Introducción a la línea de comandos

(Traducción por Nicolás Vaughan)

¿Qué es la línea de comandos?

Si nos pidieran explicarle a alguien que nunca ha visto un computador cómo hacer algo en este, muchos de nosotros explicaríamos lo que es una pantalla y un cursor, y luego mostraríamos cómo apuntar y hacer clic en los íconos. Este enfoque se basa en una interfaz gráfica de usuario o GUI (pronunciado "gúi"). Hoy vamos a explorar otra forma de hacer que tu computador haga cosas: a través de la línea de comandos. En lugar de apuntar y hacer clic, escribiremos comandos en `Git Bash` (Windows) o en `terminal` (macOS) para decirle al computador directamente qué tarea queremos que realice.

La línea de comandos es una forma de interactuar con el computador basada en texto. Es posible que oigas llamarla de diferentes maneras, como terminal, shell o bash. En la práctica, puedes utilizar estos términos indistintamente. (Sin embargo, si tienes curiosidad, puedes leer más sobre ellos en el glosario.) El shell que utilizamos (ya sea terminal, shell o bash) es un programa que acepta comandos como entrada de texto y convierte los comandos en funciones apropiadas del sistema operativo.

La línea de comandos (de los computadores actuales) recibe estos comandos como texto que se teclea.

Texto plano vs. texto formateado

Como académicos que trabajan con computadores, debemos ser conscientes de las diferencias que hay entre el texto plano y el texto formateado. Las palabras en una pantalla pueden tener un formato oculto. Muchos de nosotros hemos aprendido a utilizar un procesador de textos como Microsoft Word y no nos damos cuenta de todo lo que ocurre detrás de las palabras que aparecen en la pantalla. Para comunicarnos con el computador y para facilitar el movimiento entre los distintos programas, necesitamos utilizar texto sin formato oculto.

Documento de Word

Usuarios con discapacidades visuales, pincha aquí para descargar el archivo de Word.

Si se le pide a la línea de comandos que lea ese archivo `.docx`, se verá así:

Documento de Word visualizado por la línea de comandos

Usuarios con discapacidades visuales, pinchen aquí para descargar el archivo de texto.

Los documentos de Word que parecen "¡solo palabras!" se componen en realidad de un archivo de instrucciones de lenguaje de marcado extensible (XML) que solo Microsoft Word puede leer. Los archivos de texto plano pueden abrirse en varios editores diferentes y pueden leerse en la línea de comandos.

Texto sin formato

Para comunicarnos con las máquinas y entre máquinas, necesitamos que los caracteres sean lo más flexibles posible.

El texto plano muestra sus cartas: si está marcado, el marcado será legible para los humanos. El texto plano puede moverse entre programas con mayor fluidez y puede responder a las manipulaciones programáticas. Al no estar atado a una fuente, color o ubicación concretos, el texto plano puede ser estilizado externamente.

La contraparte del texto plano es el texto enriquecido (a veces indicado por la extensión de archivo de formato de texto enriquecido de Microsoft `.rtf`, y a veces visto como una opción en los programas de correo electrónico). En los archivos de texto enriquecido, el texto plano se elabora con un formato específico del programa en el que se realiza.

El texto plano tiene dos propiedades principales con respecto al texto enriquecido:

- el texto plano es el cadena de contenido subyacente a la que se puede aplicar el formato
- el texto plano es público, estandarizado y universalmente legible.

Recomendación por defecto para un editor de texto

Para nuestros talleres utilizaremos Visual Studio Code. Visual Studio Code no solo es gratuito y de código abierto, sino que también es consistente en los sistemas macOS, Windows y Linux.

Habrás descargado Visual Studio Code según las instrucciones de la página de instalaciones. No usaremos mucho el editor en este tutorial, así que no te preocupes por conocerlo ahora. En otros talleres hablaremos del resaltado de sintaxis y del control de versiones que Visual Studio Code soporta. Por ahora volveremos a trabajar en la línea de comandos.

Evaluación

¿Cuál es la diferencia entre un documento de texto plano y un documento de texto enriquecido? (Selecciona todo lo que corresponda)

- El texto plano no contiene formato, solo saltos de línea y espaciado.
- El texto plano no puede ser marcado.
- El texto enriquecido es un texto con estilo, es decir, un texto sin formato completado con información como el tamaño de la fuente, el formato y los colores.
- No se puede determinar si hay una diferencia entre los dos sin ver su contenido.

¿Por qué es útil la línea de comandos?

Al principio, para algunos de nosotros, la línea de comandos puede resultar un poco desconocida. ¿Por qué alejarse de un flujo de trabajo de apuntar y hacer clic? Al utilizar la línea de comandos, pasamos a un entorno en el que tenemos un control más minucioso sobre cada tarea que queremos que realice el computador. En lugar de pedir la comida en un restaurante, te metes en la cocina. Es más trabajo, pero también hay más posibilidades.

La línea de comandos te permite:

- Automatizar fácilmente tareas como crear, copiar y convertir archivos.

- Configurar tu entorno de programación.
- Ejecutar los programas que crees.
- Acceder a los (muchos) programas y utilidades que no tienen equivalentes gráficos.
- Controlar otros computadores de forma remota.

Además de ser una herramienta útil en sí misma, la línea de comandos te da acceso a un segundo conjunto de programas y utilidades y es un complemento para aprender a programar.

¿Y si todas estas geniales posibilidades te parecen un poco abstractas ahora mismo? No pasa nada. En un nivel muy básico, la mayoría de los usos de la línea de comandos consisten en **mostrar información** que tiene el computador, o **modificar o hacer** cosas (archivos, programas, etc.) en el computador.

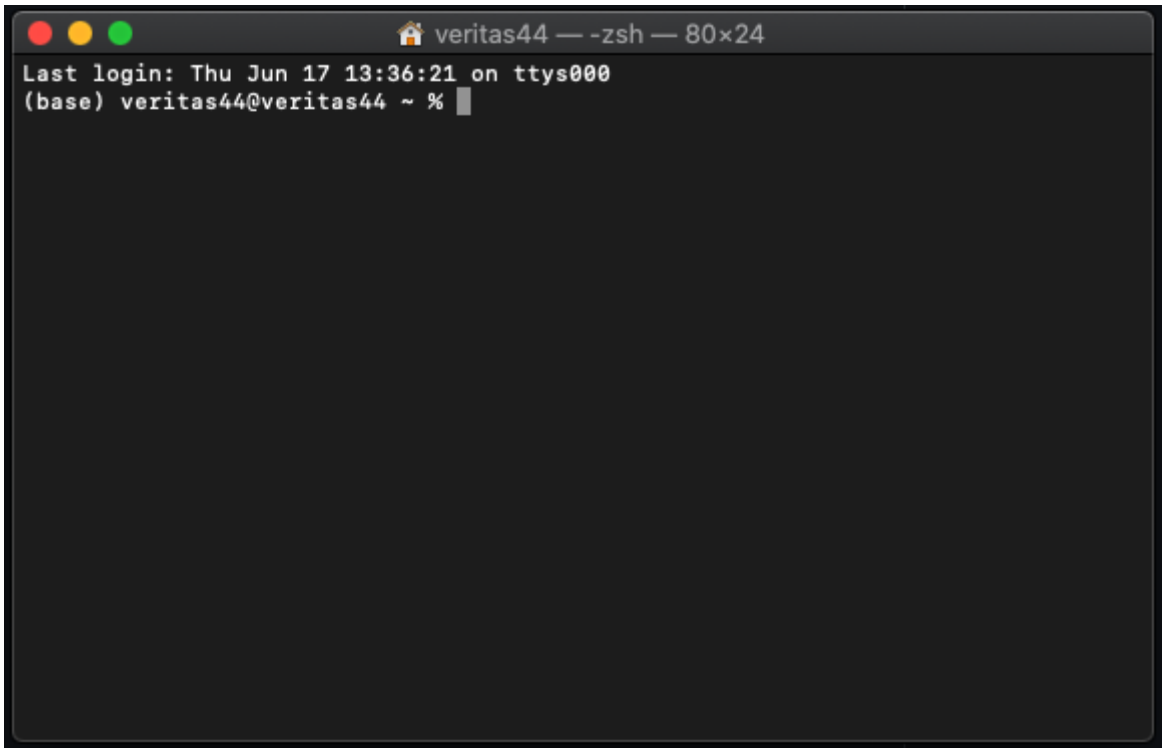
En la siguiente sección dejaremos esto un poco más claro al empezar con la línea de comandos.

Llegando a la línea de comandos

macOS

Si estás usando macOS:

1. Haz clic en el botón de búsqueda de Spotlight (la lupa) en la parte superior derecha de tu escritorio.
2. Escribe `terminal` en la barra que aparece.
3. Elige el primer ítem que aparece en la lista.
4. Cuando el Terminal se despliegue, probablemente verás o bien una ventana con texto negro sobre fondo blanco, o bien texto de color sobre fondo negro.

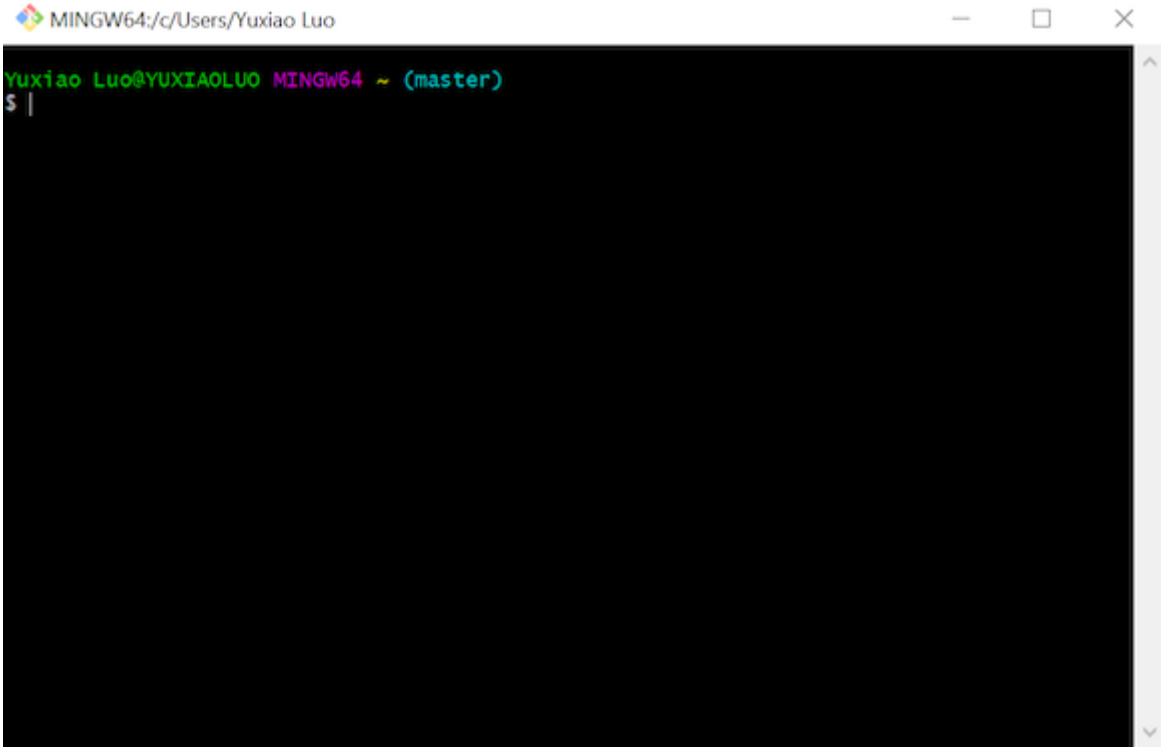


Nota: puedes cambiar el color del fondo y el texto de tu Terminal o Bash Shell seleccionando `Shell` en la barra de menú superior y luego seleccionando un tema del menú bajo `Nueva ventana`.

