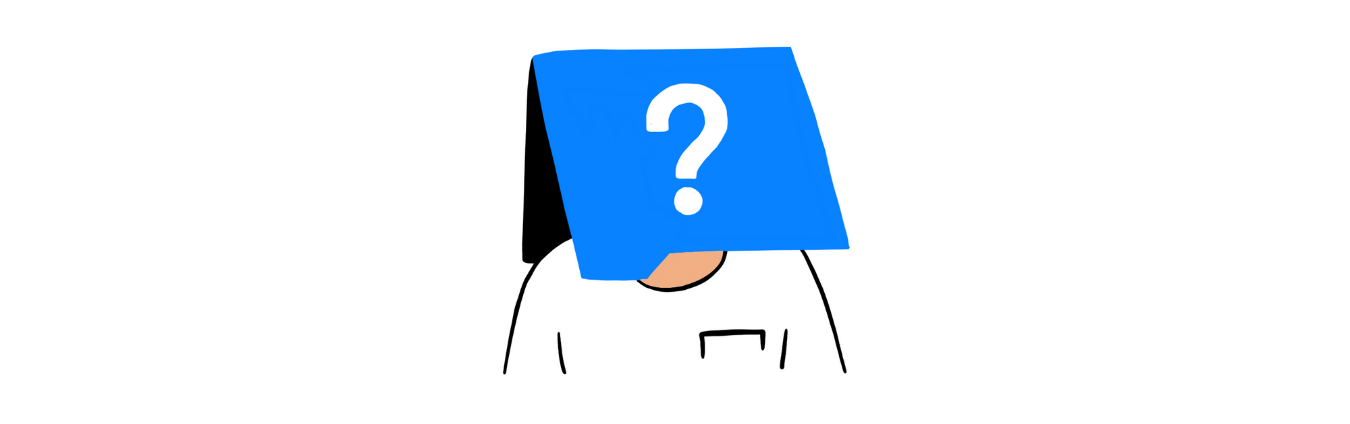
**Archivos**

En la plataforma interactiva, tu código se divide en pestañas HTML, CSS y JavaScript. Lo hemos configurado lo más simple posible para que no tengas que perder el tiempo buscando el archivo correcto, cuando lo que realmente necesitas es enfocarte en la programación en sí. En proyectos reales, sin embargo, los desarrolladores tienen que decidir por su propia cuenta dónde se coloca style.css y dónde debería ir index.html. Entonces, aprendamos a organizar nuestros archivos de front-end.

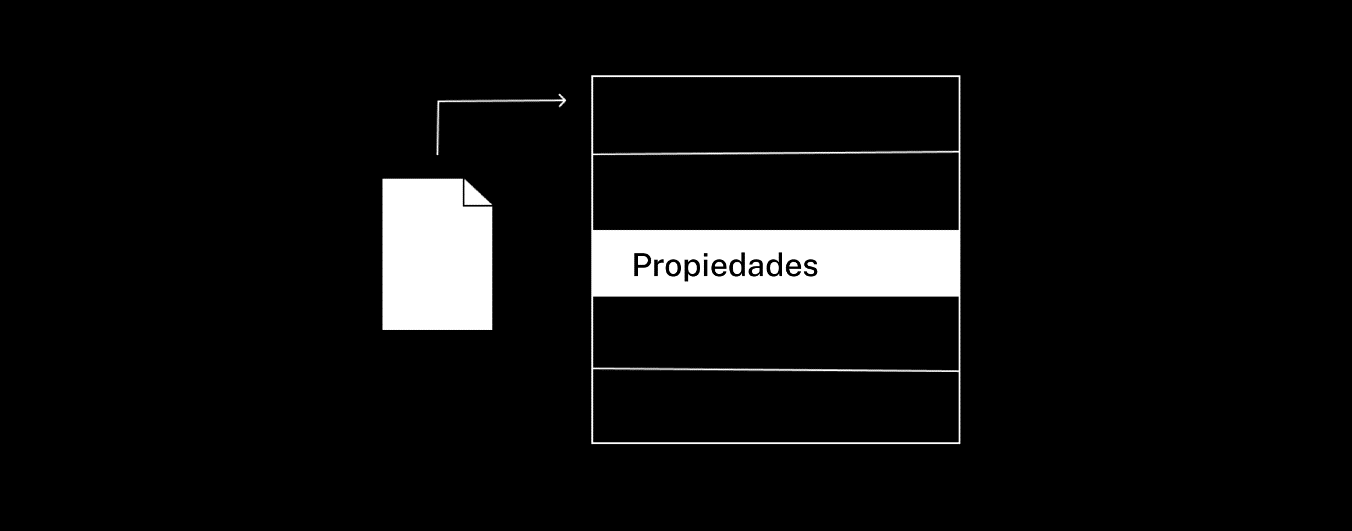
¿Qué hay en un archivo?

Pero antes de abordar el tema de la organización, veamos con más detalle qué es realmente un archivo. Un archivo es una colección de datos que un usuario o un programa puede escribir, editar o leer. Ocupan un espacio determinado en el dispositivo donde se almacenan, como el disco duro de un PC.

