




MARCH 3, 2022

INTRODUCTION HOMEWORK

HTML & CSS

BOBAN SREZOVSKI
CODE ADACEMY
Street: Anton Popov no.33



Contents

INTRODUCTION.....	2
Exercise 1	3
Exercise 2	4
Exercise 3	5
Exercise 4	7
Exercise 5	8
Exercise 6	10
Exercise 7	12
Exercise 8	14
Exercise 9	16
Exercise 10	19
Exercise 11	21
Exercise 12	23
Exercise 13	26
Exercise 14	28
Exercise 15	32
Exercise 16	35
Exercise 17	38
Exercise 18	41

INTRODUCTION

These exercises are meant to keep you practicing thing we learn and what we are about to learn in the course. If you don't know how to solve any of the exercise, please first see the solution and try to figure it out, and if you still don't know how to solve them, please contact me.

The main goal of these exercises is for you to get familiar with HTML and CSS. So, as I said earlier if you get stuck on one, please first see the solution, or continue to the next one.

Best way to learn is to practice coding

Exercise 1

After this exercise, you should be able to:

- [Define HTML](#)
- [Recognize an HTML tag](#)
- [Identify the HTML paragraph tag](#)

Example

HTML, or Hypertext Markup Language, describes a hierarchy of visual elements that a browser can display.

HTML usually looks like human-readable text surrounded by tags. We create tags using angle brackets, like so:

```
<example-tag>
  There was music in the cafes at night, and revolution in the air.
</example-tag>
```

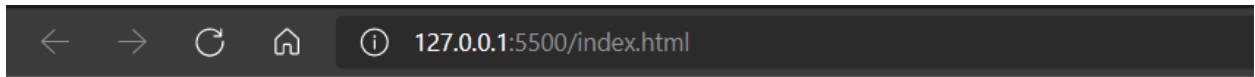
Most tags require both an opening (<example-tag>) and a closing tag (</example-tag>). A forward-slash (/) proceeds the first angle bracket of all closing tags.

The most common tag is <p>, short for a *paragraph*.

Exercise

1. Create a directory in HTML_AND_CSS_Introduction named Homework2/
2. Create a new file by clicking on the icon to the right of the Homework2/ directory in your code editor.
3. Name the new file HalloWorld.html and then click on the checkmark to save the file.
4. Create beginning html template by inserting “! + Tab” in VSCode index.html file or copy it from this link [html5-template-boilerplate-code-example](#).
5. Inside the body section insert the <p> tag.
6. Type Hello World inside the <p> tag.

- Click the Save button to save your changes. Open the HalloWorld.html you should see same result as the picture.



- Add, Commit and push the changes to GitHub

Exercise 2

After this exercise, you should be able to:

- Repeat the paragraph tag multiple times in a document

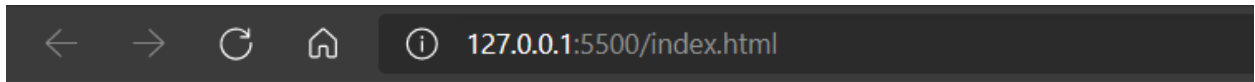
Example

You can add multiple paragraphs to the same document. Let's translate, "Hello World," into a few other languages!

Exercise

- Open HalloWorld.html
- Insert "English:" before the text in your first paragraph.
- Add three new paragraphs:
 - Hawaiian: Aloha Houna
 - Luxembourgish: Moien Welt
 - Swahili: Salamu Dunia

- Click the Save button to save your changes. Open the HalloWorld.html you should see same result as the picture.



English: Hello World
Hawaiian: Aloha Houna
Luxembourgish: Moien Welt
Swahili: Salamu Dunia

- Add, Commit and push the changes to GitHub

Exercise 3

After this exercise, you should be able to:

- Identify the HTML header tag (<h1>, <h2>, etc.)
- Use an HTML header tag in a document
- Understand when not to use <h1> tags

Example

Not all text is *paragraph* text. Documents often have headings; headings are powerful!

When writing headings, instead of using <p> (and </p>), you use <h1>, <h2>, and up to <h6>.

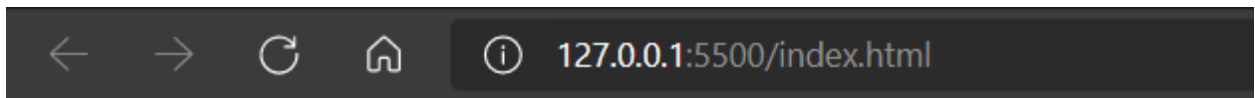
- <h1>: most important text (largest).
- <h2>, <h3>,...
- <h6>: least important text (smallest).

Your pages should have at most a single `<h1>` tag, but use other tags as many times as you like. For example, in a newspaper, "The New York Times" is in large letters on the front page. This is analogous to the `<h1>` tag.

It's not a hard and fast rule, but it keeps the importance of each heading readily distinguishable.

Exercise

1. Open `HalloWorld.html`
2. Add a `<h1>` tag at the top of your document with the text "Hello World Translations".
3. Click the Save button to save your changes. Open the `HalloWorld.html` you should see same result as the picture.



Hello World Translations

English: Hello World

Hawaiian: Aloha Houna

Luxembourgish: Moien Welt

Swahili: Salamu Dunia

4. Add, Commit and push the changes to GitHub

Exercise 4

After this exercise, you should be able to:

- Identify the `<html>` and `<body>` tags
- Describe the order of these tags
- Describe the importance of each tag

Example

The text you've written thus far belongs to the visual content of the HTML document. This is called the document's **body**, and we surrounded it with `<body>` tags.

The document's body is one part of an HTML document (we'll get to the other part — the document's **head** — later.) And we surround the `<body>` tags with `<html>` tags, like so:

```
<html>
  <body>
    <p>Some content goes here.</p>
  </body>
</html>
```

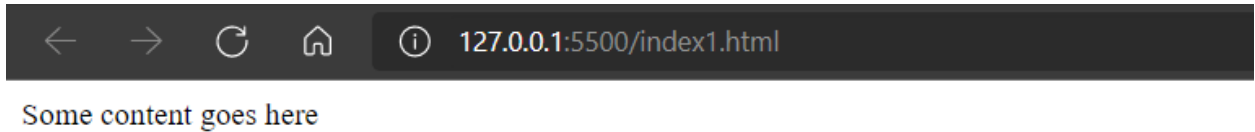
`<html>` indicates the beginning, and `</html>` indicates the end of your HTML document.

It may seem overly verbose, but these formatting rules help web browsers display your documents properly. Without them, browsers may display your content incorrectly, may take longer to display it, or may not display it at all.

Exercise

1. Create a new file "index.html" by clicking on the icon to the right of the Homework2/ directory in your code editor
2. Place `<html>` open and closing tag.
3. Inside the `<html>` tag put a `<body>` open and closing tag.

4. Inside the `<body>` tag put a `<p>` open and closing tag with the next text inside "Some content goes here"
5. Click the Save button to save your changes. Open the index.html you should see same result as the picture.



