Tabla de contenido

Unidad 1 Conceptos teóricos básicos

18 de Febrero 2022

Que es HTML

(Hyper Text Markup Language-Lenguaje de marcas de hipertexto)

Con esto le decimos a nuestras maquinas donde va ubicada cada cosa que vemos en la pantalla, se define como lenguaje de marcado que define la estructura de todo nuestro contenido. HTML consiste en una serie de elementos que usarás para encerrar diferentes partes del contenido para que se vean o se comporten de una determinada manera.

Html solo sirve para indicar como va ordenado el contenido de una página web, por medio de las marcas de hipertexto o los también llamados tags o etiquetas, es por medio de estas que logra ordenar dicho contenido, los tags o etiquetas ayudan a los buscadores a encontrar la página más fácilmente, además permite scripts (archivo con órdenes) para llevar a los navegadores que procesan el lenguaje, los más conocidos son javaScripts y PHP.

Para darle una mirada de mayor profundidad es conveniente mencionar que los tags o etiquetas son partes de lo que conocemos como elementos de html, ya que las partes principales de un elemento son:

Etiqueta o tag de apertura
Etiqueta o tag de cierre
Contenido→ que en el mayor de los casos es solo texto
Elemento→ que seria cada uno de las partes mencionadas anteriormente, tag de apertura + contenido + tag de cierre = elemento.

Ejemplo:

```
<p>hola mundo</p>
```

: tag de apertura
“hola mundo”: texto
: tag de cierre
p: nombre del elemento
 “hola mundo” : elemento

Los elementos pueden también tener atributos, dichos atributos contienen información adicional acerca del elemento que el usuario no va ver, este atributo va acompañado de un valor, es posible en un mismo elemento tener varios atributos con su respectivo valor, estos valores deben ir en comillas, además de tener un espacio entre atributos y el nombre de la etiqueta, los atributos siempre se incluyen en la etiqueta de apertura del elemento.

Ejemplo:

```
<p class="null" aria-colcount="5">hola mundo</p>
```

Anidar un elemento

Se le conoce como anidamiento a un elemento html que se encuentra dentro de otro.

Que es un elemento vacío

El elemento vacío es aquel que no contenga texto, por lo tanto no requiere etiqueta de cierre.

Estructura básica de un documento HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Doctype: indica la versión html5.
Html: encierra todo el contenido de la página entera.
Head: aquí ponemos todo lo que el navegador puede ver sin que el usuario lo pueda notar.
Meta: elemento vacío, el valor utf-8 especifica los caracteres del lenguaje humano.
Title: título de la página que aparece en la barra de navegación.
Body: este elemento encierra todo lo que quiero mostrarle al usuario.

Encabezados (titulos)

Se utilizan para especificar que ciertas partes del contenido son encabezados o sub encabezados del contenido, HTML posee 6 encabezados

```
<h1>Mi título principal</h1>
<h2>Mi título secundario</h2>
<h3>subtitulo</h3>
<h4>sub_subtitulo</h4>
<h5>sub_subtitulo</h5>
<h6>sub_subtitulo</h6>
```

Es importante resaltar que los navegadores de hoy en día están diseñados para tener una mayor semántica, en ese orden de ideas, si utilizas un h1 más de una vez en la misma página los algoritmos de los navegadores penalizan esas malas prácticas, haciendo que les sea más difícil a los usuarios encontrar tu sitio web respecto a otros.

Listas

Desordenada:

```
<p>mi carrera requiere lo siguiente</p>
<ul>
    <li>disciplina</li>
    <li>inteligencia</li>
    <li>entrega</li>
</ul>
```

Ordenada:

```
<p>mi carrera requiere lo siguiente</p>
<ol>
    <li>disciplina</li>
    <li>inteligencia</li>
    <li>entrega</li>
</ol>
```

Vínculos

Son muy importantes, aquí algunos ejemplos

```

<a>Estoy estudiando</a>
<a href="">sigo estudiando</a>
<a href="www.estudio.com">sigo estudiando</a>

```

Generalidades

Donde escribir código

Cualquier editor de texto plano, es decir, sin un formato definido como bloc de notas, también se hace en Visual Studio Code, atom, sublime text, editores web online, entre otros.

Prepara tu maquina

Lo primero que debemos hacer es elegir nuestras herramientas de trabajo, para este caso vamos a elegir Visual Studio Code, pero existen muchos otros editores de texto que podríamos elegir.