Cada archivo tiene un nombre. Esto permite que un sistema operativo (SO) identifique archivos, aunque a veces varios archivos pueden compartir el mismo nombre.

Si comparamos archivos con personas, entonces el nombre del archivo es como el nombre de una persona, mientras que la extensión del archivo es como su apellido. Las extensiones de archivo se utilizan para ayudarte al sistema operativo a decidir qué hay que hacer con un archivo cuando el usuario lo abre. Por ejemplo, el marcado de nuestro proyecto se almacena en archivos .html, nuestros estilos se guardan en archivos .css y nuestros scripts se guardan como archivos .js. Por lo general, los archivos .html se abren automáticamente en un navegador, mientras que los archivos .css y .js se abren por defecto en los editores de texto.

Las personas tienen muchos atributos, como la edad, la ocupación, la educación, etc. Del mismo modo, los archivos también contienen sus propias propiedades (atributos). La lista de atributos disponibles depende del sistema operativo y, en ocasiones, estas propiedades se pueden reasignar.



*Puedes acceder a las propiedades (atributos) de un archivo desde el menú que aparece cuando haces clic derecho sobre él*

Los atributos pueden ser de todo tipo y tamaño. Por ejemplo, si un archivo tiene el atributo "hidden", los usuarios ni siquiera podrán ver el archivo. Otro atributo, readonly, evita que los usuarios escriban o editen un archivo. Esto puede causar complicaciones, ya que no podrás editar archivos que tengan el atributo readonly cuando utilices un servidor remoto sin realizar las acciones adicionales.

Los archivos se pueden dividir en dos categorías:

* Data files: Estos son los archivos con los que trabajan los desarrolladores web. Incluyen archivos gráficos y de texto, archivos de base de datos y JavaScript, Python u otros archivos de comandos.
* Program files: Estos realizan funciones del sistema y pueden ser aplicaciones u otros archivos de utilidad. Sus extensiones dependen de su uso y del sistema operativo que estés utilizando. Por ejemplo, los archivos con la extensión .lnk indican accesos directos o enlaces en sistemas Windows.

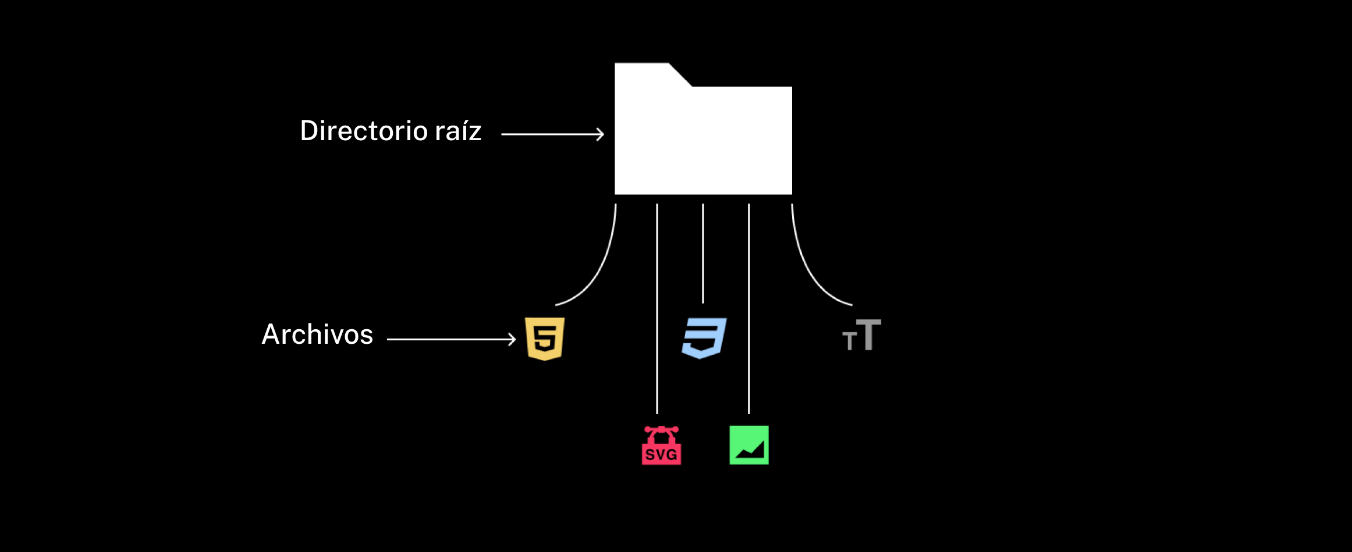
Un directorio (también llamado carpeta) puede almacenar otros archivos y carpetas. Cada proyecto en el que trabajes normalmente se almacenará en su propio directorio.

Ahora que conoces la terminología, pasemos a la creación de estructuras para nuestros archivos y directorios.

**Estructuras lineales y jerárquicas: separación de archivos por tipo**

Una estructura de archivos es simplemente una forma de organizar archivos y directorios.

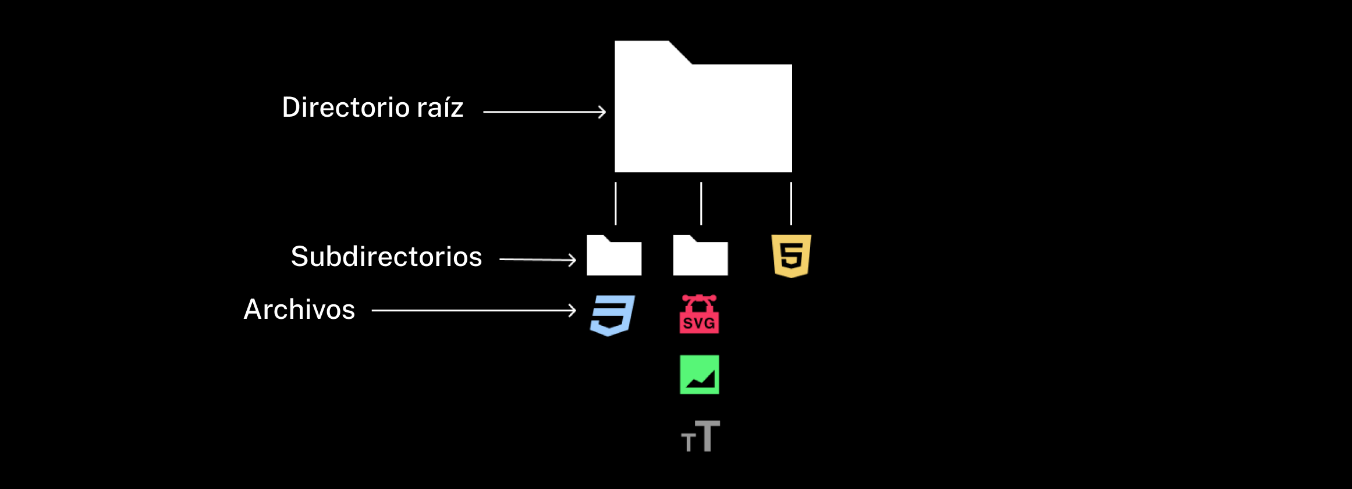
Primero veamos un ejemplo simple: una estructura de archivo lineal. Aquí todos los archivos están en el mismo nivel y eso significa que no hay directorios adicionales anidados dentro del directorio principal (también conocido como directorio raíz).



Una estructura de archivo tan simple tiene sus desventajas. Si tu proyecto tiene una gran cantidad de archivos .html, .css y .js puede convertirse en un revoltijo imposible. ¡Y eso incluso antes de que comencemos a tener en cuenta las imágenes, fuentes u otros archivos mixtos que podrían aparecer más tarde! Si usas esta estructura de archivos, puedes tardar bastante en encontrar los archivos que necesitas. Esto es un problema, ya que la velocidad es el mejor amigo del desarrollador.

Es mejor separar estos archivos en diferentes carpetas. Podemos crear directorios propios para imágenes, fuentes, hojas de estilo y scripts, lo que hará que todo sea más fácil de encontrar. Una estructura de archivos con directorios anidados se denomina estructura jerárquica.

Una estructura de archivos jerárquica es aquella que contiene archivos y subdirectorios anidados.



Los desarrolladores web a menudo estructuran sus archivos con HTML dentro del directorio raíz. Luego, anidados dentro de este directorio raíz, se encuentran comúnmente los siguientes subdirectorios :

1. El directorio /images incluye iconos, imágenes de fondo e imágenes de contenido. Esos pueden tener diferentes extensiones.
2. Dentro del directorio /styles encontraremos nuestras hojas de estilo.
3. La carpeta /scripts aloja nuestros archivos JavaScript.
4. Finalmente, el directorio /fonts almacena diferentes fuentes utilizadas en el proyecto.

Esta es una estructura de archivo relativamente básica y se usa principalmente para proyectos pequeños. En su momento construirás una estructura de archivo BEM más complicada, pero por ahora, pasaremos a vincular nuestros archivos.

# Rutas de archivos absolutas

Los proyectos de desarrollo web a menudo requieren que los archivos estén vinculados de alguna manera. Por ejemplo, cada vez que agregas la propiedad background-image a una hoja de estilo, necesitas vincular la imagen a través de la función url(). Deberás especificar dónde encontrarlo y si está ubicado en otra carpeta o almacenado en algún lugar en línea. Si pides algo por Internet y el paquete llega cuando no estás en casa, te dejarán una nota indicándote dónde puedes recogerlo. Del mismo modo, para acceder a un archivo, deberás especificar la ruta.

Esta ruta es la dirección del archivo, ya sea en tu PC o en línea. Las rutas de archivo pueden ser absolutas o relativas. ¿Qué significa esto?

Supongamos que eres una ardilla que quiere subirse a la rama favorita de su árbol. Un camino absoluto describiría cada salto que la ardilla necesitaría dar, comenzando desde la base del árbol, subiendo por el tronco y luego indicando todos los pasos necesarios para llegar a la rama. Un camino relativo permitiría a la ardilla iniciar el viaje hacia su rama favorita desde otro punto, así como desde la parte superior del tronco del árbol o desde su segunda rama favorita. En el caso de las rutas absolutas, comenzar desde el primer punto es importante, mientras que las rutas relativas son más flexibles al respecto.

Como un árbol, todos los archivos de tu PC están organizados como una estructura jerárquica en el disco duro.

Una ruta absoluta detalla todos los pasos que debemos seguir, comenzando desde la parte más profunda de nuestro directorio de inicio y terminando con el archivo que necesitamos.

En la imagen de abajo, C:/ representa el disco duro. Este es el directorio raíz, es decir, el punto de partida de nuestra ruta absoluta. Analicemos cada elemento de esta ruta de archivo. C:/Projects/html\_project/images/logo.png es la ruta absoluta completa, donde logo.png es el nombre del archivo, mientras que /Projects/html\_project/images representa los subdirectorios anidados a los que hay que acceder a lo largo de la ruta. Aquí cada directorio individual, en la ruta hacia el archivo, está dividido por un símbolo /.



Este enlace funcionará desde cualquier lugar del disco duro, pero si movemos el archivo a otro directorio, esta ruta dejará de ser válida.

Si se trata de un archivo en línea, una ruta absoluta sería un enlace al recurso solicitado. Comienza con el protocolo http:// o https://. Utilizando nuestra analogía del paquete, si queremos enviar algo a otra ciudad mediante un camión de reparto, usar el protocolo http:// sería como enviarlo en un camión de reparto convencional, mientras que https:// sería como enviarlo en un vehículo blindado de valores. La diferencia principal aquí no radica en la velocidad o el costo, sino en el nivel de seguridad que proporcionahttps://para proteger el contenido del paquete durante su transporte.



Al igual que las personas se mudan a un nuevo hogar, un archivo se puede trasladar cambiando su dominio. Si cambiamos la dirección en línea de la imagen, el archivo dejará de estar disponible a través de la ruta anterior. Así que, si el nombre de dominio del sitio web que aloja la imagen cambia de http://www.images-hosting.com/logo.png a http://www.logo-hosting.com/logo.png, y solicitamos la primera URL, ya no habrá una imagen ahí.

Por lo general, es mucho más eficiente alojar todos tus archivos en un solo lugar. Si tu página web contiene imágenes almacenadas en otro dominio, las imágenes tardarán más en cargarse. Aunque no siempre sea posible alojar todo el contenido en tu dominio, ten en cuenta que obtener tu material de muchos lugares diferentes de Internet y utilizar rutas absolutas para hacer referencia a ellos puede afectar significativamente el funcionamiento de tu sitio.

### Distinción entre mayúsculas y minúsculas

Por defecto, en Windows y macOS, los nombres de archivos y directorios no distinguen entre mayúsculas y minúsculas. Por ejemplo, puedes usar Logo.png o logo.png para referirte al mismo archivo en un disco duro. Pero si este archivo está alojado en línea, las rutas absolutas en las URLs consideran que son archivos distintos. Las URLs que distinguen entre mayúsculas y minúsculas son un [estándar web](https://www.w3.org/TR/WD-html40-970708/htmlweb.html#htmlurl) (materiales en inglés).

**Redes de distribución de contenidos (CDNs, del inglés "Content Delivery Networks")**

Ya has utilizado el elemento <link> para aplicar estilos anteriormente, pero lo has hecho dentro de una estructura de archivo simple donde el archivo style.css estaba en el mismo nivel que index.html. El <link> se veía así:

<head>

<link rel="stylesheet" href="style.css">

</head>

Por supuesto, podemos vincular no solo imágenes y hojas de estilo a nuestros archivos. También podemos usar <link> para referirnos a archivos en línea especificando su URL en el atributo href. A menudo querrás vincular librerías a tu código. Las librerías contienen código escrito por otros equipos de desarrollo que pueden proporcionar soluciones ya preparadas para ciertos problemas. Estas soluciones rápidas resultan muy útiles porque nos permiten concentrarnos en la historia principal y escribir el código para nuestro proyecto, en lugar de distraernos con misiones secundarias que consumen mucho tiempo.

Las librerías suelen alojarse en algo llamado red de distribución de contenidos. (Un grupo de servidores remotos distribuidos que permite subir más rápidamente archivos o recursos web alojados en dicho grupo de servidores.) Las CDN pueden contener tanto librerías CSS como JS. Usaremos las etiquetas apropiadas a la hora de conectarlas, por lo que las solicitudes de archivos en una CDN se pueden realizar mediante los elementos <link> o <script>.



Aquí hay una CDN bien conocida donde puedes encontrar librerías de front-end:

[cdnjs.com — La mejor CDN de FOSS (Free and Open Source Software que significa «software libre y de código abierto») para librerías relacionadas con la web para acelerar tus sitios web!](https://cdnjs.com/) *(materiales en inglés)*.

