



Desarrollo web con HTML, CSS y JavaScript

Breve descripción:

El diseño de un sitio web corresponde a un componente fundamental en el proceso de creación de aplicaciones web; por ende, este componente está orientado a la comprensión de los conceptos más importantes que se deben tener en cuenta, así como también la forma en que se deben codificar para construir de manera propia un sitio web o de apoyarse en sistemas gestores de contenidos denominados CMS.

Junio 2024

Tabla de contenido

Introducción	4
1. HTML 5	7
1.1. Etiquetas (tags).....	8
1.2. Textos	9
1.3. Imágenes	11
1.4. Hiperenlaces.....	12
1.5. Tablas.....	13
1.6. Formularios	16
2. Hojas de estilo	19
2.1. Introducción CSS3	23
2.2. Estructura	25
2.3. Formulario CSS3	26
3. JavaScript.....	31
3.1. Versiones	35
3.2. Sintaxis	36
3.3. Tipos de datos	38
3.4. Estructuras de control	40
El condicional if.....	40

Condicional if/else	41
Operador ternario	41
Condicional if múltiple.....	42
Condicional switch.....	43
Bucles e iteraciones:.....	44
Bucle while	45
4. Gestores de contenido CMS	48
Funciones	48
Características	50
Clasificación CMS.....	51
Tipos CMS.....	54
Síntesis	55
Material complementario	56
Glosario	57
Referencias bibliográficas	58
Créditos	59

Introducción

El diseño de interfaces web constituye una faceta crítica del desarrollo tecnológico moderno, actuando como el puente esencial que facilita la interacción entre los usuarios y las aplicaciones basadas en la web. A través de estas interfaces, las aplicaciones ofrecen sus servicios y funcionalidades, operando dentro de un entorno dinámico y accesible mediante navegadores web. Estas plataformas pueden ser accesibles a través de diversas redes, incluyendo LAN (Red de Área Local), WAN (Red de Área Amplia) o la omnipresente Internet, lo que les permite llegar a un público extenso y diverso.

En la era digital de hoy, la demanda por interfaces web intuitivas, eficientes y estéticamente agradables está en constante aumento. Las organizaciones y desarrolladores buscan innovar constantemente en este campo para mejorar la experiencia del usuario, reconociendo que una interacción positiva con la interfaz puede ser decisiva para el éxito de una aplicación web. Esta evolución no solo se centra en la estética visual sino también en la funcionalidad, con un enfoque particular en la adaptabilidad y la accesibilidad.

Las interfaces modernas están diseñadas para ser fluidas y eficaces en una amplia gama de dispositivos, desde computadoras de escritorio y portátiles hasta tabletas y smartphones, garantizando así una experiencia de usuario coherente y satisfactoria independientemente del medio utilizado.

El diseño de interfaces web abarca múltiples dimensiones clave para el éxito en el entorno digital actual:

- **Enfoque en usabilidad y estética**

El diseño de interfaces web se centra no solo en la funcionalidad y la apariencia visual, sino también en captar y mantener el interés de un número mayor de usuarios.

- **Importancia de la lealtad del usuario**

En un mercado digital saturado, conseguir la lealtad de los usuarios es un desafío significativo, haciendo que la capacidad de atraer y retener usuarios sea crucial.

- **Aplicación de principios fundamentales**

Comprender y aplicar los principios básicos del diseño de interfaces, junto con habilidades prácticas de programación, se ha convertido en una habilidad esencial.

- **Educación avanzada en diseño**

La inclusión de conceptos avanzados y ejemplos de código fuente en la educación de diseño prepara a los diseñadores y desarrolladores para enfrentar retos, construyendo interfaces que satisfacen y anticipan las necesidades de los usuarios.

- **Dinamismo del campo**

El diseño de interfaces web es un área de constante evolución que demanda un entendimiento profundo de las necesidades de los usuarios y la habilidad para implementar soluciones técnicas innovadoras.

- **Crecimiento impulsado por la tecnología**

La relevancia del diseño de interfaces web sigue en aumento, estimulada por el progreso tecnológico y el crecimiento de la economía digital, destacando la necesidad de enfoques de diseño que sean funcionales y visionarios.

1. HTML 5

HTML5 es la piedra angular de la creación de contenido web moderno, permitiendo a los desarrolladores y diseñadores web construir experiencias ricas e interactivas en línea. A continuación, se detalla su definición y aplicación:

Video 1. HTML 5



[Enlace de reproducción del video](#)

Síntesis del video: HTML 5

HTML: Hypertext Markup Language, es un lenguaje demarcado a la maquetación web, en esta definición es importante resaltar el término “hipertexto” como la posibilidad de tomar un texto normal que generalmente es plano y enriquecerlo con otros componentes como lo son: imágenes, videos, sonidos,

colores, formas, tipos y tamaños de letras, relaciones entre párrafos que se vinculan mediante enlaces, que también se denominan links.

Para lograr esta conjunción de elementos es necesario definir mediante un código de marcado estándar todas sus características y se escriben en un archivo con extensión .html por medio de etiquetas, tags y así visualizar la implementación de cada etiqueta siguiendo unas reglas de sintaxis con atributos en un navegador web, a este archivo creado comúnmente se le llama página web.

El conjunto de características y funcionalidades de cada etiqueta, que ha evolucionado con el tiempo, se conoce como HTML. Actualmente, está en la versión 5, la cual es regulada por el Consorcio World Wide Web (W3C).

Sabías qué?

El consorcio WWW en inglés: World Wide Web Consortium (W3C) es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento del World Web a largo plazo.

1.1. Etiquetas (tags)

Las etiquetas HTML representan el conjunto de funcionalidades que se pueden implementar en una página web, y poseen una sintaxis específica de escritura. Esto significa que existe una forma correcta en la que se deben escribir, cumpliendo con determinados atributos y manteniendo una interdependencia entre ellas. Por ello, algunas etiquetas dependen de otras para funcionar correctamente, a estas se les conoce como etiquetas padres. A continuación, se presenta un ejemplo de las etiquetas padres mínimas requeridas en un archivo .html:

- **< HTML >**
Indica el inicio del documento y que el documento está escrito en el lenguaje HTML.
- **< HEAD >**
Es el encabezado de la página y esta información va orientada al navegador.
- **< /HEAD >**
Indica el fin de la etiqueta < HEAD >. Ambas etiquetas deben quedar dentro de la etiqueta padre HTML.
- **< BODY >**
Representa el cuerpo del contenido de la información que va a visualizar el usuario.
- **< /BODY >**
Indica el fin del cuerpo.
- **< /HTML >**
Indica el final del documento HTML.

1.2. Textos

La inclusión de texto en una página web responde a la necesidad de presentar información a los visitantes. Todo texto que se añada debe estar contenido dentro de las etiquetas padres previamente mencionadas. En este contexto, se recomienda organizar el texto en párrafos, lo cual se logra mediante la utilización de la etiqueta <p>, seguido del texto deseado, y cerrando con la etiqueta de cierre </p>. De acuerdo con lo expuesto, se entiende que un conjunto de párrafos forma todo el documento, y los

saltos de línea dentro de un texto se indican con la etiqueta `
`. Además, otros elementos importantes para enriquecer el texto incluyen:

Tabla 1. Etiquetas HTML para textos

Etiqueta	Descripción
<code><blockquote/></code>	Párrafo que en sí mismo corresponde a una cita de reseña.
<code><q/></code>	Cita contenida dentro de un texto.
<code></code>	Enfatiza el texto de un párrafo.
<code></code>	Resaltado de texto.

