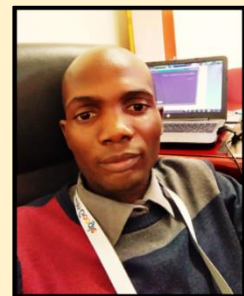
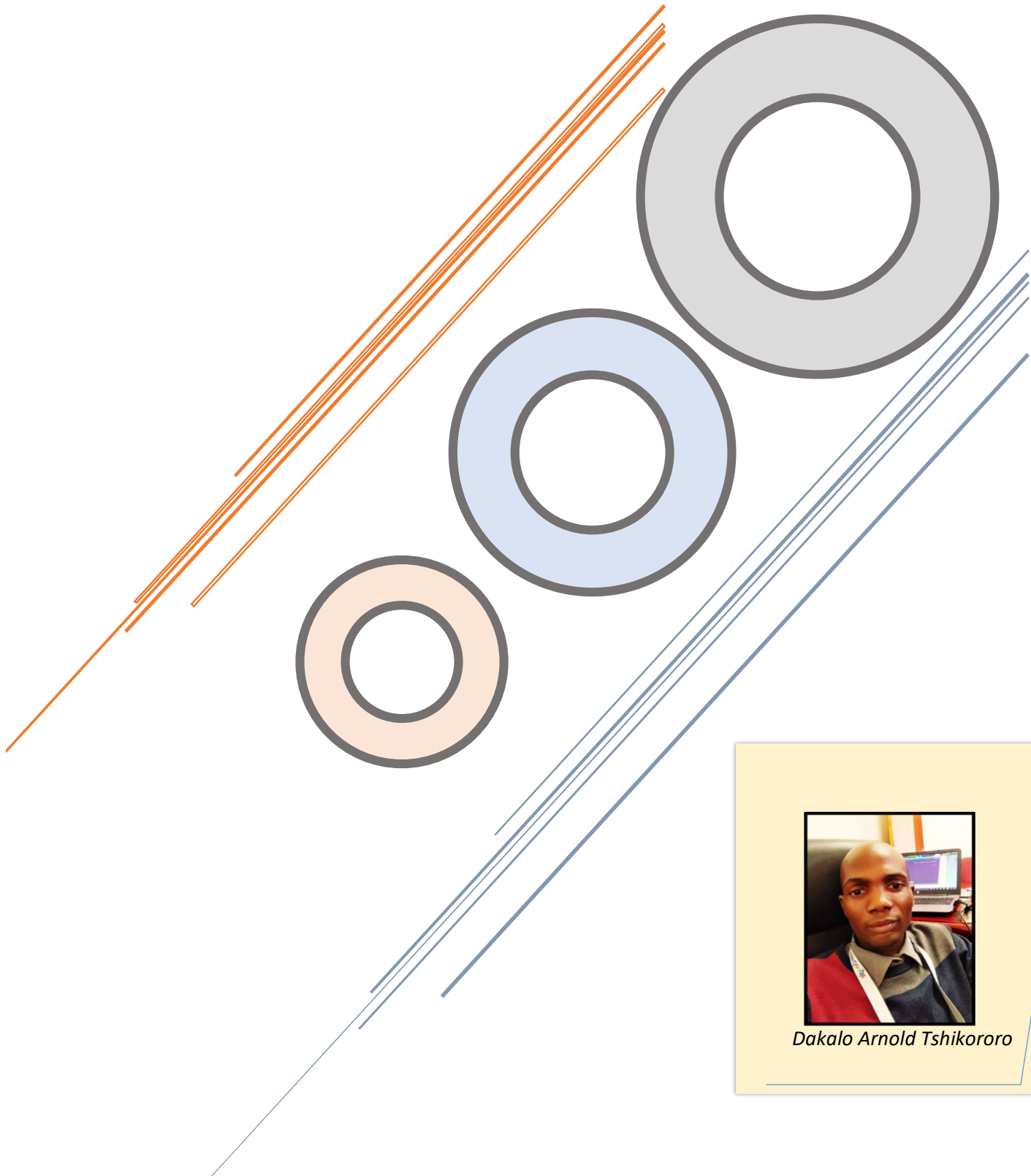




