**CoderHouse**

Full Stack

Profesores: Alejandro, Mauro y Félix.

Trabajos prácticos enviar a, Email: [fullstack**@**coderhouse.com](mailto:fullstack@coderhouse.com)

Slack

Usuario: SantiagoF

Contraseña: codersanfri15

Grupo: coder-sab-1

Clase 1

Single page aplication (ver concepto): Tiene funcionalidad y su contenido se expresa en una sola página.

Single page web: sólo muestra contenido la página.

HTML es un estándar vivo, ya que se siguen haciendo arreglos.

HTML está dividido en:

1. Cabecera o head: cierta info. Para que funcione la página.
2. Contenido o body

Etiquetas:

1. Div
2. Span
3. Section
4. Article
5. Footer

Las etiquetas pueden ser:

1. In line: no tienen salto de línea. Van pegadas, como por ej. <img> o <link>.
2. Block: ocupa todo el ancho de la página. Como por ej. <div>.

La etiqueta in line de div es span.

Otras etiquetas

1. ul: unorder list
2. ol: order list, muestra lista ordenada numéricamente.
3. Aside: es la parte secundaria de un contenedor.

Los tags que están dentro del body, son hijo del body.

Elementos que son del mismo nivel, se llaman hermanos.

El input cumple, dentro de sus funciones, la de ser un botón.

CSS

Llamar a id: #miDiv{

}

Llamar a class: .footer-izquierda{

}

Hijo directo de main

.main > a{

}

Clase 2

¿Qué en GIT y Git Hub?

C:\Program Files (x86)\Git

Git y Git Hub forman parte de lo que se denomina, “sistema de control de versiones”.

Un buen desarrollo de un producto, requiere que se pueda estar seguro antes de guardar y se puedan recuperar versiones que se van sucediendo.

**Versión**: es el estado en que se encuentra el desarrollo de cierto producto en un momento determinado.

**Control de versiones**: es la gestión de cambios de versiones.

**Programas de control de versiones:** son programas que nos facilitan esa gestión de control de versiones.

Existen varios sistemas de gestión de versiones, divididos en 2 grupos

1. Centralizado: CVS o Subversion
2. Distribuido: GIT o Mercurial

Que permiten:

* Administrar versiones
* Almacenar información
* Conservar registros

Sistema de control de versiones centralizado

**Git:** permite guardar en la nube y administrar cambios de líneas de código guardando en puntos de control (versiones). También permite convertir tu máquina en un servidor.

**GitHub:** repositorio remoto, trabajo de programación en conjunto, red social de Git.

**Terminal:** ejecutar programas y consultar cosas en el sistema operativo a través de comandos (Windows: cmd).

Trabajar con Git Bash (terminal). Todos los comandos de Git comienzan con git.

**Cd:** change directory, cambiar directorio/ruta de carpeta (el comando solo me lleva a mi carpeta de usuario).

**Tecla Tab:** trae directorio sugerido.

**Pwd:** print working directory, ver donde estoy parado en el directorio.

**Q:** salir.

**Ls:** lista contenido donde estoy parado.

**Ls –l:** listar uno por línea.

**Ls –h:** listar más amigable al usuario (KB, MB, …).

**Ls –a:** .carpeta actual, .. volver a una carpeta superior.

**Comando –help:** ayuda del comando.

**Mkdir:** crear carpeta de directorio.

**Clear:** limpiar pantalla terminal.

**Alias:** crear comando (alias nombre = ‘comando’).

**Rm:** eliminar archivo.

**Rm – r:** eliminar directorio (MUCHO CUIDADO).

**Rm – i:** eliminar interactivo, solicita confirmación antes de eliminar (y/n).

**Git help:** ayuda de git.

**Git help comando:** ayuda de un comando específico.

**[…]:** opcional, si no se indica hace la acción default.

**. :** todos los archivos de la ubicación donde estás parado.

**Git init:** crear un repositorio local git.

**Git add:** añadir archivo a Git para que sea controlado por Git.

**Git commit:** crear punto de restauración.

**Git commit –m:** incluir un mensaje al commit para seguimiento.

**Git commit –a:** commit todo lo que haya disponible para hacer commit.

**Git status:** verificar status de cambio de los archivos de mi repositorio.

**Git config:** configuración del repositorio.

**Git config user.name “nombre”:** definir nombre de usuario.

**Git config user.email “mail@mail.com”:** definir mail de usuario.

Git necesita que cada commit tenga un usuario autor (nombre y mail).