Un ejemplo de configuración de código HTML es:

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
Mi primer texto en una página web que no es un párrafo </br>
```

```
<p>Mi primer párrafo página web</p>
```

```
<blockquote>MI PRIMER PARRAFO CITA EN UNA PAGINAWEB<blockquote>
```

```
<p>Siempre se dice que este es el mejor sitio para ver videos | online
```

```
<q cite="http://youtube.com/">www.youtube.com</q>
```

```
</p>
```

En ocasiones más vale correr lentopero seguro.

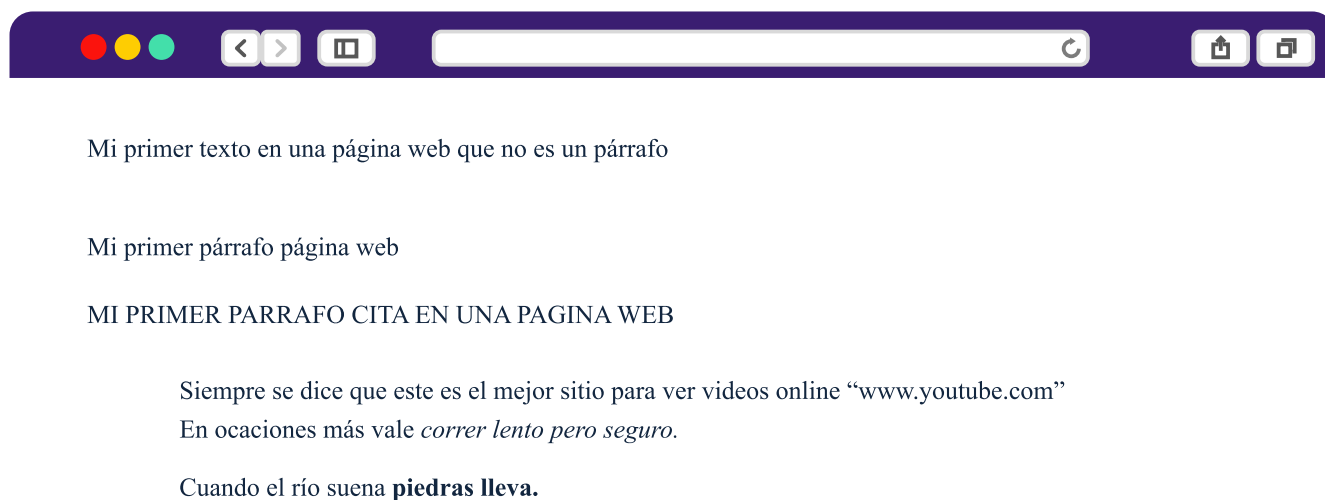
<p>Cuando el rio suena pedras lleva.</p>

</body>

</html>

El resultado de la ejecución del código anterior es:

Figura 1. Resultado del código



1.3. Imágenes

Para anexar una imagen a un documento HTML se usa la etiqueta acompañada del atributo src que corresponde a la ubicación en donde se encuentra la imagen, bien sea en formato *.jpeg, *.gif, *.png, entre otros formatos.

¡Importante!

Otros atributos importantes para la etiqueta son width y height que sirven para definir el ancho y el alto de la imagen dentro de la página web.

Un ejemplo de configuración de código HTML es:

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```

```

```
</body>
```

```
</html>
```

1.4. Hiperenlaces

Esta funcionalidad permite la navegación entre diferentes páginas web, es decir, facilita el desplazamiento de una página a otra cuyos contenidos HTML están almacenados en archivos de distintos nombres y, posiblemente, en carpetas diferentes. Además, habilita la navegación hacia otras páginas web ya publicadas en internet.

¡Ten en cuenta!

Para lograr un hiperenlace se debe utilizar la etiqueta <a> acompañada del atributo href="" que indica el destino hacia la página a la que se quiere llegar y dentro de la etiqueta el texto que describe el sitio destino.

Un ejemplo de código para hiperenlaces es:

```
<html>  
  
<head>  
  
</head>  
  
<body>  
  
<a href="http://www.google.com"> http://www.google.com </a>  
  
</body>  
  
</html>
```

1.5. Tablas

Las tablas son elementos idóneos para estructurar contenido en una página web de manera matricial, es decir, organizado en filas y columnas. Para crear una tabla, se utiliza la etiqueta `<table>`. A continuación, se define la cantidad de filas mediante la etiqueta `<tr>` y, dentro de ella, se especifica el número de columnas deseado utilizando la etiqueta `<td>` para cada celda que se quiera presentar. Al final, cada combinación de fila y columna forma una celda de información.

El siguiente código ilustra la creación de una tabla con 2 filas y 3 columnas, donde el contenido de cada celda puede variar, incluyendo texto, hiperenlaces, imágenes, entre otros elementos.

El código para una tabla de datos es:

```
<html>  
  
<head>
```

```
</head>

<body>

<table>

<tr>

<td>Celda Uno </td>

<td>Celda Dos <a href="http://www.google.com">www.google.com</a></td>

<td>Celda Tres </td>

</tr>

<tr>

<td>Celda Cuatro</td>

<td>Celda Cinco</td>

<td>Celda Seis</td>

</tr>

</table>

</body>

</body>

</html>
```

El resultado ejecución del código de la tabla es:

Figura 2. Tabla de datos



Calda Uno	Calda Dos	www.google.com	Calda tres
Calda Cuatro	Calda Cinco		Calda Seis

¡Importante: Divs!

Si bien es cierto que ya vimos el funcionamiento de las tablas, en el diseño de hoy se recomienda usar la etiqueta div que corresponde a un contenedor y permite realizar lo mismo de las tablas, pero con mayores opciones en funcionalidad y dinamismo.

Un ejemplo de tabla usando Div:

```
<div class="tabla">

<div> Cabecera con el logo de la página "header" </div>

<div>

<div class="columna1"> Columna1 </div>

<div class="columna2"> Columna2 </div>

<div class="columna3"> Columna3 </div>

</div>
```

<div> Pie de página </div>

</div>

1.6. Formularios

Los formularios son componentes HTML diseñados para facilitar la interacción y el intercambio de información con usuarios. Se caracterizan por agrupar elementos interactivos que permiten introducir datos de diversos tipos, como numéricos, alfanuméricos, caracteres, listas de datos, entre otros. La creación de formularios puede realizarse de manera tanto dinámica como estática.

La etiqueta <form> se utiliza para definir un formulario y debe incluir, como mínimo, el atributo action, que especifica el destino al que se enviarán los datos. Además, es esencial definir el método de envío de los datos, que puede ser post o get, para determinar cómo se transmitirán estos datos al servidor.

A continuación, se describen algunos tags que comúnmente conforman un formulario, estos se deben encerrar dentro de la etiqueta principal del formulario <form>:

- **<input>**

Entrada de datos: estos pueden ser texto, número o alfanumérico; sus atributos más importantes son type para definir el tipo de entrada que puede ser text, radio, submit, reset, el atributo id permite identificar de manera única el objeto en el archivo HTML, el atributo name es para darle un nombre a la entrada o variable y value para recibir el dato del componente.

- **<label>**

Permite la definición de un nombre para una entrada de datos.

Un ejemplo de código de un formulario es:

```
<html>

<head>

</head>

<body>

<form action="http://misitio.com/prog/usuarioNuevo" method="post">

<p>

<label for="nombre">Nombre:</label>

<input type="text" id="nombre"><br/>

<label for="apellido">Apellido:</label>

<input type="text" id="apellido"><br/>

<label for="email">email:</label>

<input type="text" id="email"><br/>

<input type="radio" name="sexo" value="Varón">Varón<br/>

<input type="radio" name="sexo" value="Mujer">Mujer<br/>

<input type="submit" value="Enviar"><input type="reset">

</p>
```

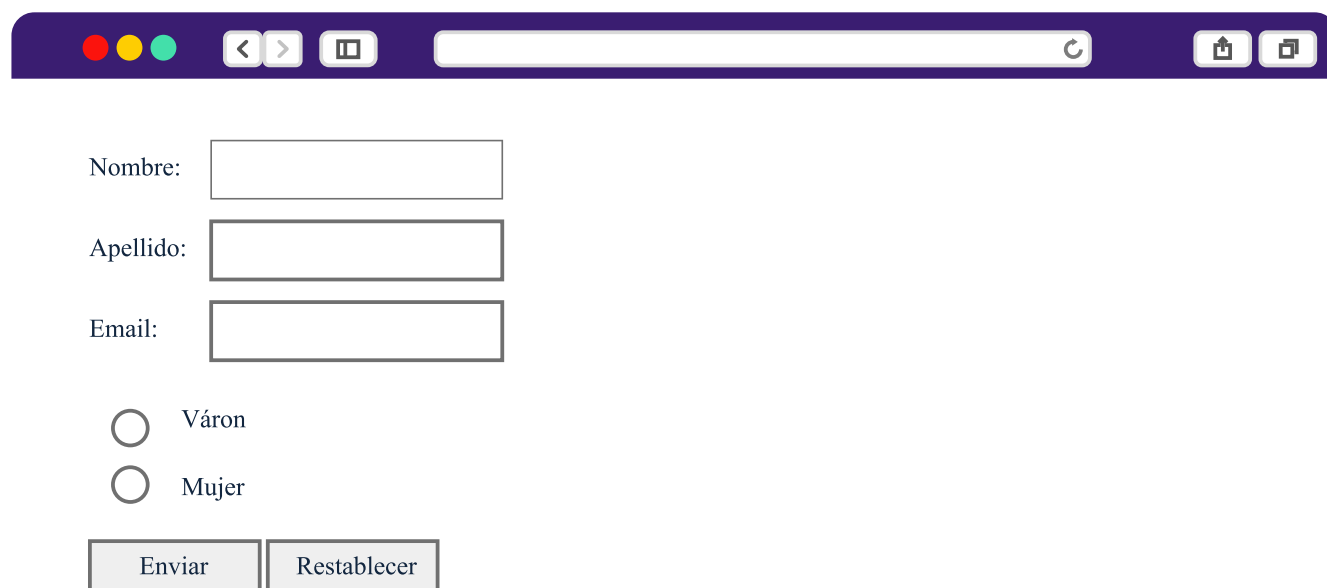
</FORM>

</body>

</html>

El resultado del código del formulario es:

