

Estructura HTML

El DOCTYPE

Cuando escribimos nuestro documento HTML, lo primero que tenemos que escribir es el **DOCTYPE**. El doctype declara el tipo de documento, es decir, nos sirve para indicar que nuestro documento está escrito siguiendo la estructura determinada por un DTD concreto.

¿Qué es el DTD que utilizamos en el Doctype?

Un **DTD** es la definición del tipo de documento. El doctype que es la forma de declarar el tipo de documento.

Así que, el DTD es dónde se define la estructura que debe tener el documento y utilizamos el doctype para informar qué DTD usamos.

El doctype en HTML5

A partir de HTML5 la declaración del doctype es notablemente más sencilla, sólo basta con encabezar nuestro documento con la siguiente etiqueta.

```
<!DOCTYPE html>
```

Estructura básica de un HTML

Usaremos 3 tags para definir la estructura principal de un HTML, ellos son:

`<html>`: Esta etiqueta delimita el contenido del documento e indica en que lenguaje está escrito.

`<head>`: Es la parte privada del documento, se utiliza como un espacio de comunicación entre el sitio web y el navegador para configuración de la visualización del sitio.

`<body>`: Encierra el contenido propiamente dicho del sitio.

Una vez definida la estructura principal, veremos otras etiquetas básicas de HTML.

`<title>`: La etiqueta title define el título de nuestra página, que será visualizado en la solapa del navegador.

`<meta>`: Con esta etiqueta definiremos que codificación de caracteres utilizará nuestro sitio.

Utilizaremos todas estas etiquetas en nuestro primero ejemplo de estructura web:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Mi primer sitio web</title>
  </head>
  <body>
    ¡Este es mi primer sitio web!
  </body>
</html>
```

Sintaxis HTML

El elemento principal del html es el de etiqueta, éstas sirven para especificar el tipo de contenido al navegador.

Las etiquetas se dividen en cerradas y abiertas.

Las cerradas encierran un contenido, por lo general texto. Las abiertas no encierran contenido, y sirven, entre otras cosas, para incluir elementos como imágenes, líneas, etc.

```
<p>Este es un párrafo con texto en su interior</p>  
<hr/>
```

En el ejemplo tenemos una etiqueta cerrada párrafo que engloba un texto y una etiqueta abierta para incluir una línea horizontal.

El signo / se utiliza para las etiquetas de cierre; en las etiquetas cerradas se pone a continuación del signo <, en las abiertas se pone delante del signo >.

Etiquetas Semánticas

HTML5 incorpora etiquetas semánticas que no sólo generan estructura, si no que también define su contenido.

Etiquetas

- `<section>`: Define una sección en un documento.
- `<nav>`: Define un bloque que contiene enlaces de navegación, como por ejemplo el menú.
- `<article>`: Define contenido autónomo que podría existir independientemente del resto del contenido.
- `<aside>`: Define contenidos vagamente relacionados con el resto del contenido de la página.
- `<main>`: Define el contenido principal o importante en el documento. Solamente existe un elemento `<main>` en el documento.
- `<header>`: Define la cabecera de una página o sección.
- `<footer>`: Define el pie de una página o sección.

¿Qué es HTML?

EL HTML es un lenguaje de programación para la elaboración de páginas web, basado en el uso de etiquetas encerradas por corchetes angulares (< >) el cual se utiliza para traducir la estructura y la información en forma de texto, con la posibilidad de agregar objetos como imágenes, videos etc. En la actualidad el HTML es complementado por otras tecnologías como las Hojas de Estilo en Cascada (CSS), el JavaScript entre otras.

HTML y CSS

Originalmente, las páginas HTML sólo incluían información sobre sus contenidos de texto e imágenes. Con el desarrollo del estándar HTML, las páginas empezaron a incluir también información sobre el aspecto de sus contenidos: tipos de letra, colores y márgenes.

La posterior aparición de tecnologías como JavaScript, provocaron que las páginas HTML también incluyeran el código de las aplicaciones (llamadas scripts) que se utilizan para crear páginas web dinámicas.

Incluir en una misma página HTML los contenidos, el diseño y la programación complica en exceso su mantenimiento. Normalmente, los contenidos y el diseño de la página web son responsabilidad de diferentes personas, por lo que es conveniente separarlos.

CSS es el mecanismo que permite separar los contenidos definidos mediante XHTML y el aspecto que deben presentar esos contenidos.

Una ventaja añadida de la separación de los contenidos y su presentación es que los documentos XHTML creados son más flexibles, ya que se adaptan mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, impresoras y dispositivos utilizados por personas discapacitadas.

Crear una cuenta

1. Descargar e instalar la última versión de [GitHub Desktop](#). Al finalizar la instalación el sistema lo mantendrá actualizado a la última versión.
2. Desde la computadora, abrir la aplicación **Git Shell**
3. En primera instancia se debe ingresar el nombre del usuario en Git para que al momento de hacer commit, todo quede correctamente etiquetado. Se deberá escribir todo después del símbolo `$`.

