**Resumen del Sprint 3**

El Sprint 3 ahonda en CSS y presenta cómo se trabaja con el sistema de archivos y Git.

**Capítulos**

Estructura y rutas de archivo

**Archivos**

* Un archivo es una recopilación de datos que un usuario o un programa puede escribir, editar o leer.
* Un directorio (o carpeta) es un contenedor para almacenar archivos.
* Tipos de archivos — archivos de datos y archivos de programa.
  + Archivos de datos: Estos son los archivos con los que trabajan los desarrolladores web. Incluyen archivos gráficos y de texto, archivos de bases de datos y JavaScript, Python u otros archivos de script.
  + Archivos de programa: Estos archivos 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 atajos o enlaces en sistemas Windows.

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

* El directorio raíz de un proyecto o sistema de archivos es el directorio principal en el que se almacenan todos los archivos del sistema.
* Ejemplo de la estructura organizativa del proyecto
* root/
* index.html
* images/
* styles/
* scripts/
* fonts/

1. El directorio images incluye iconos, imágenes de fondo e imágenes de contenido. Pueden tener diferentes extensiones.
2. Dentro del directorio styles encontraremos nuestras hojas de estilo.
3. La carpeta scripts guarda nuestros archivos JavaScript.
4. Por último, el directorio fonts almacena diferentes fuentes que se han utilizado en el proyecto. index.html está en la raíz.

**Rutas de archivos absolutas**

Ruta a un archivo que comienza en la raíz del sistema de archivos

* En Windows, la raíz se llama C:/. En Mac y Linux, se suele llamar /.
* Anatomía de una ruta de archivo local (absoluta). Comienza con el directorio raíz.



* Anatomía de una ruta de archivo en línea (absoluta). Comienza con el protocolo.



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

Un grupo de servidores remotos distribuidos que permite subir más rápidamente archivos o recursos web alojados en dicho grupo de servidores.

* Puedes subir archivos desde CDN utilizando las etiquetas <link> y <script>.

**Rutas de archivo relativas**

Haz una ruta de un archivo que comienza en el directorio de trabajo actual

* Cuando se escriben rutas relativas
  + . se refiere al directorio de trabajo actual
  + .. se desplaza un nivel fuera del directorio actual, al directorio raíz
* Ejemplo

Estructura de archivos hipotética

index.html

styles/

styles.css

more-styles/

more-styles.css

vendor/

normalize.css

images/

image-1.png

image-2.png

icons/

icon.svg

*<!-- index.html -->*

<!DOCTYPE html>

<html lang="en">

<head>

<-- un solo . se refiere al directorio de trabajo actual (es decir, donde esté este archivo) -->

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

<link rel="stylesheet" href="./vendor/normalize.css">

</head>

<body></body>

</html>

*/\* styles.css \*/*

.some-element {

*/\* .. nos lleva al directorio raíz, donde podemos acceder al directorio de imágenes\*/*

background-image: url('../images/image-1.png');

}

*/\* more-styles.css \*/*

.some-other-element {

*/\* Un uso de .. sólo nos llevará al directorio styles/. Necesitamos una segunda aplicación para llegar al directorio raíz. \*/*

background-image: url('../../images/image-2.png');

}

**Reglas-at de CSS: @import**

La regla-at @import permite incluir archivos externos (como fuentes o fragmentos extra de código CSS) en la hoja de estilos actual.

* Sintaxis

*/\* styles.css \*/*

*/\* importar los estilos de other-styles.css a styles.css \*/*

@import url("path/to/other-styles.css");

*/\* resto de estilos ... \*/*

**Git para principiantes**

**Introducción**

Git es un [sistema de control de versiones](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control) *(materiales en inglés)*, es decir, un sistema para rastrear los cambios de estado de una recopilación de archivos a lo largo del tiempo. Git tiene una interfaz de línea de comandos, pero la interfaz gráfica la proporciona Github, una plataforma basada en la web que utiliza Git para el control de versiones, además de ofrecer otras características adicionales.