Figura 3. Resultado código del formulario



A screenshot of a web browser window with a dark purple header. The browser's address bar is empty. The page content includes a form with the following elements:

- Three colored window control buttons (red, yellow, green) on the left.
- Navigation buttons (back, forward, home) and a search icon on the right.
- Form fields:
 - Nombre: [text input]
 - Apellido: [text input]
 - Email: [text input]
- Gender selection:
 - ☐ Váron
 - ☐ Mujer
- Buttons at the bottom:
 - Enviar
 - Restablecer

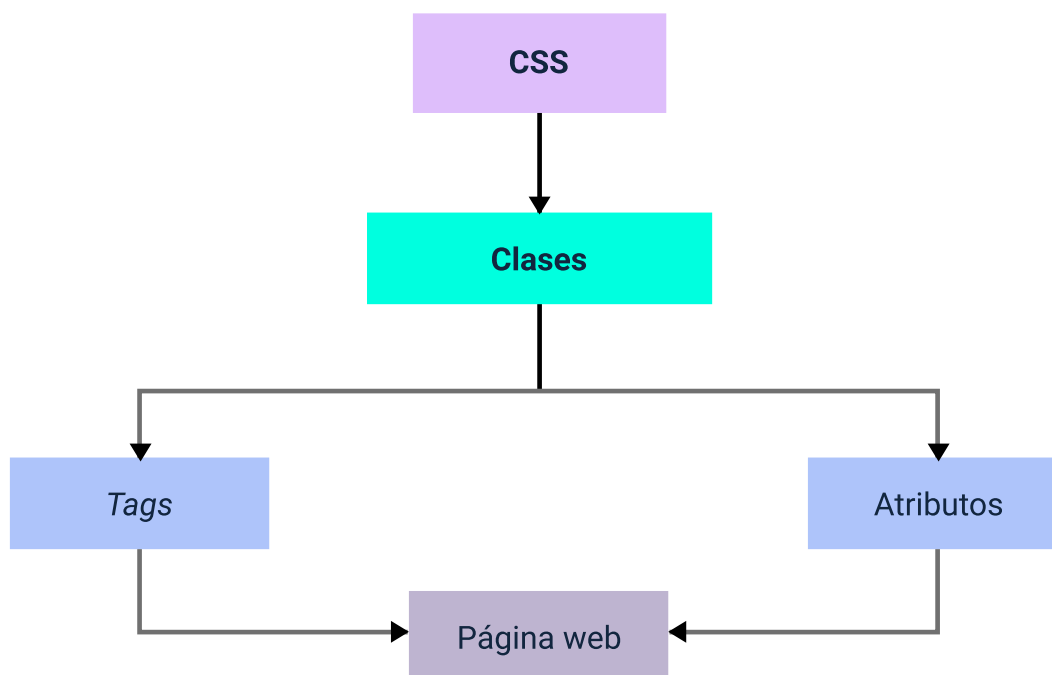
2. Hojas de estilo

Las hojas de estilo, conocidas como CSS (Cascading Style Sheets), son el mecanismo que confiere apariencia a una página web. Para ello, se crea un archivo con extensión .css, el cual contiene el código de estilización. Este código se compone de selectores (que pueden ser etiquetas HTML, clases, identificadores, entre otros) y un conjunto de propiedades y valores que definen el estilo de los elementos seleccionados.

La principal ventaja de utilizar CSS es la capacidad de reutilizar el mismo código en múltiples páginas web, lo que facilita la implementación de un diseño coherente a lo largo de todo el sitio. Además, el uso de hojas de estilo mejora significativamente el mantenimiento y control del diseño de un sitio web, permitiendo cambios globales en la presentación con modificaciones en un solo archivo.

A continuación, se muestra un esquema general de cómo se puede implementar código CSS en una página web:

Figura 4. Esquema general CSS



Al momento de implementar las CSS es necesario tener en cuenta los siguientes puntos:

- **Separación de estructura del documento:**

Se refiere a la separación del contenido del documento HTML independiente de su presentación (tipos de letras, colores, tamaños de letras, espacios, fondos de pantalla, anexo de imágenes, entre otros).

- **Optimización de carga:**

Utilizar CSS eficientemente permite reducir el tiempo de carga de las páginas web al minimizar el uso de recursos innecesarios. Esto se logra mediante la combinación de archivos, minimización de código y uso de selectores y reglas CSS adecuadas para el diseño responsive.

Un ejemplo de código de hoja interna es:

```
<head>

<style>

body{ background-color: #FFFFFF;

}

h1{

color: #000000;

text-align: center;

}

</style>

</head>

<body>

</body>
```

Un ejemplo de código de hoja externa es:

```
<head>

<link rel="stylesheet" type="text/css" href="estilo.css">

</head>

<body>

</body>
```

Un ejemplo de código de hoja en sitio web es:

```
<head>
```

```
<link rel="stylesheet" type="text/css" href="https://www.estilo.css">
```

```
</head>
```

```
<body>
```

```
</body>
```

¡Importante!

Los selectores de CSS, se utilizan para determinar los elementos a los que se va a aplicar el estilo y se clasifican por nombre, por id de selector y por clase.

- **De selector de nombre**

Donde se llama directamente al tag (h2) al que se debe aplicar el estilo.

Ejemplo de selector de nombre:

```
h2 {  
text-align: left; color: red;  
}
```

- **De selector por id**

Donde solo se aplica un elemento único. Aquí el nombre del estilo va precedido por el carácter "#". Ejemplo de selector por id:

```
#texto1 {  
text-align: center; color: blue;  
}
```

- **De selector de clase**

En este caso se pueden personalizar los estilos para aplicarlos a diferentes elementos que comparten alguna relación, para ello se debe escribir un punto (.) delante del nombre de la clase. Ejemplo de selector de clase:

```
.nombreMiclase {  
text-align: center; color: red;  
}
```

2.1. Introducción CSS3

CSS3 representa una evolución significativa respecto a sus predecesores, CSS y CSS2, introduciendo una serie de mejoras que pueden categorizarse principalmente en dos aspectos:

- **Aplicación del concepto de módulos:**

esta característica transforma la forma en que se definen los estilos de un sitio web, acercándose más a un enfoque de programación modular. Los módulos permiten organizar los estilos en contextos específicos y mantenerlos en archivos separados, lo que facilita su gestión y reutilización. Aunque la mención de archivos con extensión .less se asocia con un preprocesador CSS específico, el concepto modular en sí es fundamental para organizar CSS de manera más eficiente y no se limita a una extensión de archivo específica.

- **Enriquecimiento visual y funcional de los elementos:**

CSS3 amplía las posibilidades de diseño y animación de los elementos en las páginas web. Esto incluye la implementación de animaciones, transiciones, maquetaciones con columnas, gradientes para modificar

colores, transformaciones para rotar y escalar elementos, y el uso de fuentes de texto externas a través de librerías. Estas herramientas permiten a los diseñadores y desarrolladores web crear experiencias de usuario más ricas y dinámicas.

Estas innovaciones hacen de CSS3 una herramienta poderosa para el diseño web moderno, permitiendo la creación de sitios web más atractivos, funcionales y accesibles.