6. Add, Commit and push the changes to GitHub

Exercise 5

After this exercise, you should be able to:

- Identify the `<head>` and `<title>` tags
- Describe the order of these tags
- Describe why these tags are important
- Explain where the text in the `<title>` tag is used
- Identify the DOCTYPE instruction
- Explain why the DOCTYPE instruction is used

Example

HTML documents must include a body and a **head** section. The head contains non-visible page elements which the web browser or search engine crawlers may need.

At a minimum, you must include the <title> tag which defines the title of the HTML document. The title is used in a few places, like the browser's title bar and the bookmarks menu. It's also used in search engine results.

Here's a look at a sample head section with a title:

```
<html>
  <head>
    <title>An Eye-Catching Title</title>
  </head>
  <body>
    More HTML goes here.
  </body>
</html>
```

We must have these elements in place to comply with HTML5 standards. And to declare our page an HTML5 document, we insert a DOCTYPE tag at the top, like so:

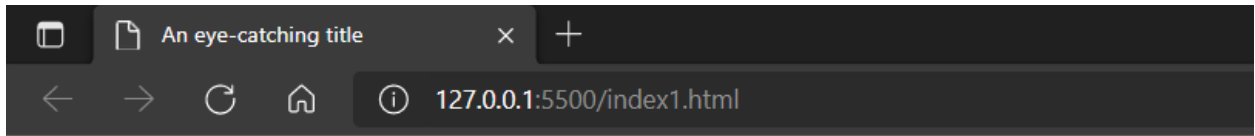
```
<!DOCTYPE html>
<html>
  <head>
    ...
```

Unlike other tags we've worked with, DOCTYPE isn't an HTML tag. It's a special web browser instruction that indicates what kind of document this is. This helps browsers render your pages faster because they skip the process which determines the version of HTML your document requires (in this case, we know it's 5).

Exercise

1. Open index.html
2. Add <head> tags to your document after the open <html> tag.
3. Add a <title> tag, make it anything you like.
4. Add the DOCTYPE instruction.

5. Click the Save button to save your changes. Open the index.html you should see same result as the picture.



6. Add, Commit and push the changes to GitHub

Exercise 6

After this exercise, you should be able to:

- Identify tag attributes
- Identify tags that don't close
- Explain the difference between absolute and relative URLs
- Describe the `` tag
- Discuss the `alt` and `src` attributes
- Explain why the `alt` attribute is important

Example

Let's spruce things up a bit by adding an image. We add images using the `` tag. It's different from other tags we've used in two ways:

- I. It requires *attributes*
- II. We don't need to close it (no `` at the end)

An image tag can look like this:

```

```

src specifies the URL of the image to display. The above example is a *relative* URL, but you can use absolute URLs like `http://example.com/images/flower.png`.

You can [read more about relative vs. absolute URLs on webreference](#).

alt provides a brief description of the image. The browser renders the alt text if it fails to display the image. Causes for this include:

- The user is on a slow connection
- The user is vision-impaired and they interact with their browser using audio
- The user has an old browser that doesn't support images
- The image cannot be found (broken link)

Exercise

1. Open index.html
2. Add an image under the <body>; you may use any image you like!
 - If you can't find an image, please contact me.
 - Make sure to provide an alt attribute with your image
3. Add, Commit and push the changes to GitHub

Exercise 7

After this exercise, you should be able to:

- Use paragraphs, headings, and images to build a basic profile

Example

Practically every website on the planet allows users to create a profile. Profiles like those on Facebook aggregate an enormous amount of personal information: age, gender, eye color, height, weight, and the total number of hairs on your body (56,783,238 for me!).

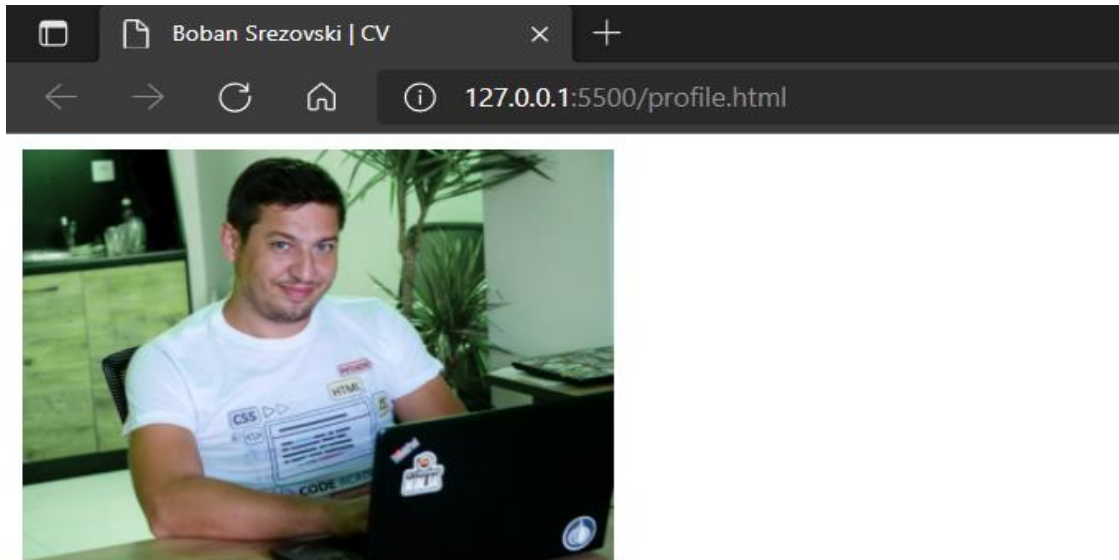
In this exercise, you will use your brand new HTML knowledge to create your profile that reflects the best things about you!

The workshop doesn't currently support image upload. If you want to experiment with other images, try an image hosting service like [imgur](https://imgur.com/).

Exercise

1. Create and open a file named profile.html
2. Add the necessary HTML structure: doctype, html, head, body, etc.
3. Choose a title for your profile
4. Use a combination of images, headings, and paragraphs to represent yourself. For example:
 - Profile Image
 - Heading (your name)
 - Paragraph (about you!)
 - Heading (My Secrets)
 - **Several paragraphs divulging all of your deepest, darkest secrets** - We won't read these (out loud)
5. Save the file whenever you're satisfied with your profile, no rush. And most importantly, **have fun!**

- Click the Save button to save your changes. Open the profile.html you should see same result as the picture



Boban Srezovski

I am a fullstack DotNet developer and mentor at Code Academy

My Secrets

1. working hard
2. Patience

7. Add, Commit and push the changes to GitHub

Exercise 8

After this exercise, you should be able to:

- Define the difference between ordered and unordered lists
- Implement the ``, ``, and `` tags
- Execute a conversion from paragraph text to lists

Example

Our page currently displays a list of information in three separate paragraphs. It makes more sense to display this information in a list.

HTML supports two types of lists: ordered lists (``) and unordered lists (``). Each list item is defined by a `` tag.