Windows

No usaremos la versión propia de Windows de la línea de comandos (`cmd`), que no es UNIX. En su lugar, utilizaremos Git Bash. Si aún no lo has instalado, puedes seguir estas instrucciones. La razón por la que usamos Git Bash como línea de comandos en Windows es que permite ejecutar los mismos comandos que en un computador con macOS o Linux. Git Bash incluye utilidades básicas disponibles en Linux que no están disponibles en Windows.

1. Busca Git Bash en el menú de programas y ábrelo.
2. Si no puedes encontra la carpeta git ahí, simplemente escribe `git bash` en el campo de búsqueda y selecciónale cuando aparezca.
3. Abre el programa.
4. Cuando el terminal aparezca, es probable que veas o bien una ventan con texto negro sobre un fondo blanco, o bien texto coloreado sobre fondo negro. Sabrás que estás en el lugar correcuando cuando vea el `$` (que también puede sr un `%` o un `#` dependiendo del sistema operativo).



El símbolo del sistema `$`

El `$`, al que nos referiremos como el "símbolo del sistema", es el lugar donde escribes los comandos que deseas que el computador ejecute. Ahora aprenderemos algunos de los comandos más comunes.

Cuando veas el `$`, estás en el lugar correcto. Sin embargo, como se ha señalado anteriormente, el signo varía un poco entre los sistemas, y a veces el signo es un `%` o un `#`. Llamamos al signo el *símbolo del sistema* (en inglés, *command prompt*); nos permite saber, entre otras cosas, que el computador está listo para recibir un comando.

En las siguientes lecciones, nos referiremos al símbolo del sistema por medio de un `$`. Nota ahora su signo, si difiere del signo de dólar. Podrás seguir adelante sin problemas siempre que entiendas que todas son formas

diferentes de saber que estás "en el *símbolo del sistema*".

Algunos consejos preliminares

Antes de empezar, quisiera darte un par de consejos de cosas a tener en cuenta.

Primero, ve despacio al principio y revisa tu ortografía. Si al principio algo no funciona, ¡revisa tu ortografía! A diferencia de la lectura humana, en la que las letras funcionan simultáneamente como símbolos atomísticos y como contingencias complejas (consulta Johanna Drucker sobre el alfabeto), en la codificación, cada carácter tiene una función discreta, incluidos (¡especialmente!) los espacios.

En segundo lugar, ten en cuenta que la línea de comandos y los sistemas de archivos de macOS y Unix son sensibles a las diferencias entre las mayúsculas y minúsculas, por lo que las mayúsculas también son importantes a la hora de escribir los comandos y los nombres de archivos y carpetas.

En tercer lugar, aunque copiar y pegar de este práctico tutorial puede ser tentador para evitar errores ortográficos y otras cosas, ¡te animamos a que no lo hagas! Escribir cada comando te ayudará a recordarlos y a saber cómo funcionan.

Ahora, estamos listos para empezar.

Navegación

Preparación: ¡conócete a ti mismo!

También puedes ver tu nombre de usuario a la izquierda del símbolo del sistema `$`. Probemos nuestro primer comando. Escribe lo siguiente y presiona `enter` en tu teclado:

```
$ whoami
```

El comando `whoami` imprime tu nombre de usuario. ¡Felicitaciones, has ejecutado tu primer comando! Este es un patrón básico de uso en la línea de comando: escribe un comando, presiona `enter` en el teclado, y recibe una salida.

(Tip: si quieres obtener información y detalles sobre casi cualquier comando, escribe `man` seguido del nombre del comando que quieras investigar. Por ejemplo: `man whoami` o incluso `man man`.)

Cómo orientarse en la línea de comandos: Carpetas

Bien, vamos a probar otro comando. Pero primero vamos a asegurarnos de que entendemos algunas cosas sobre cómo funciona el sistema de archivos de tu computador.

Los archivos de tu computador están organizados en lo que se conoce como un sistema de archivos jerárquico. Eso significa que hay una carpeta de nivel superior o "raíz" en tu sistema. Esa carpeta tiene otras carpetas dentro, y esas carpetas tienen carpetas dentro, y así sucesivamente.

Puedes dibujar estas relaciones en un árbol:

Ejemplo de un sistema de archivos jerárquico](https://github.com/DHRI-Curriculum/command-line/raw/v2.0/images/hierarchical-filesystem-example.png)

La carpeta raíz o de mayor nivel en macOS se llama simplemente `/`. Sin embargo, no necesitaremos entrar ahí, ya que en su mayoría son solo archivos del sistema operativo. En Windows, el directorio raíz suele llamarse `C:`. (Si tienes curiosidad por saber por qué `C:` es el nombre por defecto en Windows, puedes leerlo aquí. Si tienes más discos, recibirán nombres sucesivos: `D:\`, `E:\`, etc.)

Te en cuenta que estamos utilizando la palabra "directorio" indistintamente con "carpeta"; ambos se refieren a la misma cosa.

Bien, probemos un comando que nos diga en qué parte del sistema de archivos estamos:

```
$ pwd
```

Deberías obtener una salida como `/Users/tu-nombre-de-usuario`. Eso significa que estás en el directorio `tu-nombre-de-usuario` en la carpeta `Users` dentro del directorio `/` o raíz. Este directorio se llama a menudo el directorio "home".

En Windows, tu salida sería en cambio `C:/Users/tu-nombre-de-usuario`. La carpeta en la que se encuentra se llama directorio de trabajo, y `pwd` significa "imprimir el directorio de trabajo". La palabra "imprimir" puede ser algo engañosa. El comando `pwd` no imprimirá realmente nada excepto en tu pantalla. Este comando es más fácil de entender cuando interpretamos "print" como "display" en inglés.

Ahora sabemos "dónde" estamos. ¿Pero qué pasa si queremos saber qué archivos y carpetas están en el directorio `tu-nombre-de-usuario`, también conocido como el directorio de trabajo? Prueba introducir:

```
$ ls
```

Deberías ver un número de carpetas, probablemente incluyendo `Documentos`, `Desktop`, y así sucesivamente. También puede ver algunos archivos. Estos son los contenidos del directorio de trabajo actual. `ls` "listará" el contenido del directorio en el que te encuentras.

¿Te preguntas qué hay en la carpeta `Desktop`? Intentemos navegar hasta ella con el siguiente comando:

```
$ cd Desktop
```

El comando `cd` nos permite "cambiar de directorio". (Asegúrate de que la "D" de "Desktop" está en mayúsculas.) Si el comando tuvo éxito, no verás ninguna salida. Esto es normal. A menudo, la línea de comandos tendrá éxito en silencio.

Entonces, ¿cómo sabemos que ha funcionado? Usemos nuestro comando `pwd` de nuevo. Deberíamos obtener:

```
```

```
$ pwd
/Users/Tu-nombre-de-usuario/Desktop
```

Ahora prueba de nuevo con `ls` para ver qué hay en tu escritorio. Estos tres comandos —`pwd`, `ls` y `cd`— son los más utilizados en el terminal. Con ellos puedes orientarte y moverte.

Otro comando que puede resultarte útil es `cd ..`, que te hará subir un directorio en el sistema de archivos. Es un `cd` con dos puntos después:

```
$ cd ..
```

(nota el espacio luego de `cd`. La expresión `..` siempre apunta al directorio inmediatamente superior).

Cuando termines, puedes regresar a tu carpeta "home" (donde están los archivos del usuario en el computador) con:

```
$ cd ~
```

Esa virgulilla (o "tilde") `~`, que introduces con una tecla en tu teclado, es una abreviatura conveniente del nombre de la carpeta "home" del usuario actual.

Evaluación

¿Qué comando ejecuta si intenta identificar en qué parte del sistema de archivos se encuentra/trabaja actualmente?

- $ `ls`
- $ `pwd`
- $ `cd`
- $ `whoami`

¿Cuándo y por qué querrías usar la línea de comandos en lugar de la GUI de tu sistema operativo?

Palabras clave

¿Recuerdas los términos del glosario de esta sección?

- Sistema de archivos
- GUI
- Raíz

Creación de archivos y carpetas

Crear un Archivo

Hasta ahora, solo hemos realizado comandos que nos dan información. Vamos a utilizar un comando que cree algo en el computador.

Primero, asegúrate de que estás en tu directorio personal:

```
$ pwd
/Users/your-username
```

Vamos a movernos a la carpeta `Escritorio` (o `Desktop`, si el sistema está en inglés), o a "cambiar de directorio" con `cd` (de *change directory*):

```
$ cd Escritorio
```

Una vez que te has asegurado de que estás en la carpeta `Desktop` con `pwd`, vamos a probar un nuevo comando:

```
$ touch foo.txt
```

El comando `touch` se utiliza para crear un archivo sin contenido. Este comando se puede utilizar cuando no tienes ningún dato todavía para almacenar en él.

Si el comando tiene éxito, no verás ninguna salida. Ahora mueve la ventana de la terminal y mira tu escritorio "real", el gráfico. ¿Ves alguna diferencia? Si el comando tuvo éxito y estabas en el lugar correcto, deberías ver un archivo de texto vacío llamado `foo.txt` en el escritorio. Muy bonito, ¿verdad?

Consejo útil: la flecha arriba

Digamos que te ha gustado tanto ese archivo `foo.txt`, sino que te gustaría tener otro. En la ventana del terminal, pulsa la `flecha arriba` de tu teclado. Notarás que esto rellena la línea con el comando que acaba de escribir. Puedes pulsar `enter` para crear otro `foo.txt,` (nota: el comando `touch` no sobrescribirá su documento ni añadirá otro documento al mismo directorio, pero actualizará la información sobre ese archivo.) o puedes utilizar las `flechas izquierda/derecha` para mover el cursor de inserción por la pantalla de modo que pueda, por ejemplo, cambiar el nombre del archivo a `foot.txt` para crear un archivo diferente.

A medida que empezamos a escribir comandos más complicados y largos en nuestro terminal, la `flecha arriba` es un gran atajo para no tener que pasar mucho tiempo escribiendo.

Creación de carpetas

Muy bien, así que vamos a hacer un montón de trabajo en este Instituto de Investigación de Humanidades Digitales. Vamos a crear una carpeta `proyectos` en nuestro escritorio, donde podemos mantener todo nuestro trabajo en un solo lugar.

En primer lugar, vamos a comprobar que todavía estamos en la carpeta `Desktop` con `pwd`:

```
$ pwd
/Usuarios/nombre-de-usuario/Desktop
```

Una vez que hayas comprobado que estás en `Desktop`, usaremos el comando `mkdir` ("make directory") para crear una carpeta llamada `proyectos`:

```
$ mkdir proyectos
```

Ahora ejecuta `ls` para ver si ha aparecido una carpeta de proyectos. Una vez que confirmes que la carpeta de proyectos se ha creado con éxito, `cd` en ella.

```
$ cd proyectos
$ pwd
/Users/your-username/Desktop/proyectos
```

Bien, ahora tienes una carpeta de proyectos que puedes utilizar en todo el Instituto. Debería ser visible en tu escritorio gráfico, al igual que el archivo `foo.txt` que creamos anteriormente.