**Línea de comandos**

* En Windows, utiliza Git Bash; en Mac/Linux utiliza la Terminal.
* Comandos básicos ([ver ficha de ayuda](https://tripleten.com/trainer/web/lesson/bc4b2664-c3ea-4ad7-b623-824d456e7a10/))
  + .: una ruta de archivo; hace referencia al directorio actual de trabajo.
  + ..: en una ruta de archivo, se refiere al directorio *padre*.
  + pwd: muestra el cwd ("current working directory", directorio actual de trabajo en español).
  + ls: hace una lista de los archivos en el directorio actual de trabajo.
  + ls path/to/somewhere: hace una lista de los archivos en path/to/somewhere.
  + cd some-folder: cambia el cwd a some-folder (donde some-folder fue una carpeta en el cwd original)
  + cd ..: cambia el cwd al directorio padre.
  + mkdir directory-name: crea un directorio llamado directory-name en cwd.
  + touch file.ext: crea un archivo llamado file.ext.
  + code file.ext: crea file.ext y lo abre en VSCode.
    - Nota: touch y code toman argumentos adicionales separados por espacios.
  + rm file.ext: elimina file.ext.
  + rmdir dir-name: elimina el directorio *vacío*dir-name.
  + rmdir -rf dir-name: elimina el directorio *y* su contenido. *¡Ten cuidado; esto es irreversible!*
* Observaciones:
  + Si el nombre de un archivo o carpeta tiene espacios, debe ir entre comillas.
  + Por esta razón, no hagas archivos/carpetas con espacios en sus nombres.
  + Recuerda utilizar el tabulador para autocompletar la función de la ruta de archivo de la mayoría de los programas de terminal. Si pulsar una sola vez el tabulador no completa la ruta del archivo, púlsalo una segunda vez para ver una lista de todas las posibles terminaciones.

**Configuración de Git**

* Cambia el nombre de usuario y el correo electrónico.

git config --global user.name "Estudiante de TripleTen"

# introduce tu nombre y apellidos o tu alias utilizando letras latinas entre comillas

git config --global user.email tripleten\_student@gmail.com

# introduce tu dirección de correo electrónico real aquí

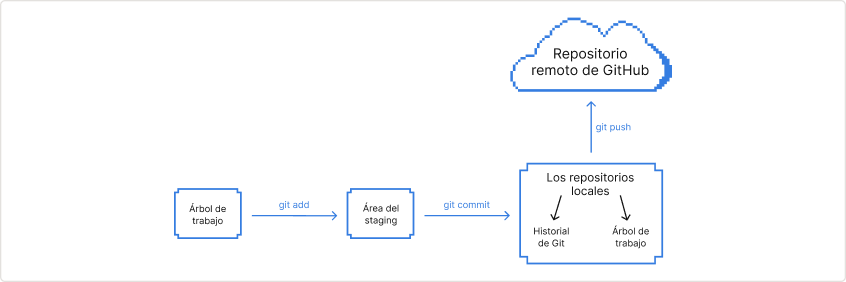
* Cambia la rama predeterminada a main

git config --global init.defaultBranch main

* Enumera los ajustes de configuración con git config --list

**El flujo de trabajo de Git/GitHub**

Un resumen del flujo de trabajo básico.



