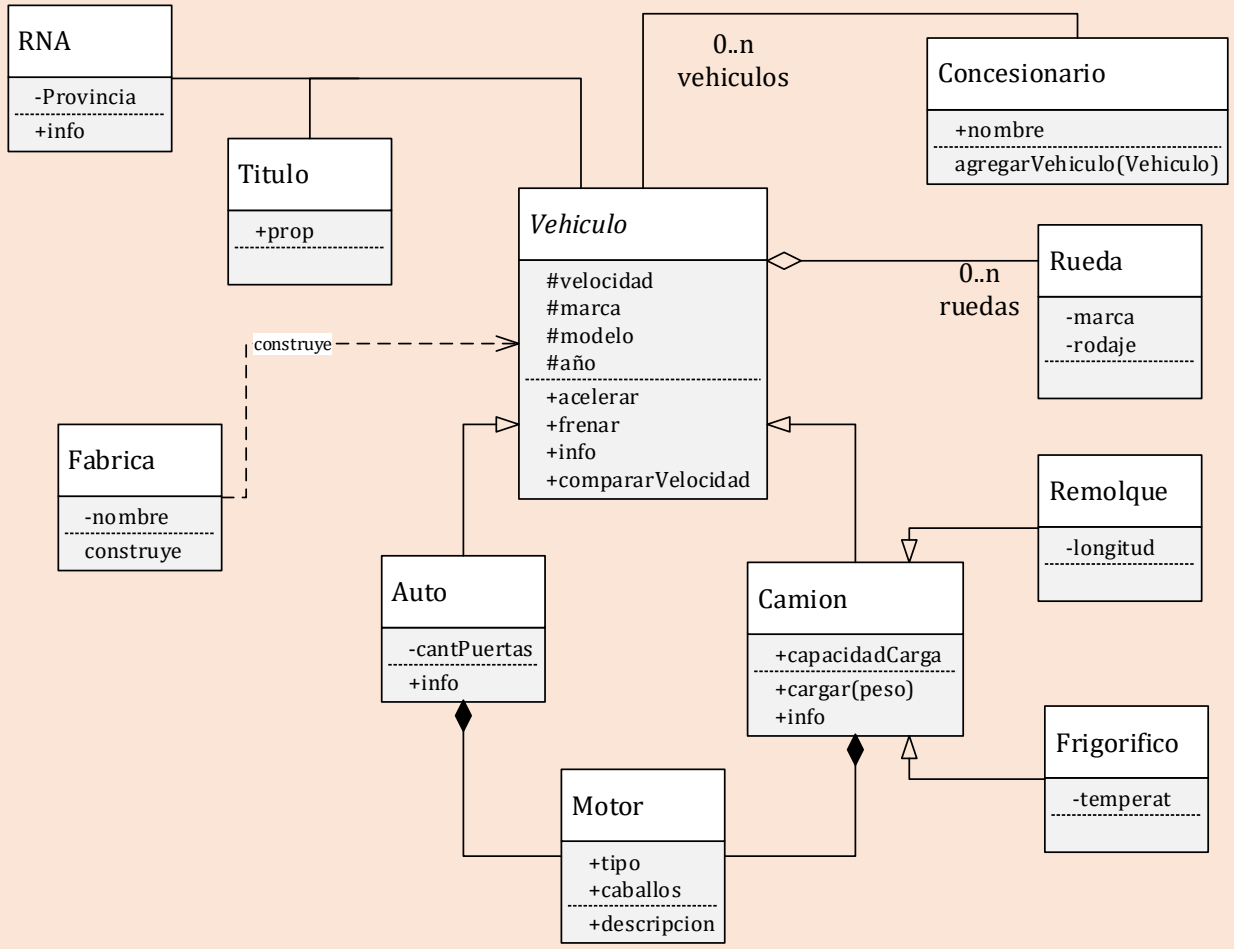
Agregación y Composición



Ver modelo_uml1.html !!!

Práctica a4_5. Clases en Javascript

Codificar en Javascript/html el siguiente modelo de clases.



Ver contador.html !!!

Práctica a5. Repaso javascript

Realizar la siguiente página WEB, donde los botones incrementar y decrementar hacen lo propio con un contador, el cual es renderizado tras cada cambio.

Futuro

- Evolución constante de frameworks/librerías: React, Vue y Angular siguen siendo populares, pero aparecen opciones nuevas como **Svelte**, **SolidJS**, **Qwik**.
- Tendencia a librerías **más ligeras, rápidas y reactivas**.
- Node.js sigue fuerte, pero surgen nuevas alternativas como Deno (del creador de Node).
- Serverless y edge computing (Vercel, Cloudflare Workers) impulsan JavaScript para procesamiento en el borde.

•**Mejor soporte para inteligencia artificial.** Integración con APIs de IA, procesamiento de imágenes, chatbots, etc. Frameworks como TensorFlow.js permiten correr modelos directamente en el navegador.