Primeros pasos en la Web

Introducción a la Web es una serie concisa que te presenta los aspectos prácticos del desarrollo web. 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 público objetivo 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 popular 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 funcionalidad lista para usar para que la utilices en tu 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!

¿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.

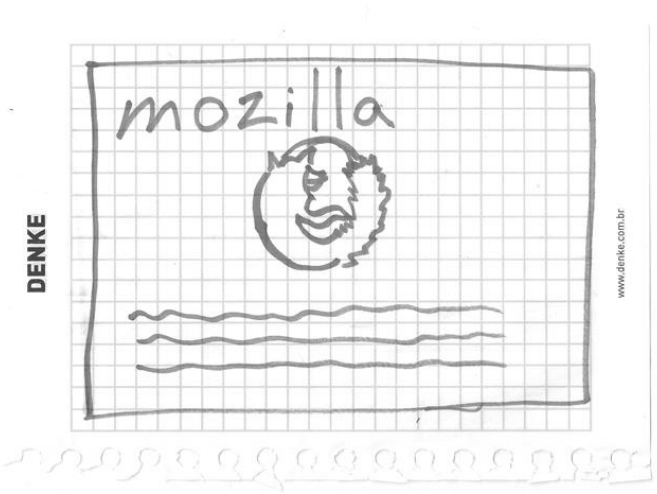
Para comenzar, deberás responder estas preguntas:

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, y puedes ver un ejemplo en el Sistema de diseño de fotones de Firefox.

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

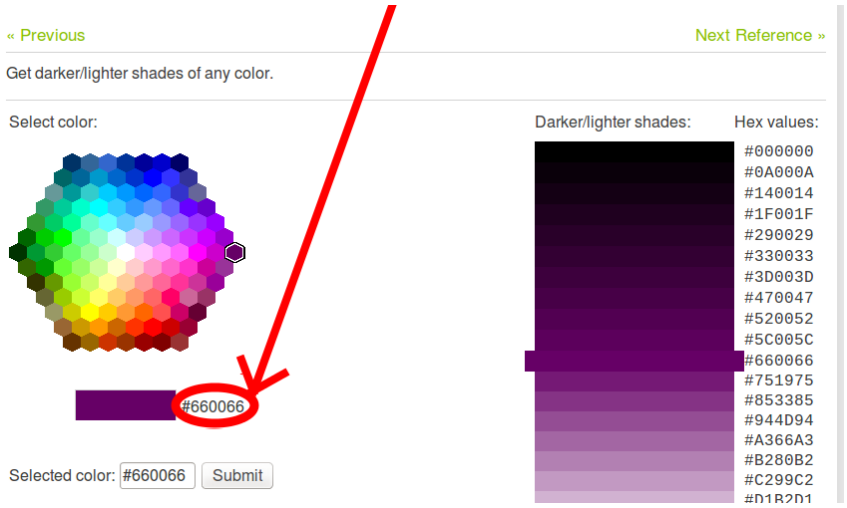
En este punto, es bueno comenzar a reunir el contenido que eventualmente aparecerá en tu página web.

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 cómo `#660066`. Eso se llama *código hexadecimal* (abreviatura de hexadecimal) y representa tu color. Copia el código en un lugar seguro por ahora.



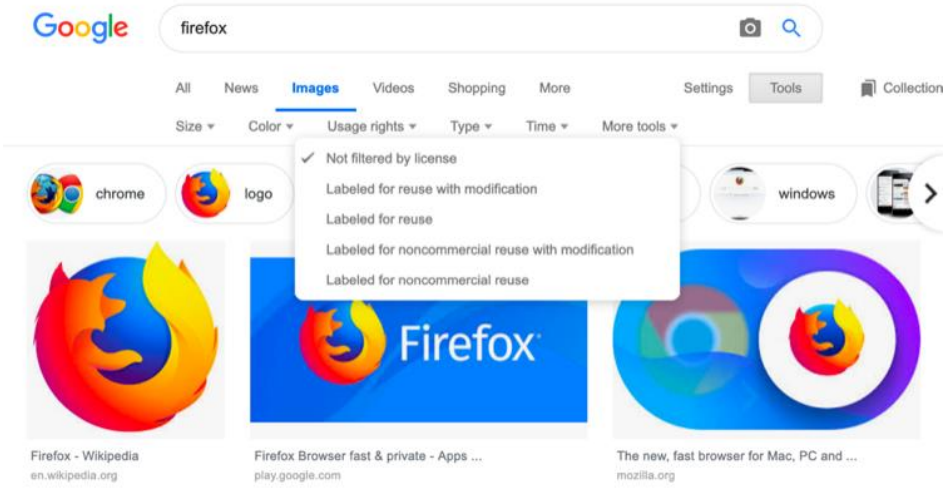
Imágenes

Para elegir una imagen, ve a Imágenes Google y busca algo adecuado.

1. Cuando encuentres la imagen que deseas, haz clic en la imagen para obtener una vista ampliada de la misma.
2. Haz clic con el botón derecho en la imagen, 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

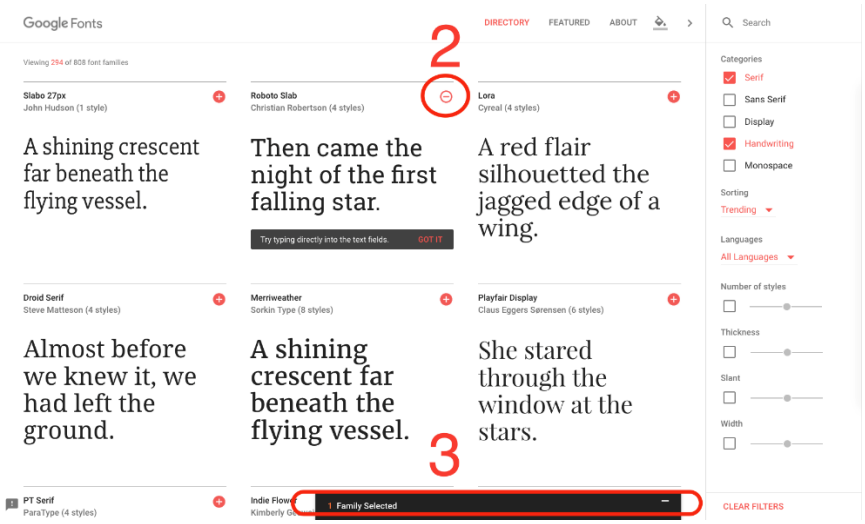
de Google. Haz clic en el botón *Herramientas* y luego en la opción *Derechos de uso* resultante que aparece a continuación. Debes elegir una opción como *Etiquetado para reutilización*.

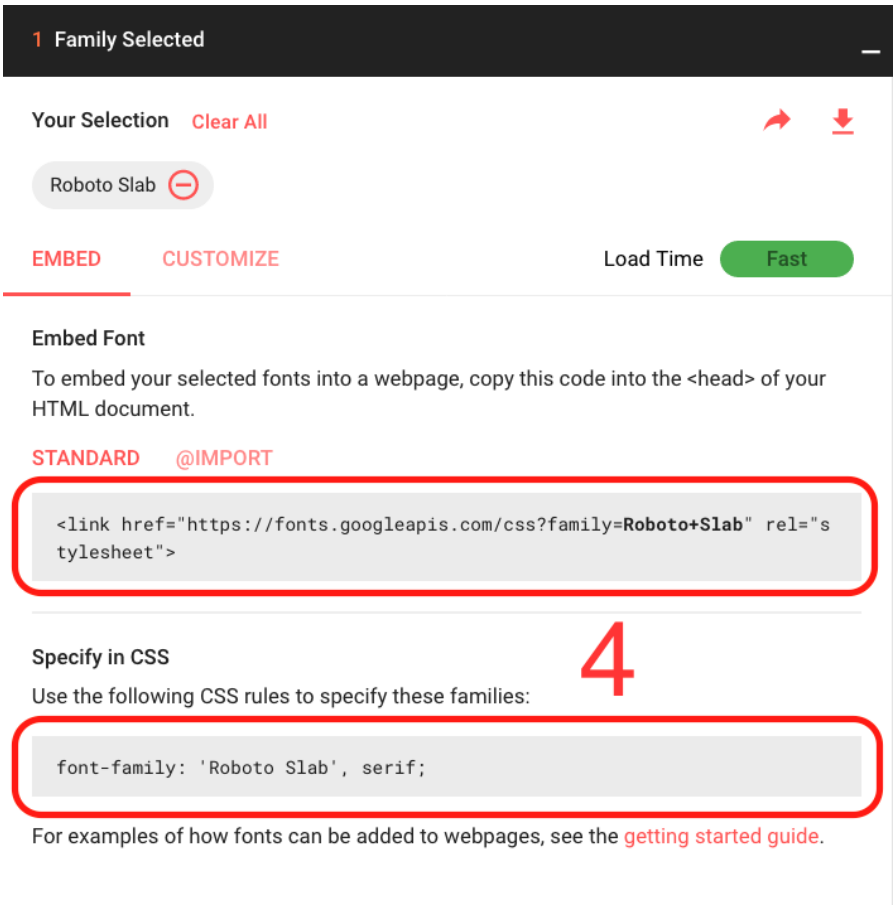


Tipos de letra

Para elegir un tipo de letra:

1. Ve a Google Fonts y desplázate hacia abajo en la lista hasta que encuentres una que te guste. También puedes utilizar los controles de la derecha para filtrar aún más los resultados.
2. Haz clic en el icono "más" (Agregar a) junto al tipo de letra que desees.
3. Haz clic en el botón "**Familia seleccionada*" en el panel en la parte inferior de la página ("*" depende de cuántos tipos de letra hayas seleccionado).
4. En el cuadro emergente, puedes ver y copiar las líneas de código que Google te brinda en tu editor de texto para guardarlas para más adelante.





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 (publicar tu sitio web). 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.

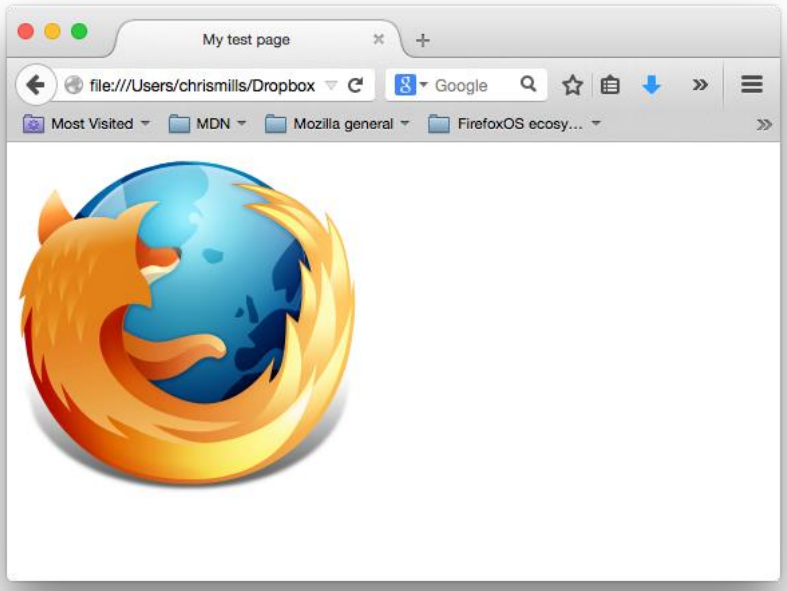
Rutas de archivo

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 `<img src="" alt="Mi imagen de prueba">` 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!



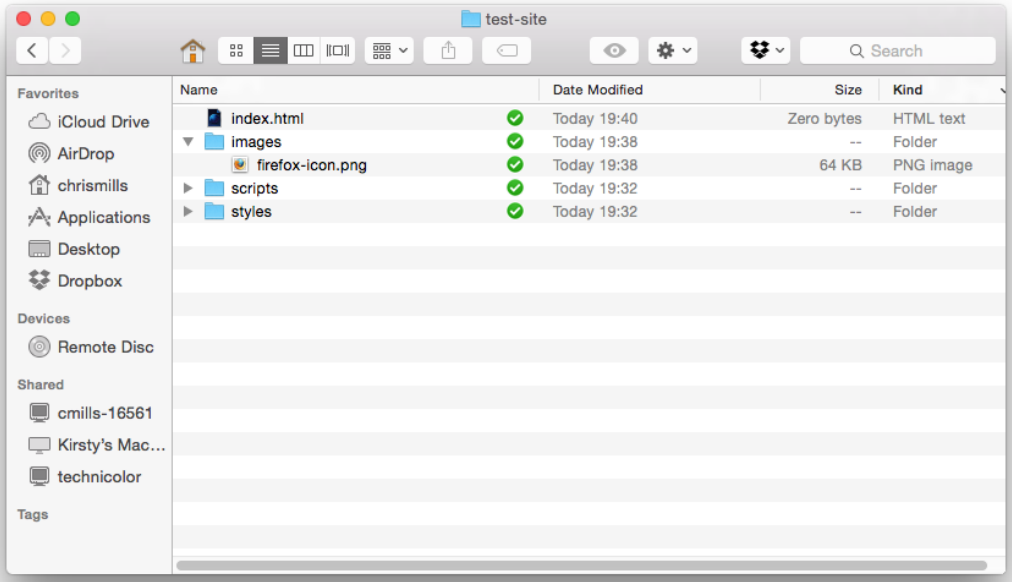
Algunas reglas generales para las rutas de archivo:

- 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.

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:

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

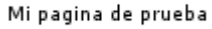
Presta atención nuevamente al elemento *imagen* `<img>`:

```
<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:



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»*.

Intenta 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.

Conclusión

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

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

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 (*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;

}
```

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>`:

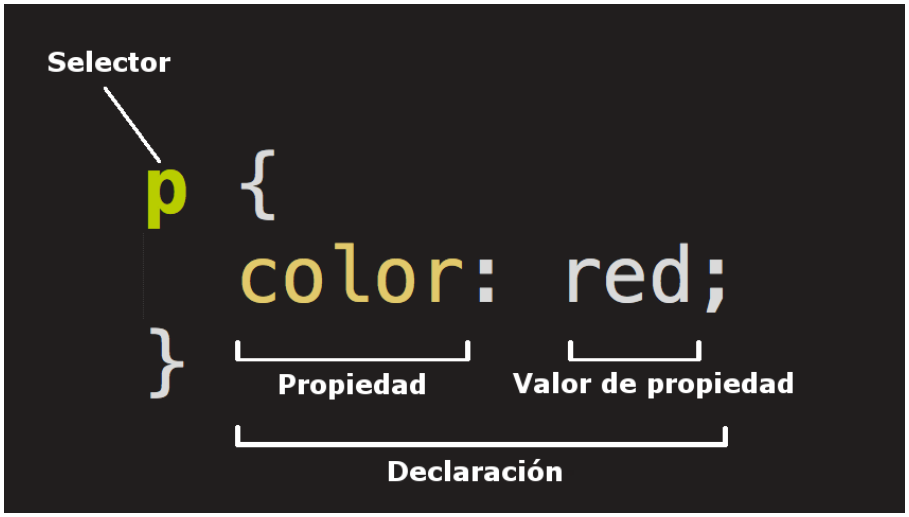
```
<link href="styles/style.css" rel="stylesheet" type="text/css">
```

2. 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

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;

  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 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:

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">`

Nombre del selector	Qué selecciona	Ejemplo
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.

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">
```

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`):

```
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 `/*` 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.