Un ejemplo de estructura de uso de módulos:

.styles/

..controlador.less

..módulos/

...configuraciones/

....menu.less

....comentarios.less

....right/

.....right.less

....left/

.....left.less

....center/

.....center.less


```
...ayudas/

....ayudas.less

/**** Contenido de archivo controlador.less *****/

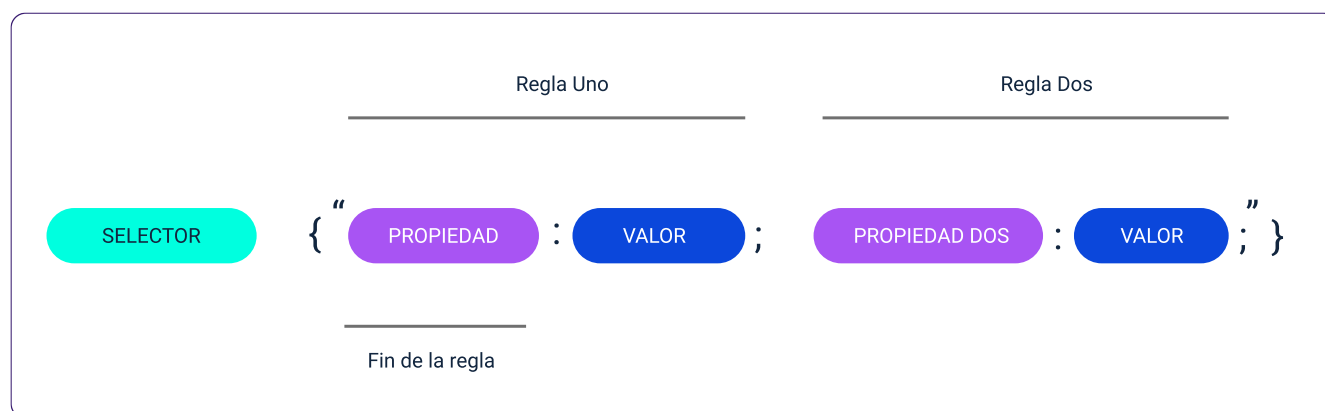
@import módulos/configuraciones/menu.less

@import módulos/ayudas/ayudas.less
```

2.2. Estructura

La estructura de CSS3 se basa en un conjunto de reglas almacenadas en un archivo. Cada regla consta de un selector, utilizado para identificar el elemento HTML al que se aplicarán los estilos, seguido de una o más declaraciones, que se separan entre sí por punto y coma (;). Una declaración incluye una propiedad y un valor asignado a esa propiedad. Además de definirse en un archivo CSS externo, estas declaraciones de estilo también pueden aplicarse directamente sobre un elemento HTML específico utilizando el atributo style. Para ello, se coloca la propiedad, seguida de dos puntos y su valor correspondiente, todo ello entre comillas dentro del atributo style en la etiqueta HTML deseada.

Figura 5. Ejemplo de estructura CSS3



Un ejemplo con uso del atributo style

```
<html>

<head>

<title>

</title>

</head>

<body>

<h1 style="color:red; size:30 px">Texto1</h1>

<h2 style="color:blue; size:40 px">Texto2</h2>

<h3 style="color:green; size:50 px">Texto3</h3>

</body>

</html>
```

2.3. Formulario CSS3

Los formularios en HTML se construyen siguiendo los principios básicos discutidos anteriormente. Sin embargo, con la introducción de HTML5, se han agregado nuevas funcionalidades que enriquecen los formularios web, incluyendo nuevos tipos de datos (como color, tel, email, date, url, time, month, number, week, datetime-local, range, y search), nuevos controles (output) y nuevos atributos (placeholder, autofocus, min, max, required, step, y pattern). Según la recomendación del W3C, es importante verificar la compatibilidad de estas nuevas características con diferentes navegadores. Aunque inicialmente algunas de estas innovaciones se soportaban principalmente en

navegadores específicos como Opera, la compatibilidad ha mejorado significativamente con el tiempo, y actualmente, muchos navegadores modernos soportan estas características.

Un ejemplo de formulario CSS3 es:

```
<html>
```

```
<head>
```

```
<meta charset="utf-8" /> <title>Nuevos componentes</title>
```

```
</head>
```

```
<body>
```

```
<form action="." oninput="range_control_value.value =  
range_control.valueAsNumber">
```

```
<p>
```

```
Nombre: <input type="text" name="name_control" autofocus required />
```

```
<br/>
```

```
Email: <input type="email" name="email_control" required />
```

```
<br/>
```

```
URL: <input type="url" name="url_control" placeholder="Escribe la URL de tu  
página web personal" />
```

```
<br/>
```

```
Fecha: <input type="date" name="date_control" />
```


Tiempo: <input type="time" name="time_control" />

Fecha y hora de nacimiento: <input type="datetime" name="datetime_control" />

Mes: <input type="month" name="month_control" />

Semana: <input type="week" name="week_control" />

Número (min -10, max 10): <input type="number" name="number_control" min="-10" max="10" value="0" />

Intervalo (min 0, max 10): <input type="range" name="range_control" min="0" max="10" value="0" />

<output for="range_control" name="range_control_value" >0</output>

Teléfono: <input type="tel" name="tel_control" />

Término de búsqueda: <input type="search" name="search_control" />

```
<br />
```

```
Color Favorito: <input type="color" name="color_control" />
```

```
<br/>
```

```
<input type="submit" value="Submit!" />
```

```
</p>
```

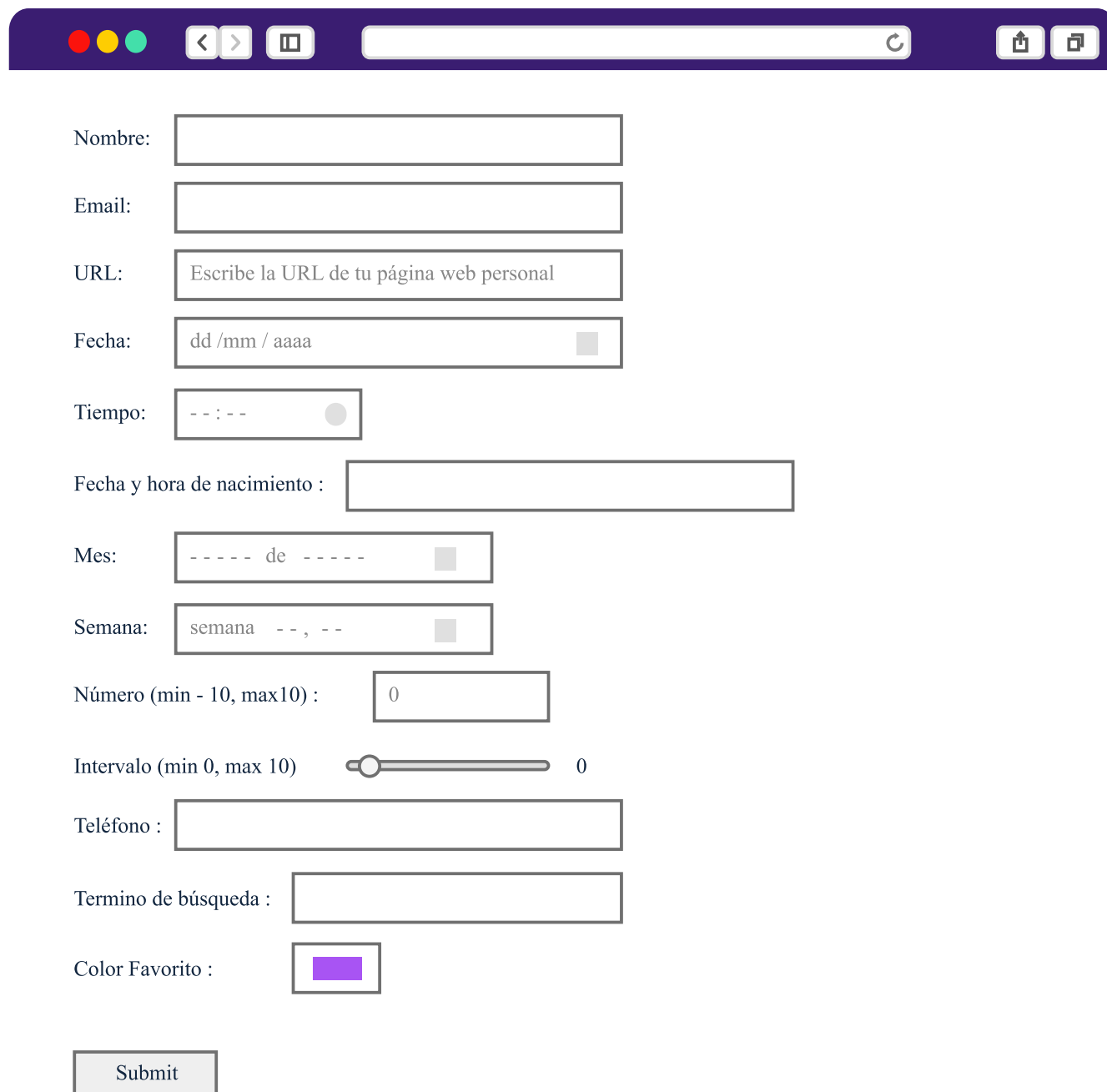
```
</form>
```

```
</body>
```

```
</html>
```

El resultado del código anterior de formulario CSS3 es:

Figura 6. Resultado código del formulario.



The image shows a web browser window with a dark purple header. The browser's address bar is empty. The form contains the following elements:

- Nombre:** A text input field.
- Email:** A text input field.
- URL:** A text input field with the placeholder text "Escribe la URL de tu página web personal".
- Fecha:** A date input field with the placeholder "dd /mm / aaaa" and a calendar icon.
- Tiempo:** A time input field with the placeholder "-- : --" and a clock icon.
- Fecha y hora de nacimiento :** A text input field.
- Mes:** A date input field with the placeholder "-- -- de -- --" and a calendar icon.
- Semana:** A date input field with the placeholder "semana -- , --" and a calendar icon.
- Número (min - 10, max10) :** A text input field with the value "0".
- Intervalo (min 0, max 10)** A range input field with a slider and the value "0".
- Teléfono :** A text input field.
- Termino de búsqueda :** A text input field.
- Color Favorito :** A color input field with a purple color swatch.
- Submit** A button.

3. JavaScript

JavaScript es un lenguaje de programación que se integra en páginas web y es interpretado por los navegadores. La característica de ser “interpretado” se refiere a que el código se ejecuta de manera secuencial, línea por línea, por el intérprete del navegador, primordialmente en el lado cliente. Sin embargo, con el desarrollo de tecnologías como Node.js, JavaScript también se puede ejecutar en el lado del servidor, a diferencia de otros lenguajes de programación que requieren compilación.

Este lenguaje está especialmente diseñado para permitir la creación de contenido HTML de forma dinámica. Esto significa que los programas escritos en JavaScript pueden generar y manipular etiquetas HTML automáticamente, enriqueciendo significativamente el diseño de las interfaces web.

Es crucial distinguir entre JavaScript y Java, ya que, a pesar de la similitud en sus nombres, son lenguajes de programación completamente diferentes. Esto puede llevar a confusiones entre quienes no están familiarizados con ellos. Para integrar código JavaScript en una página web, existen varias maneras, entre ellas:

- Con código que se agrega dentro de la misma página.
- Con archivos independientes que se almacenan en rutas diferentes a la página.
- Con código referenciado en otras páginas web ya publicadas en la red internet.

Explorando la integración de JavaScript en la web, esta guía visual proporciona ejemplos claros y concisos de cómo insertar código JavaScript directamente en el

HTML, así como cómo incluirlo desde archivos externos, y los resultados esperados de dicha implementación.

- **Código interno**

Aquí se incluye el código dentro del tag `<script></script>` después del tag `<title>`.

Ejemplo de código interno:

```
<html>
```

```
<head>
```

```
<title>
```

Página web con código JavaScript embebido

```
</title>
```

```
<script type="text/javascript"> insertar tag con atributo type indicando el uso del  
lenguaje java JavaScript alert("Mi primer mensaje usando JavaScript");
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p>Contenido de la página web</p>
```

```
</body>
```

```
</html>
```

- **Inclusión de código**

Desde archivos con extensión `.js` en rutas diferentes a la de la página.

Ejemplo inclusión código con extensión `.js`

```
<html>
```

```
<head>
```

```
<title>
```

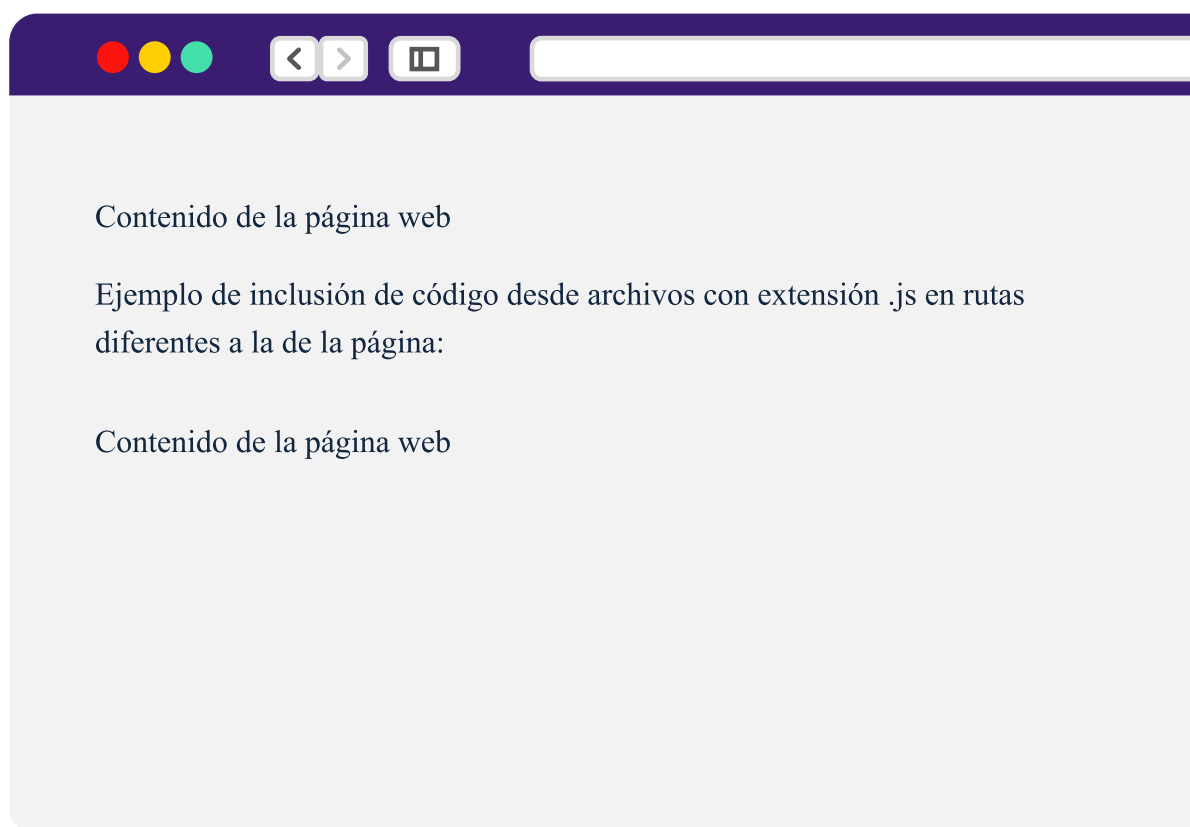

Página web con código JavaScript embebido desde otros archivos fuera de la página actual.

```
</title>
<script src="static/js/miCodeUno.js"></script>
<script src="static/js/miCodeDos.js"></script>
<script src="static/js/miCodeTres.js"></script>
</head>
<body>
<p>Contenido de la página web</p>
</body>
</html>
```

- **Resultado**

El resultado de la ejecución de los códigos anteriores se muestra a continuación:

Figura 7. Resultado de la ejecución de los códigos



- **Ejemplo inclusión de código con URL**

A continuación, se presenta cómo incluir código JavaScript desde archivos externos con extensión .js, alojados en URL diferentes a la de la página:

```
<html>
```

```
<head>
```

```
<title>
```

Página web con código JavaScript embebido desde otros archivos fuera de la página actual.

```
</title>
```

```
<script src="https://www.sitiouno.com/developmentUno.js"
crossorigin></script>
```

```
</head>
<body>
<p>Contenido de la página web</p>
</body>
</html>
```

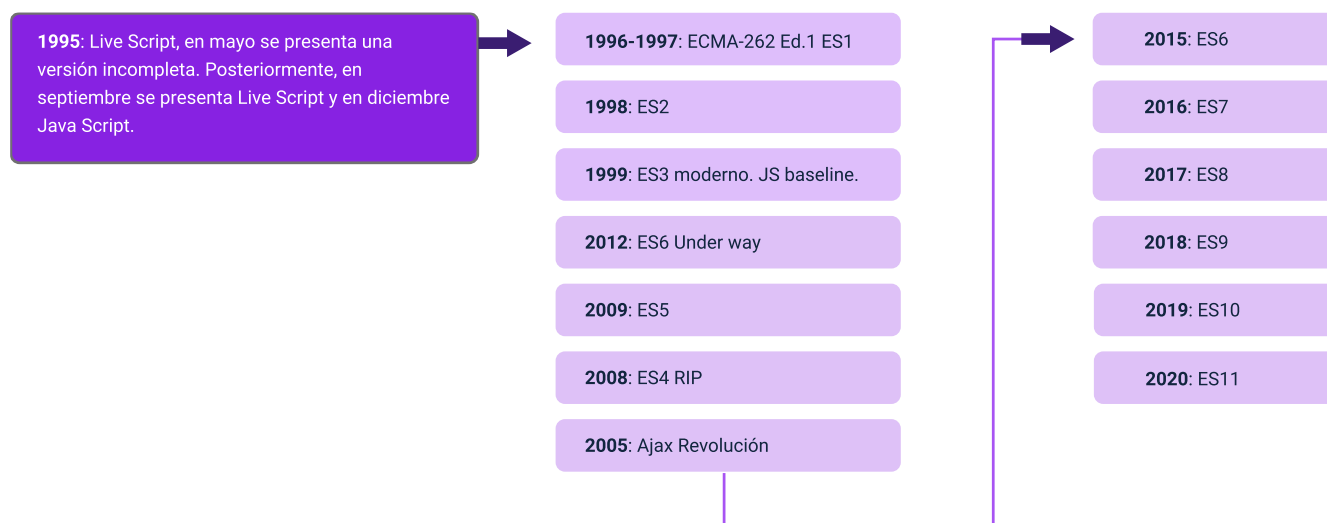
3.1. Versiones

El lenguaje JavaScript fue creado a mediados de la década de 1990 por Brendan Eich, con el objetivo de mejorar la experiencia del usuario. Durante esa época, la velocidad de internet era considerablemente lenta, y surgió la necesidad de un lenguaje de programación ejecutado del lado del cliente para agilizar las peticiones web.

Al año siguiente de su creación, se reconoció la importancia de estandarizar el lenguaje, dado que cada navegador lo interpretaba de manera diferente. Esto llevó a la propuesta y adopción de un estándar conocido como ECMA-262 Edición 1, también referido como ES1 (ECMAScript 1).

La evolución del lenguaje se muestra en la siguiente línea de tiempo:

Figura 8. Línea del tiempo del lenguaje JavaScript



3.2. Sintaxis

En JavaScript, se dispone de varios elementos fundamentales para la construcción de programas, tales como variables, constantes, operadores y expresiones. Las expresiones se forman mediante la combinación de variables, constantes y operadores. También se utilizan sentencias, que organizan el flujo del programa en un orden lógico, y funciones, que agrupan conjuntos de sentencias para resolver problemas específicos. Para manejar adecuadamente estos elementos, existen ciertas reglas de sintaxis y buenas prácticas que se deben seguir al escribir el código, las cuales son:

- **Variables**

Se definen como espacios de memoria reservada para almacenar algún dato y que se identifican con un nombre para definirlas, se debe utilizar la palabra “var” seguida del nombre, el símbolo igual y un valor finalizado con punto y coma.

Ejemplo de variable:

```
var nombreVariable= 20;
```

```
var a=5;
```

```
var b=3;
```

```
var multiplicar=a*b;
```

- **Las constantes**

Como su nombre lo indica, sirven para definir valores únicos que se pueden utilizar en diferentes partes del programa; se definen con la palabra “const” seguida del nombre de variable, el símbolo igual y termina con punto y coma.

Ejemplo de constante:

```
const gravedadTierra=9.8;
```

- **Los operadores**

Se definen en tres grupos: operadores de asignación, operadores de relación y los operadores lógicos.

- a) Los operadores de asignación son igual =, suma +=, resta -=, multiplicación *=, división /= y módulo %.
- b) Los operadores de relación son mayor >, mayor igual >=, menor <, menor igual <=, igual que == y diferente de !=.
- c) Los operadores lógicos para evaluar verdadero o falso son AND && , OR || y NOT !.

¡Nota! Las sentencias se conforman como la combinación de constantes, variables y operadores.

```
var nombreVariable= 20;
```

```
cons gravedadTierra=9.8;
```

```
var resultado = nombreVariable-gravedadTierra;
```

- **Las funciones**

Permiten identificar con un nombre una acción dentro de un programa, es la combinación de sentencias, variables, constantes y expresiones; se escribe utilizando la palabra function seguida de una lista de parámetros de entrada, separados por comas, encerrados entre paréntesis (), en donde el conjunto de sentencias que operan esas entradas se encierran entre llaves {}. Ejemplo de función:

```
function sumarDatos(parametroUno, parametroDos, parametroTres) {  
var a= parametroUno;  
var b= parametroDos;  
var c= parametroTres;  
var resultadoSuma=a+b+c;  
}
```

3.3. Tipos de datos

En JavaScript, una característica fundamental es su tipado dinámico, lo cual significa que el tipo de dato de una variable puede cambiar según el dato asignado por el usuario. Esto ocurre cuando se define una variable utilizando la palabra reservada var, let o const.

- **Variable de caracteres**

Permite almacenar cadenas de texto dentro de las comillas, para bajar un renglón en el texto se debe utilizar \n, para agregar un tabulador \t, para comilla

simple \', para comillas dobles \", y para barra inclinada \\. Ejemplo de variable de caracteres:

```
var nombreApellidos = "JUAN ESTEBAN GOMEZ \n";
```

```
var mensaje = "Este es mi primer mensaje de bienvenida \t \'VALOR\' ";
```

- **Variables numéricas**

Se utilizan para definir variables enteras (integer), reales (float), hexadecimal (0x) y octal (0). Ejemplo de variable numérica:

```
var formatoEntero=45;
```

```
var formatoFlota=7845.123;
```

```
var formatoHexadecimal=0xF34;
```

```
var formatoOctal=0568;
```

- **Variables tipo arreglo**

Denominados también como matrices o vectores y se usan para definir variables que se relacionan en un mismo conjunto; se definen entre corchetes separados por coma y valores entre comillas.

Ejemplo de variable tipo arreglo:

```
var diasSemana = ["Lunes", "Martes", "Miercoles", "Jueves", "Viernes","Sabado",  
"Domingo"];
```

- **Variables tipo lógico**

Permiten definir variables booleanas falso o verdadero. Ejemplo de variable de tipo lógico:

```
var esCasada=true;
```

```
var tieneHijos=false;
```

- **Variables tipo objeto**

Son variables que, a diferencia de los arreglos, se definen dentro de llaves con el nombre de propiedad y su valor. Ejemplo de variable tipo objeto:

```
var miCarro={  
  placa='X45-56',  
  modelo='2015',  
  marca='HONDA',  
};  
alert("Esta es la placa de mi carro..." + miCarro.placa);
```

3.4. Estructuras de control

En un programa JavaScript, las líneas de código se ejecutan de forma secuencial, es decir, en el orden en que aparecen: primero la primera línea, luego la segunda, la tercera, y así sucesivamente, nunca invirtiendo este orden, lo cual determina el flujo del programa. Durante este flujo, puede surgir la necesidad de tomar decisiones basadas en condiciones específicas, momento en el cual se emplean las estructuras condicionales, que pueden derivar en acciones diferentes dependiendo de si la condición evaluada resulta verdadera o falsa. Asimismo, en ciertos puntos del programa puede ser necesario iterar sobre estructuras de datos, como arreglos, para lo cual se utilizan bucles o ciclos.

El condicional if

El más conocido de los condicionales, si se cumple la condición continúa ese camino. Se escribe entre paréntesis después de la palabra reservada if. Ejemplo con condicional if:


```
var edad=17;

//Condicional para saber si es mayor de edad

if(edad > 18 ){

    alert("Ya puede solicitar la cédula");

}
```

Condicional if/else

Este condicional indica el camino alternativo que puede seguir el programa en caso de que no se cumpla la condición de la palabra reservada if. Ejemplo con condicional if/else:

```
var edad=17;

var mensaje;

//Condicional para saber si es mayor de edad

if(edad > 18 ){

    mensaje="Ya puede solicitar la cédula";

}else{
```

Operador ternario

Es una forma de escribir el condicional if/else pero de una manera más corta; se identifica con el símbolo "?". Ejemplo con operador ternario.

```
var edad=17;
```

```
//Operador ternario: (condición ? Verdadero : Falso)
```

```
var mensaje = edad>18 ? 'Cédula': 'Tarjeta Identidad';
```

Condicional if múltiple

Sirve para evaluar múltiples condiciones conservando la estructura del condicional if y para el caso alternativo se escribe la palabra reservada if else acompañada de la nueva condición a cumplir. Ejemplo con condicional if múltiple.

```
var edad=17;
```

```
var mensaje;
```

```
//Condicional para saber si es mayor de edad
```

```
if(edad == 18 ){
```

```
    mensaje="Ya puede solicitar la cédula";
```

```
    } if else (edad > 18 ){
```

```
        mensaje="Tu cédula ya fué tramitada";
```

```
    } if else (edad < 18 ){
```

```
        mensaje="Aún eres menor de edad";
```

```
    }else{
```

```
        mensaje="Verifica el número de edad ingresado";
```

```
    }
```

```
    alert(mensaje);
```

Condicional switch

Define casos específicos a realizar en el caso de que la variable expuesta como condición sea igual a los valores que se especifican mediante los case. Ejemplo con condicional switch:

```
var nota = 7;

switch (nota) {

case 10:

calificacion = "Sobresaliente";

break;

case 9:

case 8:

calificacion = "Notable";

break;

case 7:

case 6:

calificacion = "Bien";

break;

case 5:

calificacion = "Suficiente";

break;
```

```
case 4:

case 3:

case 2:

case 1:

case 0:

calificacion = "Insuficiente";

break;

default:

// Cualquier otro caso

calificacion = "Nota errónea";

break;

}

alert("Resultado"+calificacion);
```

Bucles e iteraciones:

Los bucles son herramientas valiosas para automatizar tareas repetitivas, especialmente cuando se manejan grandes volúmenes de datos. Para implementar un bucle eficazmente, es crucial entender los siguientes conceptos:

- **Iteración**

Representa el número de veces que el bucle ha ejecutado la condición.

- **Contador**

Es una variable que registra el número actual de iteraciones del bucle.

- **Incremento**

Es el valor que se añade al contador tras completarse un ciclo, lo cual es esencial para acercar el bucle a su condición de finalización.

- **Bucle infinito**

Se produce cuando el incremento al contador no se realiza, llevando a que el bucle nunca finalice. Es imperativo evitar esta situación para prevenir problemas de rendimiento en el programa

A continuación, se presentan varios ejemplos de bucles:

Bucle while

- Antes de entrar en el bucle while, se inicializa la variable i.
- Antes de realizar la primera iteración del bucle, se comprueba la condición.
- Si la condición es verdadera, se hace lo que está dentro del bucle.
- Se muestra por pantalla el valor de i y luego incrementamos el valor actual de i en 1.
- Se vuelve al inicio del bucle para hacer una nueva iteración. Se comprueba de nuevo la condición del bucle.
- Cuando la condición sea falsa, se sale del bucle y continúa el programa.

i = 0; // Inicialización de la variable contador

```
// Condición: Mientras la variable contador sea menor de 5
```

```
while (l < 5) {
```

```
    console.log("Valor de l:", i);
```

```
    i = i + 1; // Incrementamos el valor de i
```

```
}
```

```
// for (inicialización; condición; incremento)
```

```
for (i = 0; i < 5; i++) {
```

```
    console.log("Valor de l:", i);
```

```
}
```

```
for (i = 0, j = 5; i < 5; i++, j--) {
```

```
    console.log("Valor de l y j:", i, j);
```

```
}
```

```
var dato = [2, 6, 5, 1, 18, 44];
```

```
var msgForNormal = "";
```

```
var msgForIn = "";
```

```
//for normal
```

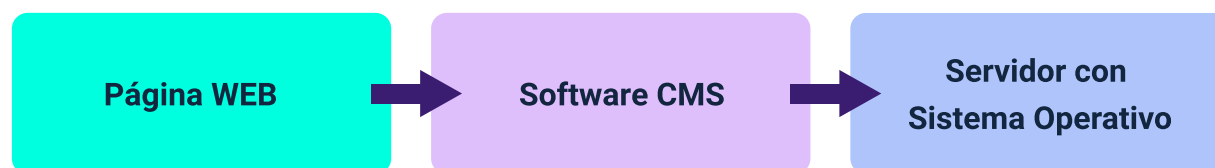
```
for (var l=0; l  
  
msgForNormal = msgForNormal + dato[i] + ' - '  
  
}
```

4. Gestores de contenido CMS

Los gestores de contenidos, conocidos como CMS (Content Management System), surgieron para simplificar la labor de los administradores de sitios web, comúnmente llamados webmasters. Su desarrollo ha avanzado significativamente, posicionándolos como herramientas clave para publicar información y contenido propio en la web. Esto permite a los usuarios gestionar sus sitios sin la necesidad de externalizar el servicio o contratar a un profesional, evitando así costos innecesarios.

A continuación, se presenta una figura que ilustra la integración general de un CMS en un sitio web:

Figura 9. Integración general del CMS



Un gestor de contenidos o CMS, es un software web diseñado para facilitar la creación, gestión y publicación de contenido en páginas web, sin requerir conocimientos avanzados en desarrollo de software o programación informática. Estas herramientas proporcionan una plataforma de administración virtual que permite a los usuarios crear, editar y publicar contenido web de manera eficiente.

Funciones

A continuación, se describen las principales funciones que debe cumplir un CMS.

- **Independencia entre código y contenido**

Permite separar la gestión de la estructura y funcionalidad de la web de la administración de contenidos, facilitando el trabajo de perfiles distintos: desarrolladores y creadores de contenido.

- **Plantillas en diseño web**

Facilitan modificar la apariencia del sitio web usando plantillas para colores, tipos de letra, imágenes, etc. Hay plantillas gratuitas y profesionales disponibles.

- **Extensión de capacidades**

Los CMS permiten añadir nuevas funcionalidades mediante plugins, cubriendo así necesidades específicas del sitio web.

- **Enfocados al SEO**

Facilitan la implementación de estrategias de SEO para mejorar el posicionamiento en motores de búsqueda.

- **Gestión de enlaces**

Permiten una gestión eficiente de los permalinks asociados a cada contenido, importante para técnicas SEO.

- **Flujos de trabajo**

Posibilitan definir responsabilidades dentro del equipo de administración, con flujos de aprobación para la publicación de contenidos.

- **Comentarios**

Incluyen sistemas de gestión de comentarios que permiten validar autores y administrar los comentarios recibidos.

- **Control de versiones**

Permiten versionar el contenido, facilitando la recuperación de versiones anteriores y la publicación de versiones específicas.

- **Múltiples idiomas**

Ofrecen gestión de contenido en varios idiomas, adaptándose a las preferencias del visitante.

- **Múltiples sitios**

Permiten gestionar varios sitios o secciones desde una plataforma centralizada, optimizando el mantenimiento.

- **Seguridad**

Enfocan en perfilar el contenido según el usuario y proveen protección general al sitio web.

Características

A continuación, se presentan las características de los gestores de contenido CMS:

- El frontend corresponde a la parte visible del CMS para los usuarios que acuden como invitados o usuarios registrados.
- Conformación modular para la configuración y personalización de los sitios web.
- Administración de usuarios, roles, perfiles y permisos de acceso.
- Componentes para la creación, edición y publicación de contenidos.
- Plantillas para distribuir y mantener el diseño del sitio web.

- Posibilidad de extensiones que complementan las funcionalidades en el gestor de contenidos.
- Características de manejo de internacionalización en varios idiomas.

Clasificación CMS

Se clasifican normalmente teniendo en cuenta los siguientes tres aspectos, de acuerdo al lenguaje de programación, a la distribución de la licencia y a las funcionalidades que presenta:

Teniendo en cuenta el lenguaje de programación:

- **Java**

jAPS, Liferay, DSpace, Fedora, Nuxeo EP, Magnolia, Hippo CMS, Calenco, Polopoly, IBM Lotus Web Content Management, Day Communiqué WCM, Jarimba, Vignette, etc.

- **php**

Drupal, CMS Made Simple, Joomla!, Mambo, PHP-Nuke, TikiWiki, TYPO3, aWordPress, Xoops, Zikula, Jadu, ExpressionEngine, Accrisoft Freedom, CMS 10, Dim Works CMS, Content-SORT, Prodigia Easy Site Manager, PipePS, SiteAd CMS, etc.