Here's an ordered list:

```
<ol>
  <li>Beyoncé</li>
  <li>Mary Virginia Cook Parrish</li>
  <li>Harriet Tubman</li>
</ol>
```

These are typically rendered with numbers, like so:

1. Beyoncé
2. Mary Virginia Cook Parrish
3. Harriet Tubman

Here's an unordered list:

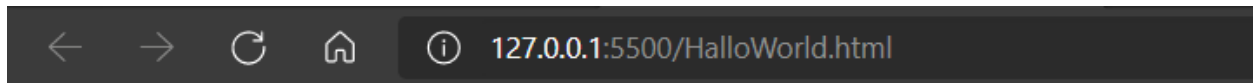
```
<ul>
  <li>Archduke Charles, Duke of Teschen</li>
  <li>Edward Lear</li>
  <li>Harriet Tubman</li>
</ul>
```

These are typically rendered with bullet points, like so:

- Archduke Charles, Duke of Teschen
- Edward Lear
- Harriet Tubman

Exercise

1. Open HelloWorld.html
2. Convert the four paragraphs into one unordered list with four list items.
3. Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture



Hello World Translations

- English: Hello World
- Hawaiian: Aloha Houna
- Luxembourgish: Moien Welt
- Swahili: Salamu Dunia

4. Add, Commit and push the changes to GitHub

Exercise 9

After this exercise, you should be able to:

- Define an HTML table
- Understand and implement the `<table>`, `<th>`, `<tr>`, and `<td>` tags
- Convert lists to tables

Example

Lists are good for showing a series of items in a single column. But some information is better represented on a grid.

For example, it's difficult to express this information in a simple list:

Country	National Animal	Motto	Population
Algeria	Fennec fox	"By the people and for the people"	40,400,000
Mexico	Xoloitzcuintli	"The Homeland is First"	119,530,753
Moldova	Aurochs	"Our Language is a Treasure"	2,913,281
Scotland	Unicorn	"In my defence God me defend"	5,347,600
United States	Bald eagle	"Out of many, one"	323,625,762

`<table>` works similarly to `` and ``, but with a bit more hierarchy. Here are the tags you'll typically find inside:

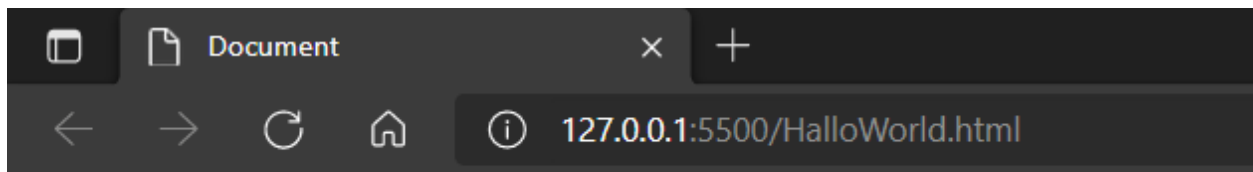
- `<tr>` ("table row") surrounds each row
- `<th>` ("table header") surrounds each header cell
- `<td>` ("table data") surrounds each regular cell

Here's the HTML that produces the above example. Notice how the header cells are styled with more emphasis than the data cells.

```
<table>
  <tr>
    <th>Country</th>
    <th>National Animal</th>
    <th>Motto</th>
    <th>Population</th>
  </tr>
  <tr>
    <td>Algeria</td>
    <td>Fennec fox</td>
    <td>"By the people and for the people"</td>
    <td>40,400,000</td>
  </tr>
  <tr>
    <td>Mexico</td>
    <td>Xoloitzcuintli</td>
    <td>"The Homeland is First"</td>
    <td>119,530,753</td>
  </tr>
  <tr>
    <td>Moldova</td>
    <td>Aurochs</td>
    <td>"Our Language is a Treasure"</td>
    <td>2,913,281</td>
  </tr>
  <tr>
    <td>Scotland</td>
    <td>Unicorn</td>
    <td>"In my defence God me defend"</td>
    <td>5,347,600</td>
  </tr>
  <tr>
    <td>United States</td>
    <td>Bald eagle</td>
    <td>"Out of many, one"</td>
    <td>323,625,762</td>
  </tr>
</table>
```

Exercise

1. Open HelloWorld.html
2. Convert your unordered list into a table with the following specifications:
 - One header row with the titles Language and Translation
 - Four data rows, with two columns:
 - Language (like Luxembourgish)
 - Translation (like Moien Welt)
3. Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture



Hello World Translations

Language	Translation
English:	Hello World
Hawaiian:	Aloha Houna
Luxembourgish:	Moien Welt
Swahili:	Salamu Dunia

4. Add, Commit and push the changes to GitHub

Exercise 10

After this exercise, you should be able to:

- Define "hyperlink"
- Understand and use the `<a>` tag

Example

Tables are good for showing a lot of data, but what if the user wants more information? What type of grammatical structure does Hawaiian have? Are there alternative spellings of "Luxembourgish?" Which languages affected the development of Swahili?

To help them, we can provide links (short for "hyperlinks") to other pages on the internet.

A link takes the following format:

```
<a href="http://www.example.com">Example Website</a>
```

It looks like this when rendered:

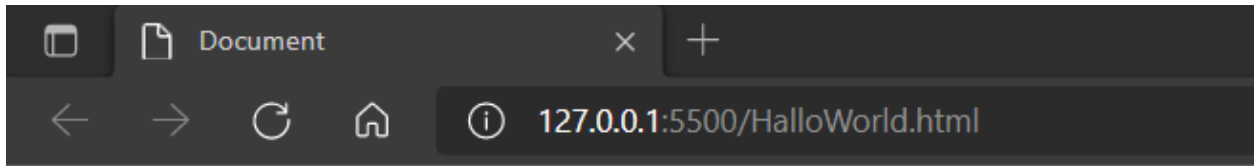
[Example Website](#)

The href attribute, like src in the `` tag, contains a relative or absolute URL. The text between the tags represents the link's title.

Exercise

1. Open HelloWorld.html
2. Link Swahili to https://en.wikipedia.org/wiki/Swahili_language.
3. After the table, add a paragraph that reads, "For additional translations, visit Bing Translator."
4. Link Bing Translator to <https://www.bing.com/translator>.

5. Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture



Hello World Translations

Language	Translation
English:	Hello World
Hawaiian:	Aloha Houna
Luxembourgish:	Moien Welt
Swahili:	Salamu Dunia

For additional translations, visit [Bing Translator](#).