4. 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;

    letter-spacing: 1px;

}
```

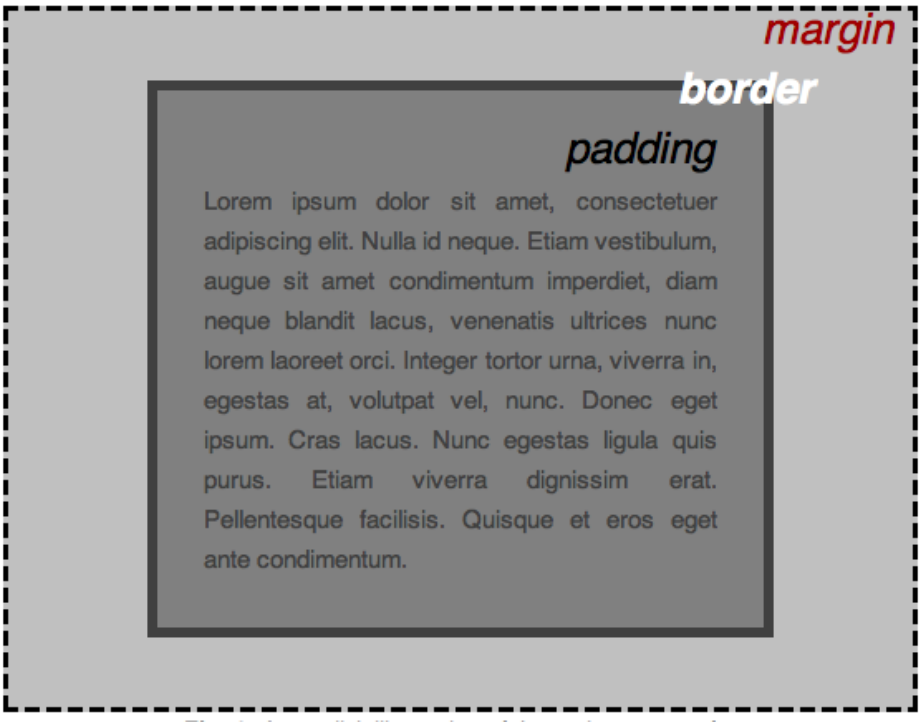
Puedes ajustar estos valores en px para lograr que tu diseño luzca como desees, pero por lo general tu diseño debe verse así:

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 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;

}
```

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 seleccionas dos valores dentro de propiedades como `margin` o `padding`, el primer valor afectará los lados superiores (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;

}
```

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;

}
```

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.

Conclusión

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

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

Fundamentos de 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.

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).

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>
```

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`:

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

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

5. 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;
```

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';
```

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';
```

Advierte que las variables tienen distintos tipos de datos:

Variable	Explicación	Ejemplo
String	Esto es una secuencia de texo conocida como cadena. Para indicar	`let miVariable = 'Bob';`

Variable	Explicación	Ejemplo
	que la variable es una cadena, debes escribirlo entre comillas.	
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
```

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	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';`

Operador	Explicación	Símbolo(s)	Ejemplo
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...');

}
```

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:

```
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 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;

}
```

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». 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!');
```

```
}
```

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 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');

    }

}
```

5. 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>
```

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 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.

4. 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:

```
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` del 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 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;

  }

}
```

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.

Conclusión

Si has seguido las instrucciones en este artículo, tendrás una página que luzca como esta :

Aquí solo has rozado la superficie de JavaScript. Si has disfrutado aprendiendo y deseas avanzar más, visita la Guía de JavaScript.

Publicar 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.

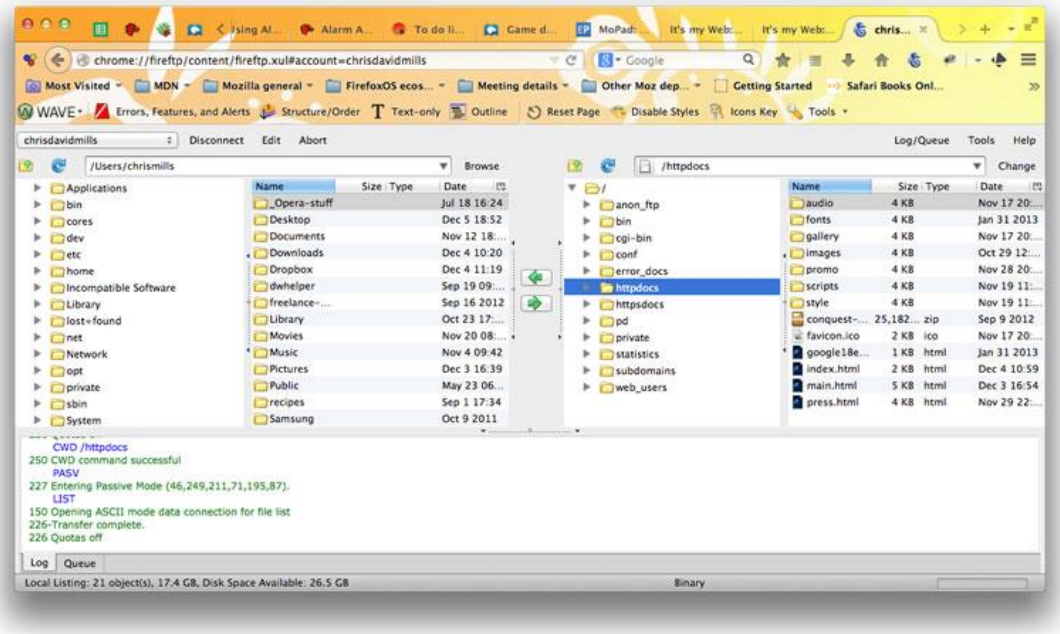
Obtener alojamiento y un nombre de dominio

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 compañia 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

- Hay 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 gratiuitos 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 versiónes** 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? (en inglés).

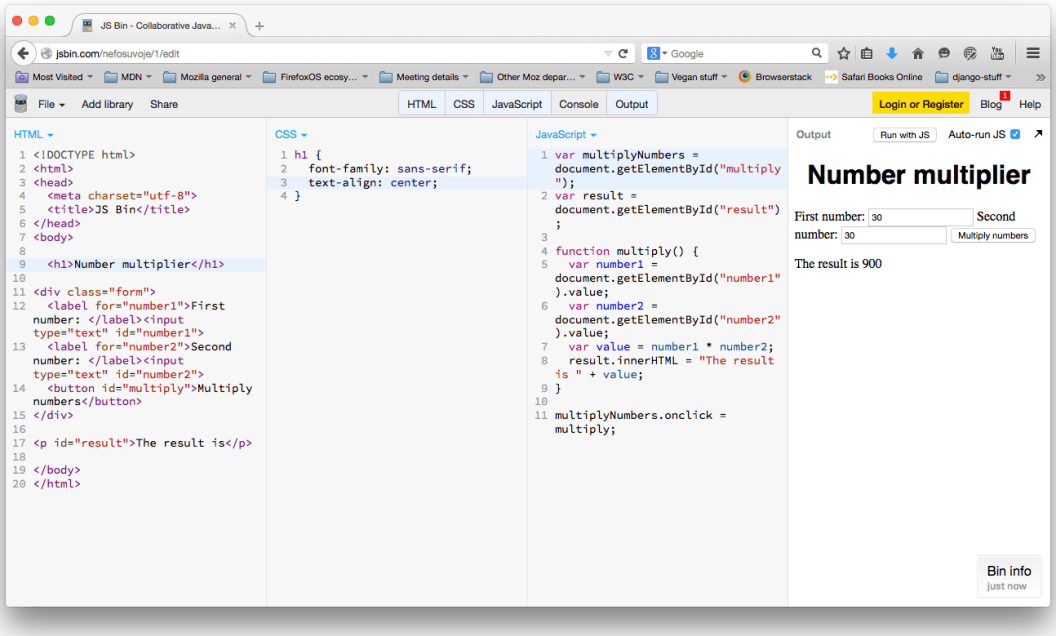
A diferencia de la mayoría de los alojamientos (servicios de *hosting*), tales herramientas son por lo general libres de utilizar, pero solo permiten un conjunto de funciones limitadas.

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

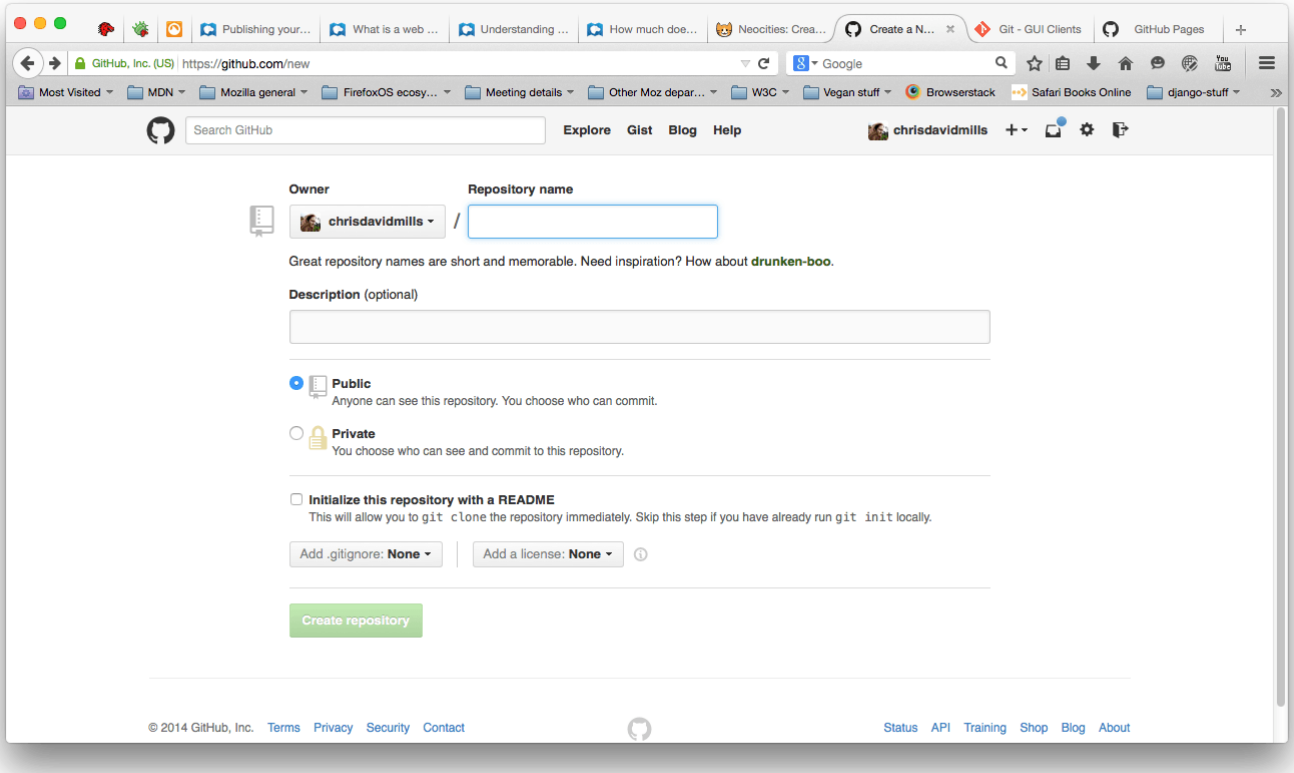
Explicados estos tres métodos veamos ahora cómo publicar fácilmente, de forma muy visual e intuitiva, o bien por medio de comandos, tu sitio a través de GitHub Pages (en inglés).

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 hechoda 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 (como se explica en la sección ***Subir tus archivos a GitHub a través de la línea de comandos***, a continuación), 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`.

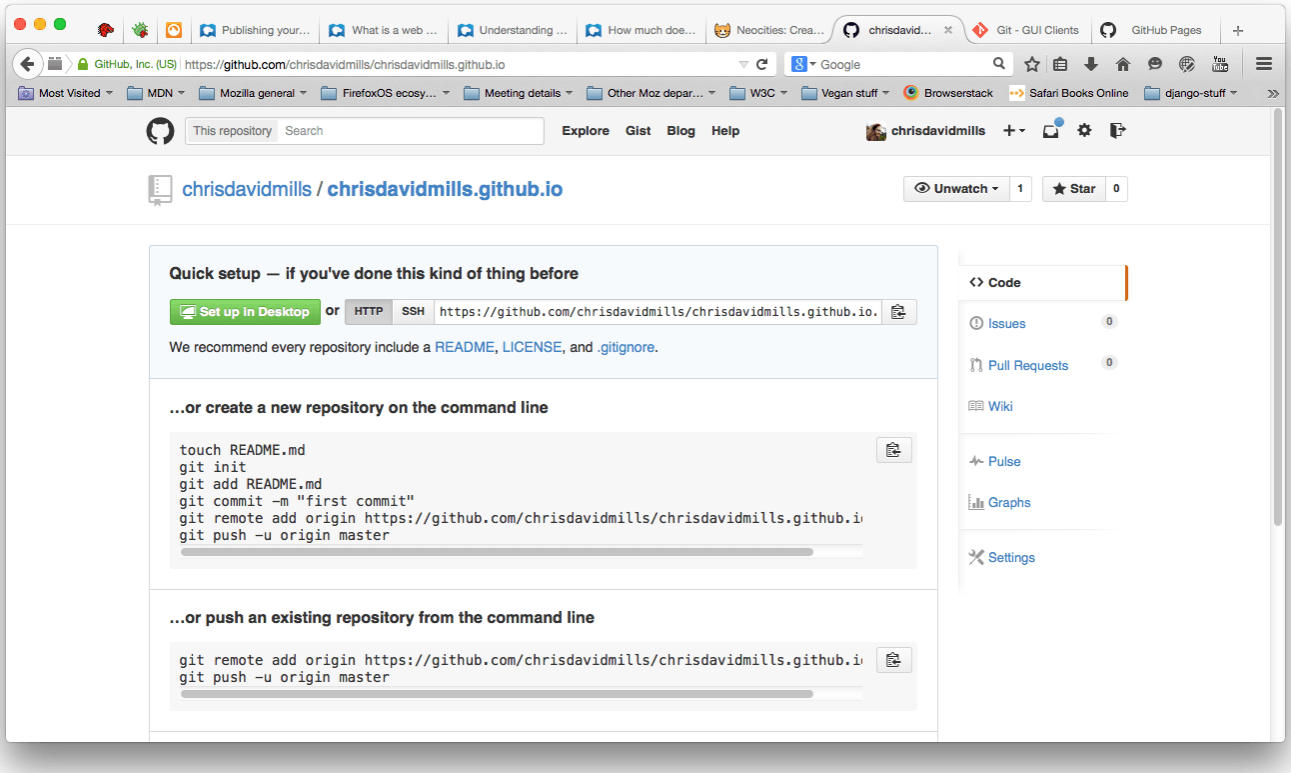
> **Nota**: puede que tu página web tarde unos minutos en entrar en funcionamiento. Si tu sitio web no se muestra inmediatamente, espera unos minutos e inténtalo de nuevo.

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

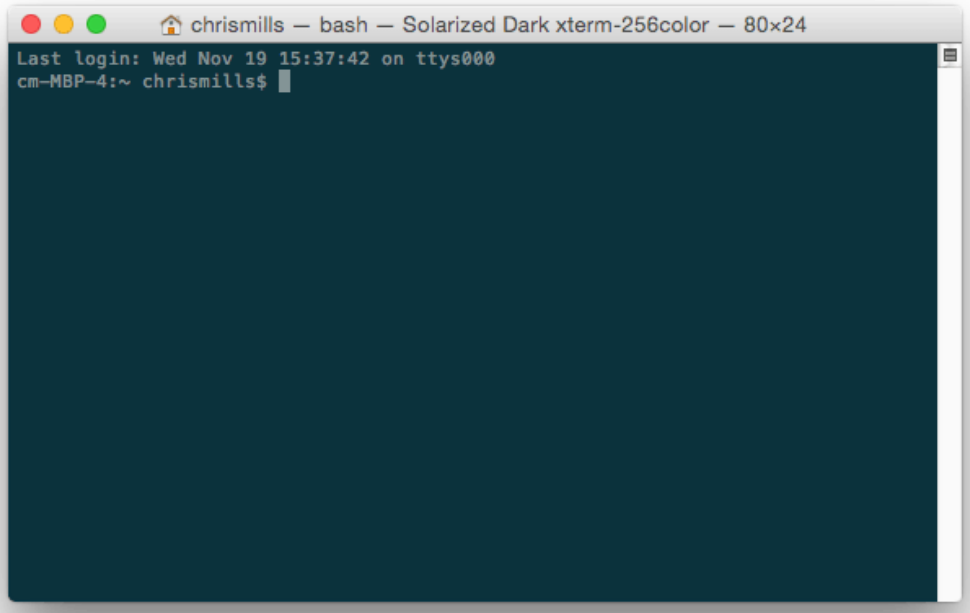
No estamos diciendo que esta es la única manera, o la mejor, de publicar tu sitio, pero es gratis, decentemente simple y abarca algunas nuevas habilidades que encontrarás útiles en adelante.

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:

```
cd Desktop/sitio-prueba
```

   En Windows sería:

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

2. 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
```

3. 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
```

4. 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 Intro


git commit -m 'agregando archivos a mi repositorio' Intro
```

5. 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
```

6. 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 tu maestría.

Nota: has tocado apenas la superficie de Git. Si te quedas atascado la ayuda de GitHub en español te será de gran apoyo.

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 Intro
git commit -m 'otro commit' Intro
git push Intro
```

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!

