EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Dirección General de Educación Tecnológica Industrial y de Servicios

Dirección Académica e Innovación Educativa

Subdirección de Innovación Académica

Departamento de Planes, Programas y Superación Académica

Cuadernillo de Aprendizajes Esenciales

Módulo III

Programación

Curso intersemetral

México 2021 Año de la Independencia

<table>
<tr><td colspan="4">Cuadernillo de Aprendizajes esenciales</td></tr>
<tr><td>Carrera:</td><td>Programación</td><td>Semestre</td><td>Cuarto</td></tr>
<tr><td colspan="2">Módulo/ Submódulo</td><td colspan="2">Módulo III. Desarrolla aplicaciones WEB.
Submódulo 1.- Construye páginas WEB.
Submódulo 2.- Desarrolla aplicaciones que se ejecutan en el cliente.
Submódulo 3.- Desarrolla aplicaciones que se ejecutan en el servidor.</td></tr>
<tr><td>Aprendizajes esenciales o Competencias esencias 1er parcial.</td><td colspan="2">Estrategias de aprendizaje (Actividades)</td><td>Productos a evaluar</td></tr>
<tr><td>El alumno
Emplea HTML para diseño de paginas WEB

El alumno
Emplea JavaScript para control de eventos y el formulario de datos

Diseño de proyecto fina
mediante las etapas:
• Creación de BD en servidor
• Diseño Front-end (formularios)
• Diseño Back-end (conexión con bd y ejecución en servidor)</td><td colspan="2">Actividad 1.- Instalación de editor de código para HTML.
Instala el sublime text para codificar en HTML y PHP y otros lenguajes.
Para hacerlo puedes ayudarte de un video tutorial de tu preferencia.
https://www.youtube.com/watch?v=4lq43AbONKc

Actividad 2.- Elabora un documento HTML que contenga lo siguiente:
Imágenes
Encabezados
Párrafos
Listas
Hipervínculos

Conceptos básicos de HTML

El Lenguaje de Marcado de Hipertexto (HTML) es el código que se utiliza para estructurar y desplegar una página web y sus contenidos. Por ejemplo, sus contenidos podrían ser párrafos, una lista con viñetas, o imágenes y tablas de datos. Como lo sugiere el título, este artículo te dará una comprensión básica de HTML y cúal es su función.

Entonces, ¿qué es HTML en realidad?</td><td>Un documento HTML que contenga:

Imágenes
Encabezados
Párrafos
Listas
Hipervínculos
Colores
Eventos
botones</td></tr>
</table>

HTML no es un lenguaje de programación; es un *lenguaje de marcado* que define la estructura de tu contenido. HTML consiste en una serie de elementos que usarás para encerrar diferentes partes del contenido para que se vean o comporten de una determinada manera. Las etiquetas de encierre pueden hacer de una palabra o una imagen un hipervínculo a otro sitio, se pueden cambiar palabras a cursiva, agrandar o achicar la letra, etc. Por ejemplo, toma la siguiente línea de contenido:

```
Mi gato es muy gruñon
```

Si quieres especificar que se trata de un párrafo, podrías encerrar el texto con la etiqueta de párrafo (`<p>`):

```
<p>Mi gato es muy gruñon</p>
```

Anatomía de un elemento HTML

Explora este párrafo en mayor profundidad.

Las partes principales del elemento son:

1. **La etiqueta de apertura**: consiste en el nombre del elemento (en este caso, p), encerrado por **paréntesis angulares** (< >) de apertura y cierre. Establece dónde comienza o empieza a tener efecto el elemento —en este caso, dónde es el comienzo del párrafo—.
2. **La etiqueta de cierre**: es igual que la etiqueta de apertura, excepto que incluye una barra de cierre (/) antes del nombre de la etiqueta. Establece dónde termina el elemento —en este caso dónde termina el párrafo—.
3. **El contenido**: este es el contenido del elemento, que en este caso es sólo texto.
4. **El elemento**: la etiqueta de apertura, más la etiqueta de cierre, más el contenido equivale al elemento.

Los elementos pueden también tener atributos, que se ven así:

Los atributos contienen información adicional acerca del elemento, la cual no quieres que aparezca en el contenido real del elemento. Aquí `class` es el *nombre* del atributo y `editor-note` el *valor* del atributo. En este caso, el atributo `class` permite darle al elemento un nombre identificativo, que se puede utilizar luego para apuntarle al elemento información de estilo y demás cosas.

Un atributo debe tener siempre:

1. Un espacio entre este y el nombre del elemento (o del atributo previo, si el elemento ya posee uno o más atributos).
2. El nombre del atributo, seguido por un signo de igual (**=**).
3. Comillas de apertura y de cierre, encerrando el valor del atributo.

Los atributos siempre se incluyen en la etiqueta de apertura de un elemento, nunca en la de cierre.

Nota: el atributo con valores simples que no contengan espacios en blanco ASCII (o cualesquiera de los caracteres `"` `'` `` ` `` `=` `<` `>`) pueden permanecer sin entrecomillar, pero se recomienda entrecomillar todos los valores de atributo, ya que esto hace que el código sea más consistente y comprensible.

Anidar elementos

Puedes también colocar elementos dentro de otros elementos —esto se llama **anidamiento**—. Si, por ejemplo, quieres resaltar una palabra del texto (en el ejemplo la palabra «muy»), podemos encerrarla en un elemento `<strong>`, que significa que dicha palabra se debe enfatizar:

```
<p>Mi gato es <strong>muy</strong> gruñon.</p>
```

Debes asegurarte que los elementos estén correctamente anidados: en el ejemplo de abajo, creaste la etiqueta de apertura del elemento `<p>` primero, luego la del elemento `<strong>`, por lo tanto, debes cerrar esta etiqueta primero, y luego la de <p>. Esto es incorrecto:

```
<p>Mi gato es <strong>muy gruñon.</p></strong>
```

Los elementos deben abrirse y cerrarse ordenadamente, de forma tal que se encuentren claramente dentro o fuera el uno del otro. Si estos se encuentran solapados, el navegador web tratará de adivinar lo que intentas decirle, pero puede que obtengas resultados inesperados. Así que, ¡no lo hagas!

Elementos vacíos

Algunos elementos no poseen contenido, y son llamados **elementos vacíos**. Toma, por ejemplo, el elemento `<img>` de nuestro HTML:

```
<img src="images/firefox-icon.png" alt="Mi imagen de prueba">
```

Posee dos atributos, pero no hay etiqueta de cierre `</img>` ni contenido encerrado. Esto es porque un elemento de imagen no encierra contenido al cual afectar. Su propósito es desplegar una imagen en la página HTML, en el lugar en que aparece.

Anatomía de un documento HTML

Hasta ahora has visto lo básico de elementos HTML individuales, pero estos no son muy útiles por sí solos. Ahora verás cómo los elementos individuales son combinados para formar una página HTML entera. Vuelve a visitar el código de tu ejemplo en `index.html` (que viste por primera vez en el artículo Manejo de archivos):

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <title>Mi pagina de prueba</title>

  </head>

  <body>

    <img src="images/firefox-icon.png" alt="Mi imagen de prueba">

  </body>

</html>
```

Tienes:

- `<!DOCTYPE html>` — el tipo de documento. Es un preámbulo requerido. Anteriormente, cuando HTML era joven (cerca de 1991/2), los tipos de documento actuaban como vínculos a un conjunto de reglas que el código HTML de la página debía seguir para ser considerado bueno, lo que podía significar la verificación automática de errores y algunas otras cosas de utilidad. Sin embargo, hoy día es simplemente un artefacto antiguo que a nadie le importa, pero que debe ser incluido para que todo funcione correctamente. Por ahora, eso es todo lo que necesitas saber.
- `<html></html>` — el elemento `<html>`. Este elemento encierra todo el contenido de la página entera y, a veces, se le conoce como el elemento raíz (*root element*).
- `<head></head>` — el elemento `<head>`. Este elemento actúa como un contenedor de todo aquello que quieres incluir en la página HTML que *no* es contenido visible por los visitantes de la página. Incluye cosas como palabras clave (keywords), una descripción de la página que quieres que aparezca en resultados de búsquedas, código CSS para dar estilo al contenido, declaraciones del juego de caracteres, etc.
- `<meta charset="utf-8">` — `<meta>`. Este elemento establece el juego de caracteres que tu documento usará en `utf-8`, que incluye casi todos los caracteres de todos los idiomas humanos. Básicamente, puede manejar cualquier contenido de texto que puedas incluir. No hay razón para no establecerlo, y puede evitar problemas en el futuro.
- `<title></title>` — el elemento `<title>` establece el título de tu página, que es el título que aparece en la pestaña o en la barra de título del navegador cuando la página es cargada, y se usa para describir la página cuando es añadida a los marcadores o como favorita.
- `<body></body>` — el elemento `<body>`. Encierra *todo* el contenido que deseas mostrar a los usuarios web que visiten tu página, ya sea texto, imágenes, videos, juegos, pistas de audio reproducibles, y demás.

Imágenes

Presta atención nuevamente al elemento *imagen* `<img>`:

```
<img src="images/firefox-icon.png" alt="Mi imagen de prueba">
```

Como ya se dijo antes, incrusta una imagen en la página, en la posición en que aparece. Lo logra a través del atributo `src` (source), el cual contiene el *path (ruta o ubicación)* de tu archivo de imagen.

También se incluye un atributo `alt` (alternative) el cual contiene un texto que debería describir la imagen, y que podría ser accedido por usuarios que no pueden ver la imagen, quizás porque:

1. Son ciegos o tienen deficiencias visuales. Los usuarios con impedimentos visuales usualmente utilizan herramientas llamadas *Lectores de pantalla (Screen Readers),* los cuales les leen el texto contenido en el atributo alt.
2. Se produjo algún error en el código que impide que la imagen sea cargada. Como ejemplo, modifica deliberadamente la ubicación dentro del atributo `src` para que este sea incorrecto. Si guardas y recargas la página, deberías ver algo así en lugar de la imagen:

Mi pagina de prueba

La frase clave acerca del texto alt de arriba es «texto que debería describir la imagen». El texto alt debe proporcionarle al lector la suficiente información como para que este tenga una buena idea de qué muestra la imagen. Por lo que tu texto actual «Mi imagen de prueba» no es para nada bueno. Un texto mucho mejor para el logo de Firefox sería: *«El logo de Firefox: un zorro en llamas rodeando la Tierra»*.

Prueba a dar con mejores textos alt para tu imagen.

Nota: Descubre más acerca de la accesibilidad en el módulo de aprendizaje sobre la accesibilidad.

Marcado de texto

Esta sección cubrirá algunos de los elementos HTML básicos que usarás para el marcado de texto.

Encabezados

Los elementos de encabezado permiten especificar que ciertas partes del contenido son encabezados, o subencabezados del contenido. De la misma forma que un libro tiene un título principal, y que a su vez puede tener títulos por cada capítulo individual, y subtítulos dentro de ellos, un documento HTML puede tenerlos también. HTML posee seis niveles de encabezados, `<h1>`–`<h6>`, aunque probablemente solo llegues a usar 3-4 como mucho:

```
<h1>Mi título principal</h1>

<h2>Mi título de nivel superior</h2>

<h3>Mi subtítulo</h3>

<h4>Mi sub-subtítulo</h4>
```

Intenta ahora añadir un título apropiado para tu página HTML, antes de tu elemento `<img>`.

Nota: verás que el encabezamiento de nivel 1 tiene un estilo implícito. No utilices elementos de encabezado para hacer el texto más grande o más oscuro, porque este elemento se utiliza por accesibilidad y otras razones como el posicionamiento en buscadores (*Search Engine Optimization, SEO*). Intenta crear una secuencia significativa de encabezados en tus páginas, sin saltarte niveles.

Párrafos

Como se explicó más arriba, los elementos `<p>` se utilizan para encerrar párrafos de texto; los usarás frecuentemente para el marcado de contenido de texto regular:

```
<p>Este es un simple parrafo</p>
```

Agrega uno o algunos párrafos a tu texto de ejemplo (deberías tenerlo de cuando estudiaste *¿Cuál será la apariencia de tu sitio web?*), colocados directamente debajo del elemento <img>.

Listas

Mucho del contenido web está dado por listas, así que HTML tiene elementos especiales para ellas. El marcado de listas se realiza siempre en al menos dos elementos. Los dos tipos de listas más comunes son las listas ordenadas y las desordenadas:

1. **Las listas desordenadas** son aquellas en las que el orden de los items no es relevante, como en una lista de compras. Estas son encerradas en un elemento `<ul>` (*unordered list*).
2. **Las listas ordenadas** son aquellas en las que el orden sí es relevante, como en una receta. Estas son encerradas en un elemento `<ol>` (*ordered list*).

Cada elemento de la lista se coloca dentro de un elemento `<li>` *(list item*).

Por ejemplo, si quieres transformar parte del siguiente párrafo en una lista:

```
<p>En Mozilla, somos una comunidad de tecnólogos, pensadores, y constructores que trabajan juntos... </p>
```

Podrías hacer lo siguiente:

```
<p>En Mozilla, somos una comunidad de</p>



<ul>

  <li>tecnólogos</li>

  <li>pensadores</li>

  <li>constructores</li>

</ul>
```

```
<p>trabajando juntos... </p>
```

Intenta agregar una lista ordenada o desordenada en tu página de ejemplo**.**

Vínculos

Los vínculos o enlaces son muy importantes —son los que hacen de la web, la web—. Para implementar un vínculo, necesitas usar un vínculo simple — `<a>` — la *a* es la abreviatura de la palabra inglesa «anchor» («*ancla*»). Para convertir algún texto dentro de un párrafo en un vínculo, sigue estos pasos:

1. Elige algún texto. Nosotros elegimos «Manifesto Mozilla».
2. Encierra el texto en un elemento <a>, así:
   ```
   <a>Manifesto Mozilla</a>
   ```
3. Proporciónale al elemento <a> un atributo href, así:
   ```
   <a href="">Manifesto Mozilla</a>
   ```
4. Completa el valor de este atributo con la dirección web con la que quieras conectar al vínculo:
   ```
   <a href="https://www.mozilla.org/es-AR/about/manifesto/">Manifesto Mozilla</a>
   ```

Podrías obtener resultados inesperados si al comienzo de la dirección web omites la parte `https://` o `http://` llamada *protocolo*. Así que luego del marcado del vínculo, haz clic en él para asegurarte que te dirige a la dirección deseada.

`href` podría parecer, en principio, una opción un tanto oscura para un nombre de atributo. Si tienes problemas para recordarla, recuerda que se refiere a ***h****ypertext* ***ref****erence* (referencia de hipertexto).

	Ahora agrega un vínculo a tu página, si es que aún no lo hiciste. Si lograste seguir todas las instrucciones de este artículo, deberías terminar con una página que se vea así (también puedes verla aquí):	


My test page
rs/chrismills/Dropbox/Work/git/web-
Google
Most Visited
MDN
Mozilla general
FirefoxOS ecosy...
Meeting details
Mozilla is cool
At Mozilla, we're a global community of
technologists
thinkers
builders
working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.
Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

Si te estancas en algún paso, puedes comparar tu trabajo con el código de ejemplo terminado en Github.

Aquí realmente solo has rasguñado la superficie de HTML. Para aprender más, ve a la página de Aprendizaje HTML.

Fuente:
https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics

Actividad 3.- Haciendo uso de hojas de estilo agrega estilo y color tu documento HTML.

CSS (*Hojas de Estilo en Cascada*) es el código que usas para dar estilo a tu página web. *CSS Básico* te lleva a través de lo que tú necesitas para empezar. Contestará a preguntas del tipo: ¿Cómo hago mi texto rojo o negro? ¿Cómo hago que mi contenido se muestre en tal y tal lugar de la pantalla? ¿Cómo decoro mi página web con imágenes de fondo y colores?

Entonces ¿qué es CSS, realmente?

Como HTML, CSS (*Cascading Style Sheets*) u Hojas de estilo en cascada en español, no es realmente un lenguaje de programación, tampoco es un lenguaje de marcado. Es un *lenguaje de hojas de estilo,* es decir, te permite aplicar estilos de manera selectiva a elementos en documentos HTML. Por ejemplo, para seleccionar **todos** los elementos de párrafo en una página HTML y volver el texto dentro de ellos de color rojo, has de escribir este CSS:

```
p {

  color: red;

}
```

Copy to Clipboard

Vas a probarlo: pega estas tres líneas de CSS en un nuevo archivo en tu editor de texto y guarda este archivo como `style.css` en tu directorio `styles` (estilos).

Pero aún debes aplicar el CSS a tu documento HTML, de otra manera el estilo CSS no cambiará cómo tu navegador muestra el documento HTML. (Si no has seguido nuestro proyecto, lee Manejo de archivos y HTML básico para averiguar qué necesitas hacer primero.)

1. Abre tu archivo `index.html` y pega la siguiente línea en algún lugar dentro del `<head>`, es decir, entre las etiquetas `<head>` y `</head>`:

```
<link href="styles/style.css" rel="stylesheet" type="text/css">
```

   Copy to Clipboard
2. Guarda el archivo `index.html` y cárgalo en tu navegador. Debes ver algo como esto:


My test page
file:///Users/chrismills/Dropbc
Google
Most Visited
MDN
Mozilla general
FirefoxOS ecosy...
Meeting details
Mozilla is cool
At Mozilla, we're a global community of
technologists
thinkers
builders
working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.
Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

Si tu texto del párrafo ahora es rojo, ¡felicitaciones, ya has escrito tu primer CSS de forma exitosa!

Anatomía de una regla CSS

Observa el código CSS de arriba, un poco más a detalle:

La estructura completa es llamada **regla predeterminada** (pero a menudo «regla» para abreviar). Nota también los nombres de las partes individuales:

Selector

El elemento HTML en el que comienza la regla. Esta selecciona el(los) elemento(s) a dar estilo (en este caso, los elementos `<p>`). Para dar estilo a un elemento diferente, solo cambia el selector.

Declaración

Una sola regla como `color: red;` especifica a cuál de las **propiedades** del elemento quieres dar estilo.

Propiedades

Maneras en las cuales puedes dar estilo a un elemento HTML. (En este caso, `color` es una propiedad del elemento `<p>`). En CSS, seleccionas qué propiedad quieres afectar en tu regla.

Valor de la propiedad

A la derecha de la propiedad, después de los dos puntos (:), tienes el **valor de la propiedad**, para elegir una de las muchas posibles apariencias para una propiedad determinada (hay muchos valores para `color` además de `red`).

Nota las otras partes importantes de la sintaxis:

- Cada una de las reglas (aparte del selector) deben estar encapsuladas entre llaves (`{}`).
- Dentro de cada declaración, debes usar los dos puntos (`:`) para separar la propiedad de su valor.
- Dentro de cada regla, debes usar el punto y coma (`;`) para separar una declaración de la siguiente.

De este modo para modificar varios valores de propiedad a la vez, solo necesitas escribirlos separados por punto y coma (;), así:

```
p {

  color: red;
```

Nombre del selector	Qué selecciona	Ejemplo
Selector de elemento (llamado algunas veces selector de etiqueta o tipo)	Todos los elementos HTML del tipo especificado.	`p` Selecciona `<p>`
Selector de identificación (ID)	El elemento en la página con el ID especificado (en una página HTML dada, solo se permite un único elemento por ID).	`#mi-id` Selecciona `<p id="mi-id">` y `<a id="mi-id">`
Selector de clase	Los elementos en la página con la clase especificada (una clase puede aparecer varias veces en una página).	`.mi-clase` Selecciona `<p class="mi-clase">` y `<a class="mi-clase">`

Selector de atributo	Los elementos en una página con el atributo especificado.	`img[src]` Selecciona `<img src="mimagen.png">` pero no `<img>`
Selector de pseudoclase	Los elementos especificados, pero solo cuando esté en el estado especificado, por ejemplo cuando el puntero esté sobre él.	`a:hover` Selecciona `<a>`, pero solo cuando el puntero esté sobre el enlace.