6. Add, Commit and push the changes to GitHub

Exercise 11

After this exercise, you should be able to:

- Use lists, tables, and links to make an even better profile

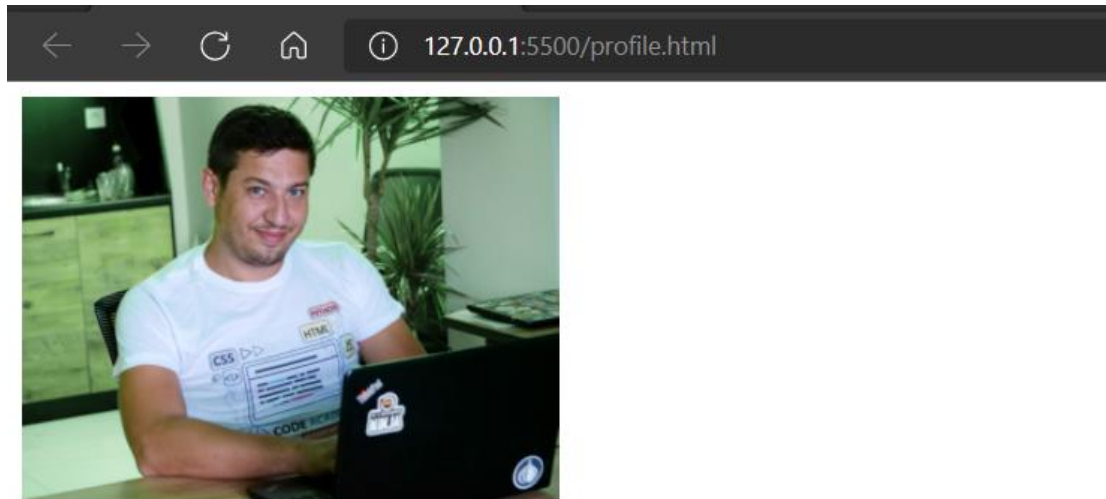
Example

Remember that excellent profile you made? Let's use lists, tables, and links to make it more excellent-er!

Exercise

1. Open "profile.html"
2. Add a heading below your last `<p>` tag titled, "My Phobias."
(use `<h2>`)
3. Below this heading, create an **ordered list** of your greatest fears, from most frightening to least. Here's mine:
4. Not breathing
5. Dying with regrets
6. Never seeing my family again
7. Ignoring cries for help
8. Hating the work I do
9. Doing my business on the toilet and I look down and there's a snake
10. Add a two-column table below your "about me" paragraph with the following rows:
 - "Favorite Book" | your favorite book
 - "Favorite Movie" | your favorite movie
 - "Favorite Band/Artist" | your favorite band or musical artist
 - "Favorite TV Show" | your favorite TV show
 - "Favorite Quote" | your favorite quote
5. For each response in the table, link to a relevant page, e.g. "Favorite Movie" | [Fifth Element](#).

6. Click the Save button to save your changes. Open the profile.html you should see same result as the picture.



Boban Srezovski

I am a fullstack DotNet developer and mentor at Code Academy

My Secrets

1. working hard
2. Patience

My Phobias.

1. Not breathing
2. Dying with regrets
3. Never seeing my family again
4. Ignoring cries for help
5. Hating the work I do

Favorite Book: [your favorite book](#)
Favorite Movie: [your favorite Movie](#)
Favorite Band/Artist: [your favorite TV Show](#)
Favorite TV Show: [your favorite TV Show](#)
Favorite Quote: [your favorite Quote](#)

7. Add, Commit and push the changes to GitHub

Exercise 12

After this exercise, you should be able to:

- Recognize a CSS ruleset
- Explain the difference between HTML and CSS
- List the elements of a CSS ruleset
- State an example of a CSS ruleset
- Understand and use the `<style>` tag

Example

We use HTML to describe the meaning and hierarchy of our content, but it doesn't say much about how that content should look.

CSS is a separate language that describes the *presentation* of HTML content.

CSS works by applying one or more **rules** to an element on the page. Together, these rules are called a **ruleset**. Here's an example:

```
h1 {  
  color: aqua;  
  text-align: center;  
}
```

In this example:

- `h1` is a **selector** (more on that later)
- `color` and `text-align` are **properties**
- `aqua` and `center` are **values**

A property paired with a value defines a single rule. And together, one selector and one or more rules make up a rule set.

Try adding this CSS to your `index.html` file. It goes inside `<style>` tags in your `<head>` section:


```
<head>
  <title>Hello World Translations</title>
  <style>
    h1 {
      color: aqua;
      text-align: center;
    }
  </style>
</head>
```

Notice how the selector, `h1`, corresponds to the `<h1>` HTML tag.

Reload the page, and the header text will appear in a soothing aqua color (color) and centered on the page (text-align).

Exercise

Let's spruce things up a bit.

1. Open [W3's CSS Reference](#) in a new tab and open profile.html.

For each part of the assignment, review the CSS Reference and experiment with different options.

You may encounter terms you're not familiar with, like *sans serif*, *margin*, and *pixel* (px for short). These terms are defined in the CSS Reference.

2. Using a body selector, create a ruleset:
 - Set background-color to black.
 - Set font-family to sans-serif.
 - Set color to white.
3. Using a selector, set the color of links to hotpink.
4. Using the `img` selector, center your image:
 - Set display to block.
 - Set both margin-left and margin-right to auto (you'll learn more about margins later).
5. Use the `body` selector to set the border to 2px solid white.

This is called a *shorthand property* because it sets multiple values at once. Review the [CSS border Property](#) reference to learn more.

For the final part of the assignment, you'll apply rules to *both* the table and td selectors, like so:

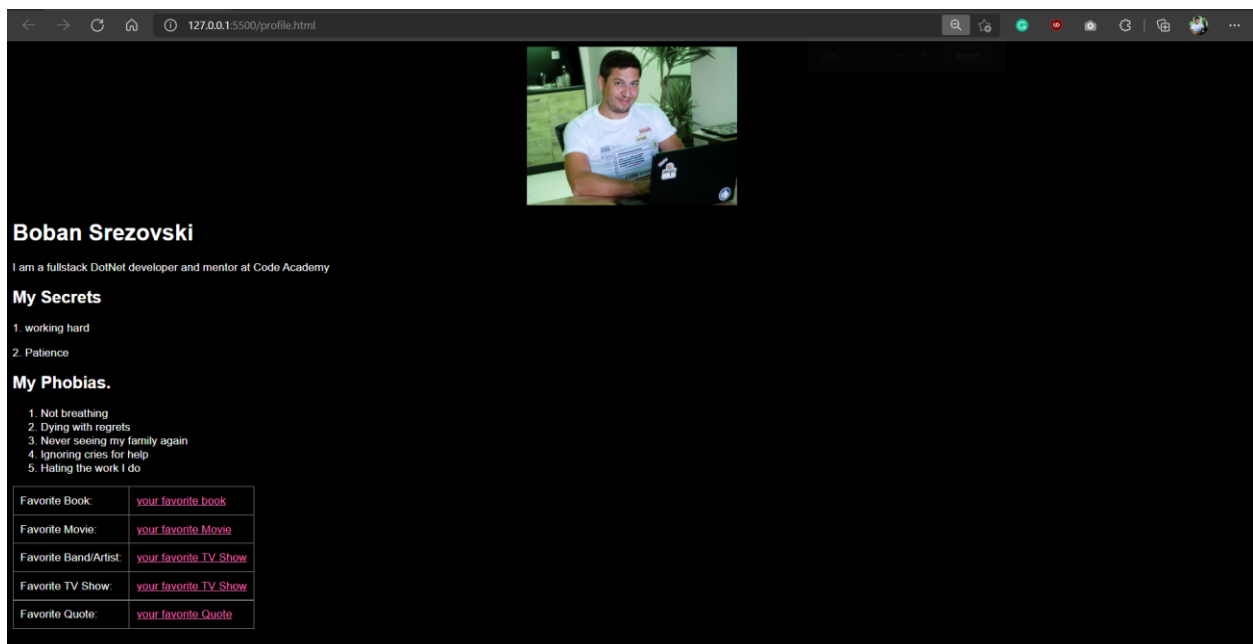
```
table, td {  
  ...  
}
```

6. Add styling to table and td:

- Set border to 1px solid gray.
- Set border-collapse to collapse.
- Set padding to 10px.