1. Descargar Visual Studio Code e instalar
2. Crea una cuenta en Github el cual es un lugar para subir archivos(repositorios)
3. Instala Git el cual es un sistema de control de versiones
4. Instala Node.Js el cual es un entorno de ejecución para JavaScrip del lado del servidor
5. Instala npm el cual es el sistema de gestión de paquetes de JavaScrip

En la instalación de Node.Js se recomienda que su sistema operativo sea superior a Windows 8 ya que Node.Js no corre bajo esta versión.

Terminal cmd, Power Shell y Git Bash

En resumen las tres son CLI (interfaces de líneas de comando), es una interfaz de usuario para comunicarse con la maquina por medio de comandos, esto permite un manejo más profundo y específico de nuestras maquinas, además de mayor rapidez, pero requiere conocer los comandos y práctica, el cmd es la versión CLI simple de Windows, aquí no todos los comandos funcionan o están disponibles, por esta razón desarrollaron la Power Shell, para brindar una mejor experiencia, posiblemente impulsado por el desarrollo del día de hoy, es decir que la Power Shell es ideal para el desarrollador actual, por otro lado contamos con Git Bash, que es un emulador el cual se integra muy bien como CLI para Windows a manera de Power Shell, este último tenemos la opción de instalarlo una vez estemos instalando Git en nuestras máquinas, por ahora ocuparemos este último CLI.

Github

Es una red para almacenar archivos, a estos archivos se les conoce como repositorios. A continuación nombraremos algunas cosas básicas que debemos ir conociendo.

```
Pipe/Mi_repo
Pipe es el nombre de usuario
Mi_repo seria el nombre de repositorio
```

```
Code
Este nos brinda la Url del repositorio para poder clonarlo

Watch
Nos permite seguir un repositorio

Star
Podemos marcar como favorito un repositorio que nos guste

Fork
Nos ayuda a crear una copia del repositorio en remoto en nuestra cuenta de
GitHub, para poder luego clonar más fácil de forma local, en otras palabras
los que hace fork es copiar archivos
```

Git

Git es un sistema de control de versiones, conocido como sistema distribuido, cada persona tiene una copia exacta donde trabajan localmente en algún archivo, al finalizar sincronizan esos archivos para obtener un archivo final, ya Git internamente posee unos protocolos que aplica para elegir la mejor versión de dicho archivo, al trabajar con muchos archivos se vuelve más complejo gestionar tanto programa, por eso nace Git.
Lo primero que debemos hacer es configurar a Git para que trabaje de manera adecuada, esto se logra configurando algunos parámetros de manera global, lo que buscamos es que nuestra configuración se realice no solo a un nuevo proyecto sino a todos, a continuación lo describiremos paso a paso:

Configuración global de Git desde la terminal

1. *Git --versión*: esto indica la versión que tenemos instalada
2. *Git config --global user.name "xxxx xxxx xxxx"*: nombre de usuario
3. *Git config --global user.email amil@xxxx.com*: correo configurado

----------configuración inicial----------

1. *Git config --global core.editor "code --wait"*: le dice a Git que Vsc es el editor y la opción de "--wait" es para que la terminal se quede esperando hasta que nosotros cerremos nuestro editor de texto
2. *Git config --global –e:* este permite ver nuestro archivo de configuración global en VSC
3. *Git config --global core.autocrlf (true ó input)*: true llenado de caracter windows=tru Linux o Mac=input
4. *Git config -h*: podemos visualizar las ayudas de configuración global

----------configuración editor de texto----------

Estados de Git

Untracked: sin seguimiento, es como verlo por primera vez en mi pc, pensaría que es como un archivo sin guardar.

Unmodified: sin modificar, ya existe en mi pc y no pienso hacerle cambios.

Modified: modificado, un archivo existente que estoy modificando.

Staged: lo que quiero guardar para que otros lo vean, es hacer un **commit,** un commit es como hacer una foto de un punto específico de mis archivos, para que queden guardados y poderlos entregar después, podría decirle que es una especie de checkpoint**.**

Aquí un procedimiento básico de Git que debemos seguir:

```
Creemos el primer repo:
Lo primero es crear una carpeta en nuestro equipo, en la ubicación que queramos, una vez estemos ubicados en el directorio, le lanzamos lo siguiente:

Git init: convierte dicha carpeta en un entorno para nuestro repositorio
Git status: estado actual del repositorio
Git add: alista los archivos para hacerles commit
Git commit –m ‘comentario’: el comentario se sugiere en ingles
Git push origin master: este me sincroniza mi trabajo local con el servidor de Github


Clonar un repositorio:
Nos ubicamos en la página de github en el repositorio que vamos a descargar, le damos en fork, luego debemos verificar que aparezca nuestro nombre de usuario y el nombre del repositorio, luego damos clic al botón code, que sirve para copiar el link de la url, una vez copiado vamos a la terminal y lanzamos el siguiente comando, todo esto dentro de alguna carpeta que tengamos o que queramos crear:
Git clone + pegar la url + enter, esto debe pegar el repo completo
```

Comandos

Comandos de la terminal

La estructura seria: comando [-opcion(s)] [argumento(s)]. No todos los comandos poseen opciones y argumentos.