```
$ git config --global user.name "NOMBRE"
```


4. Se deberá ingresar una dirección de email que quedará asociada a tus commits de Git.

```
$ git config --global user.email "EMAIL"
```

Crear un nuevo repositorio en GitHub


En GitHub, se pueden guardar toda clase de proyectos en repositorios. Los repositorios personales pertenecen a las cuentas de los usuarios, así que luego de haber creado una cuenta y entrar al sistema, ya podemos crear nuestro primer repositorio. Los pasos son los siguientes:

1. Situándonos en el menú principal, en la esquina superior derecha, podemos ver un ícono de + donde se despliegan diferentes opciones. La primera es "Nuevo repositorio".
 2. Debemos elegir un nombre para el repositorio. Se recomiendan nombres cortos y fáciles de recordar.
 3. Podemos agregar, de manera opcional, una breve descripción del repositorio. Por ejemplo: "Este es mi primer repositorio en GitHub".
 4. Luego debemos elegir si el repositorio será Público o Privado.
 5. Público: Estará visible para cualquier usuario de GitHub.
 6. Privado: Se encontrará disponible solamente para nosotros, los dueños del repositorio, y para aquellas personas con las que decidamos compartirlo.
 7. Seleccionamos la opción Iniciar el reposito con un archivo Léeme.
 8. Crear el repositorio.
- ¡Ya está! Con estos pocos pasos ya creamos el repositorio. En la siguiente imagen se verán todos los puntos mencionados anteriormente.



[Pull requests](#) [Issues](#) [Gist](#)

+



New repository


New organization

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner


Repository name




/

Great repository names are short and memorable. Need inspiration? How about **bookish-bassoon**.

Description (optional)


☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**


Add a license: **None**



Create repository

© 2016 GitHub, Inc.

[Terms](#) [Privacy](#) [Security](#) [Contact](#) [Help](#)



[Status](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#) [Pricing](#)

Clonar repositorios localmente

Cuando creamos un repositorio en GitHub, éste existe como un repositorio remoto. Existe la posibilidad de crear una copia o clon del repositorio de manera local y sincronizar el mismo entre ambos lugares.

1. Ingresamos a GitHub y nos situamos en el menú principal.
2. Debajo del nombre de un repositorio existente, clickeamos el ícono para clonar la URL del repositorio.
3. Abrimos Terminal (para usuarios de Mac o Linux), o el Símbolo de sistema (para usuarios de Windows).
4. Ubicamos el directorio donde queremos que se genere el repositorio clonado.
5. Típear `git clone / en código /`, y luego copiar la URL que tomamos en el Paso 2.

```
$ git clone https://github.com/USUARIO/REPOSITORIO
```

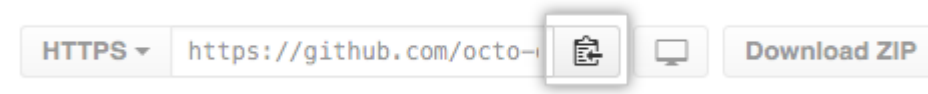
6. Presionar Enter, y el repositorio local se creará.