Before submitting this assignment, experiment with different values to observe their effects.

7. Click the Save button to save your changes. Open the profile.html you should see same result as the picture.



8. Add, Commit and push the changes to GitHub.

Exercise 13

After this exercise, you should be able to:

- Explain the difference between internal CSS and external CSS
- Execute a conversion from internal CSS to external CSS

Example

Your CSS is included within profile.html's `<style>` tags, so it only applies to that single page. But most websites reuse styles on *all* pages. For this reason, web developers discourage the use of `<style>` tags.

Instead, most web developers recommend using separate CSS files, which we can reference using the `<link>` HTML tag.

The `<link>` tag defines a relationship between your HTML document and some other file. Like the `<style>` tag, it belongs in the `<head>` section of your HTML document. Here's how it can reference a CSS file:

```
<link rel="stylesheet" type="text/css" href="filename.css">
```

- `rel` is an abbreviation for *relationship*. In this case, we specify a "style" relationship.
- `type` specifies the Internet media type. For CSS, the media type is `text/CSS`.
- `href` locates the file, just like in ``.

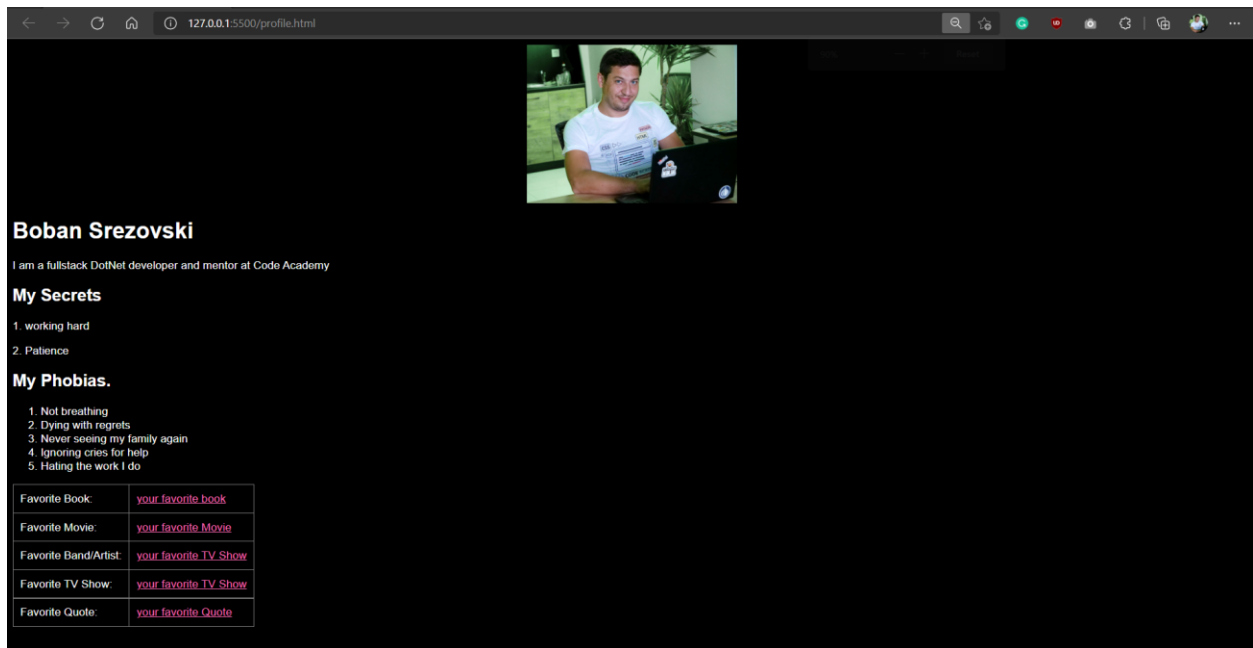
We can name CSS files anything but review these [CSS naming guidelines](#) in a new tab.

Exercise

In this exercise, move the CSS to an external file. Open profile.html.

1. Create a file named main.css
2. In profile.html, copy everything between (but not including) the `<style>` tags
3. Switch to main.css
4. Paste the style information into main.css

5. Return to profile.html
6. Replace the <style> tags (and everything between them) with a single link tag that connects your HTML file to your CSS file (href="./main.css")
(<link rel="stylesheet" href="path to css file">)
7. Click the Save button to save your changes. Open the profile.html you should see same result as the picture.



8. Add, Commit and push the changes to GitHub.

Exercise 14

After this exercise, you should be able to:

- Recognize a selector
- Explain what a selector is
- Explain the difference between "id" and "class" selectors
- Use selectors to set the style for all elements of a single type
- Use selectors to set the style for all elements with a specific class
- Use selectors to set the style of a specific element
- Use `` to style text within an element

Example

We've used a few **selectors** in our CSS file:

- **body**
- **a**
- **img**
- **h1**
- **th**
- **table, td**

As a reminder, selectors apply rules to one or more elements. There are many types of selectors ([here's a list](#)), but there are three primary selector types: element, class, and id.

Element Selectors are like those you've used already. They select all HTML elements of a certain type, like `p` or `img`.

Class Selectors, preceded by a period (`.`), define styles that you can apply to any element, regardless of type. For example, if you're making a newspaper or blog website, you would use a class selector for captions that go under an image. You could create CSS like this:

```
.captionText {  
  font-size: 11px;  
  line-height: 14px;  
  font-style: normal;  
  font-family: Georgia, "Times New Roman", Serif;  
}
```

And then you can apply this class style using the HTML class attribute, like this:

```
  
<p class="captionText">Juline Lamusga, a center from Bethesda, MD, is expected to be drafted first in Friday's NHL draft.</p>
```

You can apply as many classes to an element as you wish by separating each class with a space, e.g. `class="captionText boldText underlinedText"`.

ID Selectors, preceded by a hash (#), target a single element on the page. For example, if a page has one alert paragraph, you can define its style like this:

```
#alert {  
  color: red;  
  background: yellow;  
  border: 2px solid black;  
  font-weight: bold;  
}
```

To apply the style, you would use the id attribute:

```
<p id="alert">Warning: there is a 40% chance of rain today.</p>
```

id attributes are unique on each page. For example, the document above may not contain a second element with alert as its id.