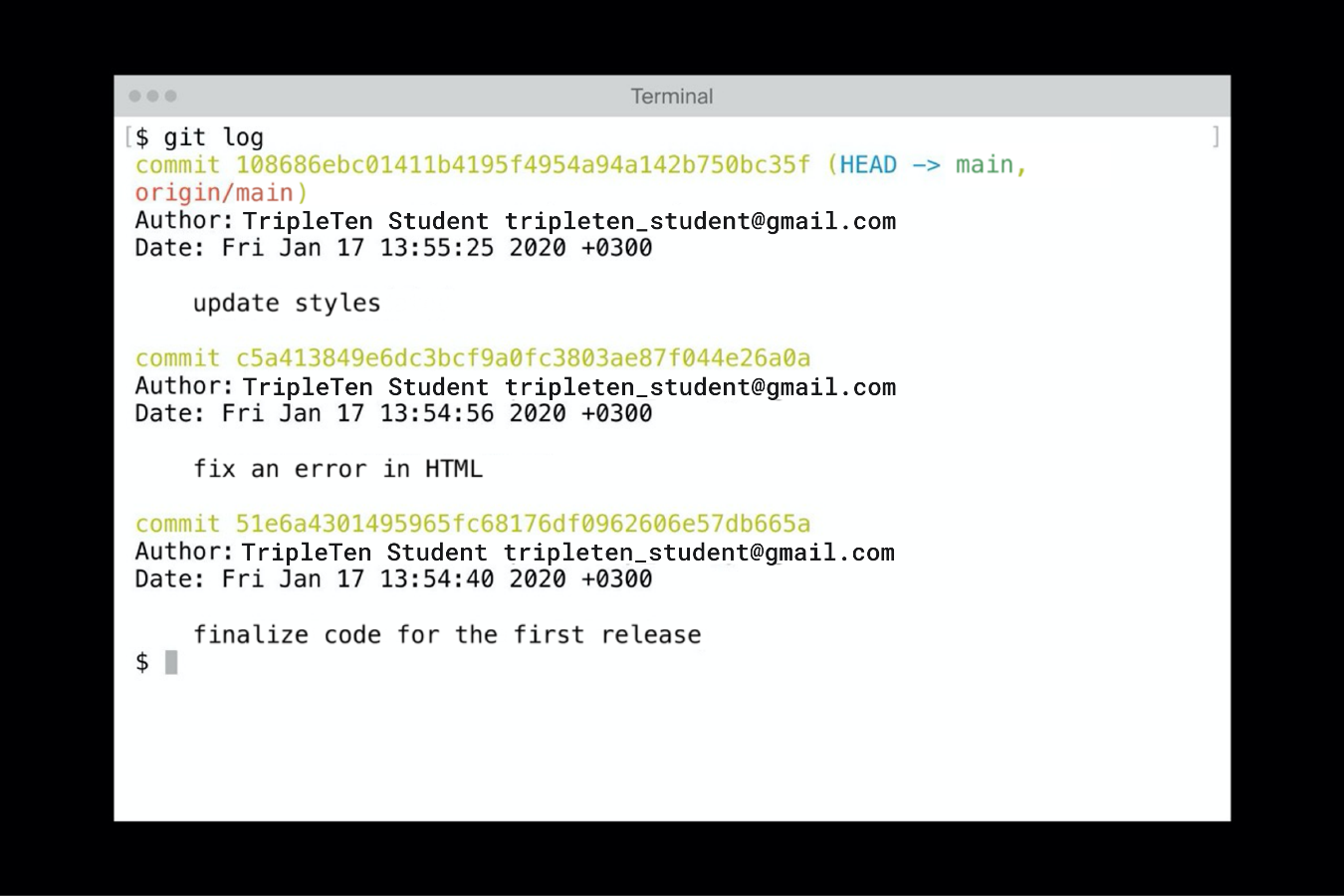
**Conectar Git al proyecto**

* git init inicializa un directorio local como un repositorio remoto.
* git add agrega archivos al "área de staging".
  + git add file1 file2 agrega los archivos de la lista.
  + git add -A agrega todos los archivos.
* git status brinda información sobre los archivos que han sido cambiados y los que están en el área de staging.

**Hacer tu primer commit**

Tras agregar archivos al área de staging con git add, "confirmamos" esos cambios con git commit, el cual hace un registro de los cambios que has realizado.

* git commit -m "Descripción del commit" hace un commit con el mensaje proporcionado.
  + Elige un mensaje descriptivo.
* git log muestra una lista de todos tus commits. Aprenderás más sobre esto en un sprint posterior.



**Vincular repositorios locales y remotos**

Esta lección incluye:

* Crear un repositorio remoto en GitHub.
* Vincular el repositorio remoto a uno local preexistente ejecutando.

git remote add origin https://github.com/username/repo-name.git

* Empujar los cambios locales a GitHub con git push -u origin main

**Sincronizar repositorios locales y remotos**

Cuando trabajes en un repositorio de Git con otras personas, ejecuta siempre git pull antes de publicar tus cambios con git push.