PERSONAL WEBSITE

[GIT | HTML5 | CSS | JAVASCRIPT]



Dakalo Arnold Tshikororo

Table of Contents

Project Requirements	2
Download and Install	2
Let's Start.....	3
CMD COMMAND / <i>cd</i> change directory – move to a specific folder.	3
CMD COMMAND / <i>mkdir</i> create a new directory.	3
CMD COMMAND / <i>ls</i> list all that is in the current directory.	3
Git Repository.....	3
Project Directory Structure.....	4
Git create version file.....	4
Landing page.....	5
Prototype for landing page.....	5
Document Appropriate Skelton.....	5
Useful resources.....	6
Creating a New Branch.....	6
Switching Branches.....	7
Basic Branching and Merging.....	7
Results on browsers <>.....	8
ADDING – CSS/3 AND JAVASCRIPT.....	9
Add CSS On HTML Document Structure.....	9
Add JavaScript On HTML Document Structure	10

Project Requirements

Alright, now it's time to make your own personal website. Design a personal webpage about yourself. The project should be subject to the following requirements:

- Your website must contain at least four different .html pages, and it should be possible to get from any page on your website to any other page by following one or more hyperlinks.
- Your website must include at least one list (ordered or unordered), at least one table, and at least one image.
- You should also have some content placed within paragraph.
- Your website must have at least one stylesheet file. The stylesheet(s) must use at least five different *CSS properties*, and at least five different types of *CSS selectors*. You must use the *#id selector* at least once, and the *.class selector* at least once.

Download and Install



<https://git-scm.com/download/win>
<https://git-scm.com/book/en/v2>



<https://notepad-plus-plus.org/downloads/>

Let's Start

[CMD COMMAND / cd change directory – move to a specific folder.](#)

The first things we will do is to create a workspace folder **c:/** driver. We will be working on **Git Bash terminal**. Now, start by opening your Git bash terminal, then clicking inside your terminal after it has open, and then type the following command: **cd c:/** followed by an **Enter** in order to move to **c:/ drive**.

```
$ cd c:/
```

Here on out, to execute (*i.e., run*) a command means to type it into a git bash terminal and then hit Enter. Commands are “case-sensitive,” so be sure not to type in uppercase when you mean lowercase or vice versa.

[CMD COMMAND / mkdir create a new directory.](#)

Now execute on Git bash terminal

```
$ mkdir workspace
```

To move yourself into (*i.e., open*) that directory. Your prompt should now resemble the below.

```
$ cd workspace
```

Now execute

```
$ mkdir dev-personal-site
```

To move yourself into (*i.e., open*) that directory. Your prompt should now resemble the command below.

```
$ cd dev-personal-site
```

[CMD COMMAND / ls list all that is in the current directory.](#)

Don't forget, to know what is on the directory, we use **ls** command.

```
$ ls
```

This will be empty

Git Repository

Make dev-personal-site a Git Repository => **git init**

```
$ echo "# Dakalo personal website " >> README.md
```

```
$ git init
```

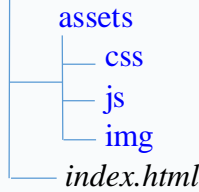
```
$ git add README.md
```

```
$ git commit -m "first commit "
```

This will create a new repository

Project Directory Structure

dev-personal-site => root folder



This will be our landing page

Your website must contain at least four different `.html` pages. Our four different page will be:

- Home : `index.html`
- About (me) : `about.html`
- Skills : `skills.html`
- Portfolio : `portfolio.html`
- Contacts (me) : `contact.html`

Now, let's create the subdirectory and files `.html` using Git bash terminal. For creating subfolder: `assets`. This folder must be created inside `dev-personal-site` folder. Inside `assets` folder: `css, js and img`.

Now execute

```
$ mkdir assets
```

| Quick |

1. To go to `assets` folder, type: `cd assets/` on your terminal.
2. Create `css, js and img` folder, Execute: `mkdir css js img`.
3. To go to `dev-personal-site` folder to create files `.html`, type: `cd ../` then, use `touch` command to create file `.html`

Git create version file

Create a file version => `Git add <filename>`

In order to begin tracking a file version, you use the command `git add`. To begin tracking the `index.php, about.html, skills.html, portfolio.html and contact.html`. Remember, we are keeping the version of a file. Version is created whenever `commit` has been performed.

```
$ git add <filename>
```

This will add a specific file

```
$ git add .
```

This will add all

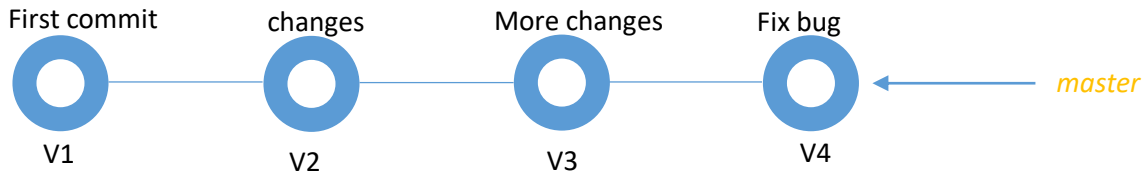
```
$ git commit -m "adding bank files .html"
```

This will commit only add files

```
$ git commit -am "adding bank files .html"
```

This will add and commit files at same time

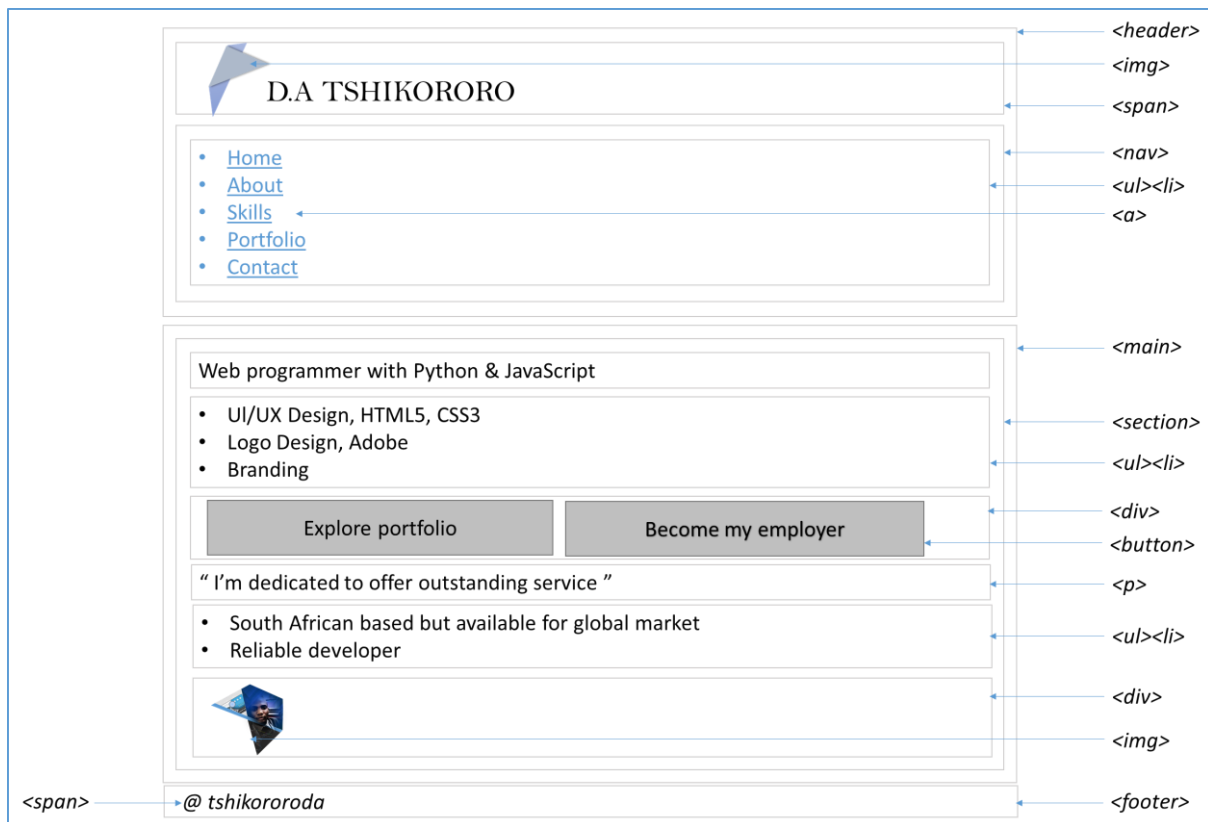
This is our second commit on a `master branch`. We will create other branch to work on, and `merge` them on `master branch` whenever we are happy with the results. In our personal websites, we will make branch for each file `.html` for the purpose of learning. We will do this after we have create our basic HTML5 document structure. Each commit creates a version of file.