If you don't want to style the entire element, you can use the `` tag to style a portion of it. For example, to make the percentage text larger:

```
#alert-percentage {  
  font-size: 18px;  
}
```

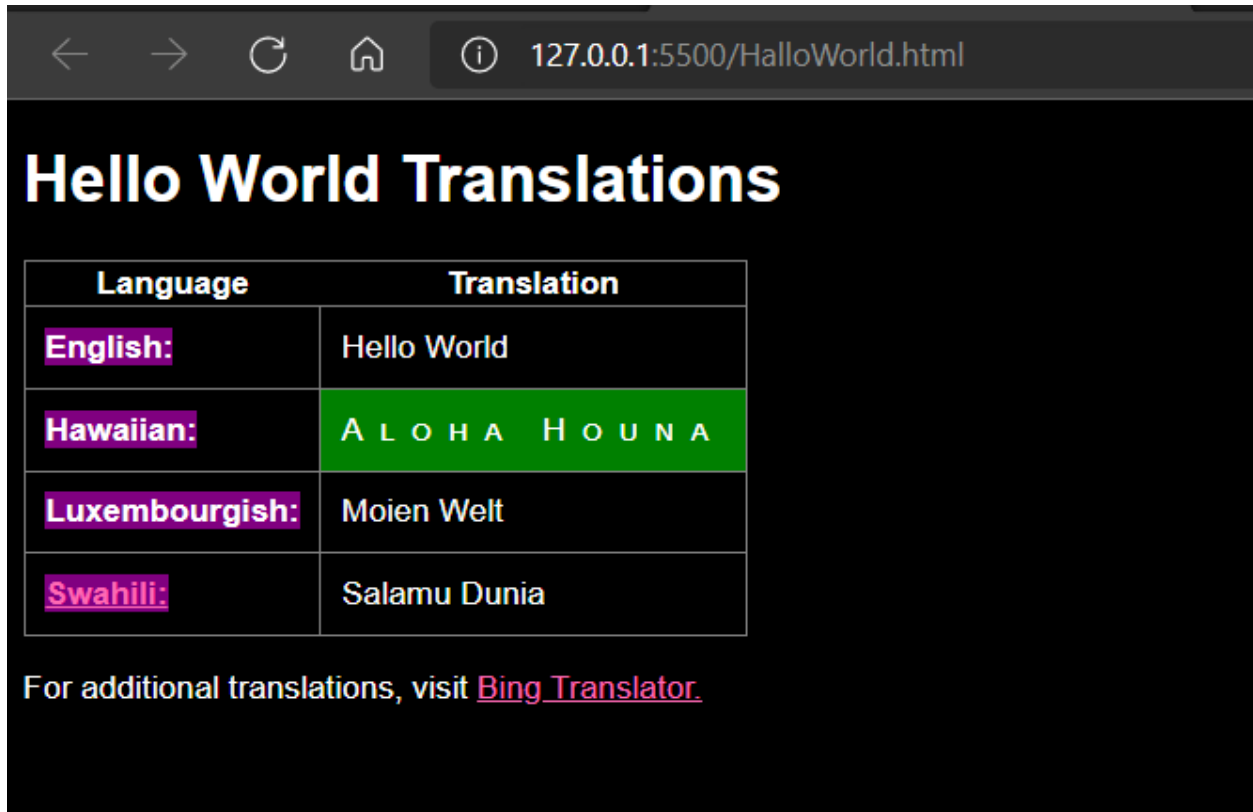
```
<p id="alert">Warning: there is a <span id="alert-percentage">40%</span> chance of rain today.</p>
```

Want more practice? Open [Try CSS Selectors](#) in a new tab to visualize how different selectors represent different areas of an HTML document.

Exercise

1. Open main.css.
2. Add a "language" class selector (it should begin with a ".").
 - Set its background to purple.
 - Set its font-weight to bold.
3. Add an "aloha" ID selector (it should begin with a "#").
 - Set its letter-spacing to 8px.
 - Set its font-variant to small-caps.
 - Set its background to green.
4. Open HalloWorld.html.
5. Link main.css. Add a `<link rel="stylesheet" href="./main.css">` Just under the `<title>` tag.
6. Use the `` tag to apply the language class to each of the four languages (like English and Hawaiian).
(Example: `<td>English:</td>`)
7. Add the aloha id attribute to the `<td>` tag surrounding Aloha Houna.
(Example: `<td id="aloha">Aloha Houna</td>`)

- Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture.



- Add, Commit and push the changes to GitHub.

Exercise 15

After this exercise, you should be able to:

- Recognize an RGB triplet color
- Recognize a hex color
- Explain the difference between hex and RGB triplet colors
- Assign colors to different elements

Example

We've used some colors so far, like green and hotpink. But millions of colors don't have a name.

On computers, colors are represented by mixing red, green, and blue. There are two ways to define these colors. The first is a **Red / Green / Blue (RGB) Triplet Color**. Using this syntax you can measure the amount of each color on a scale of 0 to 255. For example, here's pure red:

```
RGB(255,0,0)
```

Here's pure green:

```
RGB(0,255,0)
```

Here's magenta:

```
RGB(255,0,255)
```

Here's a deep blue:

```
RGB(34,55,90)
```

And a pale pink:

```
RGB(250,202,222)
```

As you can see, you can generate many different colors. Sites like [color-hex](https://color-hex.org/) offer color palettes.

Most graphic designers think RGB triplet colors are easier to read, but some prefer hex colors. A **hex color** expresses the same information differently. It's a six-character code following a #. Here's pure red in hex:

`#ff0000`

The first two characters represent red, the next two green, and the final two blue. If you want, you can [read more about hexadecimal numbering](#), but the most important thing to remember is that they're like regular numbers, but A–F represents 10–15 (F is the highest hexadecimal digit).

Notice that ff corresponds directly to 255. That's because f (15) in the left spot is multiplied by the base, 16, and f in the right spot is multiplied by 1. Added together: $15 \times 16 + 15 \times 1 = 255$.

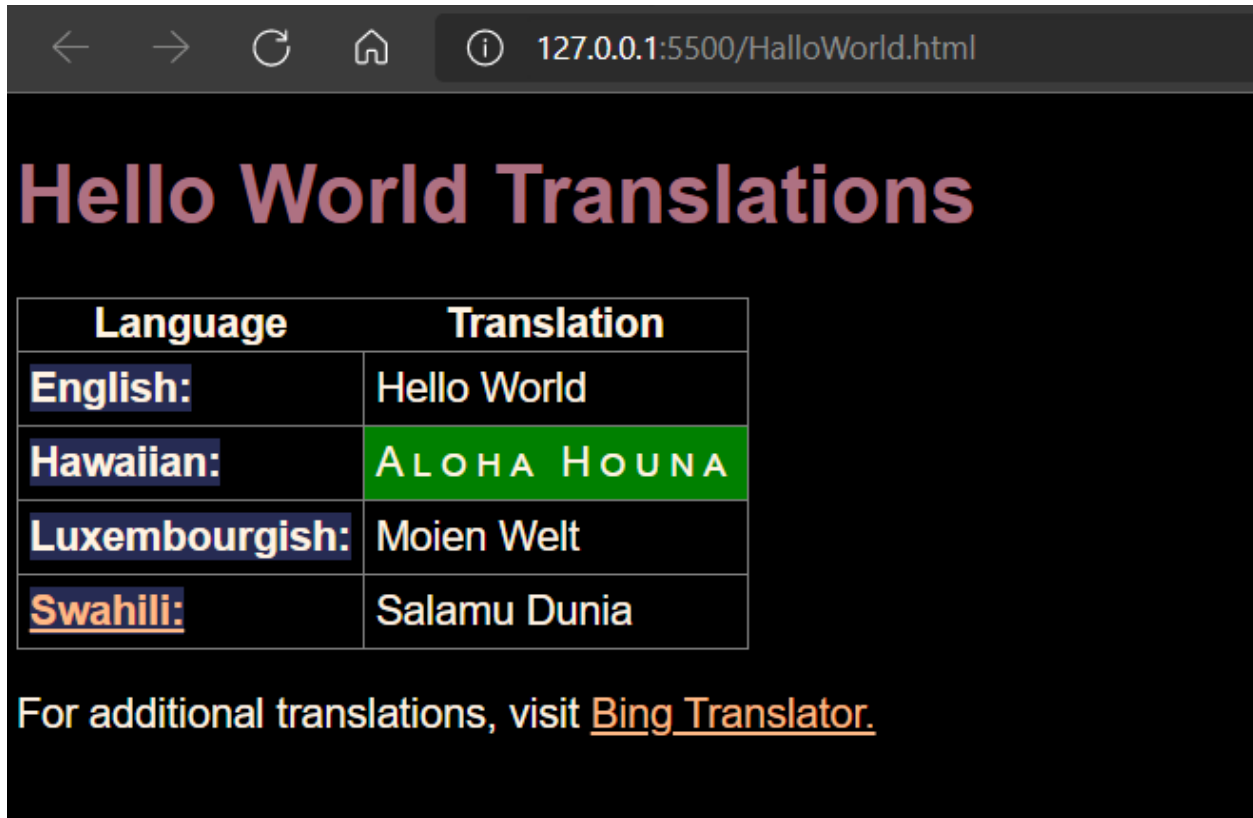
Exercise

Let's use a more soothing color scheme.

Before you type the hex codes, try to guess how the color will appear.

1. Open main.css.
2. Set the body text color and the table header borders to #fff3e3.
3. While we're updating body, set font-size to 36px and font-weight to lighter.
4. Set the link color to #ffb985.
5. Set the h1 color to #ae7182.
6. Set the language background color and the table border color to #262b53.

- Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture.



- Add, Commit and push the changes to GitHub.

Exercise 16

After this exercise, you should be able to:

- Explain the difference between pixels and ems
- Use font and text attributes to style and space text
- Use the text-shadow CSS attribute to create drop shadows

Example

You've already seen a few ways to style text:

- You can change its color
- You can center it by changing its text-align property
- You can change its font family, font-size, font-weight, and font-variant
- You can change the spacing between letters with letter-spacing

There are more ways to style text — way too many to include in this exercise. Here are a few highlights.

We've specified sizes in *pixels* (px), which count the number of dots on a computer screen. Many designers prefer to specify text sizes in *ems*:

```
p {  
  font-size: 0.75em;  
}
```

Here's an example:

This text is about to get smaller

1 em is equal to your element's default text size. In the above example, we rendered the smaller text at 75% of the normal size.

Text indents allow you to indent the beginning of a text element (like the tab at the beginning of a paragraph):

```
p {  
  text-indent: 4em;  
}
```

Here's an example:

This text is indented, good sir and/or madam!

This text is not indented, I'm afraid.

The line spacing affects the spacing between lines:

```
p {  
  line-height: 2em;  
}
```

For example:

Lines can run long sometimes, so space them out to keep the reader's eyes from tiring out.

You can add shadows (with blurs) to text using text-shadow:

```
p {  
  text-shadow: 2px 2px 8px #C0C0F3;  
}
```

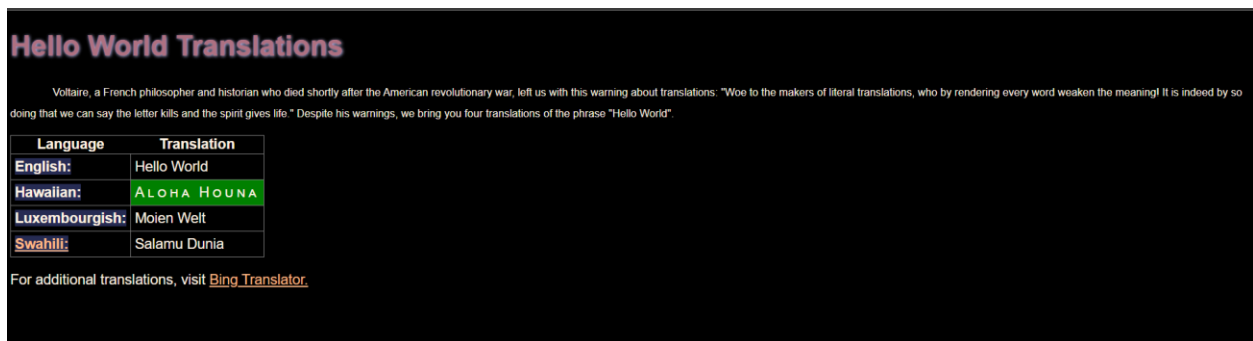
This is some shaaaaady text!

The first two numbers represent the horizontal and vertical size of the shadow. The third number represents the blur radius. The fourth value is color.

Exercise

1. Open HelloWorld.html.
 - Add this text in a paragraph tag. Place it between the header and the table:

Voltaire, a French philosopher and historian who died shortly after the American revolutionary war, left us with this warning about translations: "Woe to the makers of literal translations, who by rendering every word weaken the meaning! It is indeed by so doing that we can say the letter kills and the spirit gives life." Despite his warnings, we bring you four translations of the phrase "Hello World".
2. Add Id of "quote" to the paragraph tag.
3. Open main.css.
4. Add a new ruleset for the id "quote" (remember to put "#" in the front of the id) with:
 - font-size of 0.75em
 - text-indent of 4em
 - line-height of 2em.
5. Add a text shadow to the header text.
(h1 { text-shadow: 2px 2px 8px #C0C0F3 })
6. Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture.