Evaluación

¿Qué hace el comando `flecha arriba`?

- Sale de la Terminal/Git Bash.
- Deshace mi último comando.
- Inserta mi último comando.
- Me muestra en qué carpeta estoy trabajando.

Creación de una hoja de trucos

En esta sección, crearemos un archivo de texto que podemos usar como una hoja de trucos. Puedes usarlo para llevar la cuenta de todos los increíbles comandos que estás aprendiendo.

Echo

En lugar de crear un archivo vacío como hicimos con `touch`, vamos a intentar crear un archivo con algo de texto en él. Pero primero, aprendamos un nuevo comando: `echo`.

```
$ echo "Hola desde la línea de comandos"
Hola desde la línea de comandos
```

Redirigir (>)

Por defecto, el comando echo solo imprime el texto que le damos. Usémoslo para crear un archivo con algún texto en él:

```
$ echo "Esta es mi hoja de trucos" > cheat-sheet.txt
```

Ahora vamos a comprobar el contenido del directorio:

```
$ pwd
/Usuarios/nombre-de-usuario/proyectos
$ ls
cheat-sheet.txt
```

Bien, el archivo ha sido creado. ¿Pero qué era el `>` en el comando que hemos utilizado? En la línea de comandos, un `>` se conoce como una "redirección". Toma la salida de un comando y la pone en un archivo. Ten cuidado, ya que es posible sobrescribir archivos con el comando `>`.

Si quieres añadir texto a un archivo pero *no* sobrescribirlo, puedes utilizar el comando `>>`, conocido como comando de redirección y anexación, en su lugar. Si ya hay un archivo con texto en él, este comando puede añadir texto al archivo *sin* destruirlo y recrearlo.

Cat

Comprobemos si hay algún texto en `cheat-sheet.txt`.

```
$ cat cheat-sheet.txt
Esta es mi hoja de trucos
```

Como puedes ver, el comando `cat` imprime el contenido de un archivo en la pantalla. `cat` significa "concatenar", porque puede unir cadenas de caracteres o archivos de extremo a extremo.

Una nota sobre la denominación de archivos

Tu hoja de trucos se titula `cheat-sheet.txt` en lugar de `cheat sheet.txt` por una razón. ¿Puedes adivinar por qué?

Intenta crear un archivo titulado `cheat sheet.txt` y observa lo que ocurre.

Ahora imagina que intentas abrir un archivo de datos muy importante utilizando la línea de comandos que se titula `cheat sheet.txt`.

Para tus mejores prácticas digitales, te recomendamos que te asegures de que los nombres de los archivos no contengan espacios: puedes utilizar mayúsculas creativas, guiones o guiones bajos en su lugar. Ten en cuenta que los sistemas de archivos de macOS y Unix suelen ser sensibles a las mayúsculas, lo que significa que las mayúsculas importan cuando escribes comandos para navegar entre directorios y archivos o hacer cosas en ellos. T ambién es recomendable evitar el uso de puntos en los nombres de los archivos, ya que a veces

pueden inducir a confundirlos con archivos del sistema o extensiones de archivos (por ejemplo, el nombre completo de un archivo PDF suele ser `archivo.pdf`).

Utilizar un editor de texto

El reto de esta sección será utilizar un editor de texto, concretamente Visual Studio Code (guía de instalación aquí), para añadir algunos de los comandos que hemos aprendido a la hoja de trucos recién creada. Los editores de texto son programas que permiten editar archivos como `.txt`, `.py` (scripts de Python) y `.csv` (valores separados por comas, también conocidos como archivos de hoja de cálculo). Recuerda no utilizar programas como Microsoft Word para editar archivos de texto, ya que añaden caracteres invisibles que pueden causar problemas.

Ejercicio

Puedes usar la GUI para abrir tu editor de texto de Visual Studio Code —desde su menú de programas, a través del Finder o Aplicaciones o Launchpad en macOS, o a través del botón de Windows en Windows— y luego hacer clic en `Archivo` y luego en `Abrir` del menú desplegable y navegar a su carpeta del Escritorio y hacer clic para abrir el archivo `cheat-sheet.txt`.

Pero también puedes abrir ese archivo específico `cheat-sheet.txt` en el editor de texto de Visual Studio Code directamente desde la línea de comandos. Intentémoslo utilizando el comando `code` seguido del nombre del en la línea de comandos.

(Ten en cuenta que el comando `code` pide al computador que abra Visual Code solo si ha completado correctamente la configuración del software durante la instalación).

Una vez que tengas tu hoja de trucos abierta en el editor de texto de Visual Studio Code, escribe ahí mismo los comandos que hemos aprendido hasta ahora. Incluye descripciones de lo que hace cada comando. Recuerda que esta hoja de trucos es para ti. Escribe descripciones que tengan sentido para ti o toma notas sobre las preguntas.

Guarda el archivo.

Una vez que hayas terminado, comprueba el contenido del archivo en la línea de comandos con el comando `cat` seguido del nombre de tu archivo.

Solución

- Paso 1

```
$ code cheat-sheet.txt
```

- Paso 2

```
$ cat cheat-sheet.txt
Mi hoja de trucos del Instituto

ls
```

```
lista los archivos y carpetas de un directorio

cd ~
cambia el directorio a la carpeta de inicio

...
```

Evaluación

¿Qué efecto produce el siguiente comando?

```
$ echo "¡Hola! Mi nombre es Mark!" > intro.txt
```

- Añade la línea "¡Hola! Mi nombre es Marcos" al contenido existente del archivo `intro.txt`.
- Comprueba si el contenido del archivo `intro.txt` contiene la línea "¡Hola! Mi nombre es Mark".
- Sustituye el contenido del archivo `intro.txt` por la línea "¡Hola! Me llamo Mark".
- Ninguna de las anteriores.

Creación del contenido del plan de estudios con Markdown

(Traducción por Nicolás Vaughan)

Markdown nos permite dar formato a elementos textuales como encabezados, énfasis, enlaces y listas en un archivo de texto plano utilizando un conjunto simplificado de anotaciones que los humanos pueden interpretar sin mucho entrenamiento. Los archivos de Markdown suelen tener la extensión `.md`. Utilizaremos Markdown para escribir un programa de estudios.

Markdown es un lenguaje de marcas para darle formato al texto. Al igual que el HTML, se añaden marcadores al texto plano para dar estilo y organizar el texto de un documento.

(Nota curiosa: Markdown se convierte en HTML para poder ser representado en un navegador. De hecho, es posible usar elementos de HTML dentro de un documento Markdown y serán correctamemte mostrados.)

Markdown tiene menos opciones para marcar el texto que HTML. Ha sido diseñado para ser más fácil de escribir y editar.

En Markdown, insertamos los encabezados con un caracter de número (#) como este:

```
# Mi encabezado del plan de estudios
```

(Nótese el espacio en blanco antes del texto del encabezado.)

Un subtítulo (H2) utiliza dos marcas de verificación como esta:

```
## Lecturas
```

Las lecciones de este taller se escribieron originalmente en Markdown. Puedes ver aquí su aspecto en bruto, sin renderizar.

Compáralo con esto: el código fuente de la página web de esta lección, escrito en HTML aquí.

Markdown también es posiblemente más sostenible y accesible que formatos como `.docx` debido a su simplicidad y a su capacidad de ser leído en múltiples plataformas. El uso de Markdown también está respaldado por herramientas de conversión de documentos como Pandoc que pueden convertir un archivo Markdown en un `.epub` con un solo comando introducido en el terminal.

A continuación, te ofrecemos algunos elementos clave para que estés preparado para hacer tu propio plan de estudios en Markdown.

Para dar énfasis, pon asteriscos alrededor de algún texto:

```
*Este texto aparecerá en cursiva.*
**Este texto aparecerá en negrita.**
```

Para dar énfasis, es necesario marcar dónde debe empezar y dónde debe terminar, por lo que se necesitan asteriscos al principio y al final del texto que se está enfatizando.

Para crear una lista con viñetas, pon un guión al principio de cada elemento de la lista:

```
- Lectura uno
- Lectura dos
- Lectura tres
```

Para crear un enlace, pon el texto de anclaje (el texto que verás) entre corchetes y la URL entre paréntesis, justo después del texto de anclaje entre corchetes. No pongas un espacio entre ellos:

```
Enseño en [La Universidad de Miami](www.miami.edu).
```

Los párrafos de texto se indican poniendo una línea en blanco entre ellos:

```
Esto es un párrafo en Markdown. Está separado del párrafo de abajo con una línea
en blanco. Si sabes de HTML, es algo así como la etiqueta <p>. Eso significa que
hay un pequeño espacio antes y después del párrafo cuando se renderiza.

Este es un segundo párrafo en Markdown, que utilizaré para decirte lo que me gusta
de Markdown. Me gusta el Markdown porque se ve bastante bien, aunque sea mínimo,
tanto si se ve la versión renderizada como la no renderizada. Es como un HTML
ordenado.
```

Lo anterior se representará así:

Esto es un párrafo en Markdown. Está separado del párrafo de abajo con una línea en blanco. Si sabes de HTML, es algo así como la etiqueta

. Eso significa que hay un pequeño espacio antes y después del párrafo cuando se renderiza.

Este es un segundo párrafo en Markdown, que utilizaré para decirte lo que me gusta de Markdown. Me gusta el Markdown porque se ve bastante bien, aunque sea mínimo, tanto si se ve la versión renderizada como la no renderizada. Es como un HTML ordenado.

Crear un archivo de syllabus

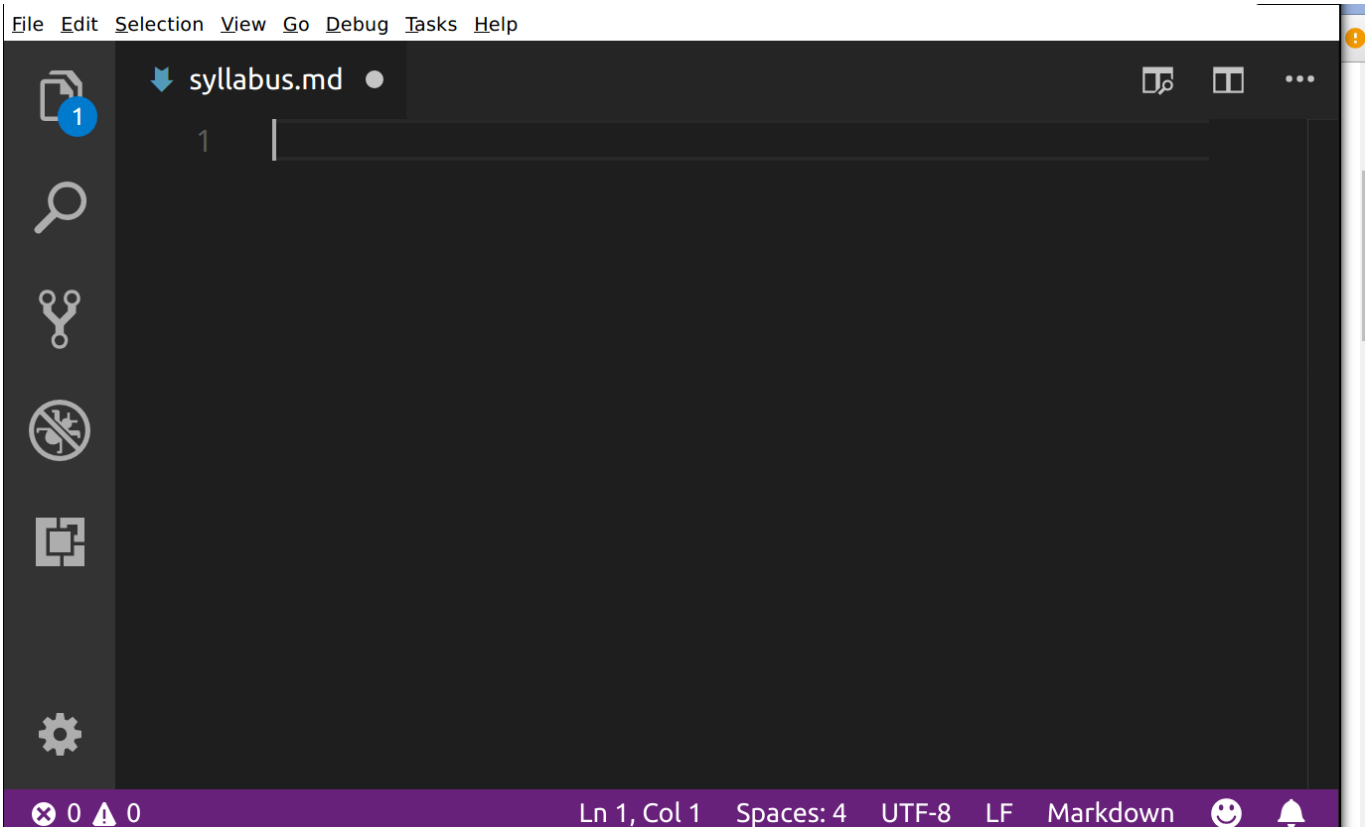
Para crear un archivo de texto plano, vamos a cambiar a nuestro editor de texto, Visual Studio Code, para crear y editar un archivo llamado `syllabus.md` y guardarlo en nuestra carpeta `projects`. La extensión `.md` indica que es un archivo Markdown.

En el terminal, comprueba que está en tu carpeta de `proyectos`. (*Tip*: utiliza `pwd` para ver en qué directorio estás actualmente).

A continuación, abre el archivo `syllabus.md` en Visual Studio Code utilizando:

```
code syllabus.md
```

Deberías ver aparecer una ventana con un aspecto similar al siguiente



Si Visual Studio Code no se abre cuando utilizas el comando `code` en tu terminal, ábrelo utilizando el Menú Inicio en Windows o la Búsqueda Spotlight en macOS como harías con cualquier otro programa. A continuación, haz clic en `Archivo > Abrir archivo` y utiliza el cuadro de diálogo para navegar hasta la carpeta `/Usuarios/<tu nombre>/Desktop/proyectos/git` y crea allí un archivo `syllabus.md`.

Escribiremos nuestro código Markdown en este archivo en la ventana de Visual Studio Code. En cualquier momento, puedes guardar tu archivo pulsando `control + s` en Windows o `⌘ + s` en macOS. También puedes hacer clic en el menú `Archivo` de la parte superior derecha, y luego seleccionar `Guardar` en el menú desplegable.

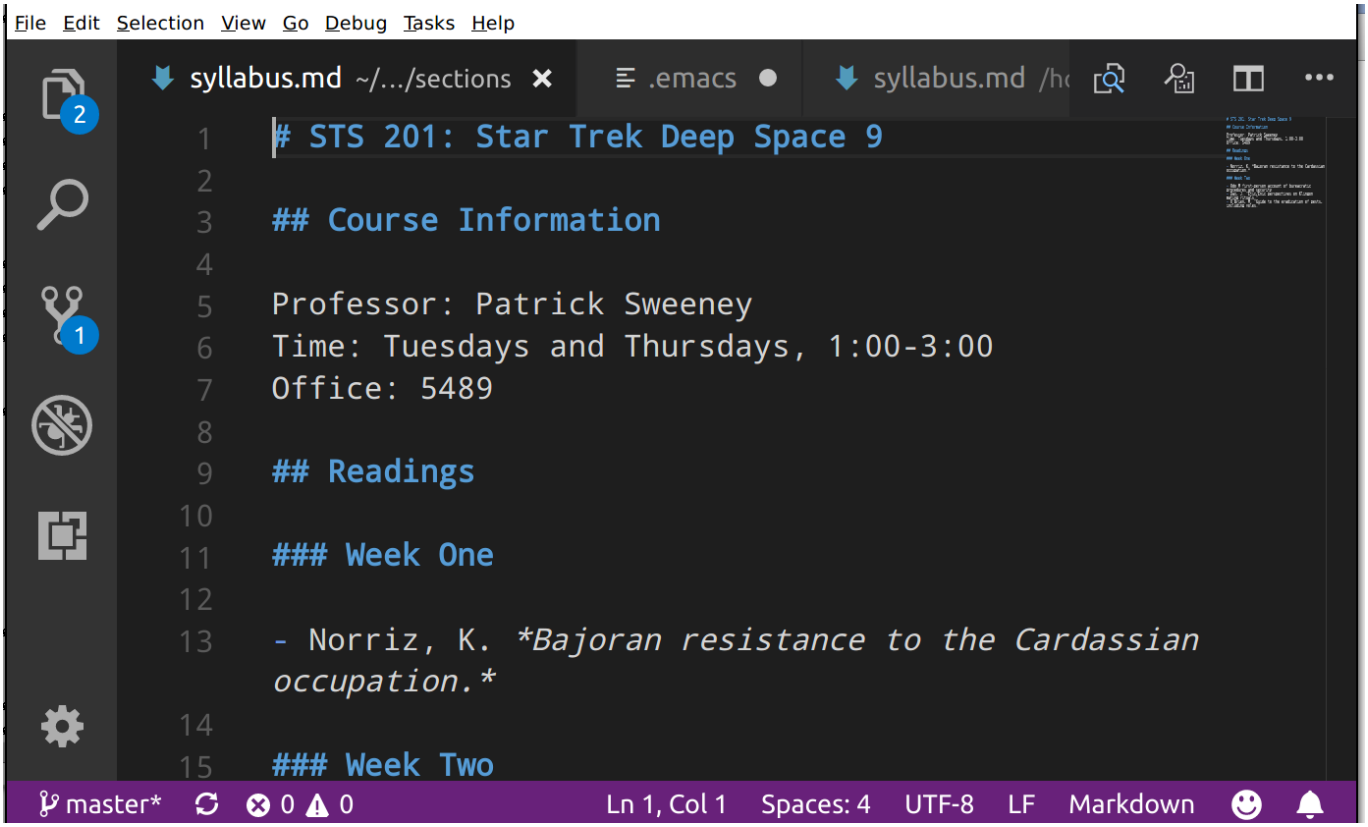
Ejercicio

Utiliza estos cinco elementos: encabezados, énfasis, listas, enlaces y párrafos, para crear un syllabus. Ten un encabezado principal que dé el título del curso (un `#`), y luego subencabezados para, al menos, la información del curso y las lecturas. Utiliza cursivas para los títulos de los libros e intenta incluir una lista en alguna parte.

Si quieres aprender más, te recomendamos encarecidamente esta hoja de trucos de Markdown para aprender elementos adicionales de Markdown y añadir algunas características extra como imágenes, citas en bloque o reglas horizontales.

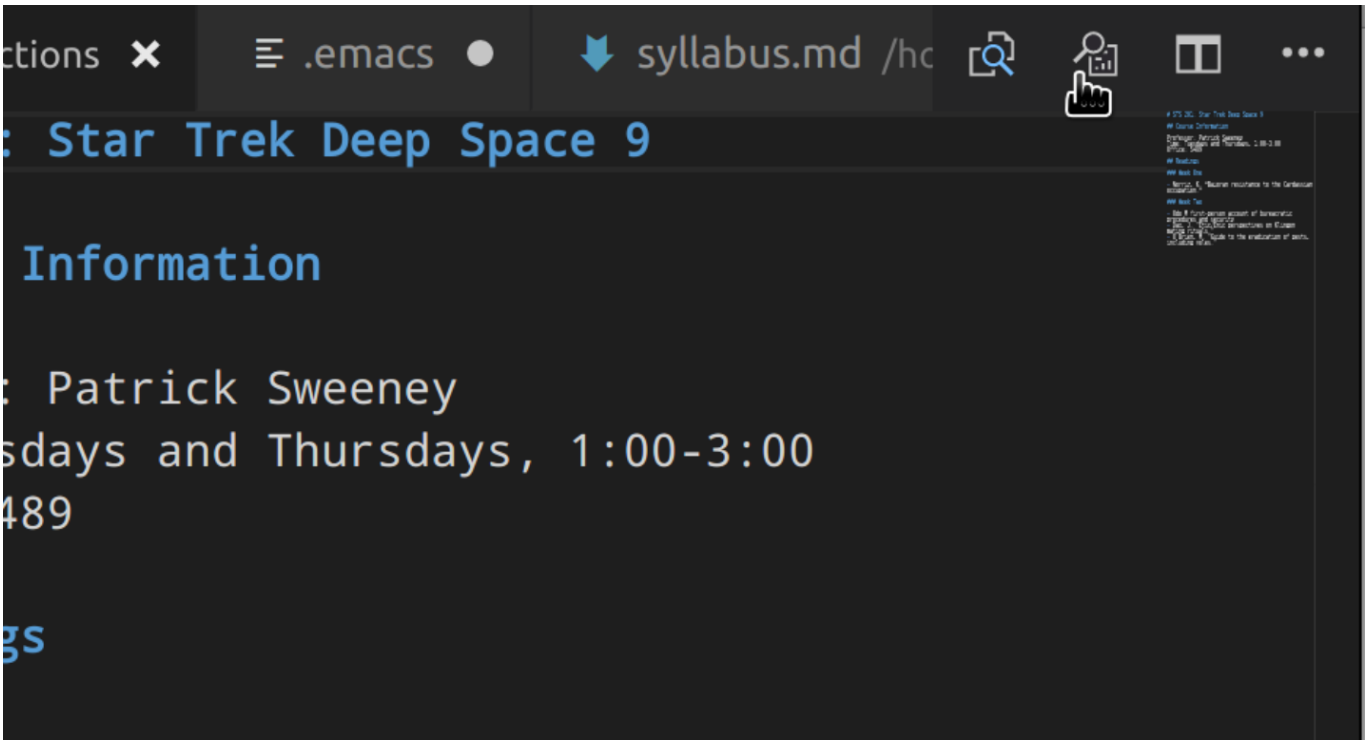
Ejemplo

Puedes ver un ejemplo de programa de estudios en forma de texto crudo aquí. Cuando es renderizado por GitHub, se ve como esto. Al editar el archivo Markdown en Visual Studio Code, podría tener este aspecto:

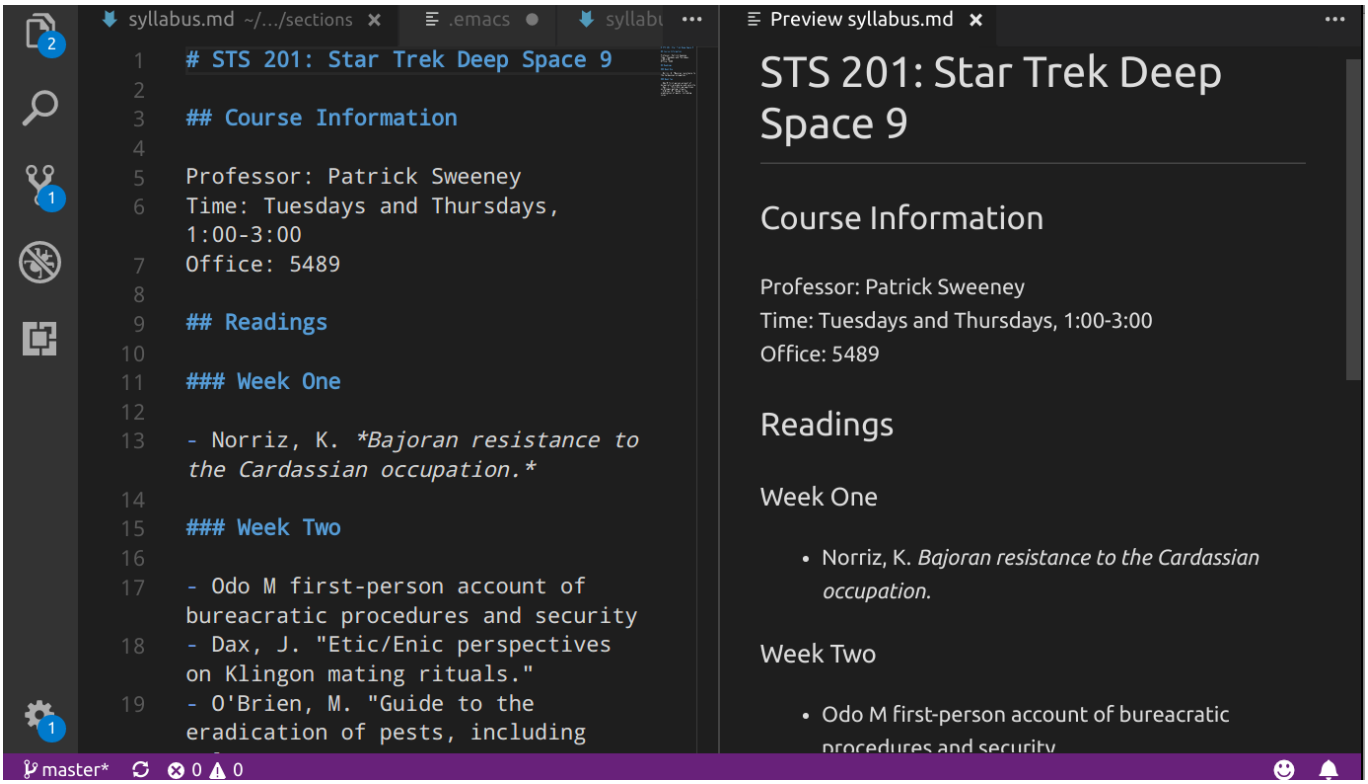


Consejos

1. Visual Studio Code también tiene una función de vista previa para Markdown. Pulsa el botón de vista previa en la parte superior derecha mientras editas tu archivo Markdown:



Obtendrás dos paneles uno al lado del otro. Tu archivo Markdown estará a la izquierda y su vista previa renderizada estará a la derecha:



2. Recuerda guardar tu trabajo (¡frecuentemente!) con `control` + `s` en Windows, o con ⌘ + `s` en macOS.

Evaluación

Qué describe mejor a Markdown:

- un software instalado en mi máquina local
- un lenguaje para formatear archivos de texto plano*

- un lenguaje que puede ser leído y renderizado por algunas plataformas basadas en la web*
- un software de control de versiones
- un software basado en la nube
- se refiere a las carpetas de proyectos como "repositorios".

Introducción a HTML

(Traducción por Nicolás Vaughan)

Introducción

Los sitios web parecen esas cosas mágicas que aparecen cuando abrimos nuestro navegador web (por ejemplo Chrome, Firefox y Safari). Sabemos que los sitios web son hipertextos, lo que significa que podemos hacer clic entre los enlaces, viajando de página en página hasta encontrar lo que necesitamos. Lo que puede resultar menos obvio sobre los sitios web es que, fundamentalmente, **los sitios web son documentos de texto plano**, normalmente escritos en HTML u otro lenguaje de marcado basado en la web, como XML o XHTML.

Dato curioso: **Más del 90% de todos los sitios web (cuyo lenguaje de marcado conocemos) utilizan HTML** (w3techs.com).

Lenguaje de marcado de hipertexto (HTML)

El HTML es un lenguaje de marcado utilizado para escribir documentos basados en la web. Nos permite proporcionar a los navegadores web información sobre el *contenido* de un documento. Podemos, por ejemplo, indicar que alguna parte de nuestro documento es un párrafo, una imagen, un título o un enlace. El navegador utiliza esta información cuando muestra el documento a los usuarios.

Lenguaje de marcado vs. Lenguaje de programación

El HTML es un lenguaje de *marcado*, no un lenguaje de programación. Los lenguajes de programación se utilizan para transformar datos, creando scripts que organizan una salida de datos basada en una entrada de datos determinada. Un lenguaje de marcado se utiliza para controlar la presentación de los datos.

Para un ejemplo práctico de esta diferencia, podemos pensar en las tablas. Un lenguaje de programación puede ayudarle a buscar en una tabla, a entender los tipos de datos que incluye la tabla, a encontrar puntos de datos concretos o a transformar su contenido en otros tipos de datos, como las frecuencias. En cambio, un lenguaje de marcado determinaría el contenido, el diseño y la presentación visual de la tabla.

Fundamentalmente, pues, un script o programa es un conjunto de instrucciones que se dan al computador. Un documento en un lenguaje de marcas determina cómo se presenta la información al usuario.

NOTA-Markup vs Markdown: Markdown y HTML son ambos tipos de lenguajes de marcado; Markdown es un juego de palabras. Los lenguajes de marcado ayudan a dar formato al contenido.

Hojas de estilo en cascada (CSS)

Las hojas de estilo en cascada (CSS) se suelen utilizar junto con el HTML. El HTML le dice al navegador cuáles son las diferentes partes de un documento *siendo*. El CSS le dice al navegador cómo deben *parecer* las partes del documento. Es esencialmente un conjunto de reglas que se aplican al renderizar un documento HTML. Su nombre -Hojas de estilo en cascada- hace referencia al hecho de que existe un orden de precedencia en la forma en que el navegador aplica las reglas CSS a un documento. Las reglas más específicas sobrescriben a las menos específicas.

¿Dónde entra Internet?

Juntos, estos lenguajes pueden utilizarse para escribir y dar estilo a un sitio web utilizando un editor de texto (como Visual Studio Code) directamente desde su computador. No se necesita acceso a Internet.

Sin embargo, el acceso a Internet es necesario si planea poner su sitio web a disposición del público. Haga clic aquí para aprender cómo llevar su sitio web desde su ordenador local a Internet.

En nuestras actividades durante este taller nos centraremos en la construcción de sitios web alojados localmente. Se trata de sitios web que puede abrir en su navegador, pero que sólo existen en su propio dispositivo y sólo son accesibles para usted. Los sitios web alojados localmente aún no están en Internet.

Evaluación

Verdadero o falso: La principal diferencia entre los lenguajes de marcado y los lenguajes de programación es que los lenguajes de marcado se utilizan para determinar el formato, la apariencia y el propósito del contenido, mientras que los lenguajes de programación se utilizan para transformar los datos.

- Verdadero*
- Falso

Palabras clave

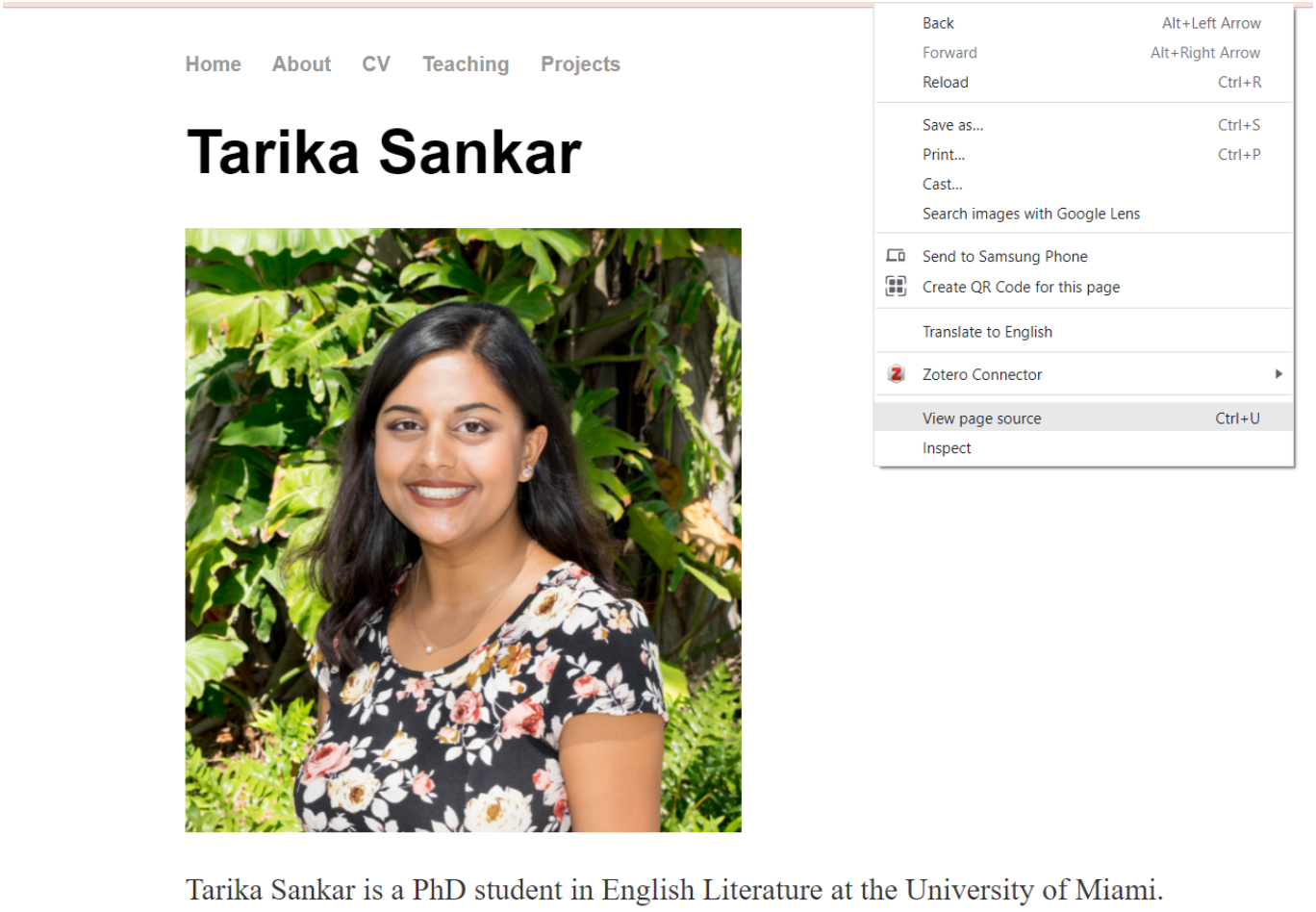
¿Recuerda los términos del glosario de esta sección?

- CSS
- HTML
- Markdown
- Lenguaje de programación

#Actividad de apertura

Nota_: por favor, utilice Firefox o Chrome. Safari no le permitirá completar esta actividad.

1. Abra un navegador web, preferiblemente Firefox.
2. Vaya a cualquier sitio web. El ejemplo siguiente es de tarikasankar.github.io.
3. Abra el menú secundario (con un ratón, sería el menú que se abre al hacer clic con el botón derecho en la página; en los ordenadores Mac, suele ser un toque con dos dedos en el trackpad, o puede pulsar el botón `control` y luego hacer clic en el trackpad).
4. Seleccione `Ver fuente de la página` en el menú desplegable.



Lo que está viendo

Una segunda pestaña debería abrirse en su navegador mostrando el código subyacente de la página. Este es el código que se utiliza para hacer y renderizar la página en su navegador.

Line wrap

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Tarika Sankar</title>
            <!-- link to main stylesheet -->
            <link rel="stylesheet" type="text/css" href="/css/main.css">
        </head>
        <body>
            <nav>
                <ul>
                    <li><a href="/">Home</a></li>
                    <li><a href="/about">About</a></li>
                    <li><a href="/cv">CV</a></li>
                    <li><a href="/teaching">Teaching</a></li>
                <li><a href="/projects">Projects</a></li>
                </ul>
            </nav>
            <div class="container">

            <div class="blurb">
    <h1>Tarika Sankar</h1>
    <img src="Professional headshot.jpg" alt="Tarika Sankar" style="width:437px;height:474px;" />
    <p>Tarika Sankar is a PhD student in English Literature at the University of Miami.</p>
</div>
<!-- /.blurb -->


            </div><!-- /.container -->
            <footer>
                <ul>
                    <li><a href="mailto:tgs46@miami.edu">email</a></li>
                    <li><a href="https://github.com/tarikasankar">github.com/tarikasankar</a></li>
                </ul>
            </footer>
        </body>
    </html>

```

En este taller, vamos a aprender a leer y escribir este código, y a renderizarlo en el navegador de su computador local. Al final discutiremos los siguientes pasos para que pueda alojar su nueva página web, haciéndola disponible para que otros puedan navegar por Internet.

Plantilla básica para HTML

A continuación se muestra una plantilla básica para un documento HTML vacío.

```
<!DOCTYPE html>
<html lang="es">

    <head>
        ...
    </head>

    <body>
        ...
    </body>

</html>
```

Los documentos HTML comienzan con una declaración `DOCTYPE` que indica qué versión de HTML se está utilizando. Esto le dice al navegador cómo leer el código que hay debajo para renderizar la página. Si la página web estuviera escrita con un lenguaje de marcado diferente (es decir, XML, XHTML), se lo diría aquí.

Después del `DOCTYPE`, vemos el comienzo del **Elemento Raíz**. TODO -todo el contenido- que quiera presentar en esta página y toda la información sobre cómo quiere que esa información se organice y tenga estilo va en el elemento raíz, y está delimitado por `<html>` y `</html>`.

El elemento raíz comienza indicando en qué idioma está escrito el documento; y en esta plantilla básica, `es` nos indica a nosotros y al computador que estamos escribiendo en inglés.

Dentro del elemento raíz de la plantilla básica anterior, observará las dos secciones principales de todos los documentos HTML: una sección de cabecera (delimitada por `<head>` y `</head>`) y una sección de cuerpo (delimitada por `<body>` y `</body>`).

La sección **head** contiene información básica sobre el archivo, como el título, las palabras clave, los autores, una breve descripción, etc. Aquí es también donde enlazará con su hoja de estilos CSS, que describe cómo quiere que sea el estilo de la página: colores, fuentes, tamaño del texto y posicionamiento de los elementos en la página.

La sección **body** contiene el contenido de la página, incluidos los párrafos, las imágenes, los enlaces, etc., e indica cómo debe estructurarse este contenido en la página.

Ejercicio

Cree una carpeta llamada `htmlpractice` en su carpeta de proyectos (`~/Desktop/projects/htmlpractice`). Dentro de esa carpeta, cree un nuevo archivo de texto y guárdelo como `index.html`.

Vamos a utilizar la línea de comandos para crear la nueva carpeta y el archivo:

1. Abra su terminal.
2. Navegue a su carpeta de proyectos usando este comando:

```
$ cd ~/Desktop/projects
```

3. Cree una nueva carpeta:

```
$ mkdir htmlpractice
```

4. Utilice su editor de texto Visual Studio Code para crear un archivo llamado `index.html`: `code index.html`.
5. Pegue la plantilla anterior (que comienza con `<< DOCTYPE html>`) en el nuevo archivo.

El archivo `index.html` es su página de inicio por defecto para el sitio web que estamos creando. Se trata de un estándar de la industria, porque los navegadores web tienden a reconocer la página `index.html` como la página de apertura al directorio que es su sitio web. Consulte aquí para obtener más explicaciones.

Una vez que haya creado su nuevo archivo, ábralo con un navegador web utilizando su interfaz gráfica de usuario:

En macOS, haga clic en el Finder en su dock (las aplicaciones en la parte inferior de la pantalla) y haga clic en Escritorio a la izquierda. Desde ahí, navegue hasta `proyectos`, y luego `htmlpractice`. Alternativamente, puede hacer clic en el icono de la carpeta de proyectos en su Escritorio y encontrarlo desde allí. Si utiliza un Mac y prefiere utilizar la línea de comandos, también puede escribir `open index.html` desde la carpeta `htmlpractice`.

En Windows, haga clic en el icono de la carpeta `projects` de su escritorio. Navegue hasta `projects`, y luego `htmlpractice`. Haga doble clic en el archivo `index.html`. Si no se abre en un navegador, haga clic con el botón derecho del ratón en el icono `index.html` y seleccione "Abrir con..." en el menú. Seleccione Firefox o Google Chrome en la lista de aplicaciones que aparece.

¿Qué ocurre?

Cuando abra la plantilla vacía, sólo verá una página web en blanco. Abra su menú secundario (haga clic con el botón derecho en Windows, mantenga pulsado `control` y haga clic con macOS) y vea la fuente de la página.

¿Qué debería ocurrir cuando abra cada uno de mis dos nuevos archivos?

Cuando "vea el código fuente de la página", debería ver el código de la plantilla básica.

No se renderiza ningún contenido en la página, porque no hay contenido en la plantilla en este momento.

Evaluación

¿Cuál de estos dos comandos HTML se conoce también como "elemento raíz"?

- `<< DOCTYPE html>`
- `<html lang="es">`*

Palabras clave

¿Recuerda los términos del glosario de esta sección?

- Elemento raíz

Etiquetas y elementos

Las etiquetas y los elementos son los componentes estructuradores de las páginas web html.

Los elementos identifican las diferentes partes de una página, como los párrafos, los encabezados, los títulos, el cuerpo del texto, las imágenes y otros. Los elementos están demarcados por etiquetas que encierran el contenido de un elemento (por ejemplo, `<head>` y `</head>` son etiquetas que denotan el elemento cabeza de su página).

Las etiquetas delimitan los elementos de dos maneras. Como en el caso del elemento de párrafo que aparece a continuación, un elemento puede tener una etiqueta de apertura y otra de cierre, con el contenido en medio.

```
<p>Esto es un párrafo.</p>

<p>
    Esto también es un párrafo.
</p>
```

Los elementos que tienen una etiqueta de apertura y cierre pueden tener otros elementos dentro de ellos. Dentro del elemento párrafo de abajo hay un elemento `<strong>`, que enfatiza el texto incluido poniéndolo en negrita.

```
<p>
    Cuando llegué a casa de la escuela, vi que había <strong>robado</strong> mi
pudín de chocolate.
</p>
```

Otros elementos tienen etiquetas de cierre automático, como ocurre con el elemento `<img>` (imagen) que aparece a continuación. Estas etiquetas también se denominan **etiquetas vacías**.

```
<img src="imagen.jpeg" />
```

Estos elementos no requieren una etiqueta de cierre separada. Las etiquetas de cierre no son necesarias porque usted no añadiría contenido dentro de estos elementos. Por ejemplo, no tiene sentido añadir ningún contenido adicional dentro de una imagen. Es una práctica común terminar las etiquetas de cierre como la de arriba con un `/` para marcar el final de la misma.

A continuación se muestra un HTML que añade un texto alternativo a una imagen -o un texto que describe la imagen-. Esta información añadida es un atributo-o algo que modifica la funcionalidad por defecto de un elemento.

```
<img alt="Esto es una imagen" src="imagen.jpeg" />
```

Añadir un texto alternativo a una imagen, como se ha hecho en este ejemplo, es de vital importancia. Esa información hace que la imagen sea más accesible para quienes ven su sitio. Por ejemplo, los usuarios con problemas de visión que no puedan ver su imagen seguirán entendiendo qué es y por qué está ahí si proporciona un texto alternativo que la describa.

Si observa la plantilla básica de su archivo `index.html`, verá que las secciones principales de su archivo tienen etiquetas de apertura y cierre. Cada uno de estos elementos principales acabará albergando muchos otros elementos, muchos de los cuales serán el contenido de nuestra página web.

Evaluación

¿Cuál de las siguientes afirmaciones es correcta?

- Los elementos tienen etiquetas de apertura y cierre.
- Las etiquetas tienen elementos de apertura y cierre.

Palabras clave

¿Recuerda los términos del glosario de esta sección?

- Etiqueta
- Elementos

Párrafos y encabezados

Los párrafos y los encabezados son los principales elementos textuales del cuerpo de sus páginas web. Como contienen el contenido que desea organizar y mostrar en su página web, se introducen en el elemento body.

Las etiquetas `<h1>`, `<h2>`, `<h3>`, etc. denotan **encabezados** y **subencabezados**, donde `<h1>` es el más grande y `<h6>` el más pequeño.

Las etiquetas `<p>` denotan **párrafos**, o bloques de texto.

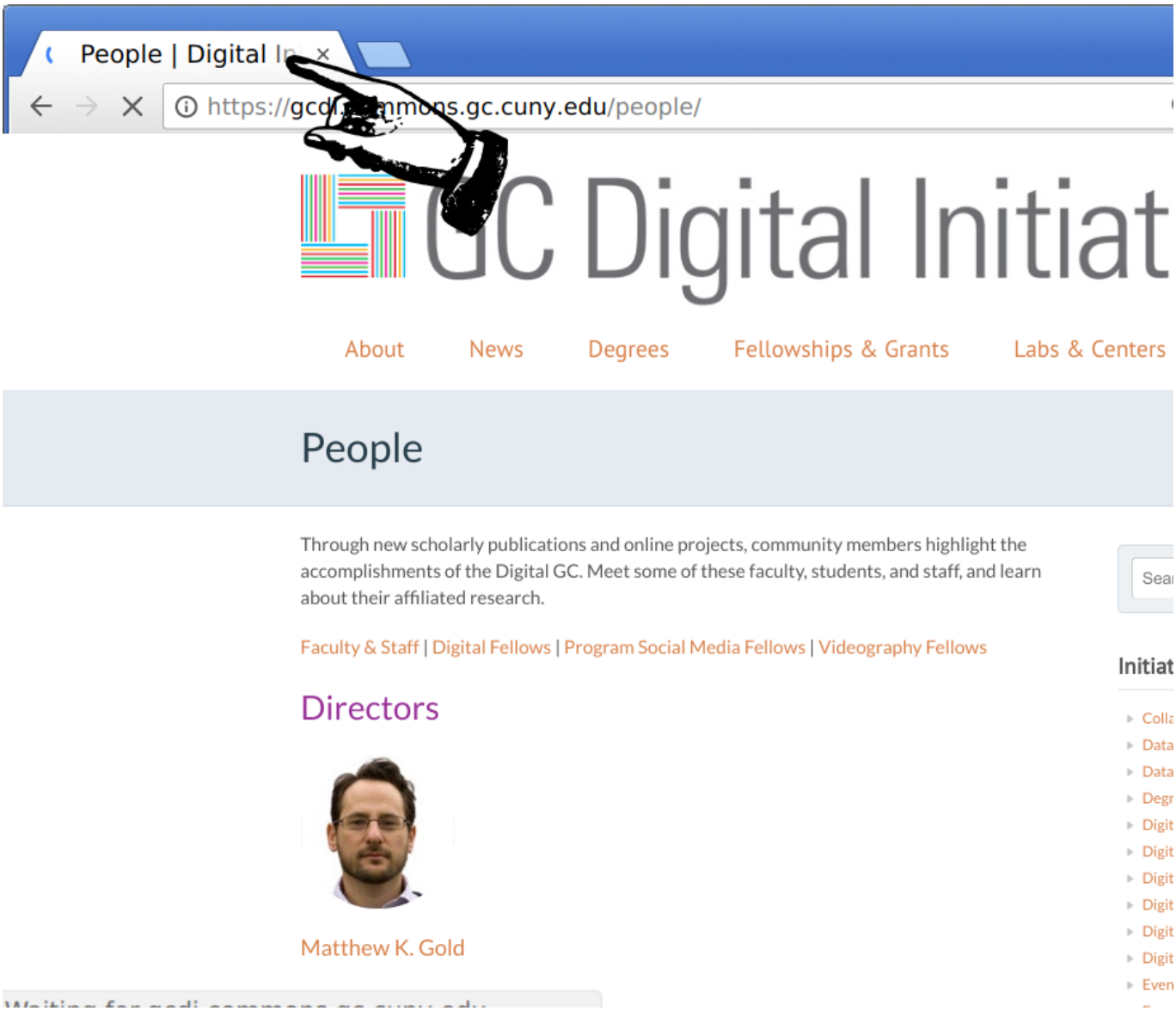
```
<!DOCTYPE html>
    <html lang="es">

    <head>
        <title>Una historia aburrida</title>
    </head>

    <body>
        <h1>
            Limpiando mi caldera
        </h1>
        <p>
            Cuando llegué a mi sótano ese día, supe que tenía que limpiar mi
caldera. Estaba demasiado sucia. Sinceramente, se estaba convirtiendo en un
peligro. Así que cogí mi cepillo de alambre y me puse mi par de monos de limpieza
de calderas más duraderos. Iba a ser un día largo.
        </p>
    </body>

</html>
```

Observe que el `<título>` está en el elemento `<head>`, que es donde va la información sobre la página web. El título no aparece en la página, sino en otra parte del navegador cuando se muestra la página. Por ejemplo, en Chrome, el título aparece en la pestaña encima de la barra de navegación.



Tenga en cuenta también que los elementos y etiquetas utilizados en HTML tienen *significado*. Proporcionan información sobre la estructura de una página web, mostrando cómo sus partes funcionan juntas. Quienes hacen uso de tecnologías de asistencia, como los lectores de pantalla, se basan en esta información semántica para navegar por la página. Por lo tanto, es importante utilizar elementos como las cabeceras sólo cuando la información marcada lo requiera. Hacer el texto grande y/o en negrita para conseguir un efecto visual debe hacerse utilizando CSS. La Red de Desarrolladores de Mozilla tiene una buena información introductoria sobre HTML semántico.

Ejercicio

Utilizando su editor de texto, añada lo siguiente a su `index.html`:

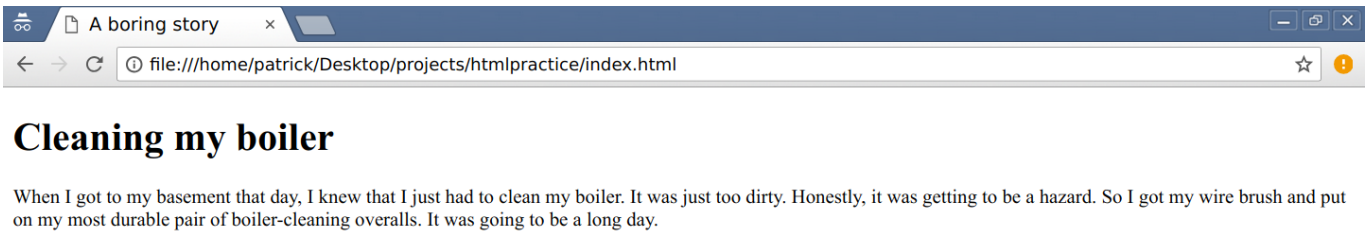
- Título
- Título
- Párrafo

Luego, vuelva a guardar el archivo. Vuelva a abrirlo en su navegador o actualice la página si sigue abierta.

¿Qué nota sobre cómo está organizada la información en la página web? En otras palabras, ¿dónde están el título, el encabezamiento y el texto del párrafo?

¿Qué debería ver?

El título debería aparecer en la parte superior de la página, seguido del texto del párrafo. El texto del encabezamiento debería ser más grande. El título debería aparecer en la pestaña de la ventana del navegador.



Evaluación

Si quisiera indicar que "Acerca de" es un subtítulo en mi página, ¿qué elemento debería utilizar?

- `<head>`
- `<h2>`*

Enlaces

Los enlaces son la base de la World Wide Web y, por tanto, son un componente importante de la mayoría de los sitios web. El texto de los hipervínculos permite a los usuarios desplazarse entre las distintas páginas web de su sitio (a veces en forma de menú o barra de navegación), o conectar con otros recursos o información de otros sitios web.

La etiqueta `<a>`, o **etiqueta de anclaje**, crea un enlace a otro documento. Puede utilizar la etiqueta `<a>` para enlazar con otros documentos o páginas web que haya creado para el mismo sitio o con documentos situados en otro lugar de la web. También puede utilizarla para enlazar con un lugar concreto de una página - veremos un ejemplo de ello en la sección sobre Clases e IDs.

Opción uno: Enlaces relativos

Los enlaces relativos toman la página actual como punto de origen y buscan otros archivos dentro de la misma carpeta o directorio. Este método es útil para crear enlaces a páginas dentro de su propio sitio.

Lo siguiente aparece como un enlace a la página `about.html` en la misma carpeta que `index.html`:

```
<a href="about.html">Acerca de</a>
```

En su página web aparecerá como

> Acerca de

Este enlace está pidiendo al navegador que busque un archivo titulado `about.html` en la misma carpeta. Si un archivo llamado `about.html` no se encuentra en la misma carpeta, al hacer clic en el enlace se producirá un error `404` ("Página no encontrada").

Opción dos: Enlaces absolutos

Un enlace absoluto incluye información que permite al navegador encontrar recursos en otros sitios web. Esta información incluye el dominio del sitio -como `google.com`- y a menudo el protocolo -como `http` o `https`.

```
<a href="https://www.google.com">Google</a>
```

En su página web aparecerá como

> Google

Esta vía está dirigiendo a su navegador para que busque en línea este documento de texto en la dirección URL proporcionada.

Más sobre los enlaces

Cada ejemplo anterior incluye una `href` -una referencia de hipertexto- que es un ejemplo de un **atributo**. Los atributos ofrecen información secundaria sobre un elemento.

La etiqueta `<a>`, o etiqueta de anclaje, crea un enlace. El texto dentro de las etiquetas `<a>` y `</a>`, el texto de anclaje, es lo que un visitante del sitio verá y podrá pulsar. El atributo `href=` indica al navegador a dónde debe dirigirse el usuario cuando haga clic en el enlace.

Hay otra diferencia técnica entre las dos opciones anteriores.

Enlaces relativos vs. absolutos: Cuándo utilizar uno u otro

Utilice los enlaces relativos cuando se refiera a páginas de su propio sitio. La principal ventaja de utilizar enlaces relativos a páginas de su sitio es que éste no se romperá si se traslada a una carpeta o entorno diferente.

Ejercicio

1. Cree un nuevo archivo de texto llamado `about.html` en su carpeta `htmlpractice`. Copie el HTML de su archivo `index.html`, pero cambie el texto del elemento `<h1>` por "Acerca de".

2. En su archivo `index.html`, añada un enlace relativo que lleve a su página "Acerca de".
3. Añada también un enlace relativo desde su página "Acerca de" a su página `index.html`. En este enlace, llame a su página `index.html` "Inicio" (Recordatorio: `index.html` es la página de inicio por defecto)
4. Por último, incluya un enlace absoluto a una página de su elección. Recuerde que un enlace absoluto incluye el protocolo (por ejemplo, `http:`) y también un dominio (por ejemplo, `cuny.edu`), como `http://cuny.edu/about`.
5. Vuelva a guardar sus archivos de texto y vuelva a abrirlos o actualizarlos en su navegador.

Compruebe si ha funcionado

Cuando sus páginas estén actualizadas, debería poder navegar desde su página "Inicio" a su página "Acerca de", y viceversa. También debería poder navegar a la página web externa.

Evaluación

¿Cuál de las siguientes opciones es un enlace relativo?

- `<a href="https://www.nytimes.com/">El New York Times</a>`
- `<a href="digitalProject.html">Proyecto digital</a>`

Palabras clave

¿Recuerda los términos del glosario de esta sección?

- Atributos

Imágenes

Las imágenes son otro componente importante de los sitios web. A veces sólo ayudan a dar vida a su sitio web, pero otras veces pueden ayudar a comunicar información a los usuarios.

Las imágenes se crean con la etiqueta `<img>`. Al igual que la etiqueta `<a>`, `<img>` requiere un atributo, en este caso `src`. El atributo `src` significa "fuente" y comunica al navegador información secundaria que identifica y localiza la imagen. A diferencia de muchas otras etiquetas, la etiqueta `<img>` no necesita cerrarse, lo que la convierte en un ejemplo de etiqueta vacía.

El siguiente elemento extrae una imagen situada en la misma carpeta que el archivo `.html`:

```
<img src="grito.jpeg" />
```

Aquí se aplican las mismas reglas que con el atributo `href`: si la imagen no se encuentra en la misma carpeta que el documento que está escribiendo, el navegador no la encontrará. Si el navegador no puede encontrar un recurso de imagen, verá un icono de imagen rota, como éste de Chrome:

Nota: Algunos sitios utilizan muchas imágenes. Cuando este es el caso, puede ser útil mantener las imágenes en una carpeta separada dentro de la estructura de su sitio. Para que el navegador pueda encontrar una imagen en ese caso, basta con añadir el directorio delante del nombre del archivo. Por ejemplo, si tiene una carpeta llamada images en la misma carpeta que su archivo index.html, y scream.jpeg está en esa carpeta, cambiaría la etiqueta void anterior por `<img src="images/scream.jpeg" />`.

Hacer que las imágenes sean accesibles

Como ya se ha señalado brevemente, el texto alternativo, o texto alt, es un "texto descriptivo asociado a una imagen que sirve para el mismo propósito y transmite la misma información esencial que la imagen" (para más información, véase Manual de estilo de Wikipedia/Accesibilidad/Texto alternativo para imágenes, y es importante para garantizar que el contenido transmitido por las imágenes sea accesible para todos.

Para añadir texto alternativo a una imagen, se añade un atributo adicional, `alt`, seguido de su texto descriptivo. Por ejemplo

```
<img src="nombredearchivo.png" alt="El texto entre estas comillas describe la imagen" />
```

Para más información sobre el uso del texto alternativo, consulte lo que dice la Administración de la Seguridad Social.

¿Qué imágenes puedo utilizar en mi sitio?

Si tiene previsto utilizar imágenes que no haya tomado o hecho usted mismo, tendrá que utilizar imágenes de "dominio público" o de "licencia abierta".

Esta guía del OpenLab de City Tech incluye más información sobre las licencias y una lista de lugares donde puede encontrar imágenes reutilizables.

Ejercicio

Descargue y guarde una imagen de la web, o mueva una imagen de su computador a la misma carpeta que su archivo `index.html`.

Sugerencia: Dé al archivo un nombre sencillo. Además, el nombre **no puede** tener espacios. Una buena práctica es utilizar guiones o guiones bajos donde de otro modo habría espacios. Por ejemplo: `esto-es-una-imagen.jpg` o `esto-es-una-imagen.jpg`.

Utilizando el código anterior como referencia, añada esa imagen en su archivo `index.html`, vuelva a guardar el archivo y abra o actualice la página en su navegador. Su imagen debería aparecer ahora en la página.

Evaluación

Verdadero o Falso: ¿Incluir el "texto alternativo" en las páginas web mejora su accesibilidad?

- Verdadero*
- Falso

Convenciones

A medida que hemos ido recorriendo los diferentes componentes de la creación de una página web, es probable que haya notado algunas convenciones comunes o estándares de la industria para crear una página web utilizando HTML. ¿Puede adivinar alguna de ellas?

Aquí tiene algunos ejemplos:

- Algunas etiquetas se cierran solas, mientras que otras requieren una etiqueta de cierre. Las etiquetas de autocierre se denominan etiquetas vacías y, por lo general, se cierran solas porque no es necesario ni se desea añadir otro elemento dentro de la etiqueta. También suelen terminar con una barra diagonal (`/`) para marcar el final de la etiqueta.
- Utilice las minúsculas. Aunque el HTML no distingue entre mayúsculas y minúsculas, facilita el escaneo del código y le da un aspecto más coherente.
- Su código debe estar anidado. Esto tampoco es una necesidad técnica: el espacio en blanco no tiene ningún significado en html. Sin embargo, esto hace que sea más fácil escanear el código rápidamente, ¡lo que es particularmente útil cuando se encuentra con errores!

Ejercicio: Crear un sitio web

En este ejercicio, comenzaremos a crear un sitio web para su portafolio personal o curso. Utilizando las etiquetas que acabamos de revisar, y otras dos adicionales (ver más abajo), haremos un sitio web básico que proporcione información sobre su perfil académico y/o cursos.

El primer paso es crear una nueva carpeta llamada `website` en su carpeta `projects` en su escritorio. Cree un archivo `index.html` así como un archivo `about.html` dentro de esa carpeta. Éstos serán la página de aterrizaje de su sitio, y una página complementaria que proporcione información sobre los organizadores de su Instituto de Humanidades Digitales.

Añada HTML a su archivo `index.html`. Esta página debe incluir lo siguiente:

- Doctype
- Elemento raíz
- Cabeza y un cuerpo
- Título de la página
- Un título
- Un párrafo
- Una imagen con texto alternativo
- Un menú o barra de navegación que enlace con sus páginas de Inicio y Acerca de

Piense en el orden de su contenido a medida que va montando el cuerpo de su página. No se preocupe por que el contenido sea el adecuado. Lo importante de este ejercicio es repasar la estructura de una página web y practicar su creación.

Etiquetas adicionales

Aquí tiene dos etiquetas adicionales que pueden ser útiles para montar su página:

Para **hacer una lista**, se abre y se cierra con las etiquetas `<ul>`, y cada elemento es una etiqueta `<li>` encerrada:

```
<ul>
    <li>Artículo 1</li>
    <li>Artículo 2</li>
    <li>Artículo 3</li>
</ul>
```

El HTML anterior producirá una lista desordenada (con viñetas). Para crear una lista ordenada (numerada) en su lugar, sólo tiene que sustituir `<ol>` y `</ol>` por `<ul>` y `</ul>`.

(Esto puede resultar útil al confeccionar su menú o barra de navegación).

Para **hacer un salto de línea** o dar espacio entre diferentes elementos:

```
<br />
```

Fundamentos de CSS

CSS son las siglas de Cascading Style Sheets (hojas de estilo en cascada). Este lenguaje funciona en coordinación con HTML, pero es un lenguaje propio con sus propias reglas y terminología. A diferencia del HTML, que es responsable del contenido de la página, el CSS es responsable de la presentación de la página.

Algunos ejemplos de lo que CSS puede ayudarle a determinar son

- Qué color de fondo quiere utilizar para la página o para un párrafo.
- Qué tipo o tamaño de letra quiere para los encabezados o el texto normal.
- Qué tamaño quiere para las imágenes y si las quiere alineadas al centro, a la izquierda o a la derecha.
- Dónde aparecen los elementos en la página.
- Si los elementos son visibles para el usuario o no.

Evaluación

¿Es CSS un lenguaje de marcado o un lenguaje de programación?

- Lenguaje de marcado*
- Lenguaje de programación

Integración de CSS y HTML

Para que el CSS informe el estilo del contenido de la página, debe integrarse con su HTML. El CSS puede integrarse en su HTML de tres maneras:

1. en línea
2. interna
3. externa (*recomendada*)

Opción 1: Inline

El estilo inline añade CSS directamente en el HTML de una página para ajustar el estilo de determinadas partes de la misma.

Por ejemplo, si quiere que el texto del primer párrafo sea rojo, pero el del segundo sea azul:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Acerca de</title>
    </head>

    <body>
        <p style="color: rojo">
            Contenido del párrafo
        </p>
        <p style="color: azul">
            Contenido del párrafo
        </p>
    </body>
</html>
```

Opción 2: Interno

El estilo interno también añade el CSS directamente en el HTML, pero lo mantiene separado del código de contenido de la página añadiéndolo en la cabecera mediante la etiqueta `<style>`. Al utilizar el estilo interno está proporcionando reglas de estilo para toda la página. Por ejemplo, si quiere que todos los encabezados sean azules

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Acerca de</title>
        <style>
            h1 {
              color: azul;
            }
        </style>
```

```
    </head>

    <body>
        <h1>
            Encabezado uno
        </h1>
        <p>
            Contenido del párrafo
        </p>
        <h1>
            Título dos
        </h1>
        <p>
            Contenido del párrafo
        </p>
    </body>
</html>
```

Opción 3: Externo (Recomendado)

El estilo externo crea un documento completamente separado para su CSS que se vinculará a su HTML en la sección head de su documento HTML utilizando el código que se indica a continuación. Este documento separado se llama *stylesheet* y a menudo se llama `style.css`. El documento se enlaza a través de una etiqueta `<link>` que vive dentro de la etiqueta padre `<head>`. Su atributo `href` es un enlace relativo al documento en algún lugar en relación con el documento que lo referencia.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo de CSS</title>
        <link rel="stylesheet" href="style.css" />
    </head>

    <body>
        ...
    </body>
</html>
```

Mejores prácticas

Es la mejor práctica utilizar la opción 3, el estilo externo, por varias razones:

1. Nos ayuda a recordar en qué se centra cada lenguaje: HTML es para el *contenido*, CSS es para el *estilo*. (Esto se conoce a veces como el principio de "separación de intereses")
2. Nos ayuda a mantener la coherencia en las distintas páginas de nuestro sitio, ya que *múltiples archivos HTML pueden enlazar con la misma hoja de estilos*.
3. Como varios archivos HTML pueden enlazar con el mismo archivo CSS, no es necesario escribir el mismo código CSS varias veces. Basta con una vez. (Esto se conoce a veces como el principio de "No te

repitas", o simplemente *DRY*).

La opción 3, el estilo externo, es la preferida por la mayoría de los desarrolladores web porque es más manejable y porque se presta a una mayor coherencia en todo el sitio.

Ejercicio

Cree una hoja de estilo en blanco utilizando la línea de comandos (siguiendo la opción 3, estilo externo, descrita anteriormente). En su documento `index.html`, vincule su hoja de estilo y vuelva a guardar el archivo.

Para enlazar su hoja de estilo con su archivo `index.html`, inserte el siguiente código en el elemento head de ese archivo `index.html`:

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

Evaluación

¿Es el siguiente fragmento de código un ejemplo de estilo en línea o de estilo interno?

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Página de inicio</title>
        <style>
            h1 {
              font-family: monospace;
            }
            p {
             font-family: fantasy;
            }
        </style>
    </head>

    <body>
        <h1>
            ¡Biblioteca en línea para todos!
        </h1>
        <p>
            ¡Libros gratis aquí!
        </p>
    </body>
</html>
```

- Estilo interno
- Estilización interna*

Conjuntos de reglas

CSS se basa en selectores y declaraciones, que juntos forman conjuntos de reglas (o simplemente "reglas"). Los conjuntos de reglas comprenden un archivo de estilo externo con una extensión `.css`. Este es el contenido de un archivo `.css` de ejemplo:

```
h1 {
    color: orange;
    font-style: italic;
}

p {
    font-family: sans-serif;
    font-style: normal;
}

#navbar {
    background-color: yellow;
    padding: 80px;
}

.intro {
    font-family: arial;
    background-color: grey;
    color: dark-grey;
}
```

La primera regla (que comienza con el selector `h1`) se aplica a todas las etiquetas `<h1>` de cada página en la que su documento HTML enlaza con su hoja de estilos, y cambia el estilo de letra y la visualización de esos encabezados.

Las líneas dentro de las llaves (es decir, `{ ... }`) se llaman **declaraciones**, y cambian el formato de los elementos en el documento HTML. Cada línea de la declaración establece el valor de una **propiedad** y termina con un punto y coma (`;`).

Observe la diferente sintaxis que se utiliza para seleccionar elementos para su estilización con conjuntos de reglas. Los dos selectores inferiores se utilizan para aplicar conjuntos de reglas a los **ID** y a las **clases**. En general, la adición de clases e IDs a los elementos HTML permite un estilo más específico - ¡más sobre esto pronto!

Ejercicio

Copie y pegue las reglas CSS anteriores en su archivo `style.css` y vuelva a guardar el archivo. A continuación, abra o actualice su archivo `index.html` en su navegador y vea lo que sucede.

**¿Qué debería ocurrir?

El formato del texto de su página debería cambiar en consecuencia. Su `<h1>` debería ser naranja y en cursiva, por ejemplo.

¿Qué otras reglas podría establecer para los diferentes elementos HTML? Haga una búsqueda rápida en Google de una regla CSS que cambie la apariencia de su página, como poner un borde alrededor de un

elemento.

Evaluación

¿Cómo asociamos un archivo CSS a una página HTML?

- Incluyendo un enlace al archivo CSS en el elemento `<head>` de la página HTML.
- Poniendo el archivo CSS en la misma carpeta que la página HTML.

Palabras clave

¿Recuerda los términos del glosario de esta sección?

- Selectores CSS
- Clase
- ID