¿Cómo funciona Internet?

En este artículo se describe lo que es Internet y cómo funciona.

Resumen

Internet es la columna vertebral de la Web, la infraestructura técnica que la hace posible. En lo más básico, Internet es una gran red de computadoras que se comunican simultáneamente.

La historia de internet es algo oscura. Comenzó en la década de 1960 como un proyecto de investigación financiado por el ejército de los EE.UU, y luego se convirtió en una infraestructura pública en la década de 1980 con el apoyo de muchas universidades públicas y empresas privadas. Las distintas tecnologías que soporta internet han evolucionado con el tiempo, pero la forma en que funciona no ha cambiado mucho: Internet es una forma de conectar las computadoras entre sí y asegurar que, pase lo que pase, encuentren una manera de mantenerse conectadas.

Aprendizaje activo

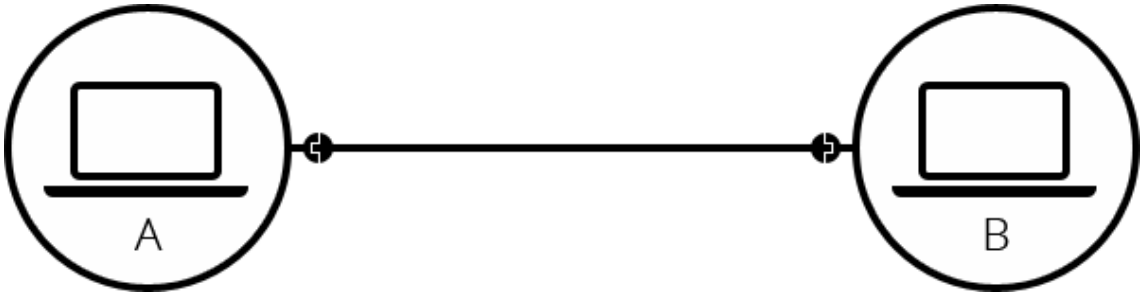
- Cómo funciona internet explicado en 5 minutos: Un vídeo de 5 minutos para entender los fundamentos de internet por Aaron Titus.
- ¿Cómo funciona Internet? Video detallado de 8 minutos de visualización.

Profundizar

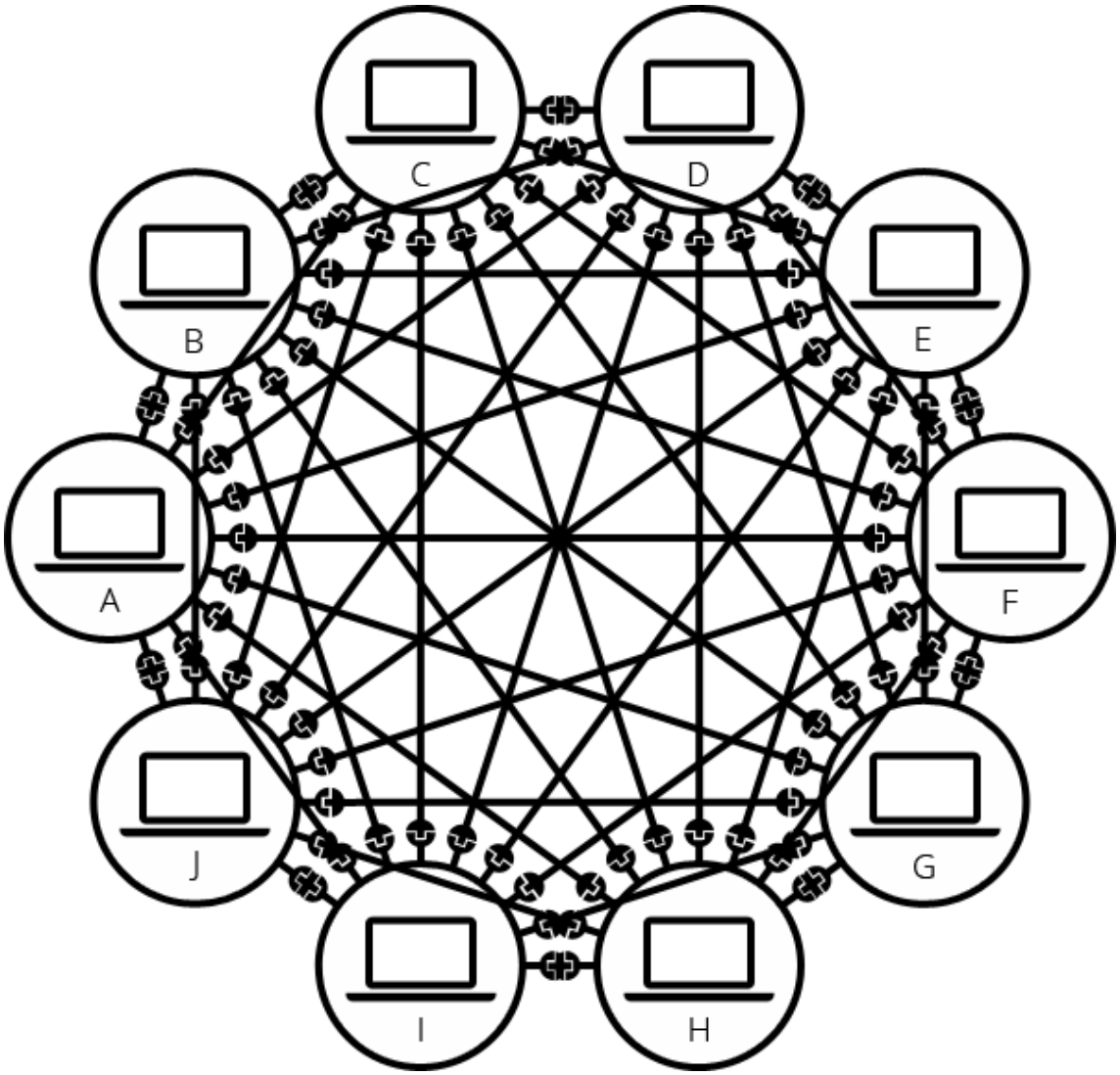
Una simple red

Cuando dos ordenadores necesitan comunicarse, tienes que vincularlos, ya sea físicamente (por lo general con un cable de Ethernet) o de forma inalámbrica (por ejemplo por WiFi o sistema de Bluetooth). Todos los ordenadores modernos pueden soportar cualquiera de este tipo de conexiones.

Nota: En el resto de este artículo, sólo nos referiremos al uso de cables físicos, pero es igualmente aplicable a las redes inalámbricas.

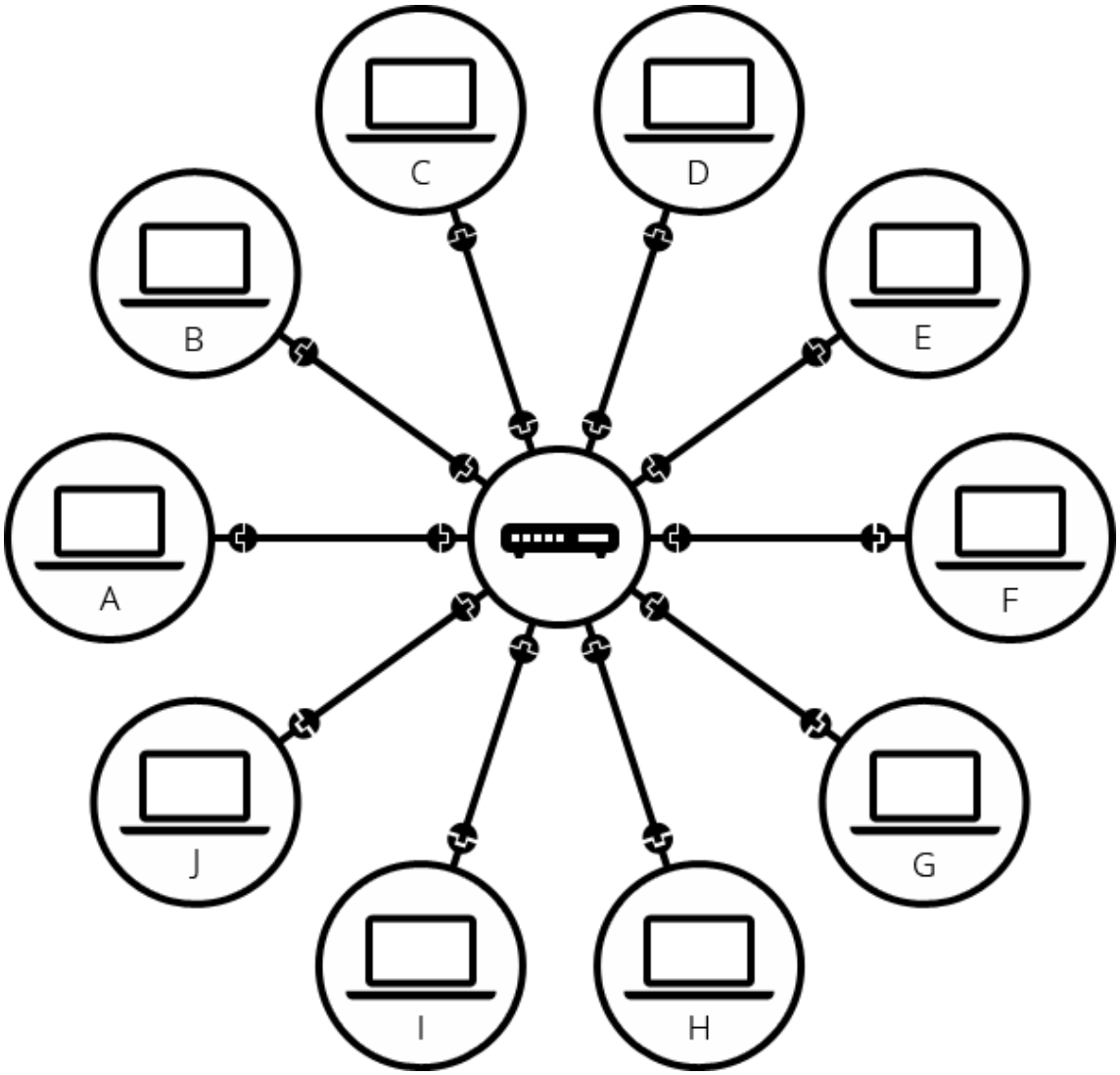


La red no se limita a dos ordenadores, se pueden conectar tantos como se deseen, aunque siendo más complicado cada vez. Por ejemplo, para conectar diez ordenadores, ¡se necesitarían 45 cables y unos nueve conectores por ordenador!



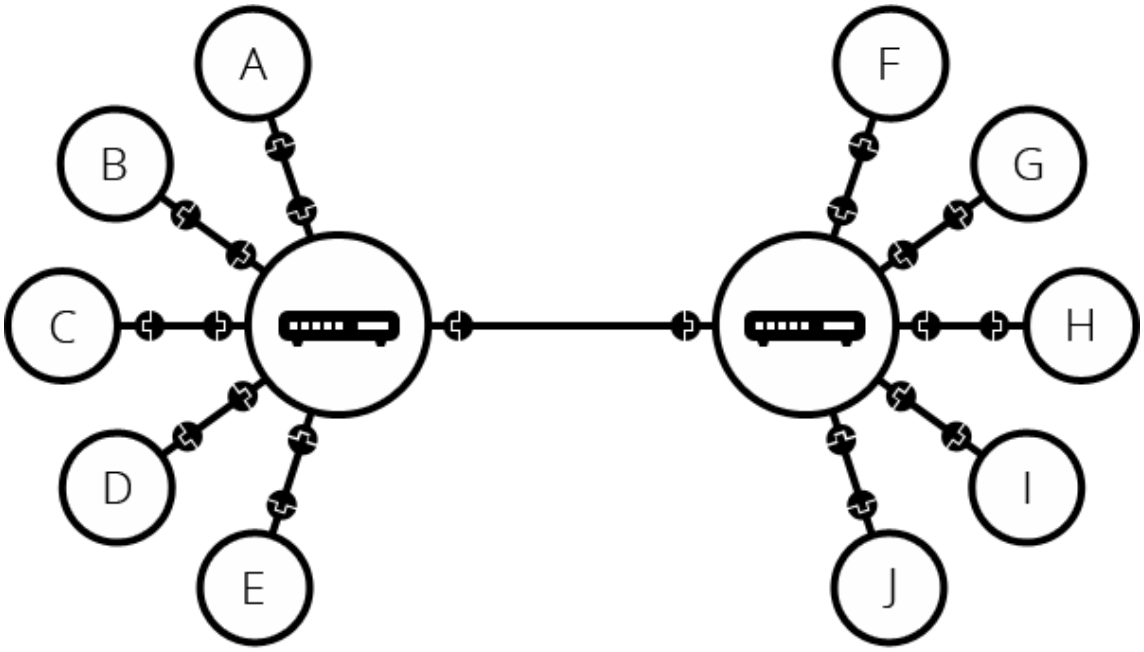
Para resolver este problema, cada ordenador en una red está conectado a una pequeña computadora especial llamada enrutador o router (en inglés). Este enrutador cumple una función: tal como hace un señalizador en una estación de tren, el router se encarga de asegurar que el mensaje enviado desde un ordenador emisor llegue al destino correcto. Para que el ordenador B reciba un mensaje desde el ordenador A, este debe enviarlo primero al router, quien a su vez lo remite al ordenador B asegurándose que dicho mensaje no sea entregado a otro ordenador C.