```
  width: 500px;

  border: 1px solid black;

}
```

Copy to Clipboard

Seleccionar varios elementos

También puedes seleccionar varios elementos y aplicar una sola regla a todos ellos. Incluye varios selectores separados por comas (,). Por ejemplo:

```
p,li,h1 {

  color: red;

}
```

Copy to Clipboard

Diferentes tipos de selectores

Existen muchos tipos diferentes de selectores. Antes, solo viste los **selectores de elementos**, los cuales seleccionan todos los elementos de un tipo dado en los documentos

HTML. Sin embargo puedes hacer selecciones más específicas que esas. En seguida están algunos de los tipos de selectores más comunes:

Existen muchos más selectores para explorar, y podrás encontrar una lista más detallada en la guía de Selectores.

Fuentes y texto

Ahora que has explorado lo básico de CSS, empieza por añadir información y algunas reglas más a tu archivo `style.css` para que tu ejemplo se vea bonito. Primero, haz que tus fuentes y texto luzcan un poco mejor.

1. Antes que nada, regresa y busca las fuentes de Google Fonts que guardaste en un lugar seguro. Agrega el elemento `<link>...` en algún lugar del *head* de tu archivo `index.html` (de nuevo, en cualquier lugar entre las etiquetas `<head>` y `</head>`). Debe verse algo así:

```
<link href="https://fonts.googleapis.com/css2?family=Open+Sans" rel="stylesheet"
type="text/css">
```

   Copy to Clipboard
2. Luego, borra la regla existente en tu archivo `style.css`. Fue una buena prueba, pero el texto en rojo en realidad no se ve muy bien.
3. Añade las siguientes líneas (que se muestran a continuación), sustituyendo la asignación de `font-family` por tu selección de `font-family` que obtuviste en ¿Cuál será la apariencia de tu sitio Web? La propiedad `font-family` se refiere a la(s) fuente(s) que deseas usar en tu texto. Esta regla define una fuente base global y un tamaño de fuente para usar en toda la página. Dado que `<html>` es el elemento primario (o padre) de toda la página, todos los elementos contenidos dentro de él heredan las propiedades `font-size` y `font-family`):

```
4. html {
```

```
  font-size: 10px; /* px quiere decir 'píxeles': el tamaño de la fuente base es ahora de 10 píxeles de altura */

  font-family: "Open Sans", sans-serif; /* Este debe ser el resto del resultado que obtuviste de Google fonts */

}
```

Copy to Clipboard

Nota: se ha añadido un comentario para explicar qué significa «px». Todo lo que está en un documento de CSS entre `/*` y `*/` es un **comentario en CSS**, el cual el navegador descarta cuando carga el código. Este es un espacio donde puedes escribir notas útiles sobre lo que estás haciendo.

7. Ahora escoge el tamaño de fuente para los elementos que contienen texto dentro del cuerpo del HTML (`<h1>`, `<li>`, y `<p>`). También centra el texto del título, escoge un ancho de línea y espaciado entre letras en el contenido del texto para hacerlo un poco más legible:

```
h1 {

  font-size: 60px;

  text-align: center;

}

p, li {

  font-size: 16px;

  line-height: 2;
```

```
16.   letter-spacing: 1px;

    }
```

Copy to Clipboard

Puedes ajustar estos valores en px para lograr que tu diseño luzca como desees, pero por lo general tu diseño debe verse así:


My test page
file:///Users/chrismills/Dropbox/Work/git/web-projects/tes
Google
Most Visited
MDN
Mozilla general
FirefoxOS ecosy...
Meeting details
Other Moz depar...
W3C
Mozilla is cool
At Mozilla, we're a global community of
technologists
thinkers
builders
working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.
Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

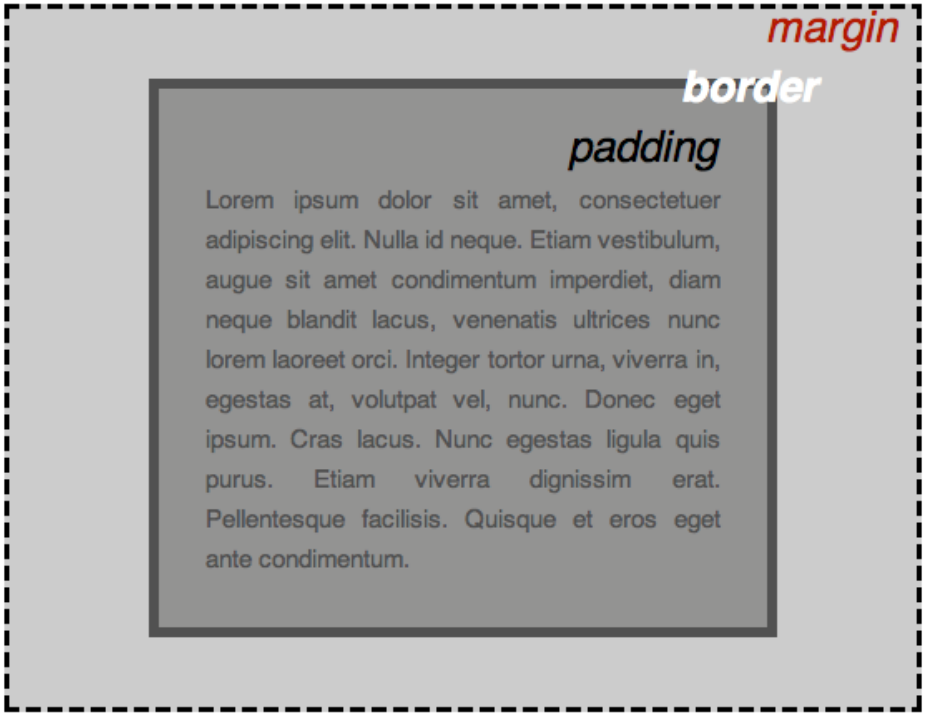
Cajas, cajas, todo se trata de cajas

Una cosa que notarás sobre la escritura de CSS es que trata mucho sobre cajas — ajustando su tamaño, color, posición, etc—. Puedes pensar en la mayoría de los elementos HTML de tu página como cajas apiladas una sobre la otra.

No es de extrañar que el diseño de CSS esté basado principalmente en el *modelo de caja*. Cada una de las cajas que ocupa espacio en tu página tiene propiedades como estas:

- `padding` (relleno), el espacio alrededor del contenido. En el ejemplo siguiente, es el espacio alrededor del texto del párrafo.
- `border` (marco), la línea que se encuentra fuera del relleno.

- `margin` (margen), el espacio fuera del elemento que lo separa de los demás.



En esta sección también se utiliza:

- `width` (ancho del elemento)
- `background-color`, el color de fondo del contenido y del relleno
- `color`, el color del contenido del elemento (generalmente texto)
- `text-shadow`: coloca una sombra difuminada en el texto dentro del elemento
- `display`: selecciona el modo de visualización para el elemento (no te preocupes de esto por ahora)

Bien, ¡continúa y agrega más código CSS a la página! Sigue añadiendo estas reglas nuevas al final de la página, y no temas experimentar cambiando los valores para ver cómo resulta.

Cambiar el color de la página

```
html {

  background-color: #00539F;

}
```

Copy to Clipboard
Esta regla asigna un color de fondo a la página entera. Puedes cambiar el código de color por cualquiera como el que elegiste usar en tu proyecto.

Dar estilo al cuerpo del documento

```
body {

  width: 600px;

  margin: 0 auto;

  background-color: #FF9500;

  padding: 0 20px 20px 20px;

  border: 5px solid black;

}
```

Copy to Clipboard

Ahora tienes varias declaraciones en el elemento body. Revisa una por una:

- `width: 600px;` — esto hará que el cuerpo siempre tenga 600 píxeles de ancho.
- `margin: 0 auto;` — cuando seleccionas dos valores dentro de propiedades como `margin` o `padding`, el primer valor afectará los lados superior (top) **e** inferior (bottom) (en este caso haciéndolo en 0), y el segundo valor los lados izquierdo (left) **y** derecho (right) (aquí, `auto` es un valor especial que divide el espacio disponible entre derecha e izquierda). Puedes usar esta propiedad con uno, dos, tres o cuatro valores como se explica en la sintaxis de padding.
- `background-color: #FF9500;` — como antes, este selecciona el color de fondo de un elemento. Se ha usado un naranja rojizo para el elemento `body` en contraste con el azul oscuro del elemento `<html>`. Sigue y experimenta. Siéntete libre de usar `white` o cualquiera que sea de tu agrado.
- `padding: 0 20px 20px 20px;` — tienes 4 valores puestos en el relleno, para dar un poco de espacio alrededor del contenido. Esta vez no pondrás relleno en la parte de arriba de `body`, 20 píxeles a la izquierda, abajo y derecha. Los valores se ponen: arriba, derecha, abajo e izquierda, en ese orden. Como con `margin` usar esta propiedad con uno, dos, tres o cuatro valores como se explica en la sintaxis de padding.
- `border: 5px solid black;` — este simplemente pone un borde de 5 píxeles de ancho, continuo y de color negro alrededor del elemento `body`.

Posicionar y dar estilo al título principal de la página

```
h1 {

  margin: 0;

  padding: 20px 0;

  color: #00539F;
```

```
  text-shadow: 3px 3px 1px black;

}
```

Copy to Clipboard
Puedes haber notado que hay un hueco horrible en la parte superior de *body*. Esto sucede porque los navegadores vienen con estilos por defecto, ¡incluso cuando aún no se ha aplicado ningún archivo CSS! Esto podría parecer una mala idea, pero se quiere que aun una página sin estilizar sea legible. Para deshacerte de este espacio elimina el estilo por defecto, agregando `margin: 0;`.

Enseguida, se ha puesto un relleno arriba y abajo del título de 20 píxeles, y se hizo que el color del texto sea el mismo que el color de fondo de `html`.

Una propiedad muy interesante que se ha usado aquí es `text-shadow`, que aplica una sombra al texto del elemento. Sus cuatro valores son como sigue:

- El primer valor en píxeles asigna el **desplazamiento horizontal** de la sombra desde el texto —qué tan lejos la mueve a la derecha—. Un valor negativo la moverá a la izquierda.
- El segundo valor en píxeles asigna el **desplazamiento vertical** de la sombra desde el texto —qué tan lejos la mueve hacia abajo—. En este ejemplo, un valor negativo la desplazaría hacia arriba.
- El tercer valor en píxeles asigna **radio de desenfoque** de la sombra —un valor grande es igual a una sombra borrosa—.
- El cuarto valor asigna el color base de la sombra.

Una vez más, trata de experimentar con diferentes valores para ver cómo resulta.

Centrar la imagen

```
img {
```

```
  display: block;

  margin: 0 auto;

}
```

Copy to Clipboard
Finalmente, centra la imagen para hacer que luzca mejor. Puedes usar nuevamente el truco de `margin: 0 auto` que usaste antes para `body`, pero existen diferencias que requieren que hagas algo más para que el código CSS funcione.

El elemento `<body>` es un elemento en nivel de bloque (**block-level**), lo que significa que tomará espacio en la página y que puede tener otros valores de espacio aplicables como margen. Las imágenes, por otra parte, son elementos **inline**, lo que quiere decir que no puedes aplicarles márgenes, debes dar a la imagen un comportamiento de *block-level* usando `display: block;`.

Nota: las instrucciones anteriores asumen que estás usando una imagen más pequeña que el ancho establecido en `body` (600 píxeles). Si tu imagen es más grande, desbordará el cuerpo, derramándose en el resto de la página. Para solucionar esto, puedes hacer lo siguiente: 1) reducir el ancho de la imagen usando un editor gráfico, o 2) usar CSS para dimensionar la imagen estableciendo la propiedad `width` en el elemento `<img>` con un valor menor.
Nota: no te preocupes si aún no entiendes `display: block;` y la diferencia entre un elemento de bloque y un elemento *inline*. Lo entenderás en tanto estudies CSS a profundidad. Puedes encontrar más en cuanto a los diferentes valores disponibles para *display* en la página de referencia de display.

Si has seguido las instrucciones de esta publicación, deberías terminar con una página que luce algo así (también puedes ver nuestra versión aquí):


My test page
file:///Users/chrismills/Dropbox/Work/git/web-projects/test-s
Google
Most Visited
MDN
Mozilla general
FirefoxOS ecosy...
Meeting details
Other Moz depar...
W3C
Mozilla is cool
At Mozilla, we're a global community of
technologists
thinkers
builders
working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.
Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

Si te atoraste, puedes comparar tu trabajo con el código del ejemplo finalizado en GitHub.

Aquí, solo has arañado la superficie de CSS. Si quieres encontrar más, puedes ir a la página de aprendizaje de CSS.

Fuente:
https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics

Actividad 4.- Agrega algunos eventos a tu documento HTML.

JavaScript

JavaScript es el lenguaje de programación que debes usar para añadir características interactivas a tu sitio web, (por ejemplo, juegos, eventos que ocurren cuando los botones son presionados o los datos son introducidos en los formularios, efectos de estilo dinámicos, animación, y mucho más). Este artículo te ayudará a comenzar con este lenguaje extraordinario y te dará una idea de qué es posible hacer con él.

¿Qué es JavaScript realmente?

JavaScript es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web. Fue inventado por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla.

Puedes hacer casi cualquier cosa con JavaScript. Puedes empezar con pequeñas cosas como carruseles, galerías de imágenes, diseños fluctuantes, y respuestas a las pulsaciones

de botones. Con más experiencia, serás capaz de crear juegos, animaciones 2D y gráficos 3D, aplicaciones integradas basadas en bases de datos ¡y mucho más!

JavaScript por sí solo es bastante compacto aunque muy flexible, y los desarrolladores han escrito gran cantidad de herramientas encima del núcleo del lenguaje JavaScript, desbloqueando una gran cantidad de funcionalidad adicional con un mínimo esfuerzo. Esto incluye:

- Interfaces de Programación de Aplicaciones del Navegador (APIs) — APIs construidas dentro de los navegadores que ofrecen funcionalidades como crear dinámicamente contenido HTML y establecer estilos CSS, hasta capturar y manipular un vídeo desde la cámara web del usuario, o generar gráficos 3D y muestras de sonido.
- APIs de terceros, que permiten a los desarrolladores incorporar funcionalidades en sus sitios de otros proveedores de contenidos como Twitter o Facebook.
- Marcos de trabajo y librerías de terceros que puedes aplicar a tu HTML para que puedas construir y publicar rápidamente sitios y aplicaciones.

Ya que se supone que este artículo es solo una introducción ligera a JavaScript, la intención no es confundirte en esta etapa hablando en detalle sobre cuál es la diferencia entre el núcleo del lenguaje JavaScript y las diferentes herramientas listadas arriba. Puedes aprender todo eso en detalle más tarde, en el Área de Aprendizaje en MDN, y en el resto de MDN.

Debajo se presentan algunos aspectos del núcleo del lenguaje y también jugarás con unas pocas características de la API del navegador. ¡Diviértete!

Ejemplo «*¡Hola Mundo!*»

La sección de arriba suena realmente emocionante, y debería serlo. JavaScript es una de las tecnologías web más emocionantes, y cuando comiences a ser bueno en su uso, tus sitios web entrarán en una nueva dimensión de energía y creatividad.

Sin embargo, sentirse cómodo con JavaScript es un poco más difícil que sentirse cómodo con HTML y CSS. Deberás comenzar poco a poco y continuar trabajando en pasos pequeños y consistentes. Para comenzar, mostraremos cómo añadir JavaScript básico a tu página,

creando un «*¡Hola Mundo!*» de ejemplo (el estándar en los ejemplos básicos de programación).

Importante: si no has venido siguiendo el resto de nuestro curso, descarga este código de ejemplo y úsalo como punto de partida.

1. Primero, ve a tu sitio de pruebas y crea una carpeta llamada `scripts`. Luego, dentro de la nueva carpeta de scripts, crea un nuevo archivo llamado `main.js` y guárdalo.
2. A continuación, abre tu archivo `index.html` e introduce el siguiente código en una nueva línea, justo antes de la etiqueta de cierre `</body>`:

```
<script src="scripts/main.js"></script>
```

Copy to Clipboard

3. Esto hace básicamente el mismo trabajo que el elemento `<link>` para CSS: aplica el código JavaScript a la página, para que pueda actuar sobre el HTML (y CSS, o cualquier cosa en la página).
4. Ahora añade el siguiente código al archivo `main.js`:

```
5. const miTitulo = document.querySelector('h1');


miTitulo.textContent = '¡Hola mundo!';
```

Copy to Clipboard

6. Finalmente, asegúrate de que has guardado los archivos HTML y JavaScript, y abre `index.html` en el navegador. Deberías ver algo así:

Nota: la razón por la que has puesto el elemento `<script>` casi al final del documento HTML es porque **el navegador carga el HTML en el orden en que aparece en el archivo**.

Si se cargara primero JavaScript y se supone que debe afectar al HTML que tiene debajo, podría no funcionar, ya que ha sido cargado antes que el HTML sobre el que se supone debe trabajar. Por lo tanto, colocar el JavaScript cerca del final de la página es normalmente la mejor estrategia. Para aprender más sobre enfoques alternativos, mira Estrategias de carga de scripts.

¿Qué ha ocurrido?

El texto del título ha sido cambiado por *¡Hola mundo!* usando JavaScript. Hiciste esto primero usando la función `querySelector()` para obtener una referencia al título y almacenarla en una variable llamada `miTitulo`. Esto es muy similar a lo que hiciste con CSS usando selectores —quieres hacer algo con un elemento, así que tienes que seleccionarlo primero—.

Después de eso, estableciste el valor de la propiedad `textContent` de la variable `miTitulo` (que representa el contenido del título) como *¡Hola mundo!*

Nota: Las dos características que has utilizado en este ejercicio forman parte de la API del Modelo de Objeto de Documento (DOM), que tiene la capacidad de manipular documentos.

Curso intensivo de fundamentos del lenguaje

Ahora se explicarán algunas de las funciones básicas del lenguaje JavaScript para que puedas comprender mejor cómo funciona todo. Mejor aún, estas características son comunes para todos los lenguajes de programación. Si puedes entender esos fundamentos, deberías ser capaz de comenzar a programar en casi cualquier cosa.

Importante: en este artículo, trata de introducir las líneas de código de ejemplo en la consola de tu navegador para ver lo que sucede. Para más detalles sobre consolas JavaScript, mira Descubre las herramientas de desarrollo de los navegadores.

Variables

Las Variables son contenedores en los que puedes almacenar valores. Primero debes declarar la variable con la palabra clave `var` (menos recomendado) o `let`, seguida del nombre que le quieras dar. Se recomienda más el uso de `let` que de `var` (más adelante se profundiza un poco sobre esto):

```
let nombreDeLaVariable;
```

Copy to Clipboard

Nota: todas las líneas en JS deben acabar en punto y coma (;) para indicar que es ahí donde termina la declaración. Si no los incluyes puedes obtener resultados inesperados. Sin embargo, algunas personas creen que es una buena práctica tener punto y coma al final de cada declaración. Hay otras reglas para cuando se debe y no se debe usar punto y coma. Para más detalles, vea Guía del punto y coma en JavaScript (en inglés).
Nota: puedes llamar a una variable con casi cualquier nombre, pero hay algunas restricciones (ver este artículo sobre las reglas existentes). Si no estás seguro, puedes comprobar el nombre de la variable para ver si es válido.
Nota: JavaScript distingue entre mayúsculas y minúsculas. `miVariable` es una variable distinta a `mivariable`. Si estás teniendo problemas en tu código, revisa las mayúsculas y minúsculas.
Nota: para más detalles sobre la diferencia entre `var` y `let`, vea Diferencia entre var y let.
Tras declarar una variable, puedes asignarle un valor:

```
nombreDeLaVariable = 'Bob';
```

Copy to Clipboard
Puedes hacer las dos cosas en la misma línea si lo necesitas:

```
let nombreDeLaVariable = 'Bob';
```

Copy to Clipboard
Puedes obtener el valor de la variable llamándola por su nombre:

```
nombreDeLaVariable;
```

Copy to Clipboard
Después de haberle dado un valor a la variable, puedes volver a cambiarlo:

```
let nombreDeLaVariable = 'Bob';

nombreDeLaVariable = 'Steve';
```

Copy to Clipboard
Advierte que las variables tienen distintos tipos de datos:

Variable	Explicación	Ejemplo
String	Esto es una secuencia de texo conocida como cadena. Para indicar que la variable es una cadena, debes escribirlo entre comillas.	`let miVariable = 'Bob';`
Number	Esto es un número. Los números no tienen comillas.	`let miVariable = 10;`

Boolean	Tienen valor verdadero/falso. `true`/`false` son palabras especiales en JS, y no necesitan comillas.	`let miVariable = true;`
Array	Una estructura que te permite almacenar varios valores en una sola referencia.	`let miVariable = [1,'Bob','Steve',10];` Llama a cada miembro del array así: `miVariable[0]`, `miVariable[1]`, etc.
Object	Básicamente cualquier cosa. Todo en JavaScript es un objeto y puede ser almacenado en una variable. Mantén esto en mente mientras aprendes.	`let miVariable = document.querySelector('h1');` Todos los ejemplos anteriores también.

Entonces, ¿para qué necesitamos las variables? Las variables son necesarias para hacer cualquier cosa interesante en programación. Si los valores no pudieran cambiar, entonces no podrías hacer nada dinámico, como personalizar un mensaje de bienvenida de un usuario que visita tu página, cambiar la imagen que se muestra en una galería de imágenes, etc.

Comentarios

Puedes escribir comentarios entre el código JavaScript, igual que puedes en CSS. El navegador ignora el texto marcado como comentario. En JavaScript, los comentarios de una sola línea se escriben así:

```
// Esto es un comentario
```

Copy to Clipboard
Pero también puedes escribir comentarios en más de una línea, igual que en CSS:

```
/*

Esto es un comentario

de varias líneas.

*/
```

Copy to Clipboard

Operadores

Un operador es básicamente un símbolo matemático que puede actuar sobre dos valores (o variables) y producir un resultado. En la tabla de abajo aparecen los operadores más simples, con algunos ejemplos para probarlos en la consola del navegador.

Operador	Explicación	Símbolo(s)	Ejemplo
Suma/concatena	Se usa para sumar dos números, o juntar dos cadenas en una.	`+`	`6 + 9;` `"Hola " + "mundo!";`

Resta, multiplicación, división	Estos hacen lo que esperarías que hicieran en las matemáticas básicas.	`-`, `*`, `/`	`9 - 3;`
			`8 * 2; // La multiplicación en JS es un asterisco`
			`9 / 3;`

Operador de asignación	Los has visto anteriormente: asigna un valor a una variable.	`=`	`let miVariable = 'Bob';`

identidad/igualdad	Comprueba si dos valores son iguales entre sí, y devuelve un valor de `true`/`false` (booleano).	`===`	`let miVariable = 3;` `miVariable === 4;`

Negación, distinto (no igual)	En ocasiones utilizado con el operador de identidad, la negación es en JS el equivalente al operador lógico NOT — cambia `true` por `false` y viceversa.	`!`, `!==`	La expresión básica es `true`, pero la comparación devuelve `false` porque lo hemos negado:
			
			`let miVariable = 3;`
			`!miVariable === 3;`
			
			Aquí estamos comprobando "`miVariable` NO es igual a 3". Esto devuelve `false`, porque `miVariable` ES igual a 3.
			
			`let miVariable = 3;`
			`miVariable !== 3;`

Hay muchos operadores por explorar, pero con esto será suficiente por ahora. Mira Expresiones y operadores para ver la lista completa.

Nota: mezclar tipos de datos puede dar lugar a resultados extraños cuando se hacen cálculos, así que asegúrate de que relacionas tus variables correctamente y de que recibes

los resultados que esperabas. Por ejemplo, teclea: `"3" + "25"` en tu consola. ¿Por qué no obtienes lo que esperabas? Porque las comillas convierten los números en "strings" (el término inglés para denominar cadenas de caracteres) y de este modo has acabado con los "strings" concatenados entre sí, y no con los números sumados. Si tecleas: `35 + 25`, obtendrás el resultado correcto.

Condicionales

Las condicionales son estructuras de código que permiten comprobar si una expresión devuelve *true* o no, y después ejecuta un código diferente dependiendo del resultado. La forma de condicional más común es la llamada `if... else`. Entonces, por ejemplo:

```
let helado = 'chocolate';

if (helado === 'chocolate') {

  alert('¡Sí, amo el helado de chocolate!');

} else {

  alert('Awwww, pero mi favorito es el de chocolate...');

}
```

Copy to Clipboard

La expresión dentro de `if (... )` es el criterio — este usa al operador de identidad (descrito arriba) para comparar la variable `helado` con la cadena `chocolate` para ver si las dos son iguales. Si esta comparación devuelve `true`, el primer bloque de código se ejecuta. Si no, ese código se omite y se ejecuta el segundo bloque de código después de la declaración `else`.

Funciones

Las funciones son una manera de encapsular una funcionalidad que quieres reutilizar, de manera que puedes llamar esa función con un solo nombre, y no tendrás que escribir el código entero cada vez que la utilices. Ya has visto algunas funciones más arriba, por ejemplo:

```
1. let nombreDeLaVariable = document.querySelector('h1');
```

Copy to Clipboard

```
2. alert('¡Hola!');
```

Copy to Clipboard

Estas funciones `document.querySelector` y `alert` están integradas en el navegador para poder utilizarlas en cualquier momento.

Si ves algo que parece un nombre de variable, pero tiene paréntesis —`()`— al final, probablemente es una función. Las funciones con frecuencia toman argumentos —pedazos de datos que necesitan para hacer su trabajo—. Estos se colocan dentro de los paréntesis, y se separan con comas si hay más de uno.

Por ejemplo, la función `alert()` hace aparecer una ventana emergente dentro de la ventana del navegador, pero necesitas asignarle una cadena como argumento para decirle qué mensaje se debe escribir en la ventana emergente.

Las buenas noticias son que podemos definir nuestras propias funciones —en el siguiente ejemplo escribimos una función simple que toma dos números como argumentos y los multiplica entre sí—:

```
function multiplica(num1,num2) {

  let resultado = num1 * num2;

  return resultado;
```

```
}
```

Copy to Clipboard
Trata de ejecutar la función anterior en la consola. Después trata de usar la nueva función algunas veces, p.ej:

```
multiplica(4, 7);

multiplica(20, 20);

multiplica(0.5, 3);
```

Copy to Clipboard
Nota: la sentencia `return` le dice al navegador que devuelva la variable `resultado` fuera de la función, para que esté disponible para su uso. Esto es necesario porque las variables definidas dentro de funciones, solo están disponibles dentro de esas funciones. Esto se conoce como «ámbito (*scope* en inglés) de la variable». Lee más sobre ámbito o alcance de la variable.

Eventos

Para crear una interacción real en tu sitio web, debes usar eventos. Estos son unas estructuras de código que captan lo que sucede en el navegador, y permite que en respuesta a las acciones que suceden se ejecute un código. El ejemplo más obvio es un clic (click event), que se activa al hacer clic sobre algo. Para demostrar esto, prueba ingresando lo siguiente en tu consola, luego da clic sobre la página actual:

```
document.querySelector('html').onclick = function() {

    alert('¡Ouch! ¡Deja de pincharme!');

}
```

Copy to Clipboard

Hay muchas maneras de enlazar un evento a un elemento; aquí hemos seleccionado el elemento `<html>` y le asignamos a su propiedad `onclick` una función anónima (función sin nombre) que contiene el código que se ejecutará cuando el evento suceda.

Nota que

```
document.querySelector('html').onclick = function(){};
```

Copy to Clipboard
es equivalente a

```
let miHTML = document.querySelector('html');

miHTML.onclick = function(){};
```

Copy to Clipboard
es solo un modo más corto de escribirlo.

Sobrecargar tu sitio web de ejemplo

Ahora vas a repasar un poco lo básico de JavaScript. Añadirás un par de funcionalidades a tu sitio para demostrar lo que puedes hacer.

Añadir un cambiador de imagen

En esta sección añadirás otra imagen a tu sitio usando la DOM API y agregarás un poco de código para cambiar entre imágenes al hacer clic.

1. Primero que todo, busca una imagen que te guste para tu sitio. Asegúrate que sea del mismo tamaño que la primera, o lo más cerca posible.
2. Guarda tu imagen en tu carpeta `images`.
3. Renombra esta imagen «firefox2.png» (sin las comillas).

4. Ve a tu archivo `main.js` y agrega el siguiente JavaScript (si tu JavaScript de *«Hola Mundo»* está aún allí, bórralo).

```
let miImage = document.querySelector('img');

miImage.onclick = function () {
    let miSrc = miImage.getAttribute('src');
    if (miSrc === 'images/firefox-icon.png') {
      miImage.setAttribute('src','images/firefox2.png');
    } else {
      miImage.setAttribute('src', 'images/firefox-icon.png');
    }

}
```

Copy to Clipboard

13. Guarda todos los archivos y carga `index.html` en tu navegador. Ahora cuando hagas clic en la imagen, ¡esta debe cambiar por otra!

Esto fue lo que sucedió: se almacena una referencia a tu elemento `<img>` en la variable `miImage`. Luego, haces que esta propiedad del manejador de evento `onclick` de la variable sea igual a una función sin nombre (una función «anónima»). Ahora, cada vez que se haga clic en la imagen:

1. El código recupera el valor del atributo `src` de la imagen.
2. El código usa una condicional para comprobar si el valor `src` es igual a la ruta de la imagen original:
    1. Si es así, el código cambia el valor de `src` a la ruta de la segunda imagen, forzando a que se cargue la otra imagen en el elemento `<img>`.

2. Si no es así (significa que ya fue modificada), se cambiará el valor de `src` nuevamente a la ruta de la imagen original, regresando a como era en un principio.

Añadir un mensaje de bienvenida personalizado

Ahora añadirás un poco más de código, para cambiar el título de la página o incluir un mensaje personalizado de bienvenida para cuando el usuario ingrese por primera vez. Este mensaje de bienvenida permanecerá luego de que el usuario abandone la página y estará disponible para cuando regrese. Lo guardarás usando Web Storage API. También se incluirá una opción para cambiar el usuario y por lo tanto también el mensaje de bienvenida en cualquier momento que se requiera.

1. En `index.html`, agrega el siguiente código antes del elemento `<script>`:

```
<button>Cambiar de usuario</button>
```

Copy to Clipboard

2. En `main.js`, agrega el siguiente código al final del archivo, exactamente como está escrito. Esto toma referencia al nuevo botón que se agregó y al título y los almacena en variables:

```
let miBoton = document.querySelector('button');

let miTitulo = document.querySelector( 'h1');
```

Copy to Clipboard

4. Ahora agrega la siguiente función para poner el saludo personalizado, lo que no causará nada aún, pero arreglarás esto en un momento:

```
function estableceNombreUsuario() {

    let miNombre = prompt('Por favor, ingresa tu nombre.');

    localStorage.setItem('nombre', miNombre);

    miTitulo.textContent = 'Mozilla es genial,' + miNombre;
```

```
}
```

Copy to Clipboard

La función `estableceNombreUsuario()` contiene una función `prompt()`, que crea un cuadro de diálogo como lo hace `alert()`; la diferencia es que `prompt()` pide al usuario un dato, y almacena este dato en una variable cuando el botón **Aceptar** del cuadro de diálogo es presionado. En este caso, pedirás al usuario que ingrese su nombre. Luego, llamarás la API `localStorage`, que nos permite almacenar datos en el navegador y recuperarlos luego. Usarás la función `setItem()` de localStorage, que crea y almacena un dato en el elemento llamado `'nombre'`, y coloca este valor en la variable `miNombre` que contiene el nombre que el usuario ingresó. Finalmente, establecerás el `textContent` del título a una cadena, más el nombre de usuario recientemente almacenado.

9. Luego, agregarás este bloque `if ... else`. Se podría llamar a esto el código de inicialización, como se ha establecido para cuando carga la app por primera vez:

```
10. if (!localStorage.getItem('nombre')) {
11.     estableceNombreUsuario();
12. }
13. else {
14.     let nombreAlmacenado = localStorage.getItem('nombre');
15.     miTitulo.textContent = 'Mozilla es genial,' + nombreAlmacenado;
}
```

Copy to Clipboard

La primera línea de este bloque usa el operador de negación (NO lógico representado por `!`) para comprobar si el elemento `'nombre'` existe. Si no existe, la función `estableceNombreUsuario()` se iniciará para crearlo. Si ya existe (como por ejemplo cuando el usuario ya ingresó al sitio), se recupera el dato del nombre

usando `getItem()` y se fija mediante `textContent` del título a la cadena, más el nombre del usuario, como hiciste dentro de `estableceNombreUsuario()`.

16. Finalmente, agrega abajo el evento `onclick` que manipulará el botón, de modo que cuando sea pulsado se inicie la función `estableceNombreUsuario()`. Esto permitirá al usuario establecer un nuevo nombre cada vez que lo desee al pulsar el botón:

```
17. miBoton.onclick = function() {

18.     estableceNombreUsuario();

    }
```

Copy to Clipboard

Ahora cuando visites tu sitio por primera vez, este te pedirá tu nombre y te dará un mensaje personalizado de bienvenida. Puedes cambiar cuantas veces quieras el nombre al presionar el botón. Y como un bonus añadido, ya que el nombre se almacena en el localStorage, este permanecerá después de que cierre el sitio, ¡manteniendo ahí el mensaje personalizado cuando abras el sitio la próxima vez!

¿Un nombre de usuario nulo?

Cuando ejecutes el ejemplo y obtengas el cuadro de diálogo que solicita que introduzcas tu nombre de usuario, intenta pulsar el botón *Cancelar*. Deberías terminar con un título que diga que *Mozilla es genial, null*. Esto sucede porque, cuando cancelas el mensaje, el valor se establece como `null`. Null (nulo) es un valor especial en JavaScript que se refiere a la ausencia de un valor.

Además, prueba a dar clic en *Aceptar* sin introducir un nombre. Deberías terminar con un título que diga que *Mozilla es genial*, por razones bastante obvias.

Para evitar estos problemas, podrías comprobar que el usuario no ha introducido un nombre en blanco. Actualiza tu función `estableceNombreUsuario()` a lo siguiente:

```
function estableceNombreUsuario() {
```

```
  let miNombre = prompt('Introduzca su nombre.');

  if(!miNombre) {

    estableceNombreUsuario();

  } else {

    localStorage.setItem('nombre', miNombre);

    miTitulo.innerHTML = 'Mozilla is genial, ' + miNombre;

  }

}
```

Copy to Clipboard
En el lenguaje humano, esto significa que si `miNombre` no tiene ningún valor, ejecute `estableceNombreUsuario()` de nuevo desde el principio. Si tiene un valor (si la afirmación anterior no es verdadera), entonces almacene el valor en `localStorage` y establézcalo como el texto del título.

Si has seguido las instrucciones en este artículo, tendrás una página que luzca como esta (también puede ver nuestra versión aquí):


My test page
file:///Users/chrismills/Dro
Yahoo
Most Visited
MDN
Mozilla general
FirefoxOS ecosy...
Mozilla is cool, Chris
At Mozilla, we're a global community of
technologists
thinkers
builders
working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.
Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.
Change user

	Si tuviste problemas, siempre puedes comparar su trabajo con el código terminado del ejemplo en GitHub. Aquí solo has rozado la superficie de JavaScript. Si has disfrutado aprendiendo y deseas avanzar más, visita la Guía de JavaScript. **Actividad 5.-** Instalación de software XAMMP. Instala el software XAMMP en la computadora, puedes ayudarte de un video tutorial https://www.youtube.com/watch?v=hlzaA_GSA8U **Actividad 6.-** Elabora una base de datos en MySql de XAMMP y posteriormente codifica en lenguaje php los siguientes códigos.	

ARCHIVO: altas.php

```
<html>
<head>
<title>alumnos</title>
</head>
<body>
<h1>Alta de Alumnos</h1>
<form action="altas2.php" method="post">

Ingrese nombre:
<input type="text" name="nombre"><br><br>
Ingrese direccion:
<input type="text" name="direccion"><br><br>
Ingrese telefono:
<input type="text" name="telefono"><br><br>
<br><br>
<input type="submit" value="Registrar">
</form>
</body>
</html>
```

ARCHIVO: altas2.php

```
<html>
<head>
<title>insertar base de datos</title>
</head>
<body>
<?php
$conexion=mysql_connect("localhost","root","12345") or die("Problemas en la conexion");

mysql_select_db("uno",$conexion) or die("Problemas en la seleccion de la base de datos");

mysql_query("insert into alumnos(nombre,direccion,telefono) values
  ('$_REQUEST[nombre]','$_REQUEST[direccion]','$_REQUEST[telefono]')",
  $conexion) or die("Problemas en el select".mysql_error());

mysql_close($conexion);

echo "El alumno fue dado de alta.";
?>
</body>
</html>
```

Actividad 7.- Codifica en lenguaje php los siguientes códigos.

ARCHIVO: bajas.php

```
<html>
<head>
<title>bajas</title>
</head>
<body>
<form action="borrar2.php" method="post">
Ingrese el nombre del alumno a borrar:
<input type="text" name="nombre">
<br>
<input type="submit" value="borrar">
</form>
</body>
</html>
```

ARCHIVO: borrar2.php

```
<html>
<head>
<title>BAJAS</title>
</head>
<body>
<?php
$conexion=mysql_connect("localhost","root","12345") or  die("Problemas en la conexion");
mysql_select_db("uno",$conexion) or die("Problemas en la selección de la base de datos");
$registros=mysql_query("select nombre from alumnos where
nombre='$_REQUEST[nombre]'",$conexion) or
 die("Problemas en el select:".mysql_error());
if ($reg=mysql_fetch_array($registros))
{
 mysql_query("delete from alumnos where nombre='$_REQUEST[nombre]'",$conexion) or
  die("Problemas en el select:".mysql_error());
 echo "Se efectuó el borrado del alumno con dicho nombre.";
}
else
{
 echo "No existe un alumno con ese nombre.";
}
mysql_close($conexion);
?>
</body>
</html>
```

Actividad 8.- Codifica en lenguaje php los siguientes códigos.

.

ARCHIVO:consulta.php

```
<html>
<head>
<title>consultas</title>
</head>
<body>
<form action="consultas2.php" method="post">
Ingrese el nombre del alumno a consultar:
<input type="text" name="nombre">
<br>
<input type="submit" value="consultar">
</form>
</body>
</html>
```

ARCHIVO: consultas2.php

```
<html>
<head>
<title>consulta2</title>
</head>
<body>
<?php
$conexion=mysql_connect("localhost","root","12345") or die("Problemas en la conexion");

mysql_select_db("uno",$conexion) or die("Problemas en la selección de la base de datos");

$registros=mysql_query("select nombre, direccion, telefono from alumnos where
nombre='$_REQUEST[nombre]'",$conexion) or
 die("Problemas en el select:".mysql_error());
if ($reg=mysql_fetch_array($registros))
{
 echo "Nombre:".$reg['nombre']."<br>";
 echo "direccion:".$reg['direccion']."<br>";
 echo "telefono".$reg['telefono']."<br>";
 }
else
{
 echo "No existe un alumno con ese nombre.";
}
mysql_close($conexion);
?>
</body>
</html>
```

Actividad 9.- Codifica en lenguaje php los siguientes códigos.

ARCHIVO: editar2.php

```
<HTML>
<FORM ACTION=editar2.php METHOD=post>
DAME EL NOMBRE A EDITAR:<INPUT TYPE=text NAME=NOMBRE><BR>
<INPUT TYPE=submit NAME=OK VALUE="BUSCAR"><BR>
</FORM>
</HTML>
<?php


if ($OK == "BUSCAR") {
// conexion al servidor de bases de datos
$dbh=mysql_connect ("localhost", "root", "12345") or die ('problema conectando porque :' .
mysql_error());
// seleccionado la base de datos
mysql_select_db ("uno",$dbh);
// preparando la instruccion sql

$q = "Select * from alumnos where nombre= '$NOMBRE'";
```

```
// ejecutando el query select regresa un rowset
$alumnos = mysql_query("$q", $dbh) or die ("problemon con query") ;
// regresando renglon con registro
if($reg = mysql_fetch_row($alumnos))
{
// construyendo forma dinamica
echo "<FORM ACTION=editar2.php METHOD=post>";
// recordar que strings se encadenan con .
echo "NOMBRE:<INPUT TYPE=text NAME=NOMBRE value= \"".$reg[0]."\"><BR>";
echo "DIRECCION:<INPUT TYPE=text NAME=DIRECCION value=\"".$reg[1]."\"><BR>";
echo "TELEFONO:<INPUT TYPE=text NAME=TELEFONO value=\"".$reg[2]."\"><BR>";
//echo "<input type=hidden name=NOMBRE value=$reg[0]>";
echo "<INPUT TYPE=submit NAME=OK VALUE=editar><BR>";
echo "</FORM>";
}
else
{
echo "No existe un alumno con ese nombre.";
}
}
if ($OK == "editar")
{
// conexion al servidor de bases de datos
```

```
$dbh=mysql_connect ("localhost", "root", "12345") or die ('problema conectando porque :' .
mysql_error());
// seleccionado la base de datos
mysql_select_db ("uno",$dbh);
// preparando la instruccion sql
$q = "UPDATE alumnos set nombre='$NOMBRE', direccion='$DIRECCION', telefono='$TELEFONO'
where nombre='$NOMBRE'";
// ejecutando el query
mysql_query("$q", $dbh) or die ("problemita con query");
// avisando
echo "REGISTRO EDITADO";
}


?>
```

Actividad 10.- Codifica en lenguaje php los siguientes códigos.

ARCHIVO: listado1.php

```
<?
//creamos el link de conexion a nuestro servidor
$link=mysql_connect("localhost","root","12345");
//seleccionamos la base de datos
mysql_select_db("uno",$link);
 //ejecutamos la consulta a la base de datos para extraer los registros
$rows=mysql_query("select * from alumnos ORDER BY nombre");
?>
<table border="2" CELLSPACING=1 CELLPADDING=1>
  <tr>
    <th> NOMBRE </th>
    <th> DIRECCION </th>
    <th> TELEFONO </th
  </tr>
<?
//esta parte es opcional.
//aqui solo extraemos el total de registros que nos devolvio nuestra consulta
$totalregistros=mysql_num_rows($rows);
 //iniciamos el recorrido de nuestros registros
while($row=mysql_fetch_array($rows)){
?>
```

Sustento teórico de PHP

PHP

El lenguaje PHP (cuyo nombre es acrónimo de PHP: Hipertext Preprocessor) es un lenguaje interpretado con una sintaxis similar a la de C++ o JAVA. Aunque
el lenguaje se puede usar para realizar cualquier tipo de programa, es en la generación dinámica de páginas web donde ha alcanzado su máxima popularidad.
En concreto, suele incluirse incrustado en páginas HTML (o XHTML), siendo el servidor web el encargado de ejecutarlo.

1. Sintaxis básica

XHTML (Extensible Hypertext Markup Language) es un lenguaje de etiquetas. Es el sucesor de HTML y se basa en la sintaxis de XML. Asegura la compatibilidad
tanto en equipos clásicos como en smartphones.
Ya conoce las etiquetas <html>, <body>, <head>...
Escriba PHP entre dos etiquetas. Se definen de la siguiente manera:
<?php: indica el comienzo del código PHP
?>: indica el final del código PHP

Una instrucción siempre termina con un punto y coma.
Ejemplo

```
<?php
echo ’<p>Hola!</p>’;
?>
```

También puede escribir este código en una sola línea:

```
<?php echo ’<p>Hola!</p>’; ?>
```

2. Inserción de etiquetas PHP en el código XHTML

Puedes insertar un código PHP en cualquier ubicación del código XHTML.

```
<html>
<head>
<title>Ejemplo de página PHP</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
Hola, hace <?php echo ’como estas’; ?>
```

</body>
</html>

3**. Envío de datos al servidor Web**

Existen varias instrucciones para enviar datos al servidor, es decir, para insertar código HTML en una página Web.

La primera instrucción es echo y se escribe de la siguiente manera:

<?php echo ’texto’; ?>

También puede escribir este código:

<?php echo "texto"; ?>

O bien:

<?php echo(’texto’); ?>

La segunda instrucción es print y se escribe de la siguiente manera:

<?php print(’texto’); ?>

Por tanto, print equivale a echo.

Existen otras variantes de print:

printf(): muestra una cadena de caracteres formateada.

sprintf(): devuelve una cadena formateada.

vprintf(): muestra una cadena formateada.

sscanf(): analiza una cadena con ayuda de un formato

fscanf(): analiza un archivo en función del formato.

flush(): vacía los búferes de salida.

También puede escribir varias instrucciones en la misma línea, siempre y cuando vayan separadas por punto y coma.

<?php echo ’texto’; ?> equivale a <?php echo ’tex’; echo ’to’; ?> y a

<?php echo ’tex’;
echo ’to’;
?>

4. Inserción del código XHTML con la instrucción echo

La función echo permite insertar cualquier código HTML, por ejemplo:

<?php echo ’<table><tr><td>texto</td></tr></table>’; ?>

Y como resultado inserta una tabla HTML.
También puede insertar una imagen de la siguiente manera:

```
<?php echo ’<img src="hoja.jpg" name="hoja" />’; ?>
```

Por tanto, puede escribir una página Web completa con la instrucción echo.

```
<?php
echo ’<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//ES" ,
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">’;
echo ’<html xmlns="http://www.w3.org/1999/xhtml">’;
echo ’<head>’;
echo ’<title>PHP </title>’;
echo ’</head>’;
echo ’<body>’;
echo ’<p>’;
echo "Hola $nombre !<br />";
echo ’La fecha es 20 de Enero del 2021 .’; //
echo ’</p>’;
echo ’</body>’;
echo ’</html>’;
?>
```

Las variables

1. Asignación

Una variable es una información que se almacena temporalmente en la memoria, es decir, es una zona de la memoria que almacena información en una página
PHP y que se destruye automáticamente cuando la página ya no se ejecuta.

Una variable PHP comienza siempre con $, seguida de una letra y de una secuencia de letras, cifras o del signo _.
Por ejemplo, $edad.

Atención: PHP distingue entre mayúsculas y minúsculas, por lo que $nombre es distinto de $Nombre.
Una variable siempre tiene un nombre y un valor.
Por ejemplo, $edad = 25, el valor 25 se asigna a la variable $edad gracias al signo =.
No es necesario definir y buscar el tipo de variable. Se hace automáticamente.