Landing page

Let's work on the landing page. On each page, the first thing is to implement or design prototype layout. We do have apps specialized for prototype which professional use to design full – scale of the websites. But for this tutorial, the prototype screens are design using Microsoft PowerPoint 2016.

Prototype for landing page



Document Appropriate Skelton

First of all, we need to open our HTML editor (*Notepad++*). When you open *index.html*, it will be clean white file which is to write your code. From there, we need to layout our page using the following tag.

```
<header> <main> <section> <a> <span> <nav> <div> <article> <footer> <p> <img />
```

All symbol should be implemented using HTML entities. E.g. `©`. If you are a new beginner on html, switch to *your simple task document* before you do the for a prototype above.

1. Document declaration : `<!DOCTYPE html>`
2. Document root : `<html>` : => `<head>` `<body>`
3. Document metadata : `<head>` : => `<link />` `<meta />` `<style>` etc.
4. Document content : `<body>` : => `<p>` `<main>` `<div>` `<a>` `` etc.

In the <https://github.com/tshikororoda/HTML-CCS-JS.git>, you will find a folder *hds* (HTML Document Structure), there is a file named *hds.index.html*. Open the file, and copy and paste everything on your *index.html* on your project and save the file.

Before we commit this file on git repository, we have to validate our mark – up language. To validate our mark-up we use w3 Mark-up validation service: <https://validator.w3.org/>. Validate *index.html* on your project, fix all the bugs, when everything is ok, create a new version of your file with the following message, “*I have added document structure for a landing page*”.

Before we code prototype layout using html tags as shown above, let’s do a bit of learning. HTML tags are group in two level: Block – level elements and inline – level elements. This is very important concept in HTML.

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Content model: Generally, block-level elements may contain inline elements and (*sometimes, not always*) other block-level elements. Inherent in this structural distinction is the idea that block elements create "larger" structures than inline elements.

An inline element does not start on a new line and only takes up as much width as necessary.

Useful resources

1. <https://devdocs.io/>
2. <https://developer.mozilla.org/en-US/docs/Web/HTML>
3. <https://html.com/>
4. <https://www.tutorialspoint.com/html/index.htm>
5. <https://ascii.cl/htmlcodes.htm>

Creating a New Branch

Now, before we do actual coding for landing page, let’s create a new branch to work on the landing page. As we said earlier, the whole purpose of this is to learn. We have already said we will create branch for each page and merge them with *master branch* whenever we are happy with the results. It’s a good practise to commit file and merge branches when we have meet project requirements at each stage. So, lets create a new branch called *home*. To do this we use the following command:

```
$ git branch home
```

This will create a new branch

```
$ git branch
```

This will show all available branch

Switching Branches

To switch to homepage branch, you run the following command.

```
$ git checkout home
```

This will move HEAD to point to the home

Basic Branching and Merging

Now, you have switch to home branch. Let's go through a simple example of branching and merging with a workflow that you might use in the real world based on our personal website project. Let's follow these steps:

1. Do some work on a website master branch: => *'You have done html5 document skeleton?'* ✓
2. Create a branch for a new personal website you're working on: => *'You have created home branch'* ✓
3. Do some work in that new created branch: => *'This what we are going to do now'*

Images you will need to include on your website. You can use the following link:

<https://github.com/tshikororoda/HTML-CCS-JS/tree/master/dev-web-tutorial/dev-00/assets/img>

but I am using this images for demonstration. So feel free to use your own designed prefer images. Now is type to code the complete home prototype as indicated above. You are doing this actual work on home branch. **DON'T MERGE HOME BRACH TO MASTER BRANCH AFTER YOU HAVE DONE.** See the output below. All the best!!!

Let's code //

Results on browsers <>

Rendered HTML on the browsers



D.A Tshikororo

- [Home](#)
- [About](#)
- [Skills](#)
- [Portfolio](#)
- [Contact](#)

Web programmer with Python & JavaScript

- UI/UX Design, HTML5, CSS3
- Logo Design, Adobe
- Branding Identity

Explore Portfolio

Become my employer

" Im dedicated to offer outstanding service"

- South African based but available for global markets
- Reliable developer



@tshikorroda

CONGRADULATION, PHASE ONE IS COMPLETED

ADDING - CSS/3 AND JAVASCRIPT

Add CSS On HTML Document Structure

- Your website must have at least one stylesheet file. The stylesheet(s) must use at least five different *CSS properties*, and at least five different types of *CSS selectors*. You must use the *#id selector* at least once, and the *.class selector* at least once.

If you are a beginner on CSS/3, let's do little bit of learning,

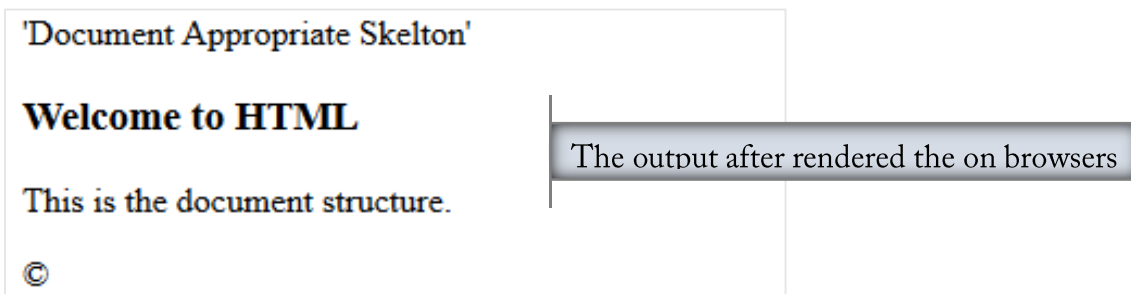
CSS properties to be used are: *width, height, border, overflow, margin, padding, font-, float, clear, display, text-, background, z-index*

On our phase one, we have done two major things: (1) we add *document structure for a landing page* and create (or commit) a vision file on *master* branch. We then create a new branch called *home* and switch to it: (2) we add *landing page prototype*. And create a file vision on home branch. **WE DIDN'T MERGE HOME BRACH TO MASTER BRANCH**. Before we merge home branch to master branch, let's add CSS only on *document structure for a landing page*. This CSS should effect anything on *home* branch. So before we add CSS files, lets switch to *master* branch: This means **HEAD** point to the *home* branch.

Now execute

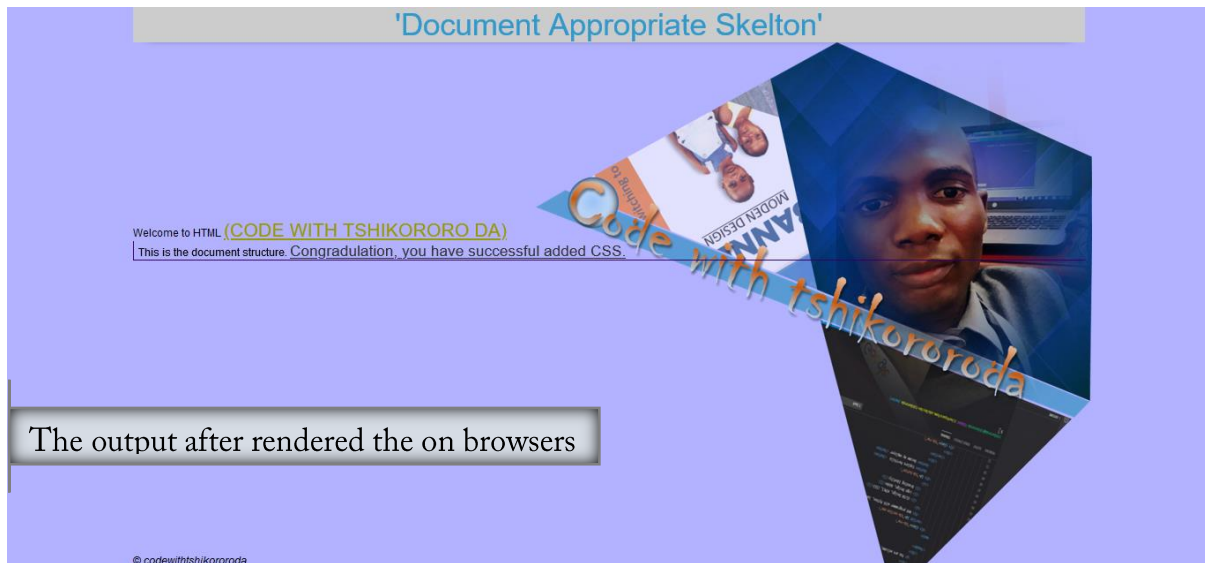
```
$ git checkout master
```

This moves **HEAD** to point to the *master* branch. And by doing so, it will switch from *landing page prototype* to *document structure for a landing page*. Now, we are going to add CSS on *document structure for a landing page* NOT TO *landing page prototype*. The *document structure for a landing page* is on the *master* branch, so is the reason we have to switch back to *master* branch. You will notice that, after switching to *master* branch: everything we have added on *home* branch disappear. render the *index.html* on the browser to view the effects:



If you check *assets/* directory on your root folder, you notice that there is only *JS* and *CSS* folder is visible but *img* folder is not visible. If we can switch to *home* branch, both folder (*JS*, *CSS* and *img*) will be visible. Now, as we follow this link:

<https://github.com/tshikororoda/HTML-CCS-JS.git>, you will find folder *hds-css/assets/css*, there are three files (*style.main.css*, *style.rest.css* and *style.general.css*) in the folder and *img* folder. Copy and paste everything on your program directory (*./assets/css/*). And then render the *index.html* on the browser to view the effects.



WELL DONE, CSS HAS BEED ADDED TO THE HDS

Add JavaScript On HTML Document Structure

While we are still here on master branch, lets add JavaScript file on *document structure for a landing page*. Now, as we follow this link: <https://github.com/tshikororoda/HTML-CCS-JS.git>, you will find folder *hds-js/assets/js*, there is a file *main.js*. Copy and paste the JavaScript *main.js* file on your program directory (*./assets/js/*). And then render the *index.html* on the browser to view the effects. Observe header and footer output: year on footer is added by JavaScript function and daytime on header is also added by JavaScript function.

WELL DONE, JAVASCRIPT HAS BEED ADDED TO THE HDS