```
$ git clone https://github.com/USUARIO/REPOSITORIO
Cloning into `Location`...
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 10 (delta 1), reused 10 (delta 1)
Unpacking objects: 100% (10/10), done.
```

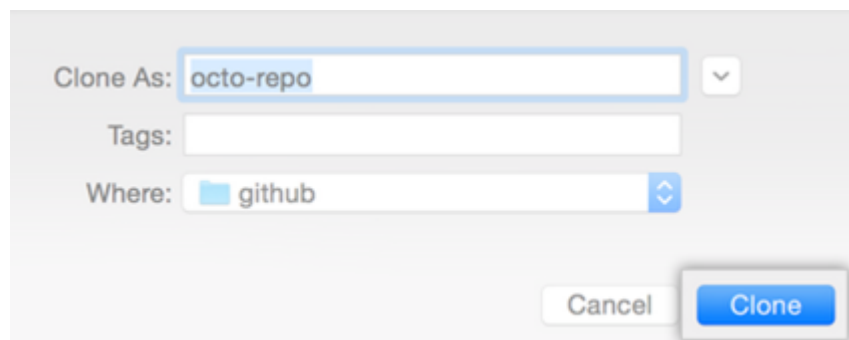
Clonar usando la aplicación de escritorio

Para esto necesitamos instalar la [aplicación de escritorio](#)

1. Ingresamos a GitHub y nos situamos en el menú principal.
2. Debajo del nombre de un repositorio existente, clickeamos el ícono para clonar la URL del repositorio.



1. Dentro de la aplicación de escritorio, luego de verificar el nombre y el directorio en la unidad de disco donde queremos que se encuentre el repositorio clonado, clickeamos el botón Clonar.



¿Qué es Git?

Git es un sistema de control de versiones, gratuito y de código abierto, diseñado para manejar desde pequeños a grandes proyectos de manera rápida y eficaz. Se entiende como control de versiones a todas las herramientas que nos permiten hacer modificaciones en nuestro código y hacen que sea más fácil la administración de las distintas versiones de cada producto desarrollado.

¿Qué es GitHub?

Github es un servicio para alojamiento de repositorios de software gestionados por el sistema de control de versiones Git. Github es un sitio web pensado para hacer posible el compartir el código de una manera más fácil y al mismo tiempo darle popularidad a la herramienta de control de versiones en sí, que es Git. Cabe destacar que Github es un proyecto comercial, a diferencia de la herramienta Git que es un proyecto de código abierto.

Conectar un repositorio local con tu repositorio en GitHub

El beneficio de tener un repositorio local es una manera efectiva de poder hacer modificaciones sin siquiera tener que estar conectado a internet. Luego, se puede subir el trabajo terminado a GitHub para que esté disponible para todos los usuarios.

Asumiendo que, por ejemplo, tenemos en GitHub un repositorio ubicado en <https://github.com/usuario/ejemplo.git>, la construcción sería:

```
git remote add origin https://github.com/username/myproject.git
```

- Modificamos "usuario" con tu usuario real de GitHub.
- Modificamos "ejemplo" con el nombre real del repositorio en GitHub.
- Para informar a Git que el repositorio remoto realmente existe en algún lugar de internet utilizamos `git add`.
- El código `origin` indica que esos archivos van a tener un nuevo destino.
- `remote` describe al código `origin` ya que va a estar subido de manera remota y no se va a tratar de un directorio local.

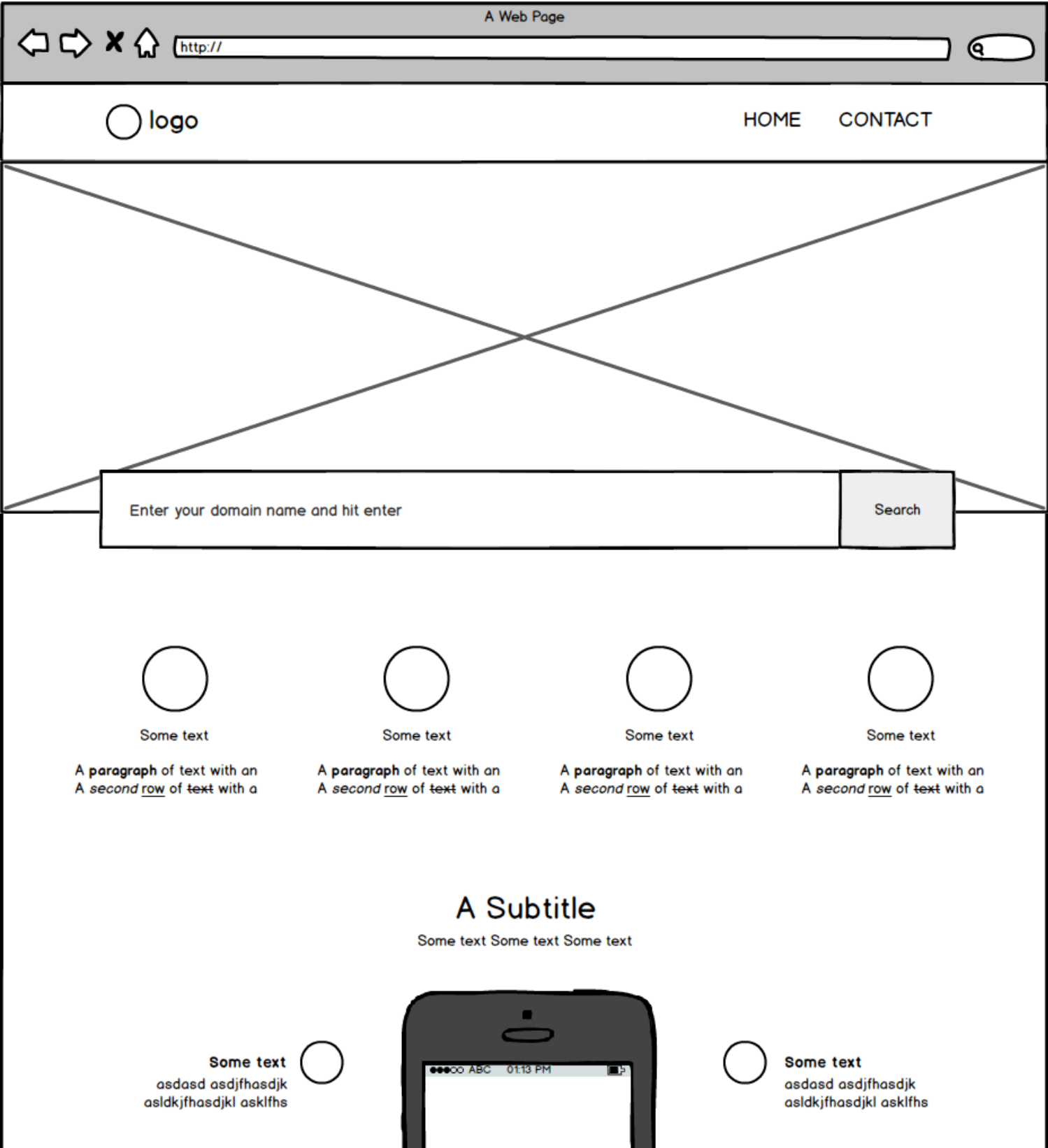
Para verificar, usamos: `git remote -v`.

Para subir ("push") nuestros cambios usamos `git push`.

Trabajo Practico 1 Git

Descripción:

- 1. Realizar el curso de git <https://try.github.io>.
- 2. Registrarte en github, y crea tu primer repositorio, en el mismo vamos a ir guardando los trabajos prácticos que realicemos.
- 3. Maquetar el mockup provisto, respetando el orden y el contenido de las secciones, solo utilizando las etiquetas semánticas de html5(section, article, footer, etc).



Some text
asdads asdjfhasdj
asldkjhasdjkl asklfhs



Some text
asdads asdjfhasdj
asldkjhasdjkl asklfhs



Some text
asdads asdjfhasdj
asldkjhasdjkl asklfhs



Some text
asdads asdjfhasdj
asldkjhasdjkl asklfhs



A Subtitle

Some text Some text Some text

Some text

\$95

Some text 1
Some text 1
Some text 1
Some text 1

Button

Some text

\$95

Some text 1
Some text 1
Some text 1
Some text 1

Button

Some text

\$95

Some text 1
Some text 1
Some text 1
Some text 1

Button

Some text

\$95

Some text 1
Some text 1
Some text 1
Some text 1

Button

Hosting use 500,000 happy customers. Want to see services?

Some text

Home Contact



Objetivo:

1. Familiarizarte con git.
2. Empezar a utilizar github como repositorio para tus proyectos.
3. Aprender a utilizar las etiquetas html5(header, footer, etc), donde y cuando utilizarlas.

Requerimientos:

1. Tener instalado git en tu computadora.
2. Tener instalado photoshop o algun software que pueda abrir archivos psd.
3. Crea en tu cuenta de github una carpeta con el nombre “trabajos prácticos” y subí la resolución de tu trabajo dentro de la subcarpeta “HTML/GITHUB”.

Bonus:

Investigar sobre el uso y aplicación del tag `<video>`.

Recursos:

1. [Tutorial Git](#)
2. [Mejores Practicas HTML](#)
3. [Primer pagina web de la historia](#)