Una vez que agregamos un enrutador al sistema, nuestra red de 10 ordenadores solo requiere 10 cables: un enchufe para cada ordenador y un enrutador con 10 enchufes.

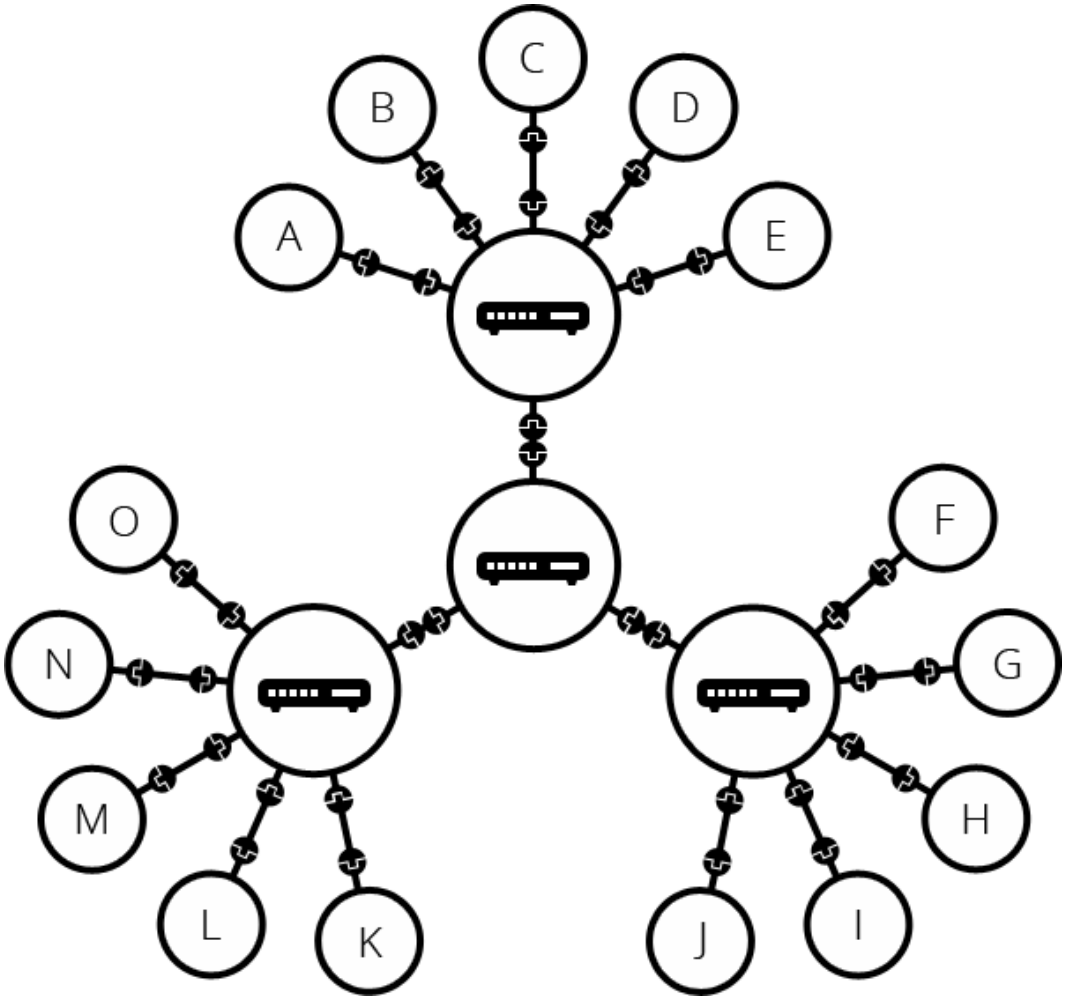


Una red de redes

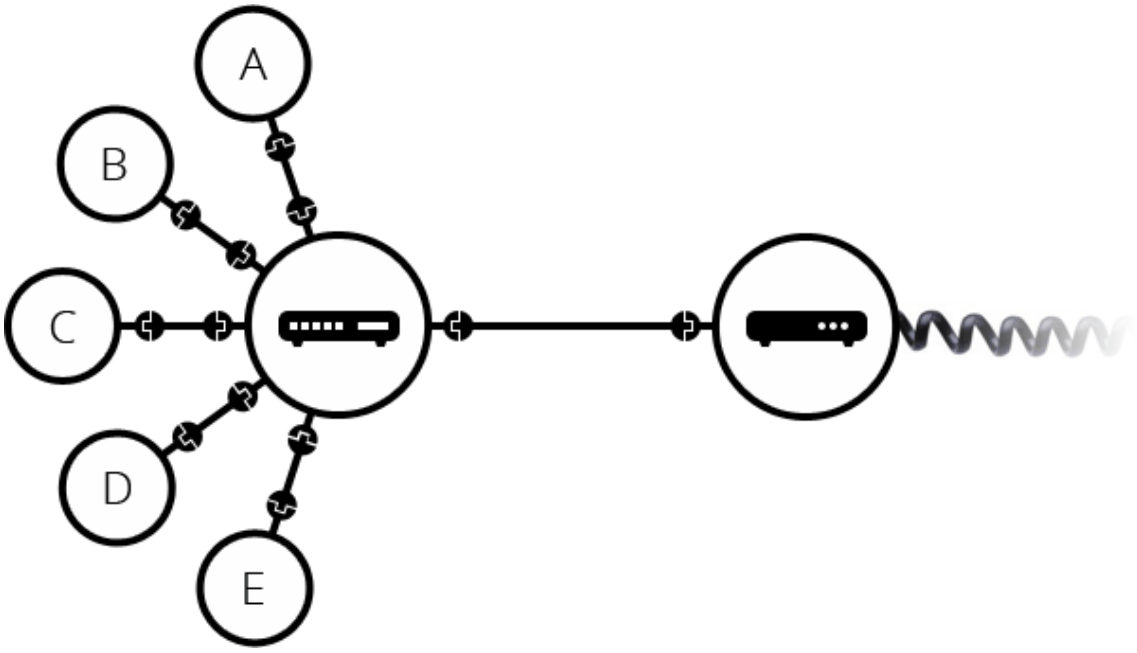
Hasta aquí todo es más o menos simple, pero ¿qué sucede al conectar cientos, miles, millones de ordenadores entre sí? Por supuesto un solo *enrutador* no puede escalar tanto, pero, si lees cuidadosamente, dijimos que un *enrutador* es como un pequeño ordenador, entonces ¿qué nos impide conectar dos *enrutadores* a la vez? Nada: hagámoslo.



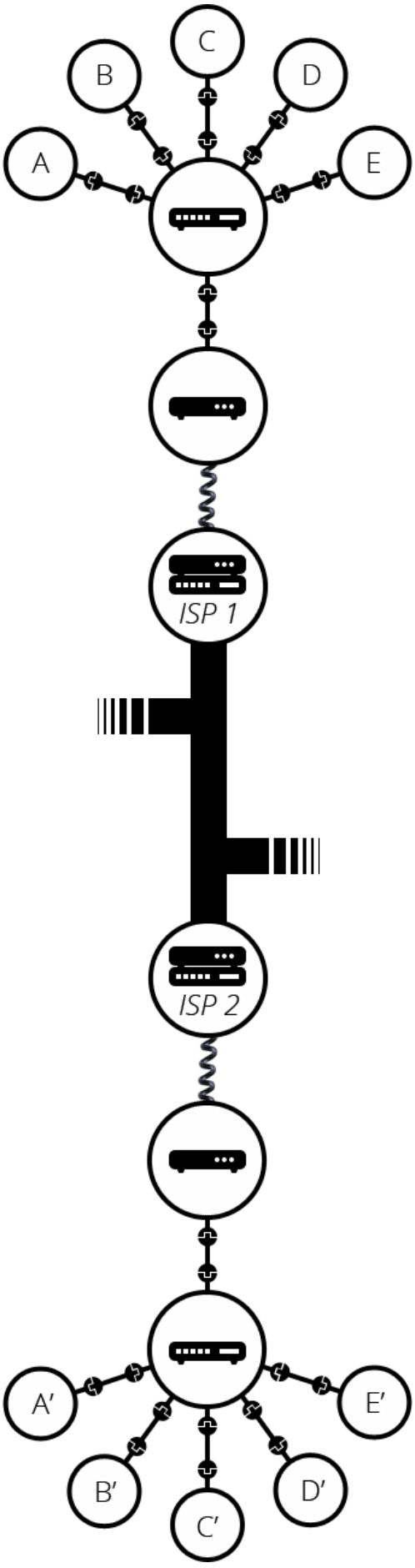
Conectando ordenadores a enrutadores y luego enrutadores entre sí, podemos escalar infinitamente.



Una red así se acerca mucho a lo que llamamos Internet, pero hay algo que nos falta. Construimos esa red para nuestros propios propósitos. Hay otras redes ahí fuera: tus amigos, tus vecinos, cualquiera puede tener su propia red de ordenadores. Pero no es posible instalar cables entre tu casa y el resto del mundo, así que ¿cómo puedes manejar esto? Bueno, ya hay cables conectados a tu casa, por ejemplo, la energía eléctrica y el teléfono. La infraestructura telefónica ya conecta tu casa con cualquier persona en el mundo, así que es el cable perfecto que necesitamos. Para conectar nuestra red a la infraestructura telefónica, necesitamos un equipo especial llamado *modem*.
Este *modem* convierte la información de nuestra red en información manejable por la infraestructura telefónica y viceversa.



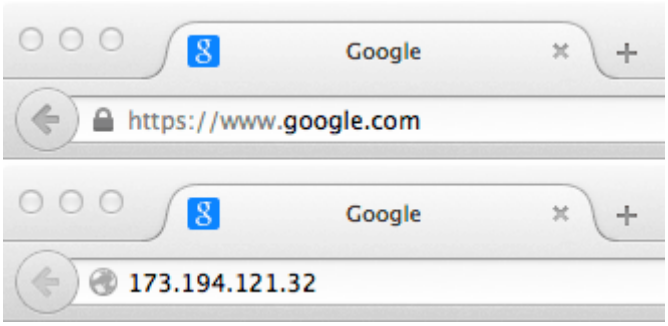
Entonces estamos conectados a la infraestructura telefónica. El siguiente paso es enviar el mensaje desde nuestra red a la red que queremos llegar. Para lograr eso, conectaremos nuestra red a un proveedor de servicios de internet (ISP de sus siglas en inglés Internet Service Provider). Un ISP es una empresa que gestiona algunos enrutadores especiales interconectados, que también pueden acceder a enrutadores de otros ISP. Así, el mensaje de nuestra red es llevada a través de la red de redes de ISP, hasta la red de destino. Internet consiste en toda esta infraestructura de redes.


A
B
C
D
E
ISP 1
ISP 2
A′
B′
C′
D′
E′

Encontrando ordenadores

Si deseas enviar un mensaje a una computadora, debes especificar a cuál. Es por ello por lo que todo ordenador conectado a una red cuenta con una dirección única que lo identifica, llamada “dirección IP” o Protocolo de Internet(IP de sus siglas en inglés *Internet Protocol*). Esta dirección está compuesta por una serie de cuatro números separados por puntos, por ejemplo: `192.168.2.10`.

Para los ordenadores es un identificador simple, pero los humanos tienen mayor dificultad a la hora de recordar y memorizar este tipo de dirección. Con el propósito de convertir esta serie numérica en algo que podamos asociar con mayor facilidad a la dirección IP se utiliza lo que conocemos como *nombre de dominio*. Por ejemplo, `google.com` es el nombre de dominio utilizado para sustituir la dirección IP `173.194.121.32`. Así, usar un nombre de dominio es la manera más fácil para nosotros de identificar un ordenador a través de Internet.



Internet y la web

Como puedes notar, cuando navegamos por la web con un navegador, normalmente utilizamos el nombre de dominio para llegar a un sitio web. ¿Significa eso que Internet y la Web son la misma cosa? No es tan simple. Como vimos, Internet es una infraestructura técnica que permite que miles de millones de ordenadores estén conectadas entre sí. Algunos de estos ordenadores, llamados *servidores web* son capaces de enviar mensajes inteligibles a los navegadores. Por tanto, *Internet* es una infraestructura, mientras que la *Web* es un servicio construido sobre dicha infraestructura. Cabe señalar que existen otros servicios soportados por Internet, como es el correo electrónico e IRC.

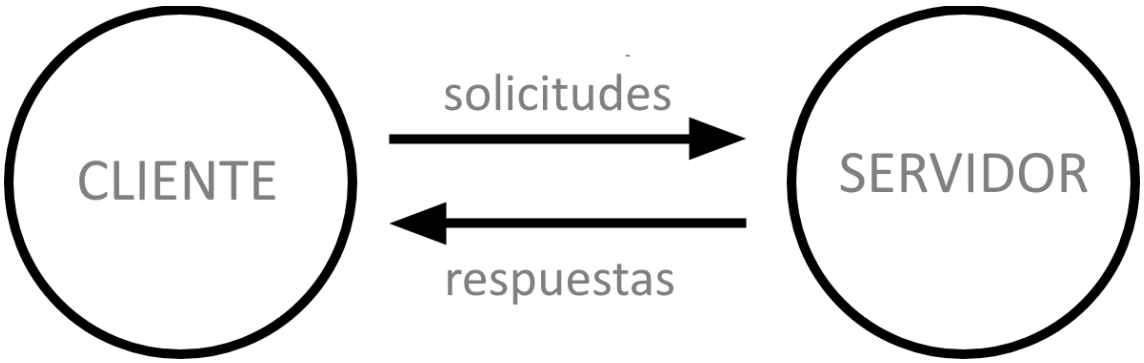
Cómo funciona la web

Cómo funciona la web proporciona una vista simplificada de lo que sucede cuando ves una página web en un navegador web de tu computador o teléfono.

Esta teoría no es esencial para escribir código web a corto plazo, pero en poco tiempo empezarás a beneficiarte realmente al entender lo que está sucediendo en el fondo.

Los clientes y servidores

Las computadoras conectadas a la web se llaman **clientes** y **servidores**. Un diagrama simplificado de cómo interactúan se vería así:



- Los clientes son dispositivos de los usuarios conectados a Internet (por ejemplo, tu ordenador conectado a la red Wi-Fi o el teléfono conectado a la red de telefonía móvil) y el software que se encuentra disponible y permite acceder a Internet en dichos dispositivos (normalmente, un navegador web como Firefox o Chrome).
- Los servidores son computadores que almacenan páginas web, sitios o aplicaciones. Cuando un dispositivo cliente quiere acceder a una página web, una copia de la página web se descarga desde el servidor en el equipo cliente y se muestra en el navegador web del usuario.

Las otras partes de la caja de herramientas

El cliente y el servidor que describimos anteriormente, no cuentan toda la historia. Hay muchas otras partes involucradas y vamos a describirlas a continuación.

Por ahora, imaginemos que la web es un camino. En un extremo de la carretera, está el cliente, que es como tu casa. En el extremo opuesto del camino, está el servidor, que es una tienda en la que deseas comprar algo.

