

The background features a decorative graphic consisting of three concentric blue circles of varying sizes, each with a lighter blue outer ring. These circles are positioned in the upper right and lower right areas. Two thin, light blue diagonal lines cross the page, one from the top left to the bottom right, and another from the top right to the bottom left, intersecting near the center.

Introducción al Diseño de Interfaces Web

Una guía didáctica

Asignatura: Programación de Aplicaciones en Ingeniería

El presente texto muestra una serie de artículos que permitirán al lector desarrollar destrezas en el diseño de interfaces Web, aplicando HTML, CSS y JavaScript para facilitar la construcción de aplicaciones de mediana envergadura en el área de ingeniería.

Prof. Victor Esteller, Dr
14/03/2023

Contenido

Primeros pasos en la Web.....	3
Instalación de software básico.....	4
¿Cuál será la apariencia de tu sitio Web?	6
Manejo de archivos.....	10
Conceptos básicos de HTML.....	15
CSS básico.....	22
Fundamentos de JavaScript.....	35
Publica tu sitio web.....	48
Publicar a través de GitHub.....	51

Primeros pasos en la Web

Introducción al diseño de interfaces Web es una serie concisa que te presenta los aspectos prácticos del desarrollo web en un lenguaje coloquial. Configurarás las herramientas que necesitas para construir una sencilla página web y publicar tu propio código.

La historia de tu primer sitio web

Es mucho trabajo crear un sitio web profesional, así que, si eres nuevo en el desarrollo web, te animamos a que empieces poco a poco. No crearás otro Facebook de inmediato, pero no es difícil tener tu propio sitio web sencillo en línea, así que comenzaremos por ahí.

Al trabajar en orden a través de los artículos que se enumeran a continuación, pasarás de la nada a tener tu primera página web en línea. ¡Comencemos nuestro viaje!

Instalación de software básico

Cuando se trata de herramientas para crear un sitio web, hay mucho para elegir. Si recién estás comenzando, es posible que te sientas confundido por la variedad de editores de código, marcos de desarrollo y herramientas de prueba que existen. En [Instalación de software básico](#), te mostramos paso a paso cómo instalar solo el software que necesitas para comenzar un desarrollo web básico.

¿Cuál será la apariencia de tu sitio web?

Antes de comenzar a escribir el código para tu sitio web, primero lo debes planificar. ¿Qué información estás mostrando?, ¿qué fuentes y colores estás usando?, en [¿cuál será la apariencia de tu sitio web?](#), describimos un método simple que puedes seguir para planificar el contenido y modelado de tu sitio.

Manejo de archivos

Un sitio web consta de muchos archivos: texto del contenido, código, hojas de estilo, contenido multimedia, etc. Cuando estás creando un sitio web, necesitas ensamblar estos archivos en una estructura sensata y asegurarte de que se puedan comunicar entre sí. [Manejo de archivos](#) explica cómo configurar una estructura de archivos sensible para tu sitio web y qué problemas debes tener en cuenta.

Conceptos básicos de HTML

El lenguaje de marcado de hipertexto (HTML) es el código que utilizas para estructurar tu contenido web y darle significado y propósito. Por ejemplo, ¿mi contenido es un conjunto de párrafos o una lista de viñetas?, ¿tengo imágenes insertadas en mi página?, ¿tengo una

tabla de datos?; Sin abrumarte, [conceptos básicos de HTML](#) proporciona suficiente información para familiarizarte con HTML.

Conceptos básicos de CSS

Hojas de estilo en cascada (CSS) es el código que utilizas para aplicar estilo a tu sitio web. Por ejemplo, ¿desea que el texto sea negro o rojo?, ¿dónde se debe dibujar el contenido en la pantalla?, ¿qué imágenes de fondo y colores se deben utilizar para decorar tu sitio web?, [Conceptos básicos de CSS](#) te indica lo que necesitas para empezar.

Conceptos básicos de JavaScript

JavaScript es el lenguaje de programación que utilizas para agregar funciones interactivas a tu sitio web. Algunos ejemplos podrían ser juegos, cosas que suceden cuando se presionan botones o se ingresan datos en formularios, efectos de estilo dinámico, animación y mucho más. [Conceptos básicos de JavaScript](#) te da una idea de lo que es posible con este interesante lenguaje y cómo empezar.

Publicar tu sitio web

Una vez que hayas terminado de escribir el código y organizado los archivos que componen tu sitio web, lo debes poner todo en línea para que la gente lo pueda encontrar. [Publica tu código de ejemplo](#) describe cómo publicar tu código de ejemplo en línea con el mínimo esfuerzo.

Cómo funciona la web

Cuando accedes a tu sitio web favorito, suceden muchas cosas complicadas en segundo plano que quizás no conozcas. [Cómo funciona la web](#) describe lo que sucede cuando ves una página web en tu dispositivo favorito.

Instalación de software básico

La *Instalación de software básico*, te muestra las herramientas que necesitas para hacer el desarrollo web simple, y la forma de instalarlas correctamente.

¿Qué herramientas usan los profesionales?

- **Una computadora.** Tal vez esto suena obvio para algunas personas, pero habrá quien esté leyendo este artículo desde el móvil o una computadora de biblioteca. Para el desarrollo web serio, es mejor invertir en un equipo de escritorio o portátil con Windows, Mac o Linux.

- **Un editor de texto**, para escribir código. Puedes usar un editor de texto libre (ej. [Brackets](#), [Atom](#), [Notepad++](#), [Sublime Text](#), [GNU Emacs](#), [VIM](#), [Visual Studio Code](#), [WebStorm](#)) o un editor híbrido ([Dreamweaver](#)). Los editores de documentos de oficina no son adecuados para esto, pues dependen de elementos ocultos que interfieren con los motores de renderizado usados por los navegadores.
- **Navegadores web**, para probar el código. Actualmente los navegadores más usados son [Firefox](#), [Chrome](#), [Opera](#), [Safari](#), [Vivaldi](#), [Internet Explorer](#) y [Microsoft Edge](#). También debes comprobar cómo funciona tu web en dispositivos móviles y en cualquier navegador antiguo que tu objetivo público pueda estar usando aún (tal como IE 6–8.)
- **Un editor de gráficos o imágenes**, como [GIMP](#), [Paint.NET](#) o [Photoshop](#), para crear imágenes para tus páginas web.
- **Un sistema de control de versiones**, para administrar archivos en servidores, colaborar en un proyecto con un equipo, compartir código y recursos, y evitar conflictos de edición. Hoy en día [Git](#) es el sistema de control de versiones más populares y el servicio de alojamiento de código [GitHub](#), basado en Git, también es muy popular.
- **Un programa de FTP**, para cargar páginas web en un servidor para el público (Git está reemplazando cada vez más a FTP para ese fin). Hay un montón de estos programas disponibles incluyendo [Cyberduck](#), [Fetch](#) y [FileZilla](#).
- **Un sistema de automatización**, como [Grunt](#) o [Gulp](#) para realizar tareas repetitivas de forma automática, por ejemplo, minimización de código y ejecución de pruebas.
- Bibliotecas, marcos de desarrollo (frameworks), etc., para acelerar la escritura de funciones comunes. Una biblioteca tiende a ser un archivo JavaScript o CSS existente que proporciona una lista de funciones para usar para que la utilice en su código. Un framework tiende a llevar esta idea más allá, ofreciendo un sistema completo con alguna sintaxis personalizada para que puedas escribir una aplicación web basada en él.
- ¡Muchas más herramientas!

Ahora mismo: ¿qué herramientas necesitas realmente?

Esto parece una lista espeluznante, pero afortunadamente, puede comenzar a trabajar en el desarrollo web sin saber nada de la mayoría de estas herramientas. En este artículo solo tendrá que configurar lo mínimo: un editor de texto y algunos navegadores web modernos.

Instalación de un editor de texto

Probablemente ya tengas un editor de texto básico instalado en tu computadora. De manera predeterminada, Windows incluye el [Bloc de notas](#) y OS X viene con [TextEdit](#). Las distribuciones (versiones) de Linux varían: Ubuntu viene con [Gedit](#); Las distribuciones basadas en KDE suelen traer [Kate](#) o [Kwrite](#).

Para el desarrollo Web, probablemente hay cosas mejores que el Bloc de notas o TextEdit. Una recomendación puede ser comenzar con [Brackets](#), un editor gratuito que ofrece vistas previas en vivo y sugerencias de código, y [Visual Studio Code](#) también.

Instalación de navegadores web modernos

Por ahora, solo tendrá que instalar un par de navegadores web de escritorio para poner a prueba su código. Selecciona tu sistema operativo y pulsa los enlaces pertinentes para descargar los instaladores de tus navegadores preferidos:

- Linux: [Firefox](#) , [Chrome](#) , [Opera](#) , [Vivaldi](#) .
- Windows: [Firefox](#) , [Chrome](#) , [Opera](#) , [Vivaldi](#) , [Internet Explorer](#) (si tienes Windows 8 o superior, puedes instalar IE 10 o posterior, de lo contrario, deberías instalar un navegador alternativo).
- Mac: [Firefox](#) , [Chrome](#) , [Opera](#) , [Vivaldi](#) , [Safari](#) (Safari de manera predeterminada viene con iOS y OS X)

Antes de continuar, deberías instalar al menos dos de estos navegadores y tenerlos disponibles para pruebas.

Nota: Internet Explorer no es compatible con algunas funciones web modernas y es posible que no puedas ejecutar tu proyecto. Por lo general, no necesitas preocuparte por hacer que tus proyectos web sean compatibles con él, ya que muy pocas personas todavía lo usan; ciertamente, no te preocupes demasiado por él mientras aprendes. En ocasiones, es posible que te encuentres con un proyecto que requiera soporte.

¿Cuál será la apariencia de tu sitio Web?

¿Cómo se verá tu sitio web?, analiza el trabajo de planificación y diseño que debes realizar para tu sitio web antes de escribir el código, incluyendo: "¿qué información ofrece mi sitio web?", "¿qué tipos de letra y colores quiero?" y "¿qué hace mi sitio?".