- **ASP.NET**

DotNetNuke Community Edition, Umbraco, mojoPortal, Kentico CMS, SharePoint Server, Telligent Community, Ektron CMS400.NET, Quantum Art QP7, webControl CMS, etc.

Teniendo como base la distribución de la licencia:

De software libre (Open Source): significa que esta herramienta no presenta costos de licencia y el código fuente es accesible para ser modificado por los desarrolladores. El soporte técnico de este tipo de CMS es asumido por la entidad que lo implementa o por medio de comunidades online que comparten y mantienen su documentación.

- **WORD PRESS**

En esta categoría, algunos de los gestores más reconocidos son WordPress, Drupal, Joomla, Plone, TYPO3, OpenCMS, PHPNuke o Moodle.

De código privado: presentan costos de licencia y servicio, el código fuente es propietario por tanto sólo puede ser accedido y modificado por la compañía o personas que lo desarrollaron. En cuanto al soporte técnico es parte de la oportunidad de negocio y normalmente es de excelente calidad, así como una gran cantidad de información documental.

- Los gestores más destacados son CMS10, Eximius2 CMS, Contendo CMS, Jarimba, CMS HYDRAportal, OnBase, IWEB, Oracle Portal, PipePS, Paloo, Smartone CMS, Vbulletin, XCM – Xeridia Content Manager, ZWeb Publisher CMS.
- **Plataformas en general**
Drupal, Gekko, E107, Joomla, Mambo, PHP-Nuke, TYPO3, TYPOLight, XOOPS, ZWeb Publisher CMS, ADSM Portal 2.0, 360 Web Manager Software, GTLive

- **Sitios de educación**

ATutor, Claroline, Dokeos, eCollege, FrogTeacher, Moodle, Sakai Project, Scholar360, Synergeia, Teletu

- **Blogs**

WordPress, bBlog, DotClear, Lifetype, Plone, Nucleus CMS, Blogger, Textpattern

- **Galerías**

Gallery, Pixelpost, Expression Engine

- **Wikis**

MediaWiki, TikiWiki, Twiki

- **Comercio electrónico**

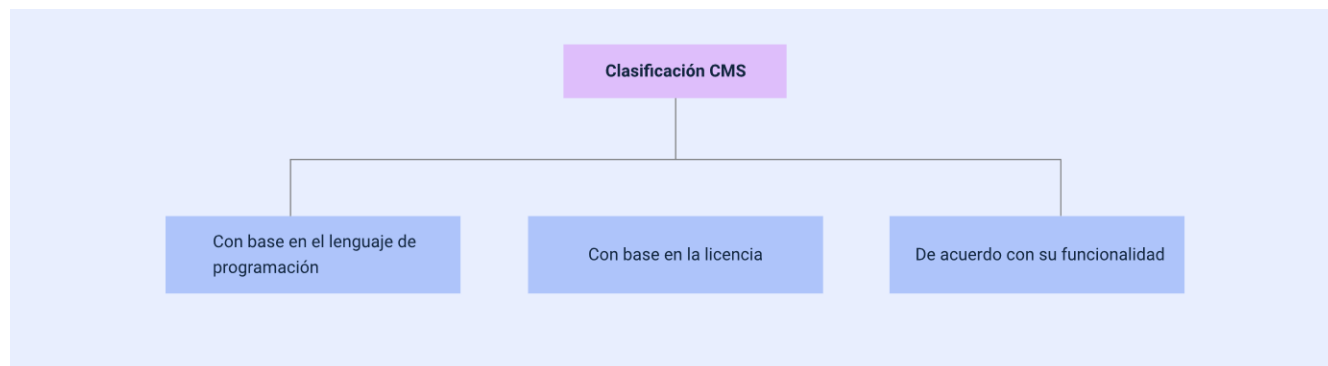
osCommerce, Magento, Zen Cart, Drupal e-Commerce, CubeCart, OpenCart, VirtueMart

- **Groupware**

Webcollab, eGroupWare, Group-Office

Como se indicó previamente, recordamos que la clasificación de los CMS se presenta de la siguiente forma:

Figura 10. Esquema de clasificación de los CMS



Tipos CMS

Los tipos de gestores de contenidos varían según los tipos de páginas web que se deseen desarrollar o crear. Para satisfacer estas necesidades, existen gestores de contenidos altamente especializados, equipados con herramientas y características diseñadas para cumplir con requerimientos específicos. A continuación, se presentan algunos de los gestores de contenidos disponibles para su uso en diferentes contextos:

- Blogs
- Páginas corporativas
- Tiendas online o ecommerce
- Sitios de e-learning
- Foros
- Wikis

Síntesis

A continuación, se muestra un mapa conceptual con los elementos más importantes desarrollados en este componente.

Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
HTML 5	HTMLeD.it. (2021). Editor HTML gratuito en línea, limpiador y convertidor.	Página web	https://htmled.it/or/
JavaScript	Cubic Factory. (2021). Ejecutar JavaScript.	Página web	https://www.cubicfactory.com/jseditor/

Glosario

CMS: sistema de gestión de contenidos.

CSS3: hojas de estilo en cascada, versión 3.

ES6: abreviación de ECMAScript, versión 6.

Frontend: capa de presentación de una aplicación web.

Get: método de envío de datos por URL web.

JavaScript: lenguaje de programación.

LAN: red de área local.

Landing Page: página web de aterrizaje, diseñada específicamente para una campaña de marketing o publicidad.

LESS: hoja de estilo dinámica, extensión que amplía las capacidades de CSS.

MAN: red de área metropolitana.

NODE.js: entorno de ejecución para JavaScript del lado del servidor.

Plugin: aplicación que extiende las funcionalidades de un programa principal.

Post: método de envío de datos por formulario web.

SEO: optimización para motores de búsqueda, estrategias para mejorar la visibilidad de un sitio web.

WAN: red de área extensa o mundial.

Referencias bibliográficas

Acibeiro, M. (2021). Qué es un gestor de contenidos y cuál es el mejor para crear una web. <https://www.lucushost.com/blog/gestor-de-contenidos/>

Beati, H. (2016). HTML5 y CSS3.

Cuervo, P. V. (2019). 10 características de un CMS.
<https://www.arquitectoit.com/cms/10-caracteristicas-cms/>

Gutierrez, R. (2018). Understanding the role of digital commons in the web; The making of HTML5.

Hverbeke, M. (2018). Eloquent JavaScript: A Modern Introduction to Programming.

Lenguaje JS. (2021). ¿Qué son los bucles? Bucles, iteraciones y repetición de código. <https://lenguajejs.com/javascript/introduccion/bucles/>

Mooc, Aprendizaje Online.(2015). Clasificación de los sistemas de gestión de contenidos. <https://mooccontenidosweb.wordpress.com/2015/05/01/clasificacion-de-los-sistemas-de-gestion-de-contenidos/>

Mora, S. L. (2021). HTML5 y CSS3: <6> HTML5 – Formularios.
<http://desarrolloweb.dlsi.ua.es/cursos/2011/html5-css3-es/html5-formularios>

Créditos

Nombre	Cargo	Centro de Formación y Regional
Milady Tatiana Villamil Castellanos	Responsable del Ecosistema	Dirección General
Olga Constanza Bermúdez Jaimes	Responsable de Línea de Producción	Centro de Servicios de Salud - Regional Antioquia
Carlos Hernán Muñoz Carvajal	Experto Temático	Centro de Teleinformática y Producción Industrial - Regional Cauca
Paola Alexandra Moya Peralta	Evaluadora Instruccional	Centro de Servicios de Salud - Regional Antioquia
Juan Daniel Polanco Muñoz	Diseñador de Contenidos Digitales	Centro de Servicios de Salud - Regional Antioquia
Jhon Jairo Urueta Álvarez	Desarrollador Fullstack	Centro de Servicios de Salud - Regional Antioquia
Edgar Mauricio Cortés García	Actividad Didáctica	Centro de Servicios de Salud - Regional Antioquia
Daniela Muñoz Bedoya	Animador y Productor Multimedia	Centro de Servicios de Salud - Regional Antioquia
Jaime Hernán Tejada Llano	Validador de Recursos Educativos Digitales	Centro de Servicios de Salud - Regional Antioquia
Margarita Marcela Medrano Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro de Servicios de Salud - Regional Antioquia
Daniel Ricardo Mutis Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro de Servicios de Salud - Regional Antioquia