```
$dia = 24; //Se declara una variable de tipo integer.
$sueldo = 758.43; //Se declara una variable de tipo double.
$nombre = "juan"; //Se declara una variable de tipo string (cadena).
```

$salida = true; //Se declara una variable boolean.
De este modo, puede escribir:
<?php
$edad = 25; //variable de tipo numérico
//después
$edad = ’25’; //variable de tipo texto

?>
1. **Operadores aritméticos**
+ Suma dos valores
- Resta dos valores (o pasa a negativo un valor)
* Multiplica dos valores
/ Divide dos valores
% Resto de dividir dos valores
2. Tipos de variables
Hay dos categorías de variables:
Escalar:
Los números enteros llamados integer son 1, 2, 3... y los números negativos, -1, -2, -3...
Los números decimales llamados float son los números positivos o negativos con comas (1.35665).
Atención: el punto se utiliza como separador.
La cadena de caracteres string: cualquiera con dobles comillas ("hola") o comillas simples
(’hola’).
Los booleanos: solo tienen dos tipos de valores: verdadero o falso, clasificados
Como true o false.
Compuesta:
3. La concatenación
Es un conjunto de cadena de caracteres. PHP permite la concatenación usando la coma o el punto.

<?php
echo ’hola ’.’lee esta ayuda’;
?>
Equivale a:
<?php
echo ’hola ’,lee esta ayuda’;
?>
Da como resultado:

Hola lee esta ayuda
Si quiere concatenar la cadena "hola" y "aquí hay un apóstrofo ’", no podrá escribir:
<?php
echo ’hola ’.’aquí hay un apóstrofo ’’;
?> marcaría error.
Ejercicios
1.- Diseña un código donde muestre tu nombre completo.
2.- Diseña un código que indique los siguientes datos : calle, numero de calle, colonia, municipio y teléfono.
3.- Diseña un código para almacenas 2 números y realice una suma.

ESTRUCTURAS DE CONTROL

1. Operadores de comparación(relacionales)

==	**Comprueba si dos números son iguales**
!=	**Comprueba si dos números son distintos**
>	**Mayor que, devuelve true en caso afirmativo**
<	Menor que, devuelve true en caso afirmativo

>=	Mayor o igual
<=	Menor o igual

1. Operadores lógicos

!	**Operador NO o negación. Si era true pasa a false y viceversa**
And	Operador Y, si ambos son verdaderos vale verdadero
or	Operador O, vale verdadero si alguno de los dos es verdadero
xor	Verdadero si alguno de los dos es true pero nunca ambos
&&	True si ambos lo son
||	**True si alguno lo es**

Cuando se pretende que el script tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, se utiliza
el conjunto de instrucciones: if, **else** y **elseif**.
La estructura base es la siguiente:
if (Condición) {

Instrucción 1;
Instrucción 2;
}
else {
Instrucción a;
Instrucción b;
}
EJEMPLO.
If
<?php
$tmp = 1;
$variable = $tmp>5 ? 'Es mayor a 5' : No es mayor a 5';
echo $variable;
?>

<?php
$nombre = ’Miguel’; //declaración de la variable $nombre
if ($nombre == ’Velez’) //comprueba la variable $nombre
{
echo ’Bienvenido’;
}
else
{
echo ’denegado’;
}
?>

Resultado es igual a “denegado”
<?php
$edad = 18; //declaración de la variable $edad
if ($edad>= 18) //comprueba la variable $edad
{
echo ’Eres mayor de edad ya puedes votar’;
}
else
{

	echo ’Eres menor de edad’;
	}
	?>
	
	Resultado es igual a “ Eres mayo de edad ya puedes votar”
	Condicionales compuestas if elseif
	if (Condicion 1)
	{
	Instrucción a;
	}
	elseif (Condicion 2)
	{
	Instrucción b;
	}
	else
	{
	Instrucción c;
	}
	<?php
	$nombre = ’Miguel’; //declaración de la variable $nombre
	if ($nombre == ’velez’) //comprueba la variable $nombre
	{
	echo ’Bienvenido’;
	}
	else if ($nombre == ’Miguel’) //comprueba la variable $nombre
	{
	echo ’Hola’;
	}
	else
	{
	echo ’Denegado’;
	}
	?>
	Resultado es igual a “Hola”	

Sentencia Switch

1. Un switch busca dentro de los “case” el valor de la variable o expresión evaluada (generalmente se evalúan variables).
2. Si lo encuentra, ejecuta el código correspondiente.
3. Si no lo encuentra, ejecuta el código por default.
4. Observaciones: El case por default es opcional.
5. Es importante poner “break;” al final de cada bloque de código dentro de cada case para que el switch no siga comparando al valor de la variable con los
case que le siguen al correcto.

Sintaxis:

```
switch(expresión) {
case "a": //Código a ejecutar;
break;
case "b": //Código a ejecutar;
break;
case "c": //Código a ejecutar;
break;
default: //Código a ejecutar por default.
}
```

Ejemplo

```
<?php
$nombre = ’miguel’; //declaración de la variable $nombre
switch ($nombre) //comprueba la variable $nombre
{
case ’miguel’:
echo ’Hola’;
break;
case ’Juan’:
echo ’Hasta pronto’;
break;
}
?>
```

Resultado = Hola

La instrucción `break` provoca la salida del `switch` y si `$nombre` es igual a "miguel" el código ejecutará `echo "Hola"` y `break`, y saldrá

del `switch` sin comprobar "Juan".

```
<?php
$edad = 25; //declaración de la variable $edad
switch ($edad) //comprueba la variable $edad
{
case 20:
echo "Tiene 25 años.";
break;
case 25:
echo "Tiene 25 años.";
break;
default:
echo "No tiene 20 ni 25 años.";
}
?>
```

Resultado = tiene 25 años

Los bucles

1. For

Un bucle permite repetir n veces la ejecución de un código.

Por ejemplo, si quiere mostrar diez veces «Hola», solo tiene que escribir el bucle for.

```
<?php
for ($i = 1; $i <= 10; $i++)
{
echo 'Hola <br />';
}
?>
```

La variable $i representa el contador del bucle.

Por tanto, la sintaxis es:

```
for ($i=número inicial; $i <= número final; aumento)
{
instrucciones
}
```

$i++ es igual a $i=$i+1 y representa el aumento de $i. Puede escribir $i=$i+2 para aumeto $i=$i-1 para disminuir.

Por ejemplo, puede escribir los números de 100 a 150 con el siguiente código:

```
<?php
```

for ($i = 100; $i <= 150; $i++)
{
echo $i.'
';
}
?>
La instrucción echo $i.'
'; se repite 50 veces y $i aumenta en 1 cada vez.

 permite saltar una línea entre cada número para no tener que mostrarlos todos.
La instrucción break permite detener el bucle.
Por ejemplo, si quiere mostrar cinco veces «Hola», solo debe escribir un bucle for:

```
<?php
for ($i = 1; $i <= 10; $i++)
{
echo 'Hola <br />';
if ($i == 5) {
break;
}
}
?>
```

Da como resultado:
Hola
Hola
Hola
Hola
Hola
El bucle se detiene cuando `$i` es igual a 5 (y no a 10).

2. While
El bucle while significa «mientras que», es decir, el bucle se ejecutará siempre y cuando una condición sea verdadera.
Por ejemplo, para mostrar diez veces «Hola», solo debe escribir un bucle while:

```
<?php
$i = 1;
while ($i <= 10)
{
$i=$i+1;
echo 'Hola <br />';
}
?>
```

La variable $i representa el contador del bucle. Pero mientras $i sea inferior o igual a 10, se repetirá el bucle. Por lo tanto, la sintaxis es:

```
$i=número inicial
while ($i <= número final)
{
aumento
instrucciones
}
```

No olvide poner el aumento de $i en las instrucciones de while; de lo contrario $i nunca valdrá 10 y tendrá un bucle infinito.

Considera que el valor de salida de $i se pone antes del bucle y que este valor debe respetar la condición del bucle ($i <= número final) para entrar en el bucle.

Si escribe:

```
<?php
$i = 11;
while ($i <= 10)
{
$i=$i+1;
echo 'Hola <br />';
}
?>
```

Nunca pasará en el bucle porque $i vale 11 en un principio, no se satisface la condición del bucle.

El bucle while es igual al bucle for; en algunas ocasiones le resultará muy útil si desconoce el número de veces que va a ejecutar un bucle, sobre todo si va a leer el bucle while en la base de datos y la condición de salida del bucle depende del valor leído en la base de datos.

3. Do while

El bucle Do while significa «hacer mientras», es decir, el bucle se ejecutará siempre y cuando una condición sea verdadera. Se diferencia del bucle while en que la expresión se ejecuta al menos una vez.

Por ejemplo, para mostrar diez veces "Hola", debe escribir el bucle Do while:

```
<?php
$i = 1;
do
```

{
$i=$i+1;
echo 'Hola
';
} while ($i <= 10)
?>

La variable $i representa el contador del bucle. Pero esta vez debe leer: ejecutar el bucle si $i es nferior o igual a 10.
Por tanto, la sintaxis es:
$i=número inicial
do
{
aumento
instrucciones
} while ($i <= número final)

```
<?php
$i = 1;
do
{
$i=$i+1;
echo 'Hola <br />';
} while ($i <= 10)
?>
```