Además del cliente y el servidor, también tenemos que saludar a:

- **Tu conexión a Internet**: permite enviar y recibir datos en la web. Básicamente es el recorrido entre tu casa y la tienda.
- **TCP/IP**: **Protocolo de Control de Transmisión** y **Protocolo de Internet**, son los protocolos de comunicación que definen cómo deben viajar los datos a través de la web. Esto es, los medios de transporte que te permiten hacer un pedido, ir a la tienda y comprar los productos. En nuestro ejemplo, podría ser un coche, una bicicleta o tus propios pies.
- **DNS:** los servidores del **Sistema de Nombres de Dominio** (DNS, por sus siglas en inglés), son como una libreta de direcciones de sitios web. Cuando escribes una dirección web en el navegador, el navegador busca los DNS antes de recuperar el sitio web. El navegador necesita averiguar en qué servidor vive el sitio web y así enviar los mensajes HTTP al lugar correcto (ver más abajo). Esto es como buscar la dirección de la tienda para que puedas llegar a ella.
- **HTTP:** el **Protocolo de Transferencia de Hipertexto** es un protocolo de aplicación que define un idioma para que los clientes y servidores se puedan comunicar. Esto es como el idioma que utilizas para ordenar tus compras.
- **Archivos componentes**: un sitio web se compone de muchos archivos diferentes, que son como las diferentes partes de los productos que

comprarás en la tienda. Estos archivos se dividen en dos tipos principales:

- **Archivos de código**: los sitios web se construyen principalmente con HTML, CSS y JavaScript, aunque te encontrarás con otras tecnologías más adelante.
- **Recursos:** este es un nombre colectivo para el resto de los materiales que conforman un sitio web, como imágenes, música, video, documentos de Word, archivos PDF, etc.

Entonces, ¿qué sucede exactamente?

Cuando escribes una dirección web en el navegador (usando nuestra analogía para ir a la tienda):

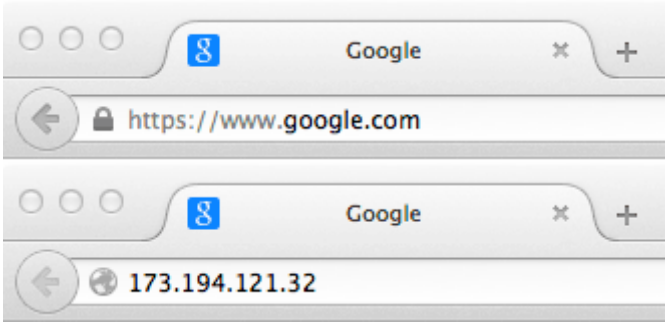
1. El navegador va al servidor DNS y encuentra la dirección real del servidor donde el sitio web vive (encontrar la dirección de la tienda).
2. El navegador envía un mensaje de petición HTTP al servidor, pidiéndole que envíe una copia de la página web para el cliente (ir a la tienda y hacer un pedido). Este mensaje y todos los datos enviados entre el cliente y el servidor se envían a través de tu conexión a Internet usando TCP/IP.
3. Siempre que el servidor apruebe la solicitud del cliente, el servidor enviará al cliente un mensaje «200 OK», que significa, «¡por supuesto que puedes ver ese sitio web! Aquí está.», y comenzará a enviar los archivos de la página web al navegador como una serie de pequeños trozos llamados *paquetes de datos* (la tienda te entrega tus productos y los llevas de regreso a casa).
4. El navegador reúne los pequeños trozos, forma un sitio web completo y te lo muestra (llegas a casa con tus nuevas compras).

Explicación de los DNS

Las direcciones webs reales no son las agradables y fácilmente recordables secuencias que tecleas en la barra de direcciones para encontrar tus sitios webs favoritos. En realidad, se trata de secuencias de números, algo como 63.245.217.105.

Lo anterior se llama dirección IP y representa un lugar único en la web. Sin embargo, no es muy fácil de recordar, ¿verdad? Por eso se inventaron los servidores de nombres de dominio. Estos son servidores especiales que hacen coincidir una dirección web tecleada desde tu navegador («mozilla.org», por ejemplo) con la dirección real del sitio web (IP).

Los sitios webs se pueden acceder directamente a través de sus direcciones IP. Intenta acceder a la página web de Mozilla escribiendo `63.245.217.105` en la barra de dirección de una nueva pestaña en tu navegador. Puedes encontrar la dirección IP de un sitio web escribiendo su dominio en una herramienta como IP Checker.



Explicación de los paquetes

Anteriormente hemos utilizado el término ***paquetes*** para describir el formato en que los datos se envían desde el servidor al cliente. ¿Qué significa esto? Básicamente, que los datos se envían a través de la web como miles de trozos pequeños, permitiendo que muchos usuarios pueden descargar la misma página web al mismo tiempo. Si los sitios web fueran enviados como grandes trozos, sólo un usuario podría descargarlos a la vez, lo que volvería a la web muy ineficiente y poco divertida.

¿Cuál es la diferencia entre la página web, el sitio web, el servidor web y el motor de búsqueda?

En este punto se describen varios conceptos referidos a la web: Páginas web, sitios web, servidores web, y motores de búsqueda. Estos términos con frecuencia son confundidos por recién llegados a la web, o son incorrectamente usados. ¡Vamos a aprender que significa de cada uno!

Resumen

Así como en cualquier área de conocimiento, la web viene con un montón de jerga. No te preocupes, no te abrumaremos con todo esto (tenemos un glosario por si tienes curiosidad). Sin embargo, unos términos básicos que necesitas entender al principio, ya que escuchará estas expresiones todo el tiempo mientras lee. A veces es fácil de confundir estos términos, puesto que hacen referencia a funcionalidades relacionadas pero diferentes. De hecho, a veces veras estos términos mal utilizados en las noticias y en otros lugares.

Cubriremos estos términos y tecnologías con más detalle mientras exploramos más, pero estas definiciones rápidas serán un gran comienzo para ti:

Página web

> Un documento que se puede mostrar en un navegador web como Firefox, Google Chrome, Microsoft Internet Explorer o Edge, o Safari de Apple, A menudo también se puede denominar "páginas web" o simplemente "páginas".

Sitio web

> Es una colección de páginas web que se agrupan y normalmente se conectan de varias maneras. A menudo llamado un "sitio web" o simplemente "sitio".

Servidor web

> Una computadora que aloja un sitio web en Internet.

Motores de búsqueda

> Un sitio web que te ayuda a encontrar páginas web, como Google, Bing o Yahoo o DuDuckGo. Normalmente se accede a los motores de búsqueda a través de un navegador web (por ejemplo, puede realizar búsquedas de motores en búsqueda directamente en la barra de direcciones de Firefox, Chrome, etc.)

Veamos una analogía simple: una biblioteca pública. Esto es lo que generalmente harías al visitar una biblioteca:

1. Buscar en un índice de búsqueda el título del libro que quieres.
2. Tomar nota del número de catálogo del libro.
3. Ir a la sección particular que contiene el libro, buscar el número de catálogo del libro y obtener el libro.

Vamos a comparar la biblioteca con un servidor web:

- La biblioteca es como un servidor web. Tiene varias secciones, que es similar a un servidor web que aloja varios sitios web.
- Las diferentes secciones (ciencias, matemáticas, historia, etc.) en la biblioteca son como sitios web. Cada sección es como un sitio web único (dos secciones no contienen los mismos libros).
- Los libros en cada sección son como páginas web. Un sitio web puede tener varias páginas web, por ejemplo, la sección de Ciencias (el sitio web) tendrá libros sobre calor, sonido, termodinámica, estadísticas, etc. (las páginas web). Las páginas web se pueden encontrar en una ubicación única (URL).
- El índice de búsqueda es como el motor de búsqueda. Cada libro tiene su propia ubicación única en la biblioteca (dos libros no se pueden mantener en el mismo lugar) que se especifica mediante el número de catálogo.

Profundizando

Entonces, vamos a profundizar en cómo estos cuatro términos serán relacionados y por qué a veces se confunden entre sí.

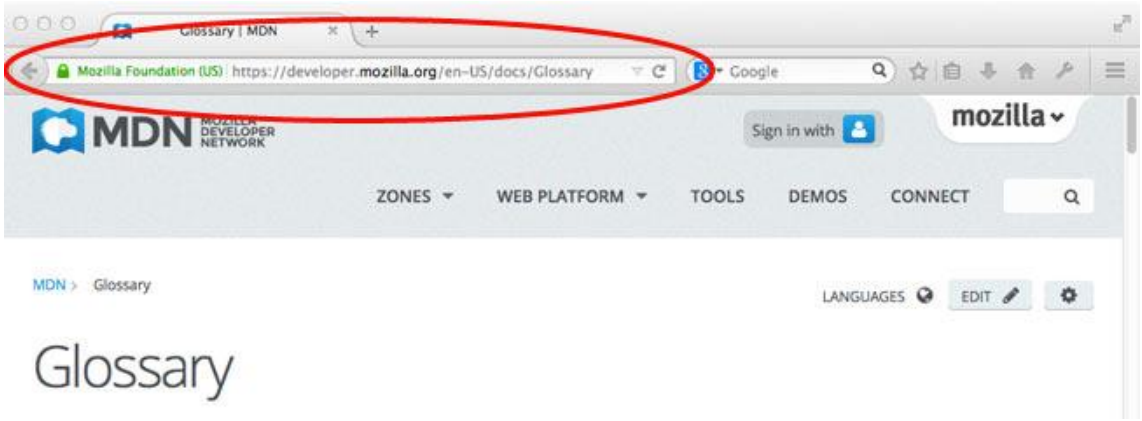
Página web

Una **página web** es un simple documento que puede ser mostrado por un explorador web. Estos documentos están escritos en lenguaje HTML. Una página web puede incluir una variedad de diferentes tipos de recursos, tales como:

- *información de estilos* — controlar la apariencia de una página
- *scripts* — que agrega interactividad a la página
- *medios* — imágenes, sonidos, y vídeos.

Nota: los buscadores pueden mostrar otros tipos de documentos como archivos PDF (en-US) o imágenes, pero el término **página web** específicamente hace referencia a documentos HTML. Por otro lado, solo usamos el termino **document**(documento).

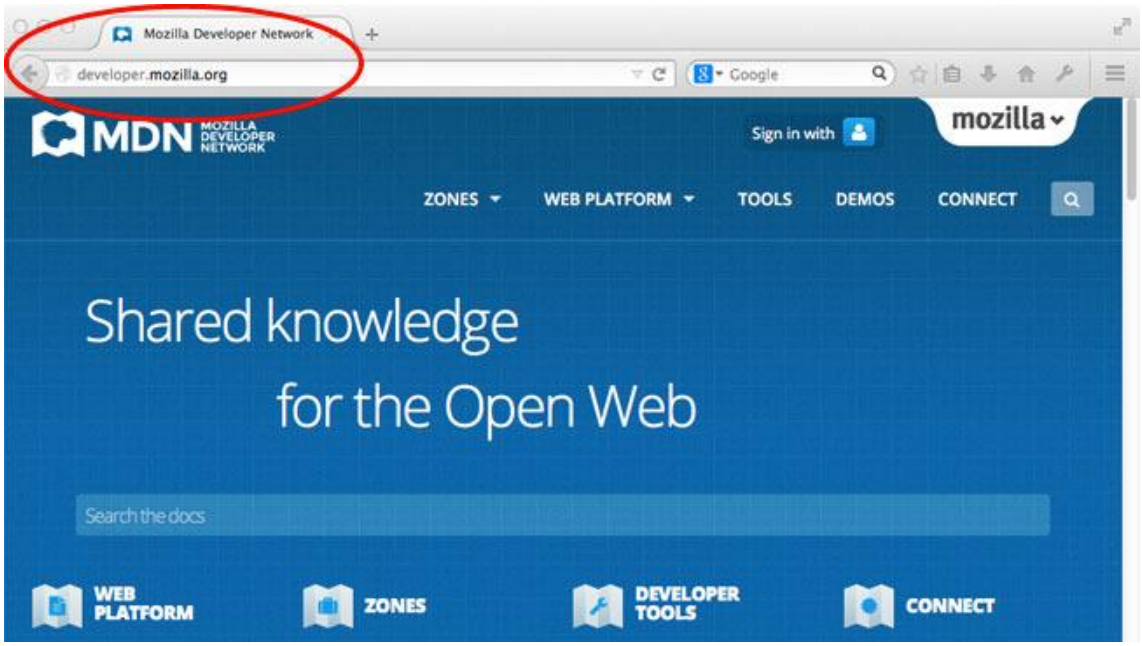
Todas las páginas web disponibles en la red son accesibles mediante una dirección única. Para acceder a una página, simplemente escribe su dirección en la barra de búsqueda de tu navegador:



Sitio web

Un **sitio web** es una colección de páginas web vinculadas (más sus recursos asociados) que comparten un único nombre de dominio. Cada página web de un sitio web determinado proporciona enlaces explícitos—la mayoría del tiempo en forma de parte del texto que se puede hacer clic—que permite al usuario moverse de una página del sitio a otra.

Para acceder a un sitio web, escribe su nombre de dominio en la barra de direcciones de tu buscador, y el mostrará la página principal del sitio web, o *homepage* (casualmente denominada "el home"):



Página web y sitio web son especialmente fácil de confundir cuando un *sitio* contiene una única *página web.* Tales sitios son denominados *sitios de una sola página.*

Servidor web

Un **servidor web** es una computadora que aloja uno o más *sitios web*. “Alojar” significa que todas las *páginas web* y sus archivos soportes están disponibles en esa computadora. El *servidor web* enviará cualquier *página web* desde el *sitio* que hospeda al navegador de cualquier usuario, por cada solicitud del usuario.

No confundir *sito web* y *servidor web*. Por ejemplo, "Mi sitio web no responde", en realidad significa que el *servidor web* no responde y, por lo tanto, el *sitio web* no está disponible. Más importante aún, ya que un servidor web puede alojar varios sitios web, el término *servidor web* nunca se utiliza para designar un sitio, ya que podría causar una gran confusión. En nuestro ejemplo anterior, si dijimos: "Mi servidor web no responde", significa que no hay sitios web en ese servidor web disponibles.

Buscador

Los buscadores son una fuente común de confusión en la web. Un buscador es un tipo especial de sitio web que ayuda a los usuarios a encontrar páginas web de sitios web.

Hay muchos por ahí: Google, Bing, Yandex, DuckDuckGo,y muchos más. Algunos son genéricos, otros están especializados en ciertos temas. Utilice los que prefiera.

Muchos principiantes en la web confunden motores de búsqueda con navegadores. Aclaremos esto: Un ***navegador*** es una parte de software que devuelve y muestra páginas web; un ***buscador*** es un sitio web que ayuda a las personas a encontrar páginas web de otros sitios web. La confusión surge porque, la primera vez que alguien inicia un navegador, el navegador muestra la página principal del motor. Esto tiene sentido, porque, obviamente, lo primero que quieres hacer con un navegador es encontrar una página web para mostrar. No confundas la infraestructura (por ejemplo, el navegador) con el servicio (por ejemplo, el buscador). La distinción te ayudará un poco, pero incluso algunos profesionales hablan imprecisamente, así que no te sientas angustiado por eso.

Aquí hay una instancia de Firefox que muestra un cuadro de búsqueda de Google como su página de inicio predeterminada:

Programación lado servidor

Este módulo contiene una serie de apartados en los que se enseña cómo crear sitios web dinámicos, sitios que entregan información personalizada como respuesta a las peticiones HTTP. El módulo ofrece una introducción genérica a la programación de lado servidor además de guías para principiantes sobre cómo usar frameworks como Django (Python) y Express(Node.js/JavaScript) para crear aplicaciones web básicas.

La mayoría de los principales sitios web utilizan alguna forma de tecnología de lado servidor para presentar dinámicamente datos cuando sean requeridos. Por ejemplo, imagina cuántos productos están disponibles en Amazon e imagina cuántas entradas han sido escritas en Facebook. Presentar todo esto usando páginas estáticas completamente diferentes sería completamente ineficiente, por lo que estos sitios en lugar de ello presentan plantillas estáticas (construidas usando HTML, CSS, y JavaScript), y actualizan dinámicamente los datos presentados dentro de esas plantillas cuando se necesiten, ej. cuando quieres ver un producto diferente en Amazon.

En el mundo moderno del desarrollo web, el aprendizaje del desarrollo de lado servidor es altamente recomendado.

Itinerario de aprendizaje

Los sitios web dinámicos tienden a efectuar un montón de operaciones muy similares (recuperar datos de una base y presentarlos en una página, validar datos introducidos por los usuarios y guardarlos en la base, comprobar los permisos e inicios de sesión de los usuarios, etc.), y se construyen usando web frameworks que hacen muy fácilmente éstas y otras operaciones comunes en un servidor web.

No hace falta ser un experto en codificación lado cliente, pero un conocimiento básico te ayudará a trabajar mejor con los desarrolladores creando tu "front-end" de web en la parte del lado del cliente.

Necesitarás entender "cómo trabaja la web". Te recomendamos que leas primero los siguientes temas:

- ¿Qué es un servidor web?
- ¿Qué software necesito para construir un sitio web?
- ¿Cómo subo ficheros a un servidor web?

Con ese conocimiento básico estarás preparado para recorrer el camino a través de los módulos de esta sección.

¿Qué es un servidor WEB?

En este apartado veremos que son los servidores, cómo funcionan y por qué son importantes.

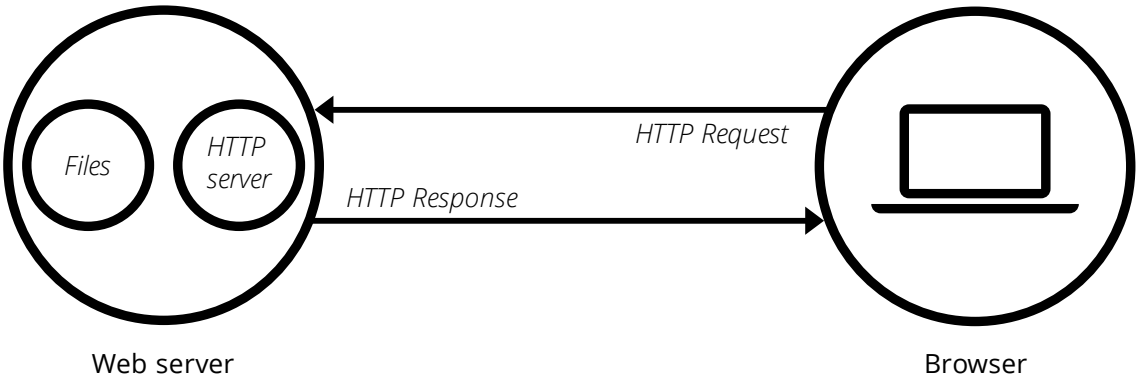
Sumario

Con **"Servidor web"** podemos referirnos a hardware o software, o a ambos trabajando juntos.

1. En cuanto a hardware, un servidor web es una computadora que almacena los archivos que componen un sitio web (ej. documentos HTML , imágenes, hojas de estilos CSS y archivos JavaScript) y los entrega al dispositivo del usuario final. Está conectado a internet y es accesible a través de un nombre de dominio como `mozilla.org`.

2. En cuanto a software, un servidor web tiene muchas partes encargadas del control sobre cómo tienen acceso los usuarios a los archivos, por lo menos un servidor **HTTP***.* Un servidor HTTP es una pieza de software que comprende URLs (direcciones web) y HTTP (el protocolo que tu navegador usa para ver las páginas web).

Al nivel más básico, siempre que un navegador necesite un archivo almacenado en un servidor web, el navegador hará una solicitud al servidor mediante la vía HTTP. Cuando la petición llega al servidor web correcto (hardware), el servidor HTTP (software) envía el archivo antes solicitado, también a través de HTTP.



Para publicar un sitio web, necesitarás un servidor web dinámico o estático.

Un **servidor web estático**, o pila, consiste en una computadora (hardware) con un servidor HTTP (software). Llamamos a este "estático" debido a que el servidor envía los archivos almacenados "tal cual" a tu navegador.

Un **servidor web dinámico** consiste en un servidor web estático con un software extra, lo común es que sea una aplicación servidor y una base de datos. Llamamos a esto *"dinámico"* por qué la aplicación servidor actualiza los archivos almacenados en la base de datos antes de enviarlos mediante el servidor HTTP.

Por ejemplo, para ver la página que ves en tu navegador finalmente, el servidor aplicación puede mostrar el diseño HTML con contenido desde una base de datos. Sitios como Facebook, Instagram o Wikipedia tienen cientos de páginas web, que no son realmente archivos HTML, si no una estructura HTML asociada a una gigantesca base de datos. Esto hace más fácil y rápido el mantenimiento y entrega del contenido.

Inmersión más profunda

Para recuperar una página web, como ya dijimos, su navegador envía una solicitud al servidor web, que procede a buscar el archivo solicitado en su propio espacio de almacenamiento. Al encontrar el archivo, el servidor lo lee, lo procesa según sea necesario y lo envía al navegador. Veamos esos pasos con más detalle.

Alojamiento de archivos (Hosting)

Un servidor web primero debe almacenar los archivos del sitio web, es decir, todos los documentos HTML y sus medios relacionados, incluidas las imágenes, las hojas de estilo CSS, los archivos JavaScript, las fuentes y videos.

Técnicamente, puede alojar todos esos archivos en su propia computadora, pero es mucho más conveniente almacenarlos en un servidor web dedicado que:

- siempre está funcionando.
- siempre está conectado a internet.
- tiene la misma dirección IP todo el tiempo (IP estática).
- es mantenido por un proveedor externo.

Por todas estas razones, encontrar un buen proveedor de alojamiento es una parte clave del desarrollo de su sitio web. Investigue a través de los diversos servicios que ofrecen las compañías y elija uno que se ajuste a sus necesidades y a su presupuesto (los servicios van desde los gratuitos hasta los miles de euros al mes).

Una vez que configura una solución de alojamiento web, solo tiene que cargar sus archivos en su servidor web (veremos cómo hacer esto más adelante).

Comunicación a través de HTTP

En segundo lugar, un servidor web brinda soporte para HTTP (**H**yper**t**ext **T**ransfer **P**rotocol ó Protocolo de Transferencia de Hipertexto). Como su nombre lo indica, HTTP especifica cómo transferir hipertextos (es decir, documentos web vinculados) entre dos computadoras.

Un *protocolo* es un conjunto de reglas para la comunicación entre dos computadoras. HTTP es un protocolo textual y sin estado.

Textual

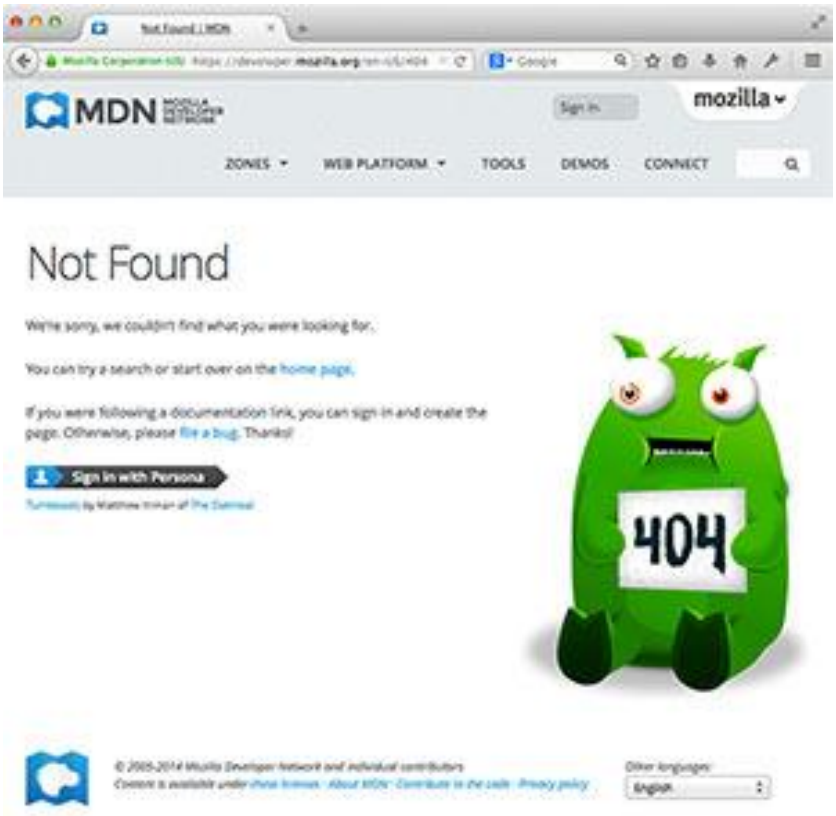
> Todos los comandos son de texto plano y legible para ser leído por humanos.

Sin estado

Ni el servidor ni el cliente recuerdan las comunicaciones anteriores. Por ejemplo, al confiar solo en HTTP, un servidor no puede recordar la contraseña que ingresó ni el paso que está realizando en una transacción. Necesita un servidor de aplicaciones para tareas como esa. (Cubriremos ese tipo de tecnología en otros módulos).

HTTP proporciona reglas claras sobre cómo se comunican un cliente y un servidor. Cubriremos el propio HTTP en unos puntos más adelante. Por ahora, sólo sé consciente de estas cosas:

- Solo los clientes pueden hacer solicitudes HTTP a los servidores. Los servidores solo pueden responder a la solicitud HTTP de un cliente.
- Al solicitar un archivo a través de HTTP, los clientes deben proporcionar la URL del archivo.
- El servidor web debe responder a todas las solicitudes HTTP, al menos con un mensaje de error.



En un servidor web, el servidor HTTP es responsable de procesar y responder las solicitudes entrantes.

1. Al recibir una solicitud, un servidor HTTP primero verifica si la URL solicitada coincide con un archivo existente.
2. Si es así, el servidor web envía el contenido del archivo de nuevo al navegador. Si no, un servidor de aplicaciones construye el archivo necesario.
3. Si ninguno de los procesos es posible, el servidor web devuelve un mensaje de error al navegador, generalmente "404 Not Found". (Ese error es tan común, que muchos diseñadores web pasan bastante tiempo diseñando páginas de error 404 como podemos ver en la imagen de arriba.)

Contenido Estático vs. Dinámico

En términos generales, un servidor puede entregar contenido estático o dinámico. "Estático" significa "servido como está". Los sitios web estáticos son los más fáciles de configurar, por lo que le sugerimos que convierta su primer sitio en un sitio estático.

"Dinámico" significa que el servidor procesa el contenido o incluso lo genera desde una base de datos. La mayoría de las veces el encargado de realizar estas tareas suele ser un servidor de aplicaciones. Esta solución proporciona más flexibilidad, pero se vuelve más difícil de manejar, lo que hace que sea mucho más complejo desarrollar el sitio web.

Hay tantos servidores de aplicaciones que resulta difícil sugerir uno en particular. Algunos servidores de aplicaciones se adaptan a categorías específicas de sitios web como blogs, wikis o tiendas electrónicas; otros, llamados CMS (Content Management Systems o Sistemas de Gestión de Contenidos), son más genéricos. Si está desarrollando un sitio web dinámico, tómese el tiempo para elegir una herramienta que se adapte a sus necesidades.

¿Qué son los hipervínculos?

En este punto, repasaremos qué son los hipervínculos y por qué son importantes.

Resumen

Los hipervínculos, generalmente llamados enlaces, son un concepto fundamental detrás de la Web. Para explicar qué son los enlaces, debemos retroceder a los conceptos básicos de la arquitectura web.

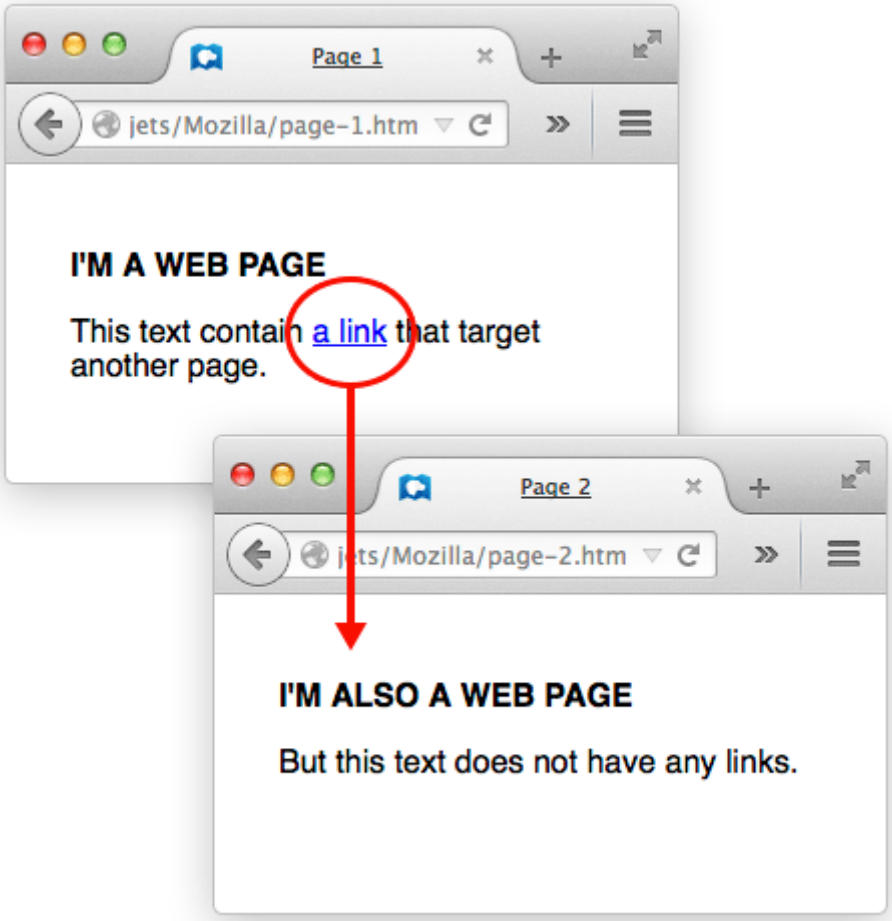
En 1989, Tim Berners-Lee, el inventor de la Web, habló de los tres pilares en los que se basa la Web:

1. URL, un sistema de direcciones que realiza un seguimiento de los documentos web
2. HTTP, un protocolo de transferencia para encontrar documentos cuando se les da su URL
3. HTML, un formato de documento que permite hipervínculos incrustados

Como puede ver en los tres pilares, todo en la Web gira en torno a los documentos y cómo acceder a ellos. El propósito original de la Web era proporcionar una manera fácil de alcanzar, leer y navegar a través de documentos de texto. Desde entonces, la Web ha evolucionado para proporcionar acceso a imágenes, videos y datos binarios, pero estas mejoras apenas han cambiado los tres pilares.