Archivos que empiezan con punto (.) son ocultos

**GitHub Fork:** copiar un repositorio de otra persona para modificarlo.

**Pull request:** proponerle al dueño original del repositorio tomar las modificaciones hechas.

**Git clone:** copiar un repositorio remoto en tu repositorio local.

**Branch:** distintas ramas del código principal.

**Master:** branch principal.

**Git branch –v:** listar todos los branch del repositorio.

**Git checkout nombre-branch:** cambiar posicionamiento a cierto branch específico.

**Git checkout -b nombre-branch:** crear nuevo branch.

**Git fetch:** bajar de repositorio remoto a repositorio local.

**Git merge:** bajar de repositorio local a directorio local.

**Git pull:** git fetch + git pull.

**Git reset --soft HEAD~#:** volver atrás cambios de un # de commits determinado. Luego hay que hacer commit para guardar los cambios.

**Package Control:** bajar para administrar plugins de Sublime Text, copiar código en consola de Sublime.

Workin directory: es mi directorio de trabajo. Es mi carpeta.

Git es todo local.

Inicializar git en working directory.

Le decimos a git, que archivo queremos que controle.

Git, tiene una interfaz gráfica y una terminal.

La terminal, es una interfaz, en la cual se tira un comando y mi sistema operativo evalúa eso y da una respuesta.

Git se va a ejecutar en la terminal.

Para Windows, la terminal es cmd.

Un comando es por ej pwd.

$ git help (tire un comando) y me responde.

Que es un comando git?

Escribo git y un comando, que esta en la lista



[directory] Es opcional

Ls Lista las carpetas que tengo

ls – lh

cd. Estoy en mi directorio actual. Quiero acceder a x carpeta

cd.. Vuelvo una carpeta para atrás.

mkdir Creo una carpeta

rm rm + nombre de archivo (borro archivo)

rm –r Borro un directorio

Git add Agrego archivo

Status:

¿Cómo inicializado git?

1. git init
2. ls –lh: tengo un registro de los cambios que pasan en ese directorio.

En Sublime: CTRL + SHIFT + P

GitHub

Usuario: santiagofrias

Contraseña: codersanfri15

Repositorio:

Forkear repositorio: copiar xx repositorio de otra persona a su usuario.

GitHub nos da un repositorio remoto (que se crea)

Git pull Entra al repositorio

Git Push Sale del repositorio

Git clone Bajo el repositorio a mi disco, lo inicializa y lo vincula a GitHub.

Branch

git log Veo historial de mis commit

CSS

Display inline: no puedo setear ancho ni alto

Display inline-block: puedo setear ancho y alto

Max-width

Style 🡪 atributo de las etiquetas para poner estilo a las mismas en el HTML definiendo ciertas propiedades (color de fondo, color de letra, tipo de letra, tamaño de letra, etc.). Se conoce como definir estilo in line.

<style type=”text/css”> … </style> 🡪 colocar estilos dentro del archivo HTML para no tener que crear una hoja de estilos (archivo CSS) aparte.

<link rel="stylesheet" type="text/css" href="./main.css"> 🡪 da estilo a un HTML relacionándolo con un archivo CSS separado.

Component CSS - Suite CSS 🡪 metodológica para aplicar estilos sistemáticamente a un HTML. Se utiliza una convención para clasificar etiquetas y que facilita su identificación y hace que la aplicación de estilos tenga una mejor performance. Se trabaja con paquetes de CSS que funcionan como módulos que pueden escalarse.

Clase componente 🡪 encapsula las declaraciones requeridas para dar estilo a un elemento en el contexto de un componente.

* *NombreClase* (elemento)
* *NombreClase-Componente* (componente del elemento)

Clase utilitaria 🡪 aplica una regla de estilo a un patrón sencillo y universal.

* u-*NombreClase* (regla genérica de estilo, ej: u-TextoRojo)

Usar class en vez de id 🡪 performance más rápida para los navegadores.

Inline no permite controlar ancho ni alto del elemento.

Definir al inicio de todo un font-size específico y controlar los tamaños de letra con la unidad de medida em como porcentajes de dicho font-size inicial.

Si no se define ancho y tiene propiedad display = block o inline-block, no se puede centrar un elemento horizontalmente.

Clase 3

CSS

Margin 0 auto: margit top 0, margin botton 0, margin right: auto, margin left: auto

Centrado vertical:

Padding: margen interno

Inspector de elemento

Network: pestaña de archivos bajados y cuanto pesan.

User agent stylesheet: es lo que el navegador pasa por default.

Box-sizing: ver propiedad

Position: posicionar un elemento. Tiene valores posibles. Posicion default: static.

Relative, es relativo al posicionamiento default.

Absolute, busca el primer parent con position distinto a static.

Float: saca del flujo normal a los elementos. Pueden flotar a la izquierda o a la derecha dentro del padre.

Clear: Corta el flotado

Clearfix: sirve para cortar el float

No usar márgenes negativos.

Los elementos inline no tienen height.

Centrado vertical: se utiliza flex box.