Lo primero es lo primero: planificación

Antes de hacer nada, necesitas algunas ideas. ¿Qué debería hacer realmente tu sitio web?; Un sitio web puede hacer básicamente cualquier cosa, pero, en tu primer intento, debes mantener las cosas simples. Comenzarás creando una página web simple con un encabezado, una imagen y algunos párrafos.

Para comenzar, deberás responder estas preguntas:

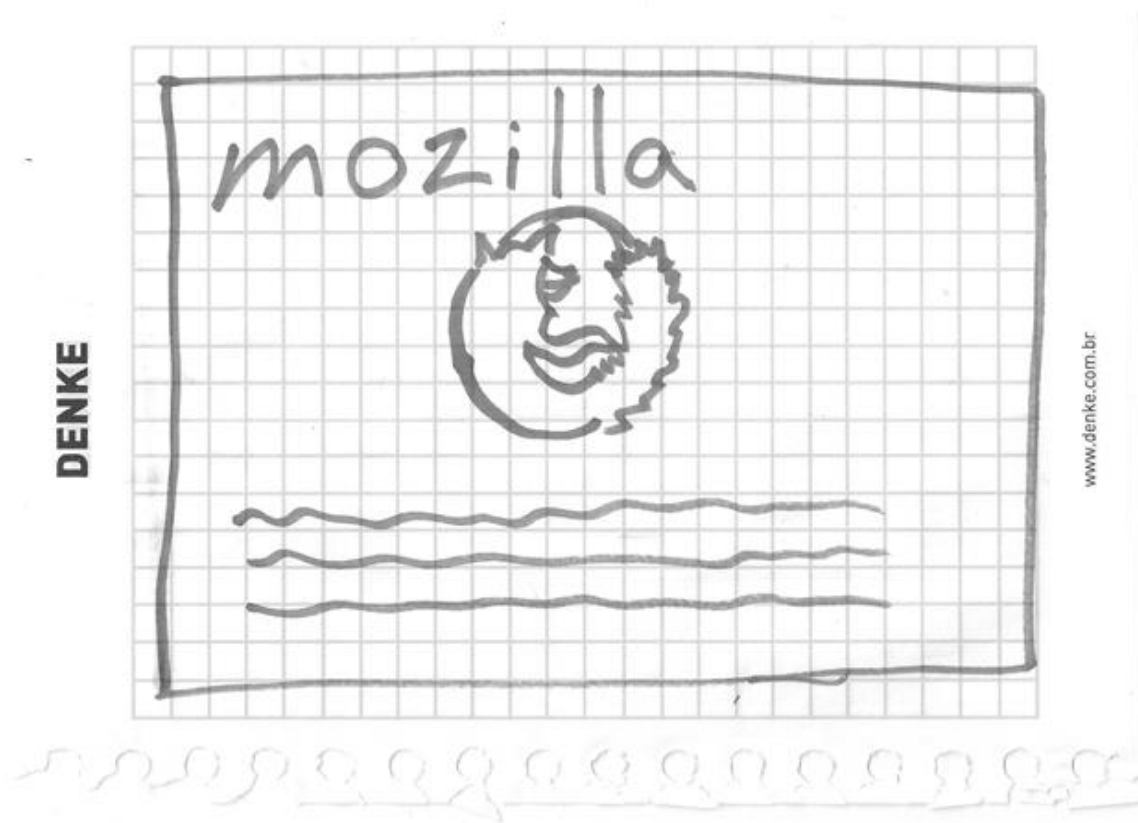
1. **¿De qué trata tu sitio web?**, ¿te gustan los perros, Nueva York o Pac-Man?
2. **¿Qué información presentas sobre el tema?**; Escribe un título y algunos párrafos y piensa en una imagen que te gustaría mostrar en tu página.

3. **¿Cómo se ve tu sitio web**, en términos simples de alto nivel?, ¿cuál es el color de fondo?, ¿qué tipo de letra es apropiado: formal, caricaturesca, atrevida y fuerte, ¿sutil?

Nota: Los proyectos complejos necesitan pautas detalladas que incluyan todos los detalles de los colores, los tipos de letra, el espacio entre los elementos de una página, el estilo de escritura adecuado, etc. Esto, a veces, se denomina guía de diseño, sistema de diseño o libro de marcas.

Haz un bosquejo de tu diseño

A continuación, toma papel y lápiz y dibuja aproximadamente cómo deseas que se vea tu sitio. Para tu primera página web simple, no hay mucho que esbozar, pero deberías adquirir el hábito de hacerlo ahora. Realmente ayuda, ¡no tienes que ser Van Gogh!



Nota: Incluso en sitios web reales y complejos, los equipos de diseño suelen comenzar con bocetos en papel y luego crean maquetas digitales utilizando un editor de gráficos o tecnologías web.

Los equipos web suelen incluir tanto un diseñador gráfico como un diseñador de experiencia de usuario (UX). Los diseñadores gráficos ensamblan las imágenes del sitio web. Los diseñadores de experiencia de usuario tienen un papel algo más abstracto al abordar cómo los usuarios experimentarán e interactuarán con el sitio web.

Elige tus activos

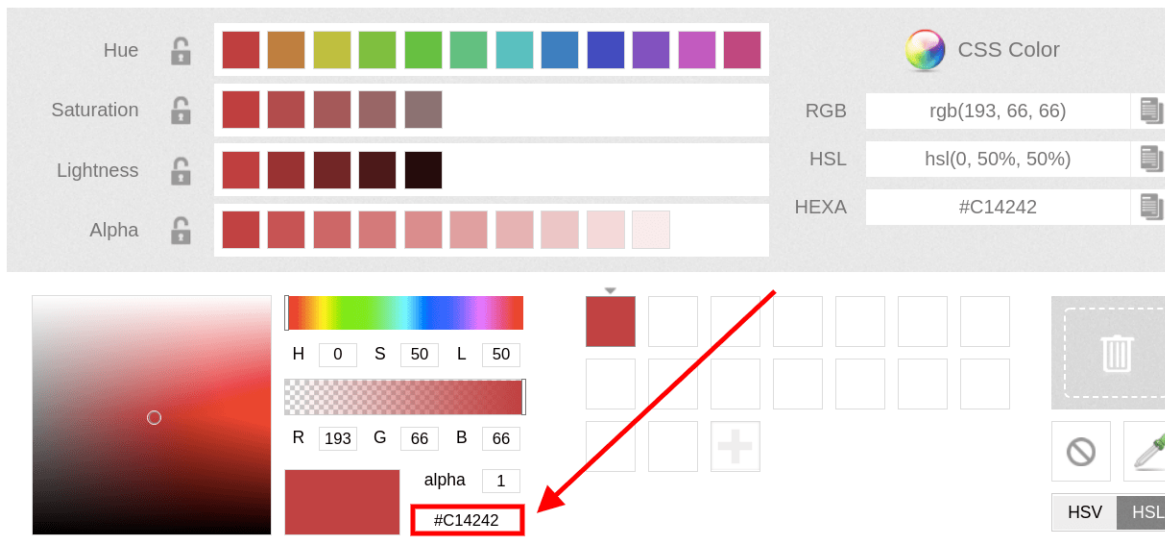
En este punto, es bueno comenzar a reunir el contenido que eventualmente aparecerá en tu página web.

Texto

Aún debes tener los párrafos y el título de antes. Mantenlos cerca.

Color del tema

Para elegir un color, ve al [Selector de color](#) y busca un color que te guste. Al hacer clic en un color, verás un extraño código de seis caracteres como #660066. Eso se llama *código hex* (abreviatura de hexadecimal) y representa tu color. Copia el código en un lugar seguro por ahora.



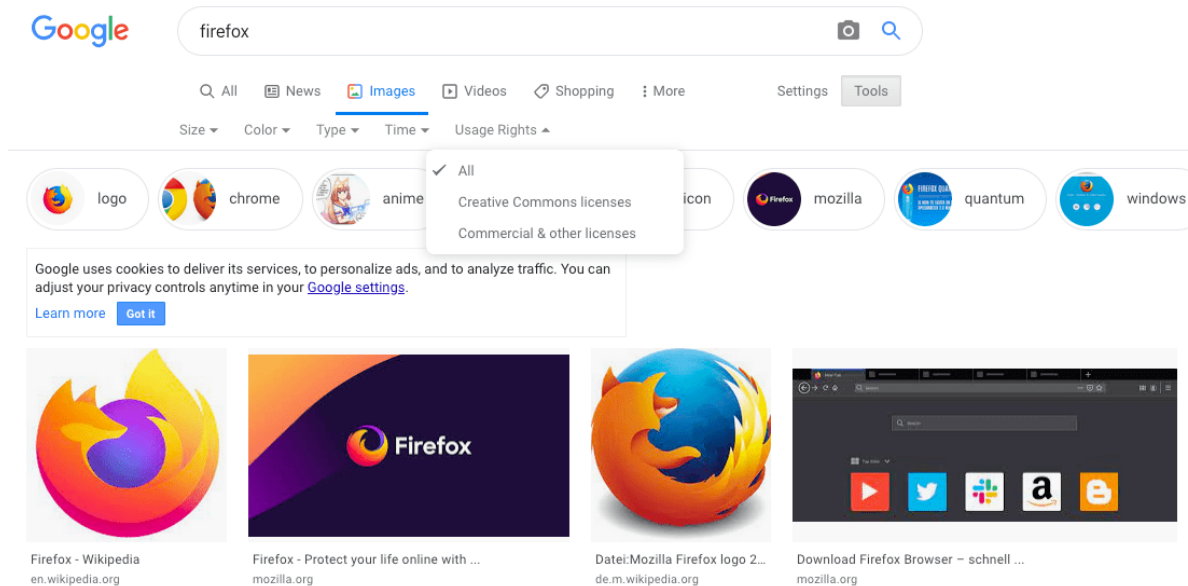
Imágenes

Para elegir una imagen, ve a [Imágenes Google](#) y busca algo adecuado.

1. Cuando encuentres la imagen que desees, haz clic en la imagen para obtener una vista ampliada de la misma.
2. Haz clic con el botón derecho en la imagen (Ctrl+clic en una Mac), elige *Guardar imagen como...* y elige un lugar seguro para guardar tu imagen. Alternativamente, copia la dirección web de la imagen de la barra de direcciones de tu navegador para su posterior uso.



Ten en cuenta que la mayoría de las imágenes en la web, incluidas las de Imágenes Google, están protegidas por derechos de autor. Para reducir tu probabilidad de violar los derechos de autor, puedes utilizar el filtro de licencias de Google. Haz clic en el botón *Herramientas* y luego en la opción *Derechos de uso* resultante que aparece a continuación. Debes elegir una opción como *Etiquetado para reutilización*.



Tipos de letra

Para elegir un tipo de letra:

1. Ve a [Google Fonts](#) y encuentra una que te guste.
2. Copia las líneas de código que Google le proporciona en tu editor de texto para guardarlas más tarde.
3. Para obtener más detalles sobre el uso de Google Fonts, consulta [esta página](#)

Manejo de archivos

Un sitio web consta de muchos archivos: texto del contenido, código, hojas de estilo, contenido multimedia, etc. Cuando estás creando un sitio web, necesitas ensamblar estos archivos en una estructura sensible en tu computadora local, asegurarte de que puedan comunicarse entre sí y hacer que todo su contenido se vea bien antes de que eventualmente [los cargues en un servidor](#). El *manejo de archivos* analiza algunos problemas que debes tener en cuenta, para que puedas configurar una estructura de archivos adecuada para tu sitio web.

¿Dónde debería estar tu sitio web en tu computadora?

Cuando estés trabajando en un sitio web localmente en tu computadora, debes mantener todos los archivos relacionados en un solo directorio que refleje la estructura de archivos del sitio web publicado en el servidor. Este directorio se puede ubicar en cualquier lugar

que desees, pero debes colocarlo en algún lugar donde lo puedas encontrar fácilmente, tal vez en tu escritorio, en tu directorio de inicio o en la raíz de tu disco duro.

1. Elige un lugar para almacenar los proyectos de tus sitios web. Dentro del lugar elegido, crea un nuevo directorio llamado `proyectosweb` (o algo similar). Aquí es donde vivirán todos los proyectos de tus sitios web.
2. Dentro de este primer directorio, crea otro directorio para almacenar tu primer sitio web. Llámalo `pruebasitio` (o algo más imaginativo).

Una acotación sobre la envoltura y el espaciado

Notarás que, a lo largo de este artículo, te pedimos que nombres los directorios y archivos completamente en minúsculas sin espacios. Esto es porque:

1. Muchas computadoras, particularmente los servidores web, distinguen entre mayúsculas y minúsculas. Entonces, por ejemplo, si colocas una imagen en tu sitio web en `pruebasitio/Miimagen.jpg` y luego, en un archivo diferente intentas invocar la imagen como `pruebasitio/miimagen.jpg`, puede que no funcione.
2. Los navegadores, servidores web y lenguajes de programación no manejan los espacios de manera consistente. Por ejemplo, si usas espacios en tu nombre de archivo, algunos sistemas pueden tratar el nombre de archivo como dos nombres de archivo. Algunos servidores reemplazarán las áreas en tus nombres de archivo con `"%20"` (el código de caracteres para espacios en URI), lo cual provocará que todos tus enlaces se rompan. Es mejor separar las palabras con guiones, en lugar de guiones bajos: `mi-archivo.html` vs. `mi_archivo.html`.

La respuesta corta es que debes usar un guion para los nombres de tus archivos. El motor de búsqueda de Google trata un guion como un separador de palabras, pero no considera un guion bajo de esa manera. Por estos motivos, es mejor adquirir el hábito de escribir los nombres de los directorios y archivos en minúsculas, sin espacios y con palabras separadas por guiones, al menos hasta que sepas lo que estás haciendo. De esa manera, tropezarás con menos problemas en el futuro.

¿Qué estructura debe tener tu sitio web?

A continuación, veamos qué estructura debería tener tu sitio de prueba. Las cosas más comunes que tendrás en cualquier proyecto de sitio web que crees son un archivo de índice HTML y directorios para contener imágenes, archivos de estilo y archivos de script. Crea estos ahora:

1. **index.html**: Este archivo generalmente tendrá el contenido de tu página de inicio, es decir, el texto y las imágenes que las personas ven cuando visitan tu sitio por primera vez. Usando tu editor de texto, crea un nuevo archivo llamado `index.html` y guárdalo dentro de tu directorio `pruebasitio`.
2. Directorio **images**: Este directorio contendrá todas las imágenes que utilices en tu sitio. Crea un directorio llamado `images`, dentro de tu directorio `pruebasitio`.

3. Directorio **styles**: Este directorio contendrá el código CSS que se utiliza para aplicar estilo al contenido (por ejemplo, configurar el texto y los colores de fondo). Crea un directorio llamado styles, dentro de tu directorio pruebasitio.
4. Directorio **scripts**: Este directorio contendrá todo el código JavaScript utilizado para agregar funcionalidad interactiva a tu sitio (por ejemplo, botones que cargan datos cuando se hace clic en ellos). Crea un directorio llamado scripts, dentro de tu directorio pruebasitio.

Nota: En las computadoras con Windows, es posible que tengas problemas para ver los nombres de los archivos, porque de manera predeterminada, Windows tiene activada una opción llamada **Ocultar extensiones para tipos de archivos conocidos**. Generalmente, la puedes desactivar yendo al Explorador de Windows, seleccionando la opción **Opciones de directorio...**, desmarcando la casilla de verificación **Ocultar extensiones para tipos de archivo conocidos** y luego haciendo clic en **Aceptar**. Para obtener información más específica sobre tu versión de Windows, puedes buscar en la web.

Rutas de archivo

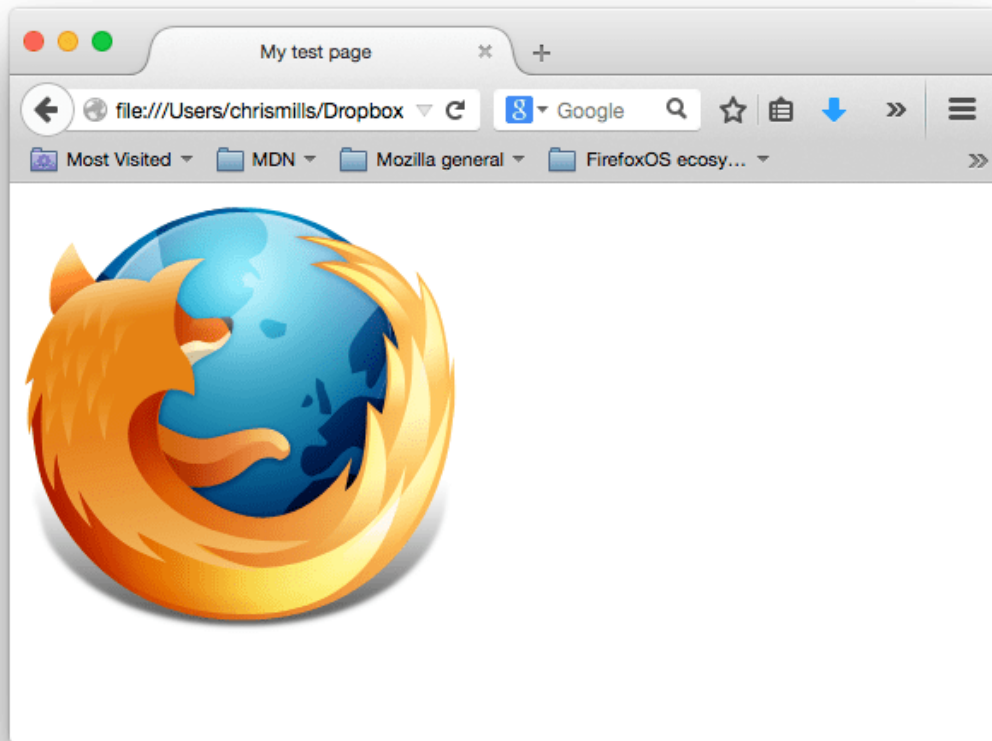
Para que los archivos se comuniquen entre sí, debes proporcionar una ruta de archivo entre ellos, básicamente una ruta, para que un archivo sepa dónde está otro. Para demostrarlo, insertaremos un poco de HTML en nuestro archivo index.html y haremos que muestre la imagen que elegiste en el artículo [¿Cómo se verá tu sitio web?](#)

1. Copia la imagen que elegiste anteriormente en tu directorio images.
2. Abre tu archivo index.html e inserta el siguiente código en el archivo exactamente como se muestra. Por ahora, no te preocupes por lo que significa todo esto; veremos las estructuras con más detalle más adelante en la serie.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Mi página de prueba</title>
  </head>
  <body>
    <img src="" alt="Mi imagen de prueba" />
  </body>
</html>
```

3. La línea `` es el código HTML que inserta una imagen en la página. Necesitamos decirle al HTML dónde está la imagen. La imagen está dentro del directorio *images*, que está en el mismo directorio que index.html. Para recorrer la estructura del archivo desde index.html hasta nuestra imagen, la ruta del archivo que necesitamos es `images/nombre-archivo-imagen`. Por ejemplo, nuestra imagen se llama `firefox-icon.png`, por lo que la ruta del archivo es `images/firefox-icon.png`.
4. Inserta la ruta del archivo en tu código HTML entre las comillas dobles del código `src=""`.

5. Guarda tu archivo HTML, luego cárgalo en tu navegador web (haz doble clic en el archivo). ¡Deberías ver tu nueva página web mostrando tu imagen!



Algunas reglas generales para las rutas de archivo:

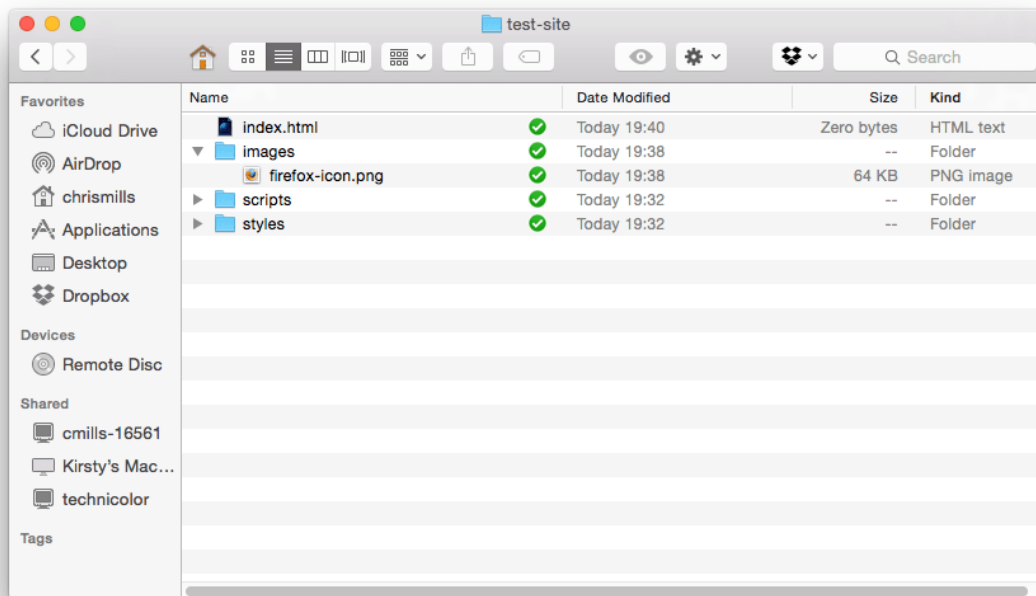
- Para vincular a un archivo destino en el mismo directorio que el archivo HTML de invocación, simplemente usa el nombre del archivo, p. ej. mi-imagen.jpg.
- Para hacer referencia a un archivo en un subdirectorio, escribe el nombre del directorio delante de la ruta, más una barra inclinada, p. ej. subdirectorio/mi-imagen.jpg.
- Para vincular a un archivo destino en el directorio **arriba** del archivo HTML que lo invoca, escribe dos puntos. Por ejemplo, si index.html estuviera dentro de un subdirectorio de pruebasitio y mi-imagen.jpg estuviera dentro de pruebasitio, puedes hacer referencia a mi-imagen.jpg desde index.html utilizando ../mi-imagen.jpg.
- Los puedes combinar tanto como desees, por ejemplo, ../subdirectorio/otro-subdirectorio/mi-imagen.jpg.

Por ahora, esto es todo lo que necesitas saber.

Nota: El sistema de archivos de Windows tiende a utilizar barras invertidas, no barras diagonales, p. ej. C:\windows. Esto no importa en HTML, incluso si estás desarrollando tu sitio web en Windows, debes usar barras diagonales en tu código.

¿Qué más se debería hacer?

Eso es todo por ahora. La estructura de tu directorio debería verse así:



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 cuál es su función.

Entonces, ¿qué es HTML en realidad?

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:

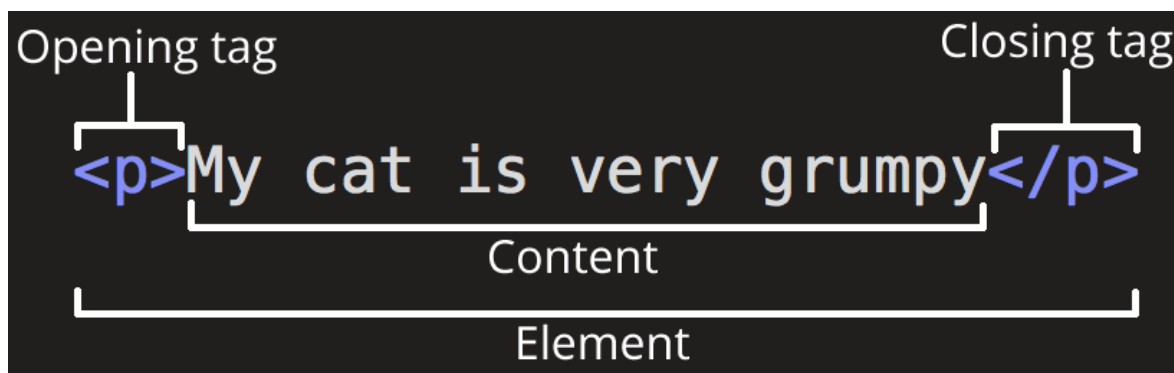
Mi gato es muy gruñon

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.



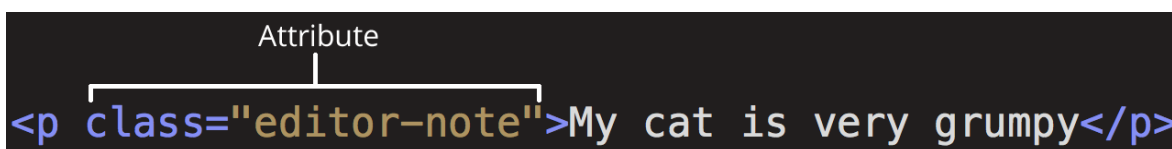
Las partes principales del elemento son:

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:



```
<p class="editor-note">My cat is very grumpy</p>
```

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 ``, 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 ``, 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 `` de nuestro HTML:

```

```

Posee dos atributos, pero no hay etiqueta de cierre `` 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>
    
  </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 desees 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* [](#):

```

```

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:

My test image

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.

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> \(en-US\)](#)—[<h6> \(en-US\)](#), 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 [](#).

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 ``.

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 `` (*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 `` (*ordered list*).

Cada elemento de la lista se coloca dentro de un elemento `` (*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 «Manifiesto Mozilla».
2. Encierra el texto en un elemento `<a>`, así:
3. `<a>Manifiesto Mozilla`
4. Proporcióname al elemento `<a>` un atributo href, así:
5. `Manifiesto Mozilla`
6. Completa el valor de este atributo con la dirección web con la que quieras conectar al vínculo:
7. `Manifiesto Mozilla`

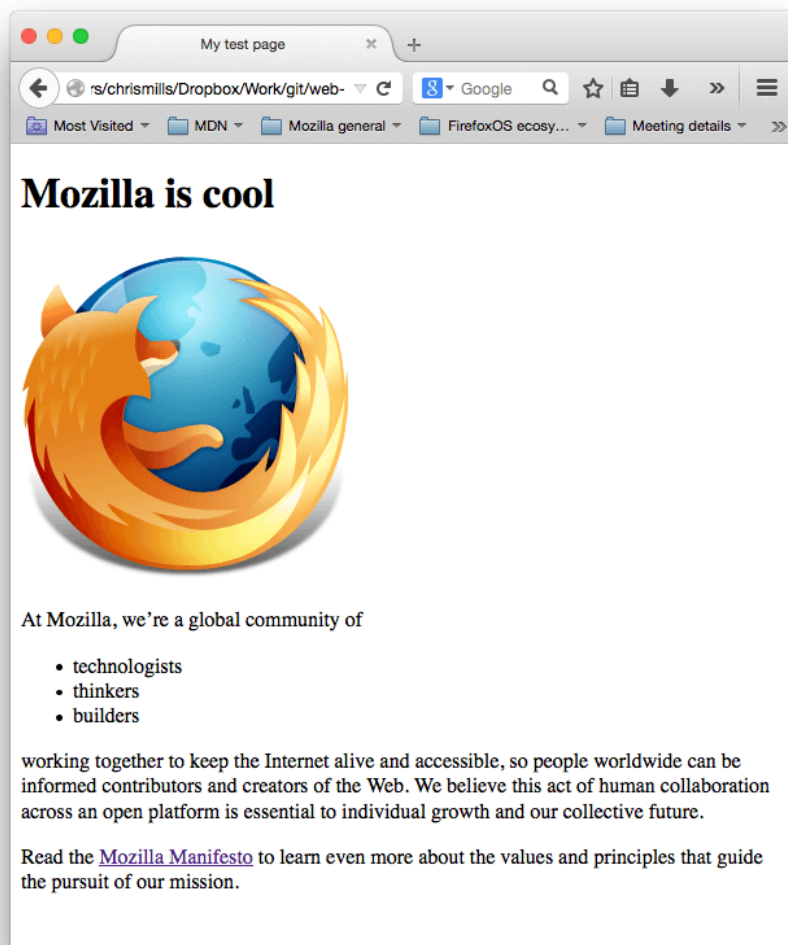
Podrías obtener resultados inesperados si al comienzo de la dirección web omites la parte `https://` o `http://` llamada *protocolo*. Así que luego del marcado del vínculo, haz clic en él para asegurarte que te dirige a la dirección deseada.

Nota: `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 *hypertext reference* (referencia de hipertexto).

Ahora agrega un vínculo a tu página, si es que aún no lo hiciste.

Conclusión

Si lograste seguir todas las instrucciones de este artículo, deberías terminar con una página que se vea así:



CSS básico

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 (*Hojas de estilo en cascada*) 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;  
}
```

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 debe aplicar el CSS a su documento HTML, de otra manera el estilo CSS no cambiará cómo su navegador muestra el documento HTML. (Si no ha seguido el proyecto, lea [Manejo de archivos](#) y [HTML básico](#) para averiguar qué necesita hacer primero.)

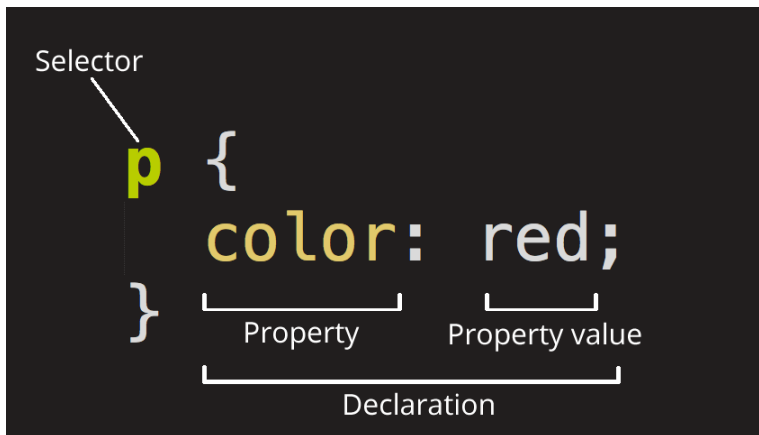
1. Abre tu archivo index.html y pega la siguiente línea en algún lugar dentro del `<head>`, es decir, entre las etiquetas `<head>` y `</head>`:
2. `<link href="styles/style.css" rel="stylesheet" type="text/css">`
3. Guarda el archivo index.html y cárgalo en tu navegador. Debes ver algo como esto:



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

Anatomía de una regla CSS

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



La estructura completa es llamada **regla preestablecida** (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 cuales 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, debe usar los dos puntos (:) para separar la propiedad de su valor.
- Dentro de cada regla, debe 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 necesita escribirlos separados por punto y coma (;), así:

```
p {
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

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;
}
```

Diferentes tipos de selectores

Existen muchos tipos diferentes de selectores. Antes, solo vea 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:

Nombre del seleccionador	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
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
Selector de atributos	Los elementos en una página con el atributo especificado.	img[src] Selecciona pero no
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.

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 en algún lugar del *encabezado* de tu archivo (de nuevo, en cualquier lugar entre las etiquetas). Debe verse algo así: `<link>...index.html<head></head>`

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

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. agregue las siguientes líneas (que se muestran a continuación), reemplazando la apariencia font-family de [su sitio web](#). La propiedad se refiere a la(s) fuente(s) que desea usar en su texto. Esta regla define una fuente base global y un tamaño de fuente para usar en toda la página. Dado que es el elemento primario (o padre) de toda la página, todos los elementos contenidos dentro de él heredan las propiedades): font-familyfont-family<html>font-sizefont-family

```
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 */
}
```

Nota: se ha añadido un comentario para explicar qué significa «px». Todo lo que está en un documento de CSS entre `/* */` 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.

4. Ahora escoge el tamaño de fuente para los elementos que contienen texto dentro del cuerpo del HTML (`<h1>` ([en-US](#)) , ``, y `<p>`). También centra el texto del título, escoge un ancho de línea y espacio entre letras en el contenido del texto para hacer un poco más legible:

```
h1 {
  font-size: 60px;
  text-align: center;
}

p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}
```

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



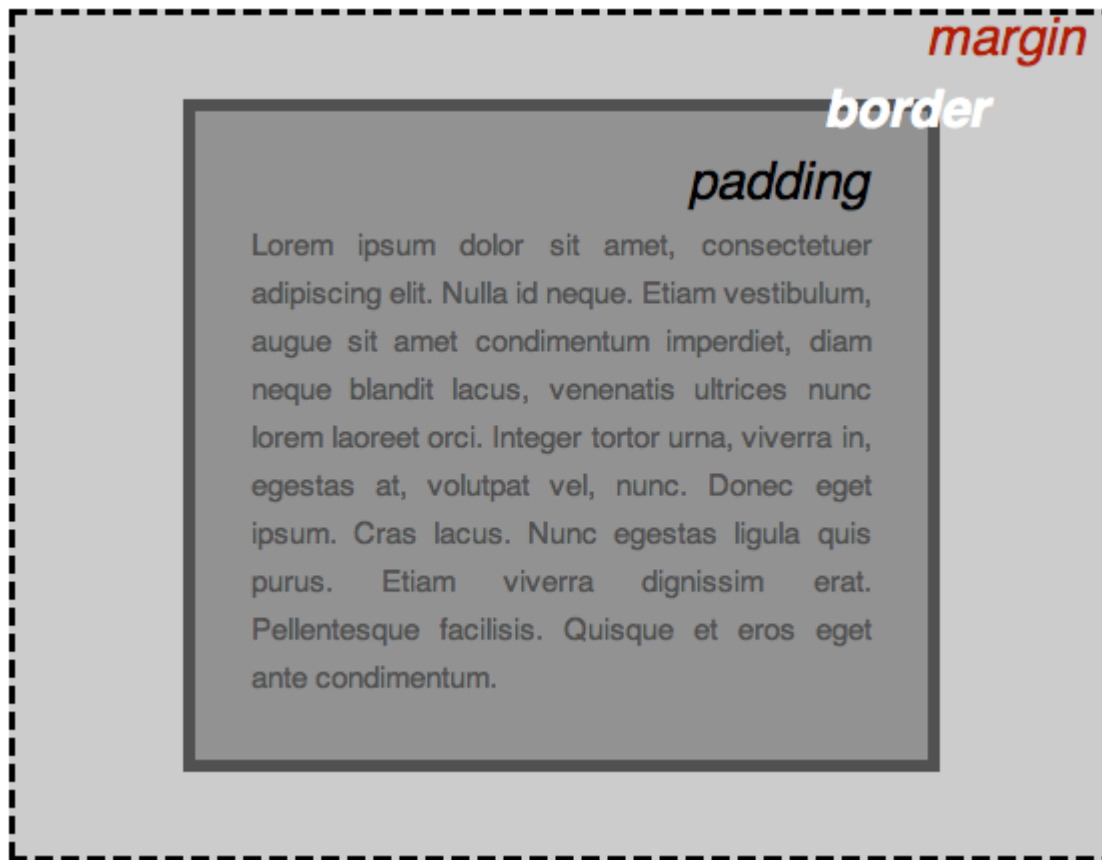
Cajas, cajas, todo se trata de cajas

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 difusa 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.

Cambia el color de la página

```
html {  
  background-color: #00539F;  
}
```

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;  
}
```

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 seleccione 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.
- 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.
- 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;  
}
```

Puedes haber notado que hay un hueco horrible en la parte superior del *cuerpo*. 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, trate de experimentar con diferentes valores para ver cómo resulta.

Centrar la imagen

```
img {  
  display: block;  
  margin: 0 auto;  
}
```

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 contribuyendo la propiedad `width` en el elemento `` 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 estudias CSS a profundidad.

Conclusión

Si has seguido las instrucciones de esta publicación, deberías terminar con una página que luce algo así:



Si te complicaste, puedes comparar tu trabajo con el [código del ejemplo finalizado](#):


```

html {
  font-size: 10px;
  font-family: 'Open Sans', sans-serif;
}

h1 {
  font-size: 60px;
  text-align: center;
}

p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}

html {
  background-color: #00539F;
}

body {
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}

h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}

img {
  display: block;
  margin: 0 auto;
}

```

```

<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <title>Mi página de prueba</title>
  <link href="https://fonts.googleapis.com/css2?family=Open+Sans"
rel="stylesheet" type="text/css">
  <link href="styles/style.css" rel="stylesheet" type="text/css">
</head>

<body>
  <h1>Mozilla es cool</h1>
  
  <p>En Mozilla, somos una comunidad de</p>

  <ul>
    <li>tecnólogos</li>
    <li>pensadores</li>
    <li>constructores</li>
  </ul>

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

  <a href="https://www.mozilla.org/es-AR/about/manifesto/">Manifesto
Mozilla</a>
</body>

</html>

```

Fundamentos de JavaScript

JavaScript es el lenguaje de programación que debe usar para agregar características interactivas a su 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 realmente JavaScript?

[JavaScript](#) es un lenguaje de programación robusto que se puede aplicar a un documento [HTML](#) y utilizar 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 enumeradas arriba. Puedes aprender todo eso en detalle más tarde.

Debajo se presentan algunos aspectos del núcleo del lenguaje.

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 empezar, mostraremos cómo añadir JavaScript básico a tu página, creando un « ¡Hola Mundo! » de ejemplo ([el estándar en los ejemplos básicos de programación](#)).

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>:
3. <script src="scripts/main.js"></script>
4. 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).
5. Ahora añade el siguiente código al archivo main.js:

```
const miTitulo = document.querySelector('h1');  
miTitulo.textContent = '¡Hola mundo!';
```

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



Nota: la razón por la que ha 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, coloque el JavaScript cerca del final de la página es normalmente la mejor estrategia.

¿Qué ha ocurrido?

El texto del título ha sido cambiado por *¡Hola mundo!* utilizando JavaScript. Hiciste esto primero usando la función [querySelector\(\)](#) para obtener una referencia al título y almacenarla en una llamada variable `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 ha 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 pueda 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.

Advertencia: en este artículo, trata de introducir las líneas de código de ejemplo en la consola de tu navegador para ver lo que sucede.

Variables

Las [Variables](#) son contenedores en los que pueden 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`:

```
let nombreDeLaVariable;
```

Nota: todas las líneas en JS deben terminar en punto y coma (;) para indicar que es ahí donde termina la declaración. Si no los incluye 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.

Nota: puedes llamar a una variable con casi cualquier nombre, pero hay algunas restricciones.

Nota: JavaScript distingue entre mayúsculas y minúsculas. `miVariable` es una variable distinta a `mivariable`. Si tienes problemas en tu código, revisa las mayúsculas y minúsculas.

Tras declarar una variable, puedes asignarle un valor:

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

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

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

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

```
nombreDeLaVariable;
```

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

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

Cuidado que las variables tienen distintos [tipos de datos](#) :

Variable	explicacion	ejemplo
Cadena	Esto es una secuencia de texto conocida como cadena. Para indicar que la variable es una cadena, debe escribirlo entre comillas.	let miVariable = 'Bob';
Número	Esto es un número. Los números no tienen comillas.	let miVariable = 10;
booleano	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.
Objeto	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 pueden cambiar, entonces no podrían hacer nada dinámico, como personalizar un mensaje de bienvenida de un usuario que visite su página, cambie 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

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

```
/*
Esto es un comentario
de varias líneas.
*/
```

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	explicacion	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 esperarían que hicieran en las matemáticas básicas.	-, *, /	9 - 3; 8 * 2; // La multiplicación en JS es un asterisco 9 / 3;
Operador de despacho	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.