Antes de la Web, era bastante difícil acceder a los documentos y pasar de uno a otro. Al ser legibles por humanos, las URL ya facilitaron las cosas, pero es difícil escribir una URL larga cada vez que desee acceder a un documento. Aquí es donde los hipervínculos revolucionaron todo. Los enlaces pueden correlacionar cualquier cadena de texto con una URL, de modo que el usuario pueda alcanzar instantáneamente el documento de destino activando el enlace.

Los enlaces se destacan del texto circundante al estar subrayados y en texto azul. Toque o haga clic en un enlace para activarlo, o si usa un teclado, presione Tab hasta que el enlace esté enfocado y presione Entrar o la barra espaciadora.



Los enlaces son el avance que hizo que la Web fuera tan útil y exitosa. En el resto de los puntos, discutimos los diversos tipos de enlaces y su importancia para el diseño web moderno.

Profundizando

Como dijimos, un enlace es una cadena de texto vinculada a una URL, y usamos enlaces para permitir saltar fácilmente de un documento a otro. Dicho esto, hay algunos matices que vale la pena considerar:

Tipos de enlaces

Enlace interno

> Un enlace entre dos páginas web, donde ambas páginas pertenecen al mismo sitio web, se denomina enlace interno. Sin enlaces internos, no existe un sitio web (a menos, por supuesto, que sea un sitio web de una página).

Enlace externo

> Un enlace desde su página web a la página web de otra persona. Sin enlaces externos, no hay Web, ya que la Web es una red de páginas web. Utilice enlaces externos para proporcionar información además del contenido disponible a través de su página web.

Enlaces entrantes

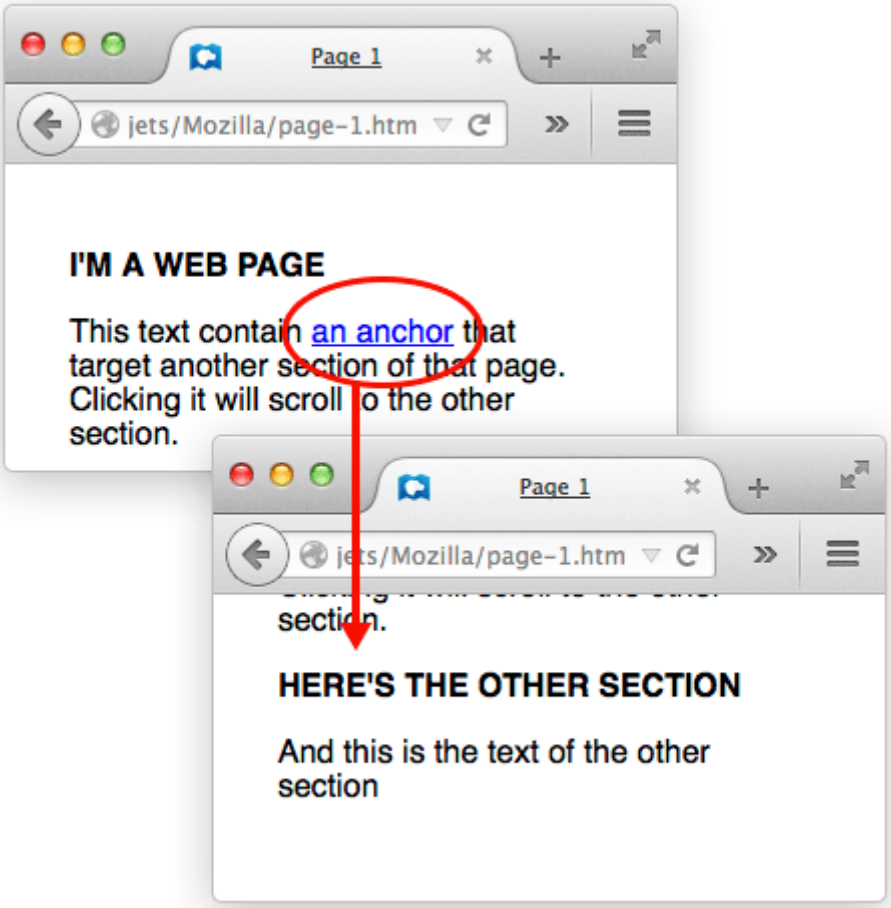
> Un enlace desde la página web de otra persona a su sitio. Es lo contrario de un enlace externo. Tenga en cuenta que no tiene que volver a vincular cuando alguien vincula a su sitio.

Cuando esté creando un sitio web, concéntrese en los enlaces internos, ya que estos hacen que su sitio sea utilizable. Encuentre un buen equilibrio entre tener demasiados enlaces y muy pocos. Hablaremos sobre el diseño de la navegación del sitio web en otra asignatura, pero como regla general, cada vez que agregue una nueva página web, asegúrese de que al menos una de sus otras páginas enlaza con esa nueva página. Por otro lado, si su sitio tiene más de aproximadamente diez páginas, es contraproducente vincular a cada página desde cualquier otra página.

Cuando comienzas, no tienes que preocuparte tanto por los enlaces externos y entrantes, pero son muy importantes si quieres que los motores de búsqueda encuentren tu sitio (ver más abajo para más detalles).

Anclas

La mayoría de los enlaces vinculan dos páginas web. Las anclas unen dos secciones de un documento. Cuando sigue un enlace que apunta a un ancla, su navegador salta a otra parte del documento actual en lugar de cargar un nuevo documento. Sin embargo, crea y utiliza anclajes de la misma manera que otros enlaces.



Enlaces y motores de búsqueda

Los enlaces son importantes tanto para los usuarios como para los motores de búsqueda. Cada vez que los motores de búsqueda rastrean una página web, indexan el sitio web siguiendo los enlaces disponibles en la página web. Los motores de búsqueda no solo siguen enlaces para descubrir las distintas páginas del sitio web, sino que también usan el texto visible del enlace para determinar qué consultas de búsqueda son apropiadas para llegar a la página web de destino.

Los enlaces influyen en la facilidad con que un motor de búsqueda se vinculará a su sitio. El problema es que es difícil medir las actividades de los motores de búsqueda. Las empresas, naturalmente, quieren que sus sitios tengan un alto ranking en los resultados de búsqueda. Sabemos lo siguiente acerca de cómo los motores de búsqueda determinan el rango de un sitio:

- El texto visible de un enlace influye en qué consultas de búsqueda encontrarán una URL determinada.
- Cuantos más enlaces entrantes pueda presumir una página web, más alto se ubica en los resultados de búsqueda.

- *Los enlaces externos influyen en el ranking de búsqueda de las páginas web de origen y destino, pero no está claro cuánto.*

SEO (optimización de motores de búsqueda) es el estudio de cómo hacer que los sitios web tengan un alto ranking en los resultados de búsqueda. Mejorar el uso de enlaces de un sitio web es una técnica útil de SEO.

¿Qué es una URL?

Este artículo habla sobre las Uniform Resource Locators (URLs), explicando qué son y cómo se estructuran.

Resumen

Junto con el Hipertexto y HTTP, las ***URL*** son uno de los conceptos claves de la Web. Es el mecanismo usado por los navegadores para obtener cualquier recurso publicado en la web.

URL significa *Uniform Resource Locator (Localizador de Recursos Uniforme)*. Una URL no es más que una dirección que es dada a un recurso único en la Web. En teoría, cada URL valida, apunta a un único recurso. Dichos recursos pueden ser páginas HTML, documentos CSS, imágenes, etc. En la práctica, hay algunas excepciones, siendo la más común una URL apuntando a un recurso que ya no existe o que ha sido movido. Como el recurso representado por la URL y la URL en si son manejadas por el servidor Web, depende del dueño del servidor web manejar ese recurso y su URL asociada adecuadamente.

Profundizando

Conceptos básicos: anatomía de una URL

Aquí hay algunos ejemplos de URL:

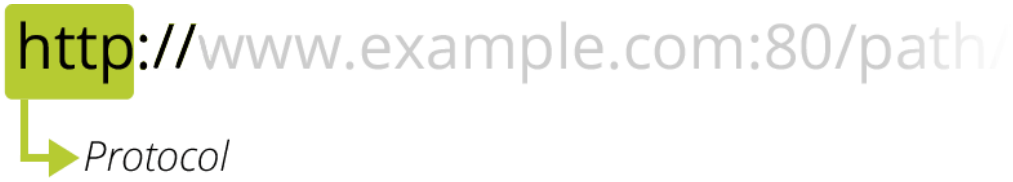
```
https://developer.mozilla.org

https://developer.mozilla.org/en-US/docs/Learn/
```

```
https://developer.mozilla.org/en-US/search?q=URL
```

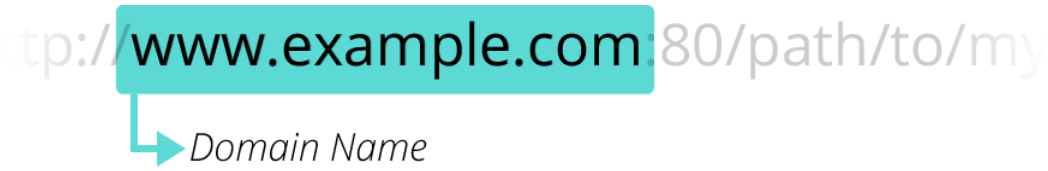
Cualquiera de esas URL se puede escribir en la barra de direcciones de su navegador para indicarle que cargue la página (recurso) asociada.

Una URL está compuesta de diferentes partes, algunas obligatorias y otras opcionales. Veamos las partes más importantes usando la siguiente URL:

```
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
```



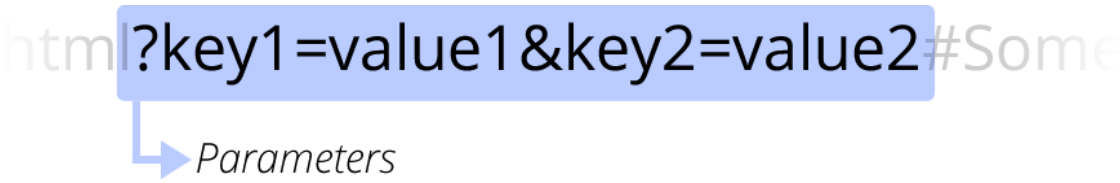
`http` es el protocolo. La primera parte de la URL indica qué protocolo debe usar el navegador. Un protocolo es un método establecido para intercambiar o transferir datos alrededor de una red informática. Por lo general, para sitios web es el protocolo HTTP o su versión segura, HTTPS. La Web requiere uno de estos dos, pero los navegadores también saben cómo manejar otros protocolos como mailto: (para abrir un cliente de correo) o ftp: para manejar la transferencia de archivos, así que no se sorprenda si ve tales protocolos.



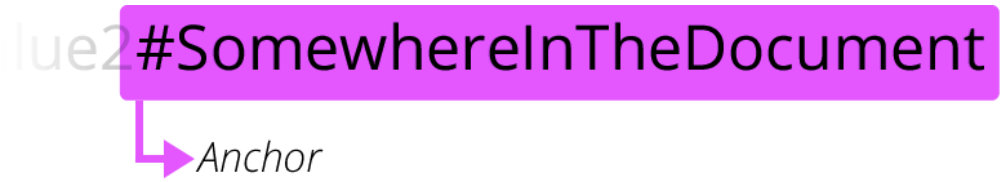
`www.example.com` es el nombre de dominio. Indica a qué servidor web se solicita. Alternativamente, es posible usar directamente una dirección IP, pero debido a que es menos conveniente, no se usa con frecuencia en la Web.

`:80` es el puerto. Indica la "puerta" técnica utilizada para acceder a los recursos en el servidor web. Por lo general, se omite si el servidor web utiliza los puertos estándar del protocolo HTTP (80 para HTTP y 443 para HTTPS) para otorgar acceso a sus recursos. De lo contrario es obligatorio.

`/path/to/myfile.html` es la ruta al recurso en el servidor web. En los primeros días de la Web, una ruta como esta representaba la ubicación de un archivo físico en el servidor web. Hoy en día, es principalmente una abstracción manejada por servidores web sin ninguna realidad física.



`?key1=value1&key2=value2` son parámetros adicionales proporcionados al servidor web. Esos parámetros son una lista de pares clave/valores separados con el símbolo &. El servidor web puede usar esos parámetros para hacer cosas adicionales antes de devolver el recurso. Cada servidor web tiene sus propias reglas con respecto a los parámetros, y la única forma confiable de saber si un servidor web específico está manejando parámetros es preguntando al propietario del servidor web.



`#SomewhereInTheDocument` es un ancla para otra parte del recurso en sí. Un ancla representa una especie de "marcador" dentro del recurso, dando al navegador las instrucciones para mostrar el contenido ubicado en ese lugar "marcado". En un documento HTML, por ejemplo, el navegador se desplazará hasta el punto donde se define el ancla; en un video o documento de audio, el navegador intentará ir a la hora que representa el ancla. Vale la pena señalar que la parte después del #, también conocido como el identificador de fragmento, nunca se envía al servidor con la solicitud.

Algunas partes y reglas extras

Puede pensar en una URL como una dirección de correo postal normal: el protocolo representa el servicio postal que desea utilizar, el nombre de dominio es la ciudad o el pueblo y el puerto es como el código postal; la ruta representa el edificio donde se debe entregar su correo; los parámetros representan información adicional como el número de apartamento en el edificio; y, finalmente, el ancla representa a la persona real a la que ha dirigido su correo.

Cómo usar las URL

Se puede escribir cualquier URL dentro de la barra de direcciones del navegador para acceder al recurso que se encuentra detrás. ¡Pero esto es sólo la punta del iceberg!

El lenguaje HTML hace un uso extensivo de las URL:

- para crear enlaces a otros documentos con el elemento `<a>`;
- para vincular un documento con sus recursos relacionados a través de varios elementos como `<link>` o `<script>`;
- para mostrar recursos como imágenes (con el elemento `<img>`), videos (con el elemento `<video>`), sonido y música (con el elemento `<audio>`), etc.;
- para mostrar otros documentos HTML con el elemento `<iframe>`.

Nota: Al especificar URL para cargar recursos como parte de una página (como cuando se usa <script>, <audio>, <img>, <video> y similares), solo debe usar URL HTTP y HTTPS. El uso de FTP, por ejemplo, no es particularmente seguro y muchos navegadores ya no lo admiten.

Otras tecnologías, como CSS o JavaScript, usan URLs ampliamente, y estos son realmente el corazón de la Web.

URL absolutas vs URL relativas

Lo que vimos arriba se llama URL absoluta, pero también hay algo llamado URL relativa. Examinemos lo que significa esa distinción con más detalle.

Las partes requeridas de una URL dependen en gran medida del contexto en el que se utiliza la URL. En la barra de direcciones de su navegador, una URL no tiene ningún contexto, por lo que debe proporcionar una URL completa (o absoluta), como las que vimos anteriormente. No necesita incluir el protocolo (el navegador usa HTTP de manera predeterminada) o el puerto (que solo se requiere cuando el servidor web de destino está utilizando algún puerto inusual), pero todas las otras partes de la URL son necesarias.

Cuando se usa una URL dentro de un documento, como en una página HTML, las cosas son un poco diferentes. Debido a que el navegador ya tiene la propia URL del documento, puede usar esta información para completar las partes faltantes de cualquier URL disponible dentro de ese documento. Podemos diferenciar entre una URL absoluta y una URL relativa mirando solo la parte de ruta de la URL. Si la parte de ruta de la URL comienza con el carácter "/", el navegador buscará ese recurso desde la raíz superior del servidor, sin referencia al contexto dado por el documento actual.