COMANDO	CARACTERISTICA
Git --version	Especifica que versión tiene instalada el equipo, o si no tiene instalada ninguna versión.
Git init	Inicializar un repositorio utilizando la terminal, nos paramos en una carpeta o creamos un directorio y lanzamos el comando, esto indica que este espacio o directorio será especial solo para ser utilizado para código, con esto Git reconoce dicha ubicación como un espacio especial para ir estructurando todos nuestros archivos relacionados con el desarrollo de algún tipo de código.
Git status	Como se encuentra nuestro repositorio en el momento.
Git add	Pasar un archivo de un estado untracked a un estado commit o staged, en pocas palabras es como si uno guardara un documento en su pc, en una ubicación especifica.
Git add .	Al lanzar este comando + (.) lo que hacemos es agregar todas las carpetas de una sola vez al futuro commit, mientras que si no le ponemos el punto, nos tocará agregar carpeta por carpeta.
Git commit -m ‘comentario’	**DUDAS DE COMO FUNCIONA** **comillas con Alt + 39, escribir el mensaje m en ingles.**
Git log	**DUDAS DE COMO FUNCIONA**
Git push	Sincroniza mi repositorio con el repositorio del servidor en la página de Github.
Node -v	Al lanzar este comando nos devuelve la versión instalada en nuestras máquinas, de esta manera podemos comprobar si node se encuentra instalado o si está funcionando de manera correcta.
Npm --version	

	Indica la versión de npm que tengamos instalado.
Start chrome	Para iniciar desde la terminal Google Chrome solo lanzamos este comando y ya está.
Ctrl + C	Esto detiene el proceso que esté ejecutando la terminal y devuelve el control al desarrollador.
Code	Al lanzar el comando code podemos abrir nuestro Visual Studio Code desde la terminal de PowerShell.
Pwd	Muestra la ubicación actual, donde estoy parado.
Date	Muestra la hora y la fecha.
Ls	Muestras los archivos de la ubicación en la que me encuentro.
Ls –la	Muestra más información, archivos ocultos.
Cd.. o CD ..	Retrocede en la carpeta, en PowerShell no requieres poner los dos puntos con espacio, mientras que en GitBash se necesita ingresar primero espacio y luego los dos puntos.
Cd /nombre	Avanza en las carpetas, se debe especificar el nombre de la carpeta al que se quiere avanzar, los nombres de las carpetas no deben contener espacios, de lo contrario no podremos ingresar por medio de la interfaz de línea de comandos (CLI).
C:	Cambia de directorio.
D:	Cambia de directorio, también probar con cd /d aunque tambien podemos usar cd espacio la unidad y dos puntos, eje: cd d: sobre todo en Git Bash.
Clear	Con este borramos el output de la pantalla.
Tab	

	La techa tab es de gran ayuda a la hora de rellenar lo que uno está escribiendo, por ejemplo si queremos escribir el nombre de una carpeta solo escribimos las letras iniciales y luego presionamos la tecla tab, esto lo que hace es terminar de rellenado del nombre de la carpeta, funciona muy bien en PowerShell.
mkdir	Crea una carpeta.
New-item (ni)	Este comando es de Windows y sirve para crear carpetas y archivos, podemos lanzar new-item o simplemente escribir ni más el nombre del archivo y le podemos poner la extensión, luego debemos confirmar si es un directory para el caso de una carpeta o file para el caso de un archivo.
Touch	Este comando es de Linux, se usa para crear archivos y funciona de la siguiente manera, se lanza el comando touch más espacio y el nombre del archivo, este comando funciona en Git Bash, mas no en Power Shell.
Notepad	Este nos abre un archivo para poder ser editado, escribimos el comando + el nombre del archivo.
rm	Borra una carpeta. En algunos momentos la carpeta o los archivos no eliminan fácilmente, por lo que podríamos forzar ese borrado con el comando rm –rf + el nombre de la carpeta o archivo, en momentos el PowerShell no realiza el borrado, por lo que es necesario cambiarnos a Git Bash.
SYSTEMINFO	Información del sistema.
Head	Muestra las n primeras líneas de un archivo de texto, pero me funciona solo con git Bash.

Clara dice….. prueba local.. no me dio

Segundo: