THE ODIN PROJECT

Notebook

Name:
León Mora Leonardo Rafael

Contents

1 Introduction

If you want to work as a web developer you will need a more rigorous understanding of the web itself than you possibly have right now. Knowing this concepts will open a new path that will help you to identify what you are doing in each line of code and also it will allow you to talk intelligently with other developers about your work.

1.1 How Does the Web Works

- **The internet** is a *network* (connection between computers) that goes around the world and is achieved through cables that traverse mountains and oceans from one computer to another. In this case I am talking about the computers that store all the information we see on websites and applications, the kind of computers that are embedded in buildings to support the heaviest part of the web. In order for the data to reach your device, you must have a connection to the web through a cable, radio waves, cellular networks, etc.

- **The IP Address** also known as the Internet Protocol address, it is a direction assigned to our device creating a path between us and the web, as an address to our home where we can request a package and be sure it will be delivered at our door. Packages are analogous to *packets*, through them you can see a web page correctly constructed. Those packets know where to go because the route created is unique thanks to the IP address.

- **Router** refers to another computer that recognized all the IP address from others connected to it. The router allows our computers to communicate with the world wide web, including those in which a website lives.

- **Packets** are letters that contain components of a web page, they travel different paths to get to the computer requesting the information. When a browser receives the packets, its job is to concatenate them all to form the web page. Basically, when a client (computer connected to the web) request a web page, a copy of it is created on the server to which it belongs, this copy is decomposed and stored in different letters, each one of them goes through different ways to reach the client and reassemble the web page (this process takes place in seconds).

- **The client** refers to a device requesting information to a server through the web.

- **Server** is a central place where information and programs are stored and accessed by applications over the network.

- **Web page** is a document which can only be displayed in a web browser. It lives in a website and contains just a section of the entire site.

- **Web server** refers to a computer hosting one or more websites. "Hosting" means that all web pages and their supporting files are available only on that computer. The web server will send all the necessary web pages from the server that contains them. Each web server has different information from each other, which means that a website cannot be registered on two or more servers.

- **Web browser** is the platform that shows a web page. The browser managed functions like the search engine and also is able to read files like HTMLs or even PDFs. A browser lets users access further pages through hyperlinks.

- **DNS request** or Domain Name System request is a hierarchical and decentralized naming system for Internet connected resources. DNS is like a directory that contains the domain names along with the resources necessary to locate a website, such as the IP addresses which are associated with them.

1.2 What is an ISP?

Internet Service Provider is a company that manages some special routers that are all linked together and can also access to others ISPs routers.

2 Command Line Basics

The command line (cmd) is the basis of all the functions that the computer executes, through the command line the entire computer can be managed. Cmd is the best way to communicate with your system, sending direct and specific orders to it.

Opening the command line could be different depending on the OS or even the computer, mine is a HP laptop using Ubuntu Budgie, to open the cmd I use `ctrl + alt + T`. Once in the command line you can start making your will.

2.1 Shortcuts

- `ctrl + L` → will delete every line that is in the terminal.
- `ctrl + C` → is used in specific kinds of trouble, when you are in a line that starts with ">" symbol you can use it to get out of it.
- `ctrl + D` → closes the terminal.

2.2 Useful Commands

- `pwd` which means Present Working Directory, this command is to find out where you are. `pwd` will list the "logical" path of your current directory.
  you should use `pwd -P` if you want to see the actual physical path. This command shows the full path to your current working directory.

- `ls` is used to list files and directories in your current directory, or another.
  `ls <path>` will let you pass it a path to work on without being in the directory.
  `ls -a` this option "`-a`" is often used to show all the files, even those that aren't visible (files that are prefixed with a dot "."). The option along with the command shows both "visible" and "hidden" files/directories.
  `ls -l` shows a list of files and directories in long form, with more details. The information displayed for each file is: file mode, number of links, owner name, group name, number of bytes in the file, month, day, hour and the path name.
  `ls -lh` if you want to see the size of the file in human readable terms, such as 1.7K or 35M. This information is modified by using the option `-h`.
  `ls -lhs` for sorting the list `-S` (Capital S), will do the job.
  `ls -lt` if you want to sort by last time the files were modified. To do this we can use the `-t` flag. `ls -lr` through the option `-r` you can sort the list in reverse.

- `ln <file_a>.<ext> <file_b>.<ext>` links, allow us to create an association from one file or directory to another. By default the `ln` command will create a hard link between the files but it cannot create a link between directories.
  `ln -f <file_a>.<ext> <file_b>.<ext>` is used to force a link. `ln -s <file_a>.<ext> <file_b>.<ext>` this flag (`-s`) creates a link to a directory. This can also be used for linking to files as well, not just directories. Symbolic links can also link to files or directories on other file systems. This makes symbolic links (symlinks) more powerful, and more common than the default hard links.

- `cd /User/home/etc` is used to change directories.
  `cd ..` go backwards, instead of passing the command a full path to where you want to go. Using ".." flag you can go to the parent directory of the current one. `cd` will navigate to your home directory.

- `mkdir <directory_name>` is used to create a directory.
  `mkdir -p <directory_a>/<directory_b>/<directory_c>` the flag `-p` will allow us to create nested directories.
  `mkdir -v <directory_name>` displays a verbose output. Adding the `-v` flag will print the results of `mkdir` to the console.

- `cp <file_a>.<ext> <file_b>.<ext>` copying files and directories with the `cp` command.
  `cp <file_a>.<ext> <file_b>.<ext> <directory_name>` `cp` will allows you to list several files you would like to copy. When you do this, however, the last argument must be a directory.
  `cp *.<ext> <directory_name>` with this option we can also pass simple patterns to `cp` copying all files with the same extension.
  `cp -v <file_a>.<ext> <file_b>.<ext>` the `cp` command supports verbose output.
  `cp -Rv <directory_a> <directory_b>` the `-R` flag is used to copy directories. This option is recursively copying the directory's contents to the new one.
  `cp -f <file_a>.<ext> <file_b>.<ext>` in this case, the flag `-f` is forcing the file to be copied, even after a "Permission denied" message that results when copying files from different users.
  `cp -i <file_a>.<ext> <file_b>.<ext>` the `-i` flag is used to confirm the overwriting of a file.

- `rm -v <file_a>.<ext>` will remove files and folders. It supports the same flags and arguments as the `cp` command.

- `mv -v <file_a>.<ext> <file_b>.<ext>` supports the same flags as `cp`.

2.3 Git Basics

Git is a Version Control System (VCS) that store its data in a series of snapshots. Git thinks about its data like a stream of snapshots. Each series is independent from one to another so, if you have 3 series and you need to modify or create a new version of the project, that modification will the fourth group of files on that sequence of series.

2.3.1 Git and Text Editors

Text editors rewrite data over the same file you are working with, the workflow is moving forward without keeping track of the old data so, if you close the file and just the you need a younger version of that document for some reason, you would have a hard time recovering that data. These kind of things are different with Git because its way of working with files. Git see every new modification as a new document but it creates a timeline that associate all those versions of the file with a date.

If you need a previous version of a project that has already a lot of modifications, using Git you can go back to the modification you are working with. This does not happen with text editors.

2.3.2 Does Git and GitHub Work at a local or remote level?

Git works at a local level, using the computers memory to store files and work with the project we are dealing with.

Otherwise, GitHub works at a remote level, stores files online and provides a platform where a team of developers can work together, allowing them to delegate specific areas to the right developer and quickly improve the project.

If you mix Git and GitHub the result would be a remote environment that you can modify trough a local level.

2.3.3 Git for Individual Developers

Working be yourself as an individual developer can be better using Git, you will need to be organized with your project and be sure that every change is correctly adapted to it. By using Git you can create a new branch alongside the main one, if you do this the project will have a save point just in case the program crashes due to a new modification. Now you can start messing things up again with no worries.

2.3.4 Git and GitHub for a Team of Developers

When there are to many people working on the same project things get a little complicate, sometimes the changes are made over old versions of the project because they get together a couple of hours per day and it is difficult to cover all the topics, so everybody goes home and start working on the project but maybe they where in the bathroom or watching the new Harley on instagram (or both) when the changes were in course, so they have lost track of the new things that took place on the project.

GitHub provides the latest version of the project every time one of the members needs it, for a team this is very useful because in addition to pulling the repository, they can see what the others are doing and focused on the right area.

When working with the project from a computer is when Git takes action to manage the repository, speeding up the modification process. Once the changes are ready, we can submit it to GitHub and call for a pull request so that the modification can be in the original project.

2.3.5 Git Command-Line Fundamentals

- **On Initial Install:**

  `git --version` → checks the current version of the installed locally Git.

  `git config --global user.name "<your name>"` → sets up the username.

  `git config --global user.email "<your email direction>"` → sets up the user's mail.

  `git config --list` → lists all the Git configurations.

- **Help with Commands:**

  `git help <verb>` or `git <verb> --help` → shows a guide explaining the verb.

- **For Initializing the Project:**

  `git init` → initializes the Git repo in the current directory.

  `touch .gitignore` → creates a Git ignore file.

  `git status` → checks the current working tree (both Git and local).

- **Adding Files:**
  `git add -A` → adds all of the files for committing.

- **Removing Files:**

  `git reset` → removes files to be committed.

  `git reset <some_file.js>` → removes some_file.js from the commit preparation.

- **Committing:**
  `git commit -m "<Insert commit message>"` → makes the commitment and adds a message.

- **Check Log:**
  `git log` → renders commit ids, authors, dates, etc.

- **Clone a Remote Repo:**
  `git clone <url> <direction>` → saves a copy of the repo indicated in the url within the directory at the address.

- **View Info about the Repo:**

  `git remote -v` → lists info about the repo.

  `git branch -a` → lists all of the branches.

- **View Changes:**
  `git diff` → shows the difference between the new file and the old one.

- **Pulling the Repo (always pull before pushing):**
  `git pull <remote repo name> <name of the branch>` → pulls the latest version of the repository.

- **Pushing the Repo (right after pulling it):**
  `git push <remote repo name> <name of the branch>` → pushes the repository.

- **First Time Push of the Branch:**
  `git push -u <remote repo name> <name of the branch>` → "-u" coordinates the two branches (local and on server).

- **Branches:**

  `git branch <name of the branch>` → creates a new branch.

  `git checkout <name of the branch>` → checks out a branch.

2.3.6 Merging and Deleting Branches

Consider both branch and the remote repository called "main" and "origin" respectively and another branch created by us called "nmod".

Merge a branch:

- `git checkout main`
- `git pull origin main`
- `git branch --merged` (see which branches are merged).
- `git merge nmod`
- `git push origin main`

Delete a branch:

- `git branch -d nmod` (this deletes it locally).
- `git branch -a` (check the repo branches).
- `git push origin --delete nmod` (this deletes it from the repo).

3 Introduction to The Front End

Front-end web development refers to the visual part of every single page of the website. To build this pages you will need some languages like HTML for the structure, CSS for the aspect and JavaScript to manage how the web page works.

3.1 Basics of HTML

Hypertext Markup Language is the code used to structure a web page and its content. HTML consist of a series of elements that dictates and distributes all the given information around the page as we want it to be. Every element inside a HTML file contains not only the content's location but also the media files and links that the page needs. Figure 1 shows an element's structure containing one small paragraph of the page.

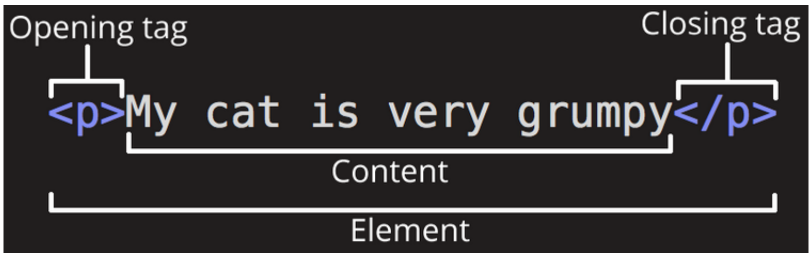

Figure 1

Elements can also have attributes that look like the following

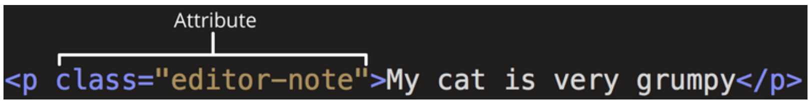

Figure 2

Attributes contain extra information about the element that you don't want to appear in the actual content. Here, `class` is the attribute *name* and `editor-note` is the attribute *value*. The `class` attribute allows you to give the element a non-unique identifier that can be used to target it (and any other elements with the same class value) with style information and other things.

A short document *.htm or *.html, with some code to begin with could be something like:

```
<!DOCTYPE html>
<html>
  <!--
  this is a comment
  -->
  <head>
    <title>A Web Page Example</title>
  </head>
  <body>
    <h1>First Heading</h1>
    <p>One paragraph of text.</p>
    <p>Another paragraph of text, containing an emphasized <em>word</em>.</p>
    <h2>A list of items</h2>
    <ul>
      <li>First item</li>
      <li>Second item</li>
      <li>Third item</li>
      </ul>
    </body>
</html>
```

You can add to this document almost everything you want, the following list contains some elements that could be in the code:

- Images: `<img src="https://www.bit.ly/fcc-relaxing-cat" alt="about the image">`

The source attribute (`src`) refers to the place where the image lives and the `alt` attribute requires some text for screen readers to improve accessibility and is displayed if the image fails to load.

- Anchor: `<a href='https://www.freecatphotoapp.com'>cat photos</a>`

  A link or reference to a page shown as "cat photos".

- New tab: `<a href="https://www.freecatphotoapp.com" target="_blank">cat photos</a>`

  The `target` attribute with the value of "_blank" creates a new tab in the browser for the specified link.

- Nest your image with an `a` element: `<a href="#"><img src="https://www.bit.ly/fcc-running-cats" alt="Three kittens running towards the camera."></a>`

  The image will be like a button for the specified link.

- Text Field: `<input type="text" placeholder="this is placeholder text" required>`

  The placeholder attribute is optional, the required attribute ask the user for an input to be submitted.

- Form Element:

  ```
  <form action="/url-where-you-want-to-submit-form-data">
    <input type="text" placeholder="Enter-Info" required>
  </form>
  ```

  This is a web form that actually submit data to a server using nothing more than pure HTML. You can do this by specifying an `action` attribute on your `form` element. The value for `action` will be the link where we want the information to be submitted.

- Button:`<button type="submit">this button submits the form</button>`

  This button goes inside the form element and it's function is to send the form to the specified link.

3.1.1 Nested Links

You can nest links within other text elements

```
<p>
  Here's a <a target="_blank" href="the_link"> link to the_link</a>.
</p>
```

Sometimes you want to add `a` elements to your website before you know where they will link. This should be easy by using the hash symbol "#" as the value for the `href` attribute. This is known as **Dead Links**.

3.1.2 Radio Buttons

You can use radio buttons for questions where you want the user to only give you one answer out of multiple options.

Radio buttons are a type of `input`.

Each of your radio buttons can be nested within its own `label` element. By wrapping an `input` element inside of a `label` element it will automatically associate the radio button input with the label element surrounding it.

All related radio buttons should have the same `name` attribute to create a radio button group. By creating a radio group, selecting any single radio button will automatically deselect the other buttons within the same group ensuring only one answer is provided by the user.

Here is an example of a radio button:

```
<label for="indoor">
    <input id="indoor" type="radio" name="indoor-outdoor" value="indoor">Indoor
</label>
```

It is considered best practice to set a for attribute on the label element, with a value that matches the value of the id attribute of the input element. This allows assistive technologies to create a linked relationship between the label and the related input element.

References

[1] How does the Internet work? - Learn web development — MDN. (2021, May 13). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/How_does_the_Internet_work

[2] What is the difference between webpage, website, web server, and search engine? - Learn web development — MDN. (2021, January 11). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Pages_sites_servers_and_search_engines

[3] How the Web works - Learn web development — MDN. (2021, May 21). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works#clients_and_servers

[4] Blum, A. (2012). Tubes: behind the scenes at the internet. Penguin UK.

[5] Bates, M. (n.d.). Conquering the Command Line. Softcover.Io. Retrieved July 21, 2021. http://conqueringthecommandline.com/book/basics

[6] Git - About Version Control. (n.d.). Git –Fast-Version-Control. Retrieved July 21, 2021 https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control