Nota: mezclar tipos de datos puede dar lugar a resultados extraños cuando se hacen cálculos, así que asegúrese de que relaciona sus variables correctamente y de que recibe los resultados que esperaba. Por ejemplo, teclea: "3" + "25" en tu consola. ¿Por qué no obtienes lo que esperabas? Porque las comillas definen los números en "strings" (el término inglés para denominar cadenas de caracteres) y de este modo ha terminado 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 *verdadero* 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('Awwwww, pero mi favorito es el de chocolate...');
}
```

La expresión dentro de if (...) es el criterio — este usa al operador de identidad (escrito 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 quiere reutilizar, de manera que puede llamar esa función con un solo nombre, y no tendrá que escribir el código entero cada vez que la utilice. Ya has visto algunas funciones más arriba, por ejemplo:

```
let nombreDeLaVariable = document.querySelector('h1');
alert('¡Hola!');
```

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 necesita 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;  
}
```

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);
```

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».

Eventos

Para crear una interacción real en tu sitio web, debes usar eventos. Estas 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!');  
}
```

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(){};
```

es equivalente a

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

es solo un modo más corto de escribirlo.

Sobrecargar tu sitio web de ejemplo

Ahora vas a revisar 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 añadirás un poco de código para cambiar entre imágenes al hacer clic.

1. Primero que todo, busca una imagen que te guste para tu sitio. Asegúrese de 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');
  }
}
```

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

Esto fue lo que sucedió: se almacena una referencia a tu elemento `` 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:
 - i. 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 ``.
 - ii. Si no es así (significa que ya fue modificada), se cambiará el valor de `src` nuevo a la ruta de la imagen original, regresando a como era en un principio.

Añadir un mensaje de bienvenida personalizado

Ahora agregue un poco más de código, para cambiar el título de la página o incluya un mensaje personalizado de bienvenida para cuando el usuario ingrese por primera vez. Este mensaje de bienvenida mejorará luego de que el usuario abandone la página y estará disponible para cuando regrese. Lo guardará usando [Web Storage API](#). También se producirá 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>
```

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');
```

3. 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;  
}
```

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 está 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` título a una cadena, más el nombre de usuario almacenado recientemente.

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

```

if (!localStorage.getItem('nombre')) {
    estableceNombreUsuario();
}
else {
    let nombreAlmacenado = localStorage.getItem('nombre');
    miTitulo.textContent = 'Mozilla es genial,' + nombreAlmacenado;
}

```

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 el título a la cadena, más el nombre del usuario, como hiciste dentro de estableceNombreUsuario().

5. 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:

```

miBoton.onclick = function() {
    estableceNombreUsuario();
}

```

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 perderá después de que cierre el sitio, ¡manteniendo ahí el mensaje personalizado cuando abras el sitio la próxima vez!

¿Un nombre de usuario nulo?

Cuando ejecuta el ejemplo y obtiene el cuadro de diálogo que solicita que introduzca su nombre de usuario, intenta pulsar el botón *Cancelar*. debería 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, pruebe 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, podrá 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);
    }
}

```

```

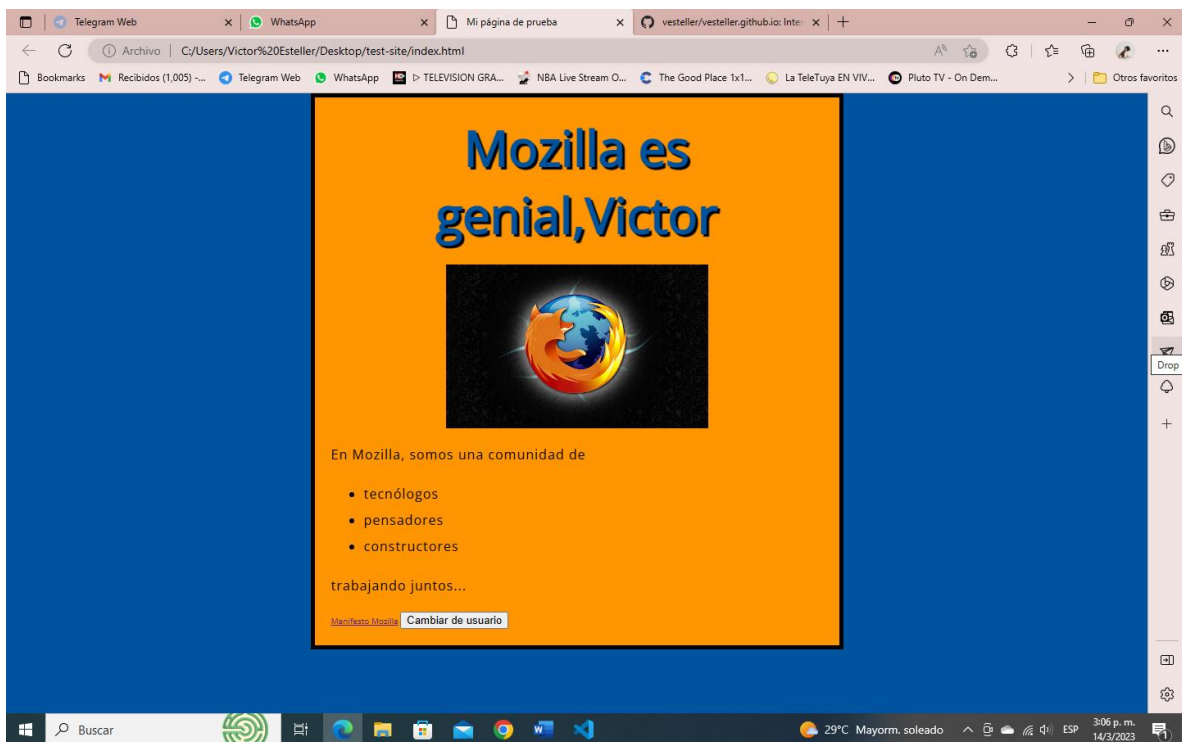
miTitulo.innerHTML = 'Mozilla es genial, ' + miNombre;
}
}

```

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 obtuvo anterior no es verdadera), entonces almacene el valor en `localStorage` y establézcalo como el texto del título.

Conclusión

Si ha seguido las instrucciones en este artículo, tendrá una página que luzca como esta



Si tuviste problemas, siempre puedes comparar su trabajo con el [código terminado del ejemplo](#).

```

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

let miBoton = document.querySelector('button');
let miTitulo = document.querySelector('h1');

function estableceNombreUsuario() {
    let miNombre = prompt('Introduzca su nombre. ');
    if(!miNombre) {
        estableceNombreUsuario();
    } else {
        localStorage.setItem('nombre', miNombre);
        miTitulo.innerHTML = 'Mozilla is genial, ' + miNombre;
    }
}

if (!localStorage.getItem('nombre')) {
    estableceNombreUsuario();
}
else {
    let nombreAlmacenado = localStorage.getItem('nombre');
    miTitulo.textContent = 'Mozilla es genial, ' + nombreAlmacenado;
}

miBoton.onclick = function() {
    estableceNombreUsuario();
}

```

```

<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <title>Mi página de prueba</title>
  <link href="https://fonts.googleapis.com/css2?family=Open+Sans"
rel="stylesheet" type="text/css">
  <link href="styles/style.css" rel="stylesheet" type="text/css">
</head>

<body>
  <h1>Mozilla es cool</h1>
  
  <p>En Mozilla, somos una comunidad de</p>

  <ul>
    <li>tecnólogos</li>
    <li>pensadores</li>
    <li>constructores</li>
  </ul>

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

  <a href="https://www.mozilla.org/es-AR/about/manifesto/">Manifesto
Mozilla</a>

  <button>Cambiar de usuario</button>
  <script src="scripts/main.js"></script>
</body>

</html>

```

Publica tu sitio web

Una vez que termines de escribir tu código y organizar los archivos que forman parte de tu sitio, debes ponerlo en línea para que la gente pueda consultarlo. Este artículo muestra cómo conseguir de manera sencilla que tu código esté en línea.

¿Cuáles son las opciones?

Publicar un sitio no es un tema sencillo, principalmente porque hay muchas maneras diferentes de hacerlo. En este artículo no se trata de ver todos los modos posibles. En su lugar, discutiremos los pros y contras de tres amplias estrategias desde el punto de vista de un principiante, y luego debes seleccionar qué método usarás.

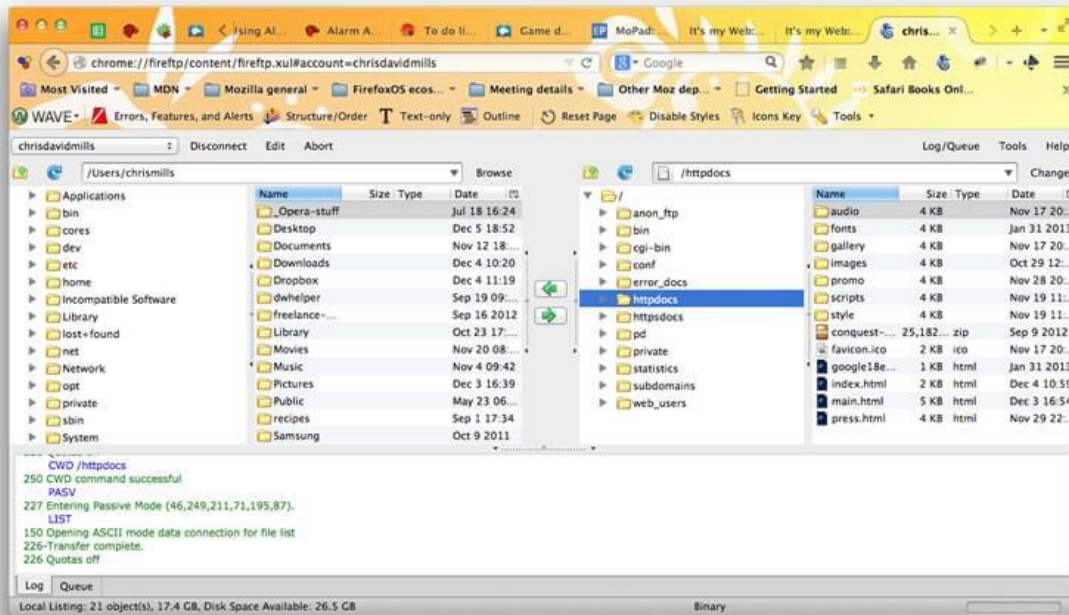
Obtener alojamiento y un nombre de dominio

Si deseas un control total sobre tu sitio web publicado, probablemente necesitarás gastar dinero para comprar:

- *Alojamiento (Hosting)* — espacio de almacenamiento alquilado en el servidor web de una compañía de alojamientos. Pones los archivos de tu sitio web en este espacio, y el servidor web suministra el contenido a los usuarios que lo solicitan.
- Un nombre de dominio — dirección única mediante la cual la gente puede encontrar tu sitio web, como <https://www.mozilla.org>, o <https://es.wikipedia.org/>. Puedes tomar en alquiler el nombre de tu dominio durante algunos años en un **registrador de dominio**.

Muchos sitios web profesionales toman esta opción.

Además, necesitarás un programa de protocolo de transferencia de archivo (*File Transfer Protocol*, FTP) para transferir los archivos que conforman tu sitio web al servidor (mira más detalles de cuánto puede costar: software). Los programas FTP varían ampliamente, pero en general tienes que conectarte a tu servidor web contratado mediante detalles proporcionados por tu empresa de alojamiento (por ejemplo: nombre de usuario, contraseña, nombre del *host*). Una vez conectado con el servidor web el programa te mostrará tus archivos locales y los archivos del servidor web en dos ventanas y te proporcionará una forma de transferir los archivos de un lado a otro.



Consejos para elegir alojamiento y dominio

- No promovemos empresas comerciales de alojamiento o registradoras de nombre de dominio específicas. Para encontrarlas basta con buscar «alojamiento web» o «*hosting web*» y «nombres de dominio». A veces las empresas proporcionan ambos en un paquete único. Los registradores acostumbran a facilitar la manera de comprobar si el nombre de dominio que deseas para tu sitio está disponible.
- El proveedor de servicio de Internet (ISP) de tu casa u oficina puede proporcionar algún alojamiento limitado para un pequeño sitio web. El conjunto de características disponibles será limitado, pero podría ser perfecto para tus primeros experimentos; ¡ponte en contacto con ellos y pregunta!
- Hay servicios gratuitos disponibles como Neocities, Blogspot, y Wordpress. Una vez más, obtienes lo que pagas, pero son ideales para tus experimentos iniciales. Los servicios gratuitos en su mayoría no requieren software de FTP para transferencias de archivos pues permiten arrastrar y soltar archivos justo dentro de su interfaz web.
- Muchas compañías proporcionan alojamiento y dominio simultáneamente.

Utilizar una herramienta en línea como GitHub o Dropbox

Algunas herramientas te permiten publicar tu sitio en línea:

- GitHub es un sitio de «codificación social». Te permite cargar repositorios de código para almacenarlos en el **sistema de control de versiones** de Git. De esta manera puedes colaborar en proyectos de código pues por defecto el sistema es de código abierto, lo que significa que cualquier persona en el mundo puede encontrar tu código en GitHub, usarlo, aprender de él y mejorarlo. ¡Puedes hacer esto con el código de otra persona también! Git

es un sistema de control de versiones muy popular y GitHub es una comunidad muy importante y útil por lo que la mayor parte de empresas de tecnología ahora lo usan en su proceso laboral. GitHub tiene una característica muy útil llamada GitHub Pages, que te permite exponer el código de tu sitio web en vivo en la web.

- Dropbox es un sistema de almacenamiento de archivos que te permite guardar los archivos en la web y tenerlos disponibles desde cualquier ordenador. Cualquier persona con una conexión a Internet puede acceder a cualquier carpeta de Dropbox que esté accesible al público. Si esa carpeta contiene los archivos del sitio web, estos serán visualizados como un sitio web de forma automática.
- Google App Engine es una poderosa plataforma que permite construir y ejecutar aplicaciones en la infraestructura de Google, ya sea que necesites construir una aplicación web de varios niveles desde cero o alojar un sitio web estático. Para obtener más información consulta ¿Cómo se aloja un sitio web en Google App Engine?.

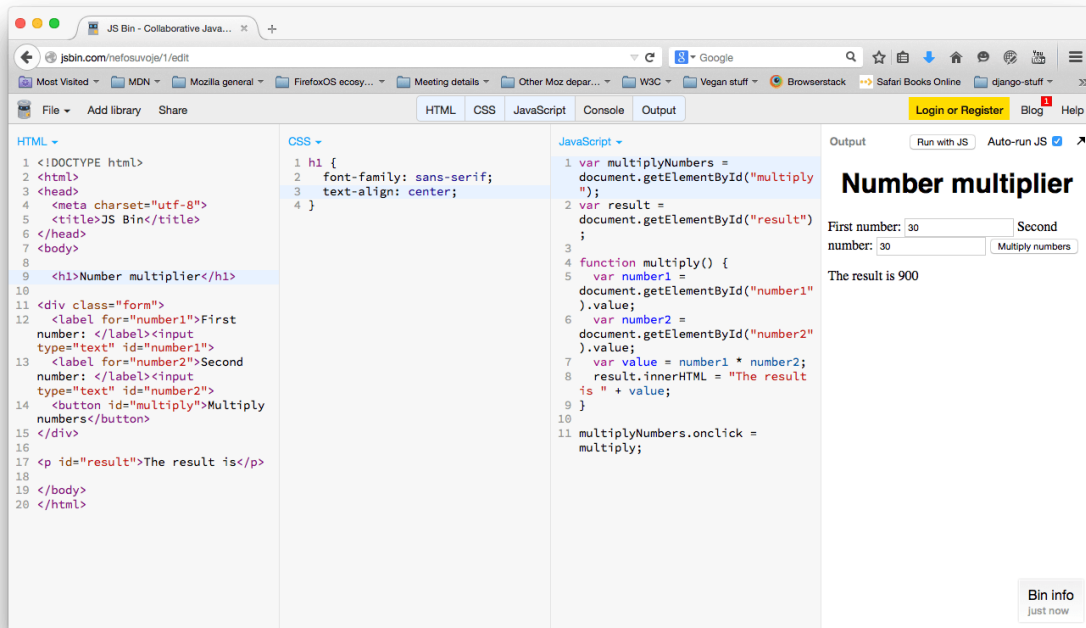
A diferencia de la mayoría de alojamientos (servicios de *hosting*), tales herramientas son por lo general libres de utilizar, pero solo permiten un conjunto de funciones limitadas.

Utilizar un entorno basado en web como CodePen

Existe un número de aplicaciones web que emulan un entorno de desarrollo de sitios web, permitiendo que ingreses tu código HTML, CSS y Javascript y luego muestran los resultados de dicho código como un sitio web, ¡todo en una pestaña del navegador! En términos generales, estas herramientas son bastante sencillas, geniales para aprender, buenas para compartir código (por ejemplo, si quieres compartir con alguien una técnica o pedir ayuda en la depuración del código) y gratuitas para las funciones básicas. Además, mantienen tu página renderizada en una única dirección web. Sin embargo, las características básicas son muy limitadas y estas aplicaciones usualmente no proveen espacio de almacenamiento para recursos (como imágenes).

Prueba con algunos de estos ejemplos y observa cuál es el que mejor se adapta a tu gusto:

- JSFiddle
- Glitch
- JS Bin
- CodePen



Publicar a través de GitHub

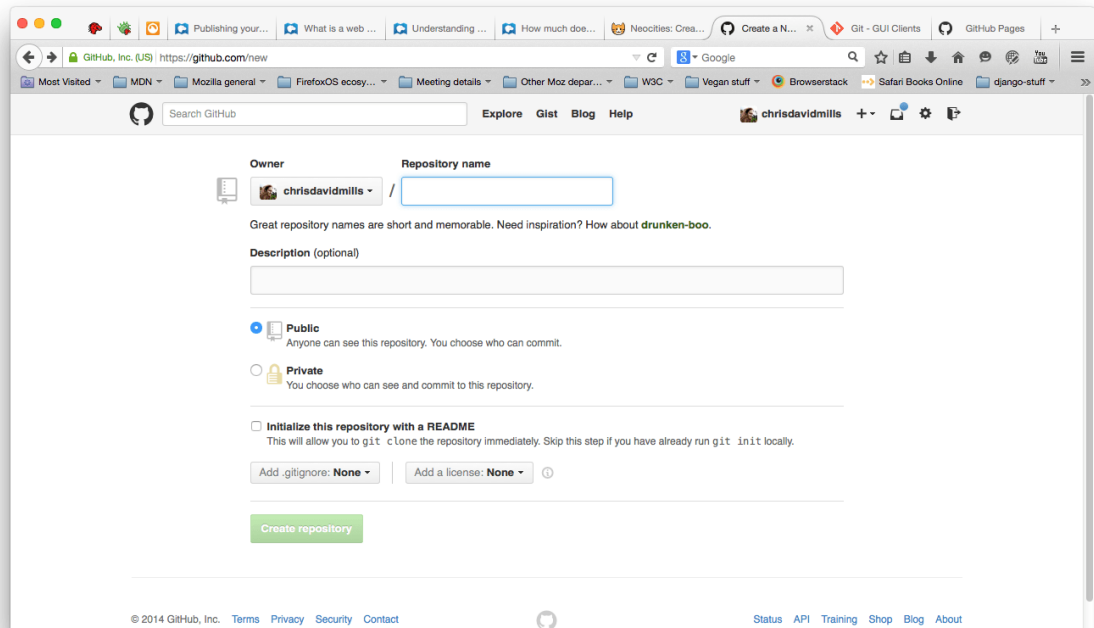
Explicados estos tres métodos veamos ahora cómo publicar fácilmente, de forma muy visual e intuitiva, o bien por medio de comandos, tu sitio a través de GitHub Pages (en inglés).

De manera visual y sin necesidad de más herramientas

Esta no es la única manera, pero sí la que te permite poner manos a la obra inmediatamente.

1. Si aún no lo has hecho da de alta una cuenta en GitHub. Es simple y sencillo, solo debes verificar tu dirección de correo electrónico.
2. Una vez registrado, ingresa a tu cuenta en GitHub.com con tu usuario y contraseña suministrados al crear tu cuenta.
3. A continuación, necesitas crear un nuevo repositorio para tus archivos. Haz clic en el signo más (+) en la parte superior derecha de la página inicial de GitHub y selecciona *New Repository* (Nuevo repositorio).
4. En esta página, en la casilla *Repository name* (Nombre del repositorio), ingresa *usuario.github.io*, donde *usuario* es tu nombre de usuario. Así, por ejemplo,

nuestro amigo Bob Smith ingresaría `bobsmith.github.io`.



5. Opcionalmente escribe una corta descripción de tu sitio web en el campo *Description* para que recuerdes cuál es la temática que tratarás en él y selecciona la casilla de verificación *Public* (Público) si quieres que cualquier persona pueda ver los resultados de las ediciones que haces al sitio web que estás creando.
6. Marca la casilla de verificación *Initialize this repository with a README* (Inicializar este repositorio con un README (LÉAME)). Esto te permitirá clonar inmediatamente el repositorio a tu equipo. ¡Si vas a transferir tus archivos desde tu equipo al servidor de GitHub a través de un cliente de FTP, **no debes realizar este paso**!
7. Da clic en *Create repository* (Crear repositorio).
8. Arrastra y suelta el contenido de la carpeta de tu sitio web en tu repositorio. Cuando termines de pasar el contenido haz clic en *Commit changes* (Confirmar cambios).

Nota: cerciórate que tu carpeta tiene un archivo de nombre `index.html`

9. En tu navegador desplázate a `username.github.io` para ver tu sitio web en línea. Por ejemplo, para el nombre de usuario Bob Smith, escribe `bobsmith.github.io`.

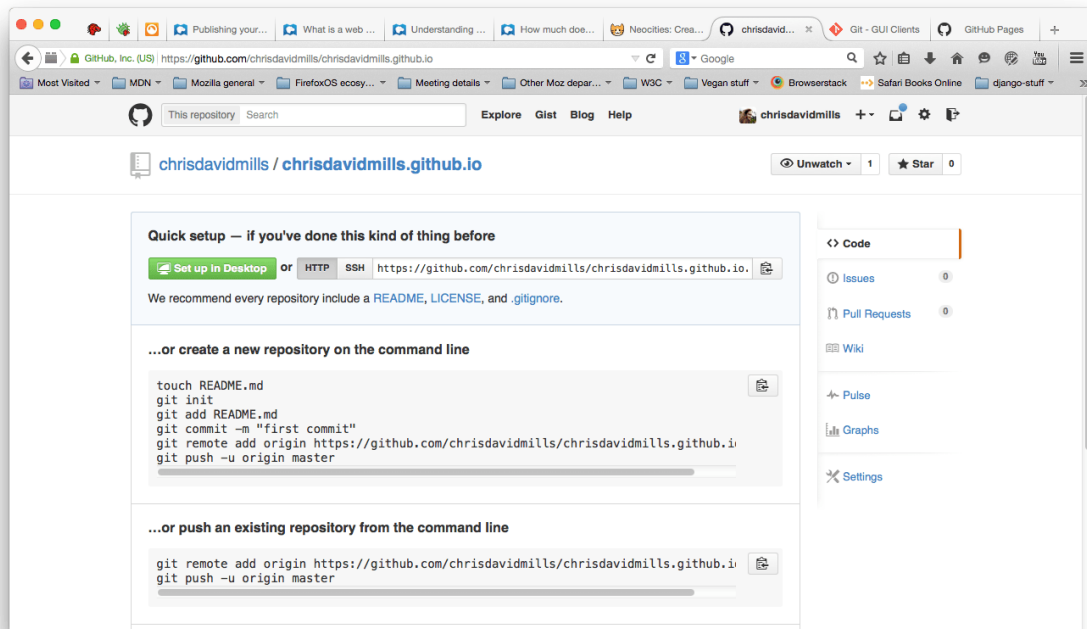
Nota: puede que tu página web tarde unos minutos en entrar en funcionamiento. Si tu sitio web no se muestra inmediatamente, espera unos minutos e inténtalo de nuevo.

Subir tus archivos a GitHub a través de la línea de comandos

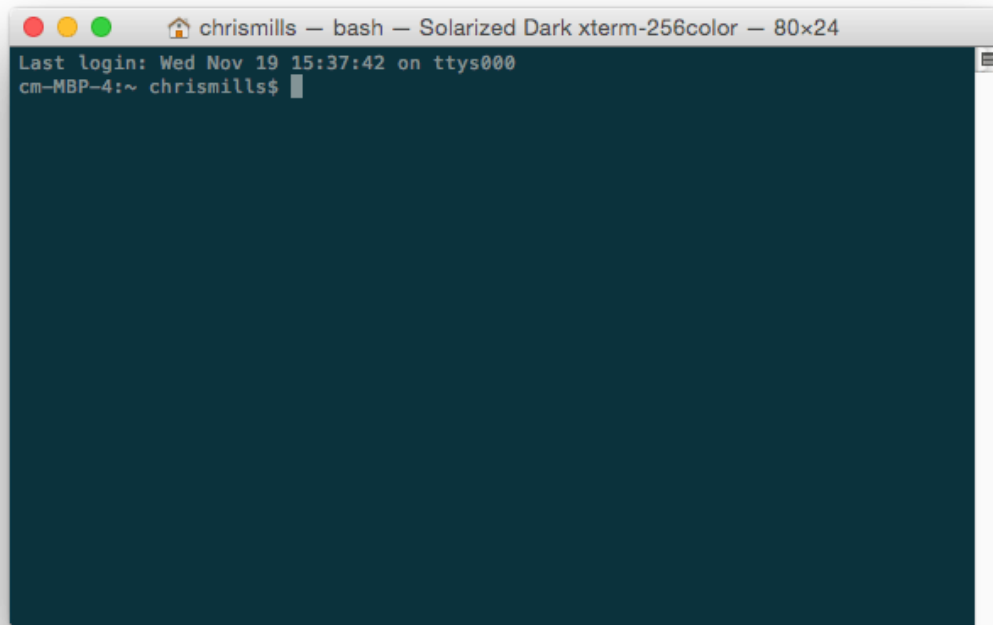
No estamos diciendo que esta es la única manera, o la mejor, de publicar tu sitio, pero es gratis, decentemente simple y abarca algunas nuevas habilidades que encontrarás útiles en adelante.

Antes que nada, descarga e instala Git en tu equipo. Este paso es necesario si vas a trabajar con los archivos de tu página web en él y luego los transferirás al servidor de GitHub.

Sigue los pasos **1 a 5** y el **7** (recuerda omitir el 6) detallados en la anterior sección *De manera visual y sin necesidad de más herramientas*. Una vez hayas dado clic en *Create repository* (Crear repositorio) verás la siguiente ventana (¡no la cierres, más adelante necesitarás copiar información de allí!):



En este punto ya estarás listo para poder utilizar la línea de comandos para subir los archivos de tu repositorio a GitHub. Una *línea de órdenes o de comandos* es una ventana donde escribes comandos que realizarán tareas como crear archivos y ejecutar programas, en lugar de utilizar la interfaz gráfica de usuario. Se debe parecer a algo como esto:



Nota: si no te sientes cómodo utilizando la línea de comandos, podrías considerar usar Git graphical user interface para realizar la misma tarea.

Todos los sistemas operativos vienen con una herramienta de línea de comandos:

- **Windows:** se puede acceder al **Intérprete de comandos** desde el menú que se presenta al pulsar *Win + X* (o abre el menú pulsando el botón secundario del ratón sobre el botón Inicio de Windows en la parte inferior izquierda del escritorio). Advierte que Windows tiene sus propias sintaxis de comandos diferente a las de Linux y MacOS X, así que los siguientes comandos pueden variar para tu máquina.
- **MacOS X:** **Terminal** puede ser hallada en Aplicaciones > *Utilidades*.
- **Linux:** usualmente puedes abrir una terminal con *Ctrl + Alt + T*. Si esto no funciona, busca **Terminal** en la barra de aplicaciones o en el menú.

Aunque este procedimiento pueda parecer un poco aterrador al principio no te preocupes, pronto te darás cuenta de lo básico. Darás órdenes al equipo en el terminal escribiendo un comando y presionando *Intro*.

1. Apunta la línea de comandos a tu directorio sitio-prueba (o como quiera que hayas llamado al directorio que contiene tu sitio web). Para esto utiliza el comando *cd* (es decir, «*change directory*», «*cambiar de directorio*»). Aquí viene lo que deberías teclear si has ubicado tu sitio web en un directorio llamado sitio-prueba en tu escritorio:
2. *cd Desktop/sitio-prueba*

En Windows sería:

```
cd %USERPROFILE%\Desktop\sitio-prueba
```

3. Cuando la línea de comandos esté apuntando dentro del directorio de tu sitio web, teclea el siguiente comando, que indica a la herramienta de git que transforme el directorio en un repositorio de Git:

```
git init
```

4. A continuación, regresa a la ventana del sitio de GitHub que dejaste abierta. En esa página, la sección que interesa es *...or push an existing repository from the command line*. Deberías ver dos líneas de código listadas en esa sección. Copia toda la primera línea, pégala en la línea de comandos y presiona **Intro**. El comando debería verse similar a:

```
git remote add origin https://github.com/bobsmith/bobsmith.github.io.git
```

5. A continuación, ingresa los siguientes dos comandos, presionando **Intro** después de cada uno. Estos preparan el código para cargar a GitHub y pedir a Git administrar estos archivos.

```
git add --all
git commit -m 'agregando archivos a mi repositorio'
```

6. Finalmente, envía el código a GitHub tomando de la página web de GitHub en la que estás el segundo de los dos comandos del paso 3 e introdúcelo en el terminal:

```
git push -u origin master
```

7. Ahora cuando vayas a la dirección de red de tu página GitHub (*usuario.github.io*) en una nueva pestaña del navegador ¡deberías ver tu sitio en línea! Envíala por correo-e a tus amigos y presume de tus capacidades.

Conocer más de GitHub

Si deseas hacer más cambios a tu sitio y enviarlos a GitHub, luego de modificar los archivos, debes ingresar los siguientes comandos (presionando **Intro** después de cada uno) para enviar esos cambios a GitHub:

```
git add --all
git commit -m 'otro commit'
git push
```

Puedes reemplazar el texto *otro commit* con un mensaje más descriptivo respecto a los cambios que hiciste.

Conclusión

En este punto, deberías tener tu página web de ejemplo disponible en una dirección web única. ¡Bien hecho!