**Clonar un repositorio**

git clone https://github.com/username/repo-name.git hace una copia del repositorio de GitHub en tu equipo local.

**Ficha de ayuda para Git**

* Comandos básicos de git:
  + git init: convierte un proyecto ya existente en un repositorio git (podría ser simplemente una carpeta vacía).
    - No necesitas hacer esto si estás empezando tu proyecto con git clone
  + git clone remote-repo-url: clona el repositorio en remote-repo-url, dándote una copia local.
  + git status: brinda información sobre el estado de tu repositorio git. Concretamente, te indica todos los archivos que no están en seguimiento, si los hay. Es decir, te dice qué cambios se han hecho desde tu último commit.
  + git log: muestra una lista de todos tus commits en la rama actual.
  + git add -all, git add .: pone todos los archivos en el área de staging (los prepara para su commit).
  + git add file.ext: sólo envía file.ext al área de staging.
  + git commit -m "blah blah": confirma (hace commits de) todos los archivos del área de staging con el mensaje "bla bla". Por favor, utiliza mejores mensajes de commit.

**Posicionamiento de elementos**

**Introducción y terminología**

La forma en que se disponen los elementos en una página web se denomina flujo. De forma predeterminada, todos los elementos de una página forman parte del flujo. Este flujo por defecto (llamado 'normal flow') se describe con detalle en [MDN](https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Normal_Flow).

La forma en que se disponen los elementos dentro de un diseño se denomina posicionamiento. El capítulo se centra en la propiedad CSS position, que indica al navegador dónde colocar los elementos en el flujo.

**Flujo de documentos y posicionamiento estático**

* El flujo del documento se refiere a cómo el navegador coloca los elementos HTML en la página:
  + Cada elemento block se coloca en una nueva línea, y
  + los elementos inline se agrupan en la misma línea
* La propiedad position controla el posicionamiento de los elementos.
  + Por defecto la posición establecida es position: static, que corresponde al flujo normal del documento.

**Posicionamiento relativo**

position: relative te permite ajustar la posición de un elemento en relación al lugar en el que se colocaría normalmente en el flujo.

* Deja el elemento en el flujo de documentos, pero.
* Permite ajustar la posición del elemento mediante las propiedades top, right, bottom y left.
* La posición se ajusta con respecto a la posición normal del elemento.

**Posicionamiento fijo**

position: fixed posiciona un elemento en relación a la ventana gráfica, para que siempre aparezca en el mismo punto de la pantalla.

* Elimina el elemento del flujo de documentos.
* Permite ajustar la posición del elemento mediante las propiedades top, right, bottom y left.
* La posición se ajusta en relación a la ventana gráfica.

**Posicionamiento absoluto**

* Saca elementos del flujo normal del documento.
* Permite ajustar la posición del elemento mediante las propiedades top, right, bottom y left.
* La posición se ajusta respecto al primer elemento padre que no tenga una posición predeterminada.
  + Si ningún elemento padre tiene una posición predeterminada, se colocará en relación con el cuerpo de la página web.

**z-index**

1. Posiciona los elementos en la tercera dimensión, apareciendo los valores mayores en la parte superior.
2. Sólo funciona en estos dos casos:
   * elementos con propiedades de posición explícitamente definidas y no predeterminadas
   * elementos dentro de un contenedor flex que al mismo tiempo posean un posicionamiento que no sea estático.

**Recursos**

* [Guía de trucos CSS para la posición](https://css-tricks.com/almanac/properties/p/position/) *(materiales en inglés)*
* [Un manual sobre la propiedad position](https://developer.mozilla.org/es/docs/Web/CSS/position)
* [Manual oficial de la propiedad ‘position’](https://www.w3.org/TR/css-position/) *(materiales en inglés)*
* [Un ejemplo del uso de la propiedad z-index](https://medium.com/hackernoon/my-approach-to-using-z-index-eca67feb079c) *(materiales en inglés)*