Sin embargo, ten en cuenta que, si bien es común incluir librerías de terceros en nuestros proyectos, este enfoque puede tener varias desventajas:

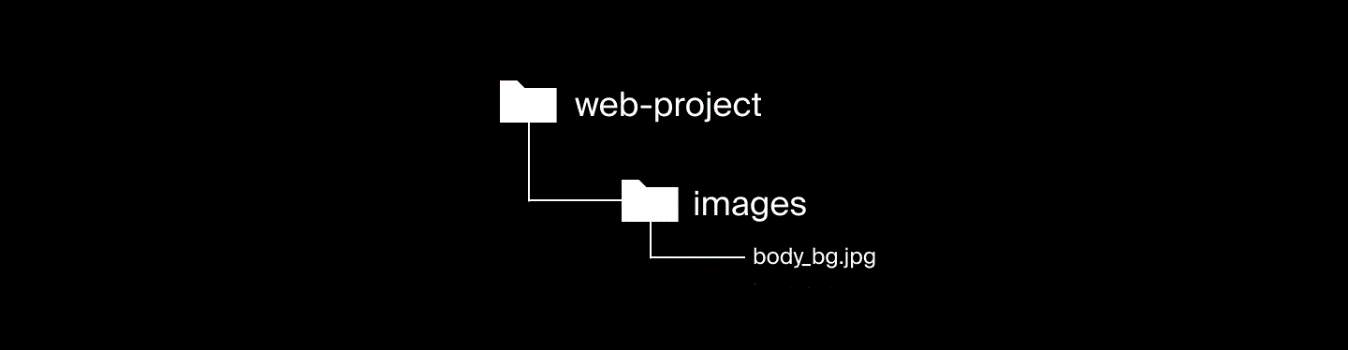
* No puedes editar archivos recibidos de una CDN.
* Las librerías de terceros son mantenidas por sus propietarios, quienes pueden agregar nuevas funciones o cambiar algún código existente. Estos cambios podrían afectar a tu código y esto estaría completamente fuera de tu control.
* Las consultas se realizan a un archivo de terceros. El servidor puede fallar y dejar de responder a las consultas, por lo que si tu sitio web se basa en un archivo de este servidor, también fallará.

Para evitar estos problemas, vamos a descargar librerías de terceros utilizando varias herramientas. Discutiremos esto con más detalle más adelante.

**Rutas de archivo relativas**

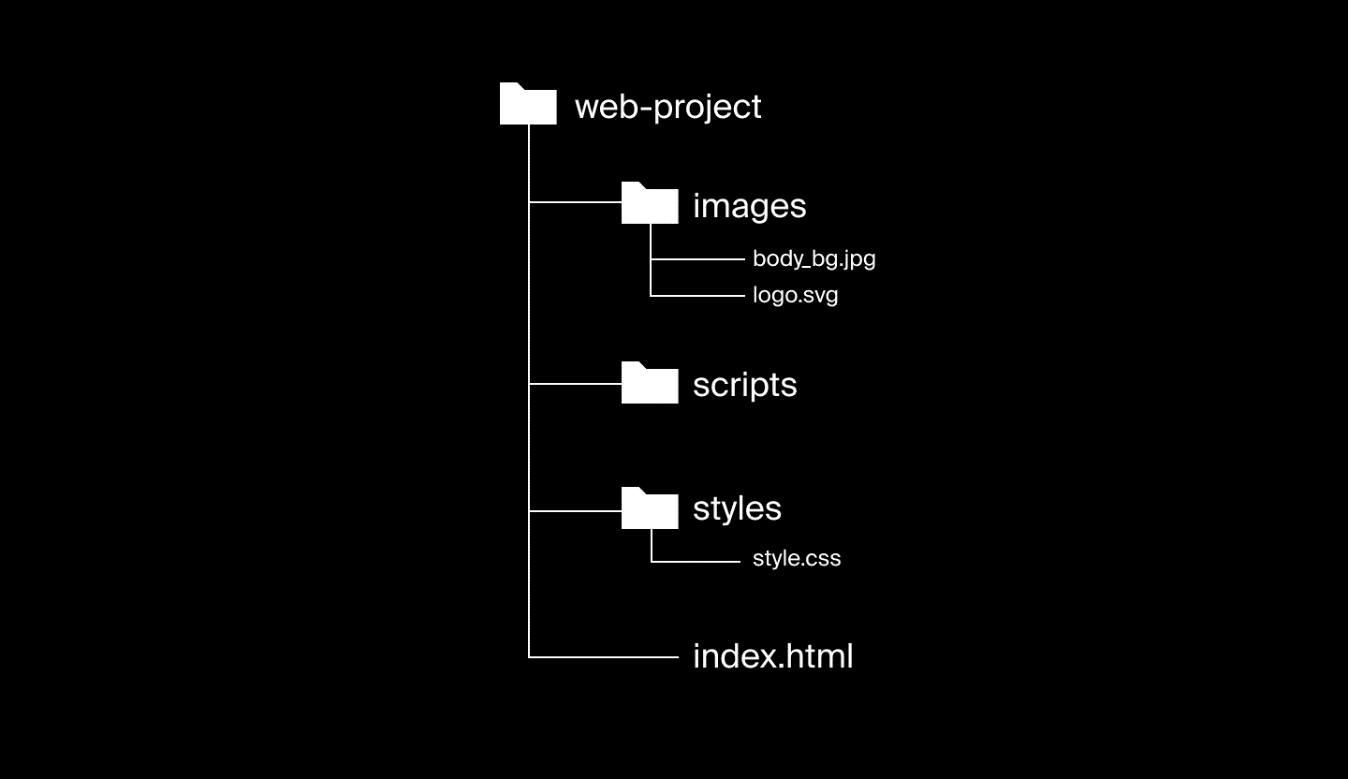
En la última lección, aprendiste a vincular archivos utilizando rutas absolutas. También hablamos sobre ciertos problemas que puede generar este método. Por ejemplo, cuando subamos un sitio a un servidor remoto, todas las rutas absolutas de archivo en el proyecto, que hacen referencia a directorios en nuestro equipo local, dejarán de funcionar. El servidor web tiene su propio disco y una estructura subyacente de archivos, que es diferente a la de tu dispositivo.

Las rutas relativas te permiten evitar este problema. Esto se debe a que especificas la ruta al archivo que necesitas, en relación con el archivo del que proviene la solicitud. Vamos a usar una imagen del directorio /images junto con la propiedad CSS background-image como ejemplo:



Imagina que nuestro proyecto está representado por una pequeña biblioteca, las carpetas del proyecto son estanterías y los libros representan archivos. Además, tenemos una pequeña tarjeta de índice que representa nuestro archivo index.html. Cuando queramos encontrar un libro, bastará con que sigamos el camino indicado en nuestra tarjeta de índice. Así que para encontrar en la biblioteca un libro sobre California iremos a la estantería sobre los estados de EE. UU., donde encontraremos el libro. Si toda esta sala fuera transportada a un edificio diferente, las instrucciones en nuestro archivo de índice seguirían siendo válidas, porque estamos comenzando nuestro viaje en relación con la propia biblioteca y no con el edificio.

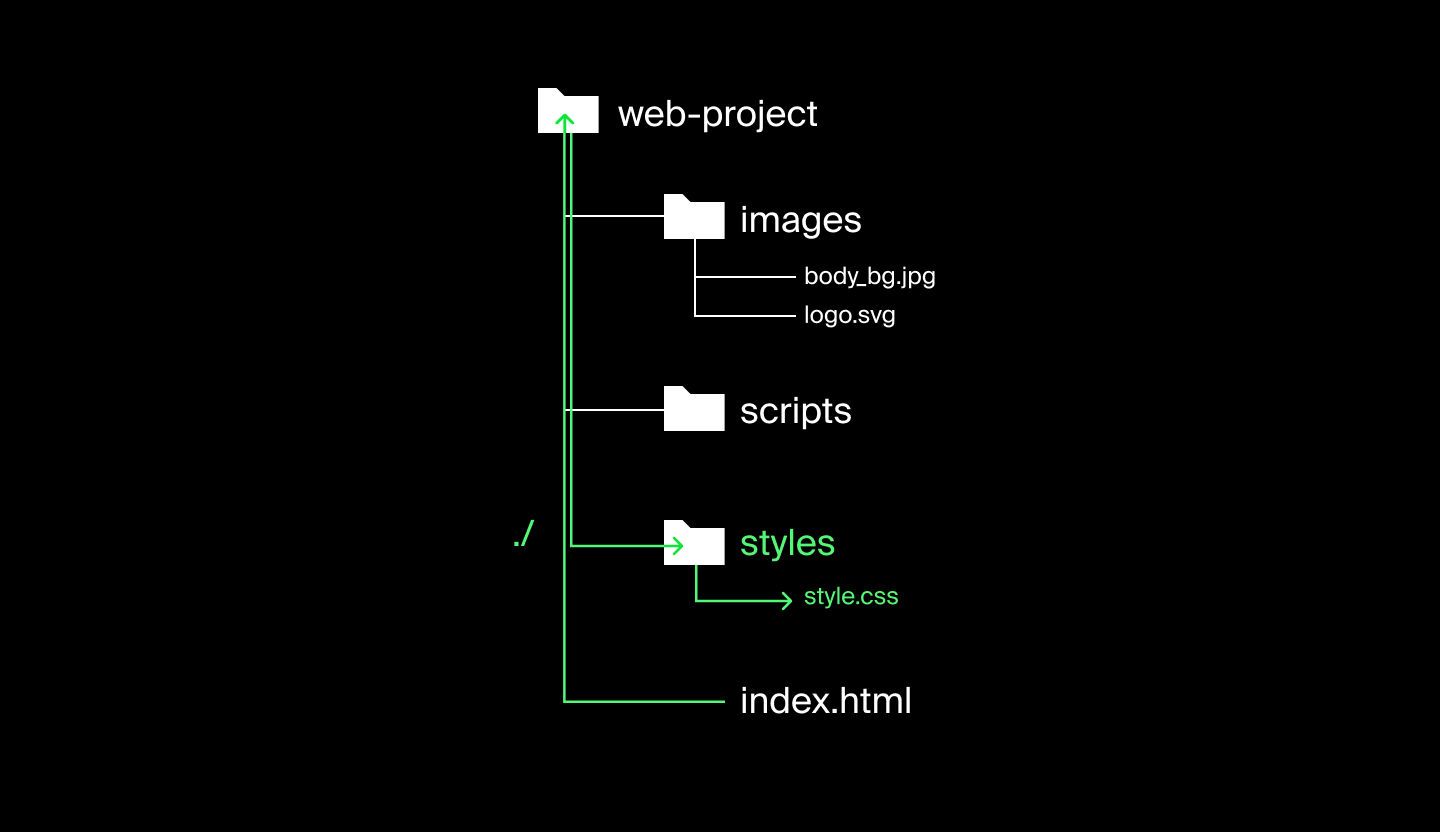
El mismo principio se aplica cuando se utilizan rutas relativas. Las rutas relativas nos muestran cómo llegar a un archivo desde el lugar en el que ya estamos, en vez de hacerlo desde el directorio raíz.

La estructura de archivos del proyecto

Por analogía con nuestra librería:

* /web-project es el directorio raíz (la librería).
* /images, /scripts y /styles son subdirectorios (estanterías) que contienen sus archivos respectivos (libros).
* index.html actuará como nuestro punto de entrada para este proyecto simple (lo representamos como una tarjeta de índice con direcciones).

En nuestro ejemplo anterior, el archivo index.html está ubicado en el directorio raíz / web-project, el archivo style.css está situado en la carpeta /styles y el archivo de imagen body\_bg.jpg se encuentra en la carpeta /images. Queremos conectar la hoja de estilo a nuestro archivo HTML y conectar la imagen a la hoja de estilo como una imagen de fondo. Veamos cómo podemos hacer esto usando rutas relativas.



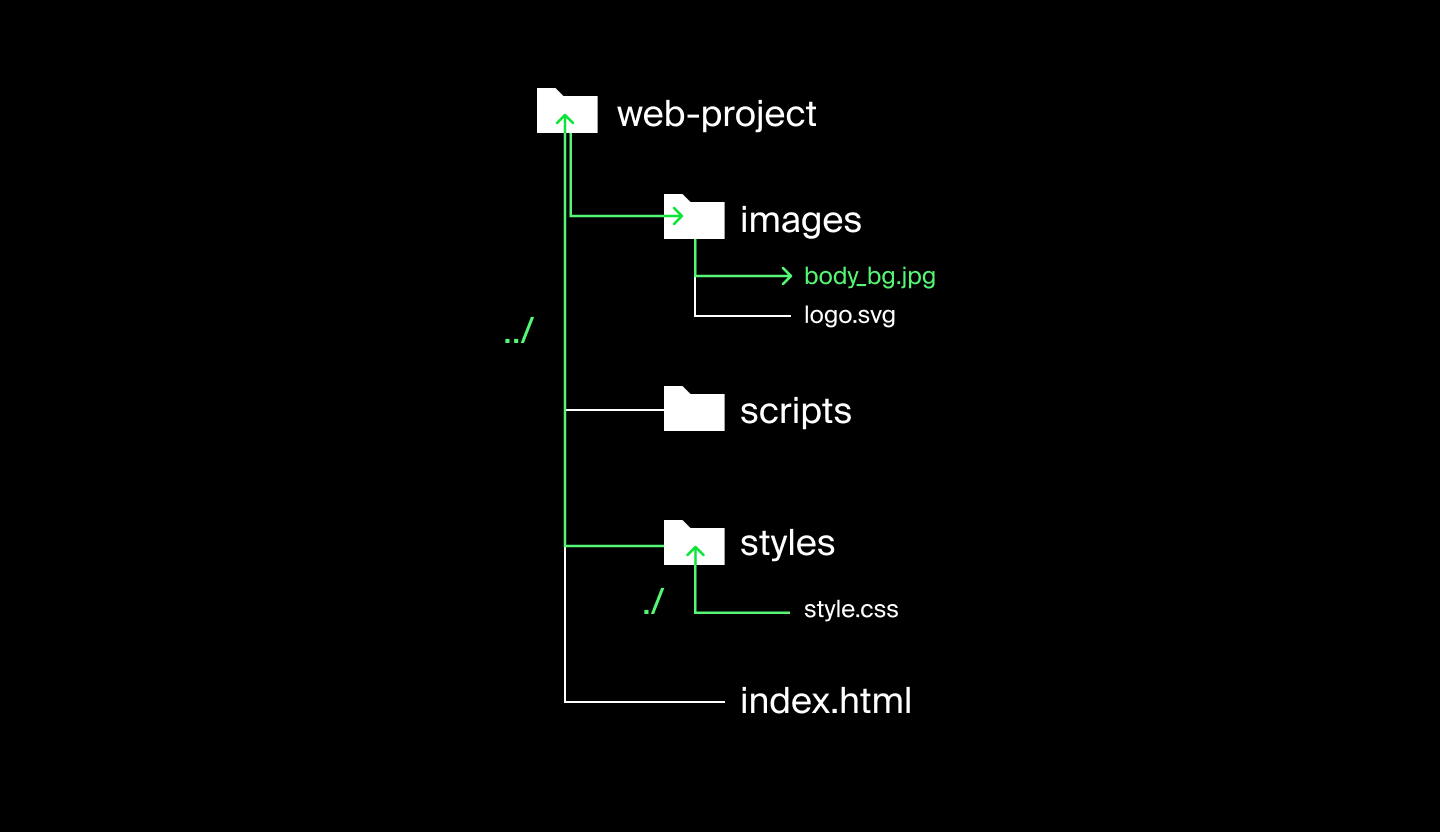
*Usamos ./ para indicar el directorio actual. Desde allí, podemos navegar al directorio /styles y luego al archivo style.css.*

En primer lugar, vinculemos la hoja de estilo a nuestro archivo HTML. Para crear una ruta relativa hacia el archivo que deseas conectar, la ruta relativa debería comenzar con ./. Si comienzas una ruta de archivo con ./ estás indicando que la ruta empieza en el directorio actual del archivo desde el que estás creando la ruta. En nuestro caso, estamos en el archivo index.html, por lo que escribir ./ llevará al directorio raíz /web-project. Desde allí, puedes escribir la ruta hacia el archivo que quieras:

*<!-- estamos entrando en la carpeta de estilos desde index.html -->*

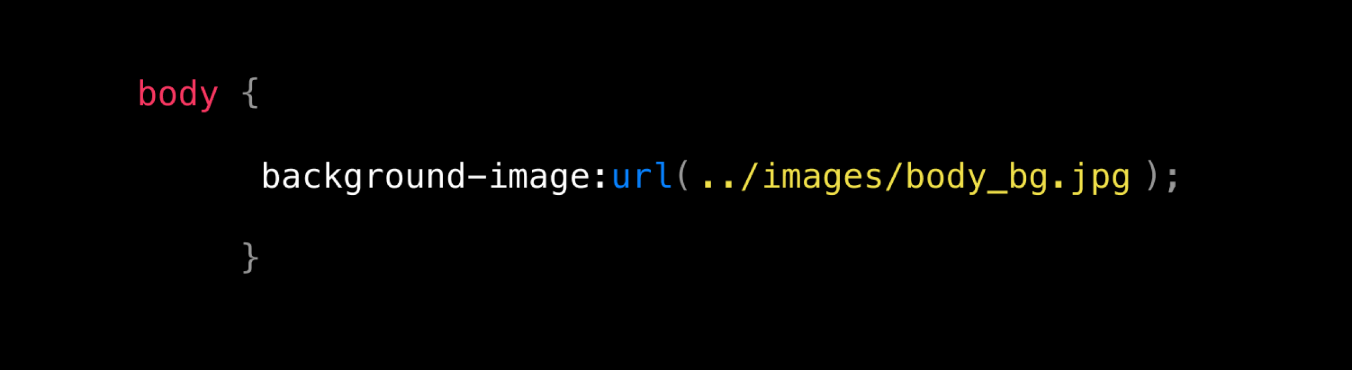
<link rel="stylesheet" href="./styles/style.css">

Ahora, para crear una ruta relativa desde style.css a body\_bg.jpg, necesitamos subir a otro nivel para llegar al directorio raíz. Para hacer esto, necesitamos usar ../ en nuestra ruta de archivo.



*Cómo acceder al archivo images/body.bg.jpg desde style.css.*

Podemos usar ../ en una ruta de archivo para indicar que queremos subir un nivel en el directorio. En este caso, lo usamos para ir al directorio padre de la carpeta /styles, que es nuestro directorio raíz. Desde allí podemos acceder a la carpeta /images y a nuestra imagen deseada:



*La ruta de archivo hacia body\_bg.jpg está en amarillo. Primero se accede al directorio raíz /web-project y después a la carpeta images.*

Por otro lado, también podemos crear rutas absolutas eliminando todos los puntos y simplemente usando / al comienzo de la ruta.

En este caso, el carácter / apuntará al directorio raíz y la ruta continuará hasta el siguiente directorio o archivo indicado, y así sucesivamente. Esto se debe a que los navegadores utilizan la notación de ruta UNIX path notation predeterminada. Una pequeña advertencia: recuerda que si inicias una ruta de archivo con al menos un punto, esta se convertirá en una ruta de archivo relativa.

Además, ten en cuenta que el uso de / para acceder a la carpeta raíz del proyecto solo funcionará en un servidor web. Si escribes una ruta como esa para los archivos en tu PC y luego abres el enlace en un navegador, te encontrarás en el directorio raíz del sistema operativo.

Podemos reescribir la ruta relativa de nuestro ejemplo anterior de la siguiente manera:



*La ruta de archivo desde el directorio raíz hacia la imagen de fondo.*

Esto también significa que si agregas otro directorio dentro del directorio images /, este cambio no causará ningún problema. Siempre que mantengas body\_bg.jpg en tu ubicación actual, tu ruta seguirá siendo válida.

¡Ahora es el momento de practicar tus habilidades de escritura de rutas!