Veamos algunos ejemplos para aclarar esto.

Ejemplos de URL absolutas

URL Completa (la misma que usamos antes)

```
https://developer.mozilla.org/en-US/docs/Learn
```

Protocolo implícito

```
//developer.mozilla.org/en-US/docs/Learn
```

En este caso, el navegador llamará a esa URL con el mismo protocolo que el utilizado para cargar el documento que aloja esa URL.

Nombre de dominio implícito

```
/en-US/docs/Learn
```

Este es el caso de uso más común para una URL absoluta dentro de un documento HTML. El navegador utilizará el mismo protocolo y nombre de dominio que el utilizado para cargar el documento que aloja esa URL. **Nota**: *no es posible omitir el nombre de dominio sin omitir también el protocolo*.

Ejemplos de URL relativas

Para comprender mejor los siguientes ejemplos, supongamos que las URL se invocan desde el documento ubicado en la siguiente URL:

```
https://developer.mozilla.org/en-US/docs/Learn
```

Sub-recursos

```
Skills/Infrastructure/Understanding_URLs
```

Debido a que la URL no se inicia con `/`, el navegador intentará encontrar el documento en un subdirectorio del que contiene el recurso actual. Entonces, en este ejemplo, realmente queremos llegar a esta URL:

```
https://developer.mozilla.org/en-US/docs/Learn/Skills/Infrastructure/Understanding_URLs
```

Volviendo en el árbol de directorios

```
../CSS/display
```

En este caso, usamos el `../` convención de escritura, heredada del mundo del sistema de archivos UNIX, para decirle al navegador que queremos subir desde un directorio. Aquí queremos llegar a esta URL:

`https://developer.mozilla.org/en-US/docs/Learn/../CSS/display`,

que se puede simplificar a:

`https://developer.mozilla.org/en-US/docs/CSS/display`

URL semánticas

A pesar de su sabor muy técnico, las URL representan un punto de entrada legible para un sitio web. Se pueden memorizar y cualquiera puede ingresarlos en la barra de direcciones de un navegador. Las personas están en el centro de

la Web, por lo que se considera una buena práctica construir lo que se llama *URL semánticas*. Las URL semánticas usan palabras con un significado inherente que cualquier persona puede entender, independientemente de sus conocimientos técnicos.

La semántica lingüística es, por supuesto, irrelevante para las computadoras. Probablemente has visto URL que parecen mashups de caracteres aleatorios. Pero hay muchas ventajas en la creación de URL legibles por humanos:

- Es más fácil para ti manipularlos.
- Aclara las cosas para los usuarios en términos de dónde están, qué están haciendo, qué están leyendo o interactuando en la Web.
- Algunos motores de búsqueda pueden usar esa semántica para mejorar la clasificación de las páginas asociadas.

URI

Un **URI** *(Identificador Uniforme de Recursos de sus siglas en inglés: Uniform Resource Identifier)* es una cadena que se refiere a un recurso. Los más comunes son URLs, que identifican el recurso dando su ubicación en la Web. URNs (en-US), por el contrario, se refiere a un recurso por un nombre, en un espacio de nombres determinados, como el ISBN(International Standard Book Number) de un libro.

Lo abordaremos más adelante cuando veamos el protocolo HTTP, pero simplemente tener en cuenta que existen las URL’s, URN’s y las URI’s.

¿Qué es un nombre de dominio?

En este módulo discutiremos acerca de los nombres de los dominios: qué son, cómo se estructuran y cómo conseguir uno.

Resumen

Los nombres de dominio son una parte clave de la infraestructura de internet. Proporcionan una dirección legible para cualquier servidor web disponible en Internet.

Cualquier computadora conectada a Internet puede ser alcanzada a partir de una dirección IP pública, la cual puede estar formada por 32 bits para el protocolo IPv4 (por lo general se escribe con 4 números separados por puntos entre el 0 y 255, por ejemplo, `173.194.121.32`) o por 128 bits para la versión IPv6 (formada por 8 grupos de 4 números hexadecimales separados por dos puntos, ejemplo `2027:0da8:8b73:0000:0000:8a2e:0370:1337`).

Las computadoras pueden manejar estas direcciones fácilmente, pero a las personas les cuesta saber de quién es el servidor o que servicio ofrece, ya que un número por sí solo no dice mucho. Además, las direcciones IP son difíciles de recordar y pueden cambiarse en cualquier momento. Para resolver estos problemas se usan direcciones que las personas pueden leer, que son intuitivas, fáciles de recordar y dicen mucho sobre el servicio web que ofrecen, se denominan **nombres de dominio**.

Análisis del tema

Estructura de los nombres de dominio

Un nombre de dominio tiene una estructura simple formada por varias partes (puede tener solamente una parte, dos, tres, ...), separadas por puntos y **se leen de derecha a izquierda**:

Cada una de estas partes provee información específica sobre el nombre de dominio completo.

- TLDs locales como `.us`, `.fr`, o `.se` pueden requerir el servicio en un determinado idioma o que esté alojado en un país específico - está hecho para indicar que un recurso está en un idioma o país en particular.

- Los TLDs que contienen `.gov` son solamente permitidos para ser usados por los departamentos de gobierno.
- Los TLDs como `.edu` y `.ac` `.uk` se supone que se usen solamente en instituciones educacionales o académicas.

TLD (Top-Level Domain) Dominio de primer nivel.

El TLD proporciona la información más genérica. Los TLDs les dicen a usuarios el propósito general del servicio que se esconde tras el nombre de dominio. Los TLDs más genéricos (.com, .org, .net) no requieren que los servicios web cumplan ningún criterio particular, pero algunos TLDs hacen cumplir políticas más estrictas por lo que es más claro su propósito. Por ejemplo:

Etiqueta (o componente)

Las etiquetas son lo que siguen al TLD. Una etiqueta puede ser cualquier cosa desde una letra hasta una oración completa. La etiqueta localizada a la derecha antes del TLD puede ser llamada también Dominio de Nivel Secundario, en inglés *Secondary Level Domain* (SLD). Un nombre de dominio puede tener muchas etiquetas (o componentes), no es obligatorio ni necesario tener tres etiquetas para formar un nombre de dominio. Por ejemplo, www.inf.ed.ac.uk es un nombre de dominio correcto. Para cualquier dominio sobre el que se tenga control (por ejemplo mozilla.org), uno puede crear otros nombres de dominio (a veces llamados "subdominios", por ejemplo developer.mozilla.org o iot.mozilla.org).

Comprar un nombre de dominio

¿Quién es propietario de un nombre de dominio?

No se puede “comprar un nombre de dominio”. Se paga por el derecho de usar un nombre de dominio por uno o más años. Se pueden renovar los derechos y la renovación tiene prioridad sobre las aplicaciones de otras personas. Pero nunca se podrá apropiarse un nombre de dominio. Una vez que deja de pagarlo queda libre para que otras personas puedan utilizarlo.

Las compañías llamadas registradores utilizan los registros de nombres de dominio para realizar un seguimiento de la información técnica y administrativa que lo conecta con su nombre de dominio.

Nota : Para algunos nombres de dominio pudiera no ser un registrador quien esté a cargo de mantener el seguimiento. Por ejemplo, cada nombre de dominio bajo `.fire` es manejado por Amazon.

Encontrar un nombre de dominio disponible

Para encontrar si un nombre de dominio dado está disponible,

- Ir a un sitio web de registro de nombres de dominio. La mayoría de ellos, tienen un servicio "whois" que te dice si un nombre de dominio está disponible.
- Alternativamente, si usted usa un sistema con un shell incorporado, escriba el comando `whois`, como se muestra aquí para `mozilla.org`:

```
$ whois mozilla.org

Domain Name:MOZILLA.ORG

Domain ID: D1409563-LROR

Creation Date: 1998-01-24T05:00:00Z

Updated Date: 2013-12-08T01:16:57Z

Registry Expiry Date: 2015-01-23T05:00:00Z

Sponsoring Registrar:MarkMonitor Inc. (R37-LROR)

Sponsoring Registrar IANA ID: 292

WHOIS Server:

Referral URL:

Domain Status: clientDeleteProhibited

Domain Status: clientTransferProhibited

Domain Status: clientUpdateProhibited

Registrant ID:mmr-33684

Registrant Name:DNS Admin
```

```
Registrant Organization:Mozilla Foundation

Registrant Street: 650 Castro St Ste 300

Registrant City:Mountain View

Registrant State/Province:CA

Registrant Postal Code:94041

Registrant Country:US

Registrant Phone:+1.6509030800
```

Como se observa, no se puede registrar `mozilla.org` porque la fundación de Mozilla ya ha sido registrada.

Por otra parte, veamos si se puede registrar `afunkydomainname.org`:

```
$ whois afunkydomainname.org

NOT FOUND
```

Como se observa, el dominio no existe en la base de datos de `whois` (en el momento que se escribió), por lo que pudiéramos pedir registrarlo. ¡Bueno para saber!

Obtener un nombre de dominio

El proceso es bastante sencillo:

1. Ir a un sitio de registro.
2. Generalmente hay un letrero que llama la atención que dice “Get a domain name”. Hacer click en él.
3. Rellenar el formulario con todos los detalles requeridos. Asegúrese de no haber escrito incorrectamente el nombre de dominio deseado. ¡Una vez que esté pagado, es muy tarde!.
4. El registrador te permitirá conocer cuando un nombre de dominio esté correctamente registrado. Dentro de unas pocas horas, todos los servidores DNS habrán recibido su información de DNS.

Nota: En este proceso se le pide su dirección real. Asegúrese de escribirla correctamente, ya que en algunos países los registradores pueden verse obligados a cerrar el dominio si no pueden proporcionar una dirección válida.

Actualización de DNS

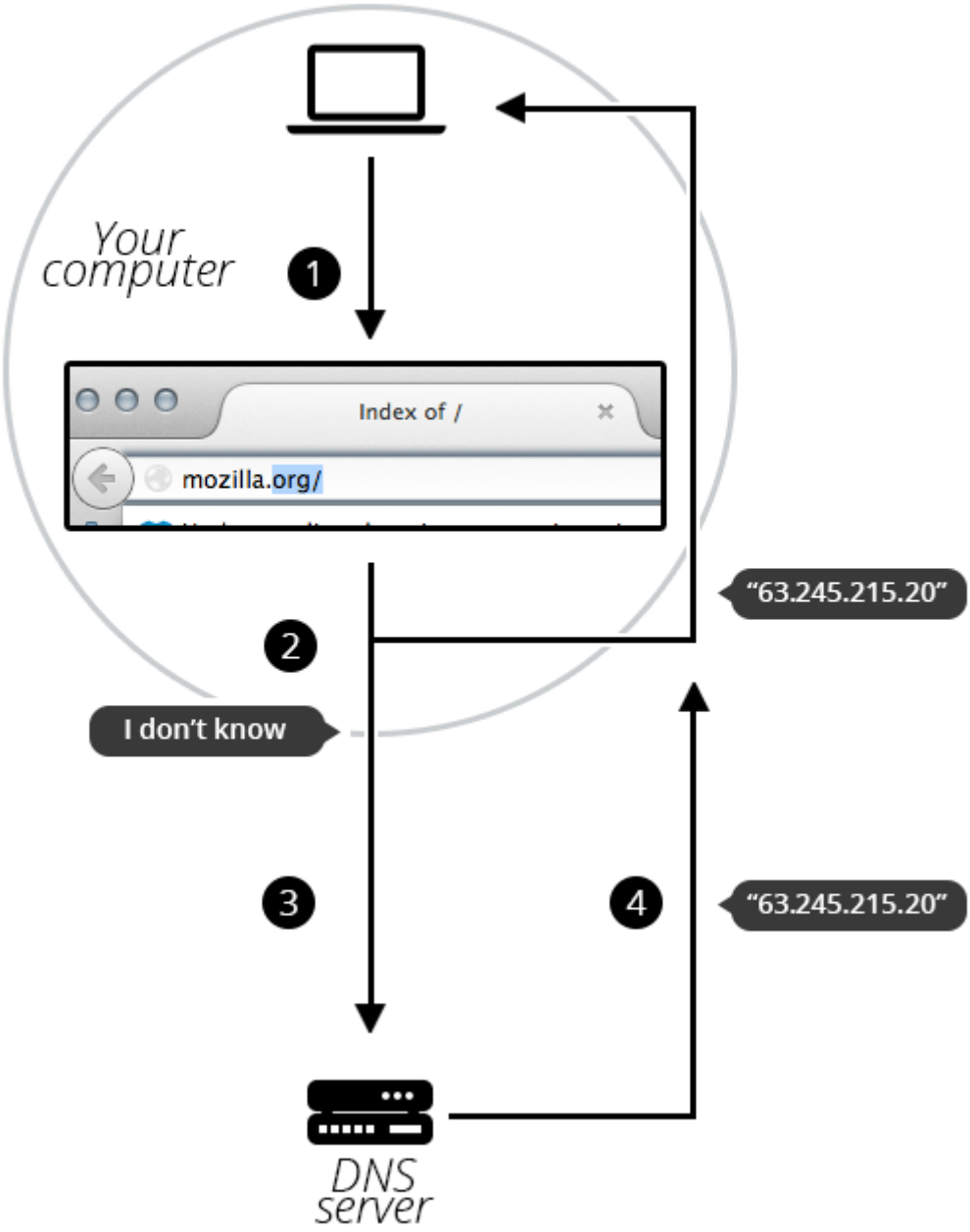
Las bases de datos DNS son almacenadas en cada servidor DNS del mundo, y todos ellos hacen referencia a unos pocos denominados "servidores de nombre autoritario" o "servidores DNS de primer nivel". Cuando su registrador crea o actualiza alguna información para un dominio dado, la información tiene que ser actualizada en cada base de datos DNS. Cada servidor DNS que conoce sobre un dominio dado almacena la información por algún tiempo antes de que sea automáticamente invalidada y luego actualizada (el servidor DNS consulta un servidor autoritario nuevamente). De esta manera, a los servidores DNS que conocen este nombre de dominio les toma algún tiempo poner la información al día.

Nota : Este tiempo es amenudo llamado **tiempo de propagación** . Sin embargo, este término no es preciso puesto que la actualizaciónn no se está propagando en sí (primer nivel → nivel inferior). Los servidores DNS consultados por su computadora (nivel inferior) son los que obtienen la información del servidor autoritario(primer nivel) cuando lo necesitan.

¿Cómo funciona una petición DNS?

Como ya hemos visto, cuando usted quiere visualizar una página web en su navegador es más simple escribir un nombre de dominio que una dirección IP. Echemos un vistazo al proceso:

1. Escriba `mozilla.org` en la barra de direcciones de su navegador.
2. Su navegador le pregunta a su computadora si reconoce la dirección IP identificada por este nombre de dominio (usando una caché DNS local) Si lo hace, el nombre es traducido a la IP y el navegador gestiona el contenido con el servidor web. Fin de la historia.
3. Si la computadora no sabe qué IP está detrás del nombre `mozilla.org`, hay que pedírselo a un servidor DNS, cuyo trabajo es precisamente decirle a la computadora cuál es la dirección IP de cada nombre de dominio registrado.
4. Ahora que la computadora conoce la dirección IP requerida, su navegador puede gestionar contenidos con el servidor web.


Your computer
1
Index of /
mozilla.org/
"63.245.215.20"
2
I don't know
3
4
"63.245.215.20"
DNS server