7. Add, Commit and push the changes to GitHub.

Exercise 17

After this exercise, you should be able to:

- Understand the difference between margins, padding, and borders
- Explain the CSS Box Model
- Understand the difference between `` and `<div>` tags
- Use float to make elements snap to the side of the screen

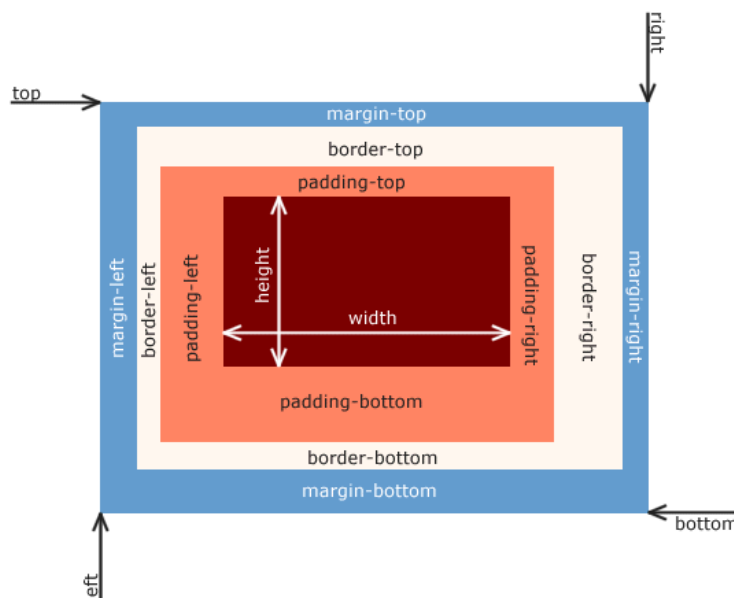
Example

This course uses CSS for relatively simple layouts. As you start to use more advanced layout techniques, you must understand the **CSS Box Model**.

The box model follows these rules:

1. The height and width specified in a CSS rule apply to the content itself.
2. All content is surrounded by a border, which can be invisible.
3. **Padding** represents the space between the border and the content.
4. **Margin** represents space outside the border, which pushes other content away.

Here's an illustration from [the Wikipedia page](#) on the topic:



The CSS box model attributes are margin, border, padding, height, and width. And more specific versions of these attributes exist: margin-top, padding-right, etc.

It's common to use a <div> tag to group elements together into a box. And you can modify the box with CSS like this:

```
div {  
  width: 500px;  
  padding: 5px;  
  border: 2px solid hotpink;  
  margin: 15px;  
}
```

A <div> is like a that applies to a whole block of content.

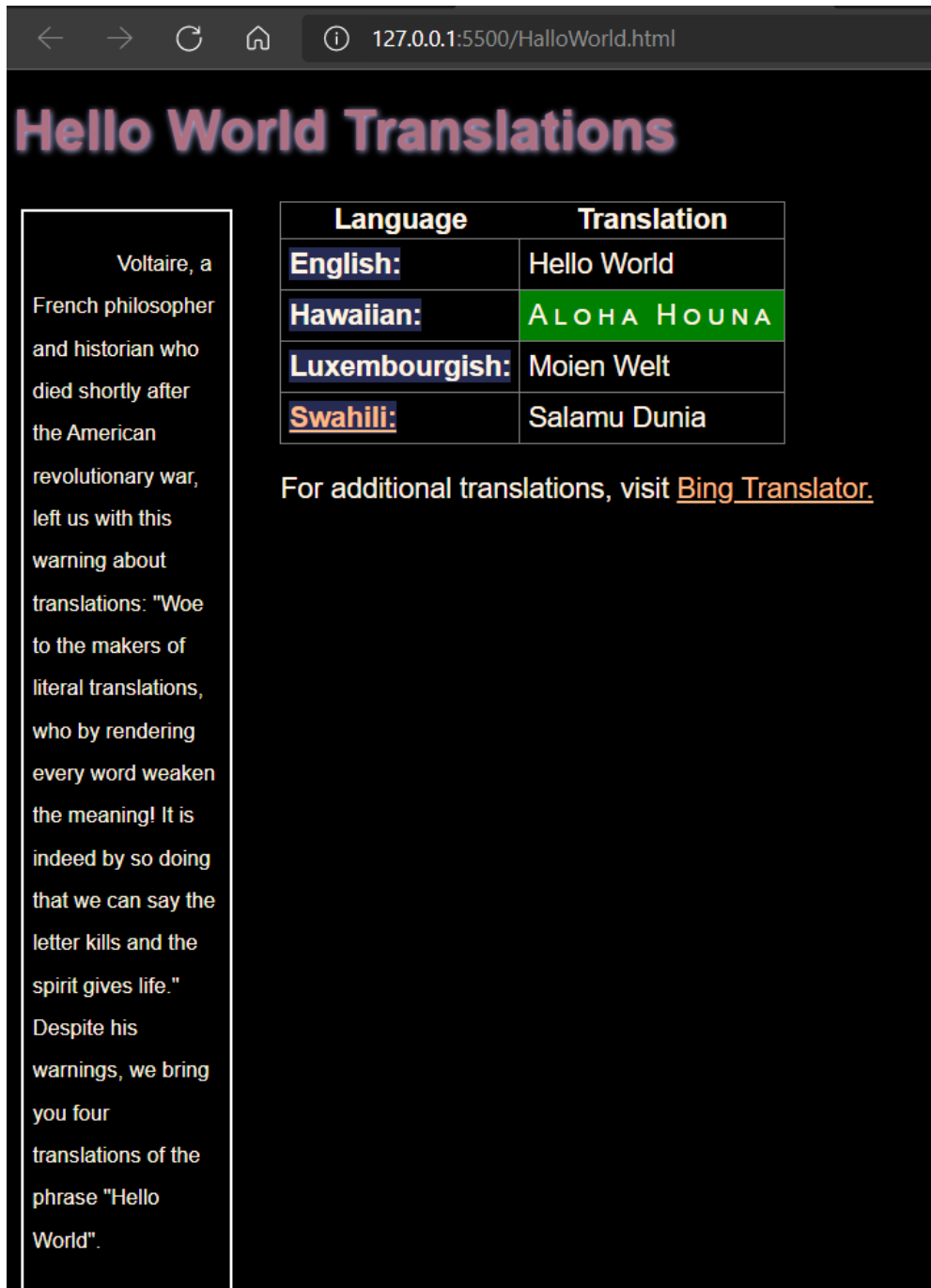
Exercise

1. Open HelloWorld.html.
 - Add <div> tags around the <p> tags with the introductory text.
2. Open main.css.
3. Add a new CSS ruleset with the div selector and the following attributes:
 - Set width to 240px.
 - Set padding to 10px.
 - Set border to 5px solid white.
 - Set margin to 10px.
 - Set margin-right to 60px to override the right margin.
 - Notice that the new styling pushes the table far down. To resolve this, add a div class selector called float-left:

```
div.float-left {  
  float: left;  
}
```

- This class selector applies only to float-left classes assigned to <div> tags.
4. Open HelloWorld.html.

- Add class="float-left" to your <div>.
5. Click the Save button to save your changes. Open the HelloWorld.html you should see same result as the picture.



6. Add, Commit and push the changes to GitHub.

Exercise 18

After this exercise, you should be able to:

- Use CSS to improve the look of your profile

Example

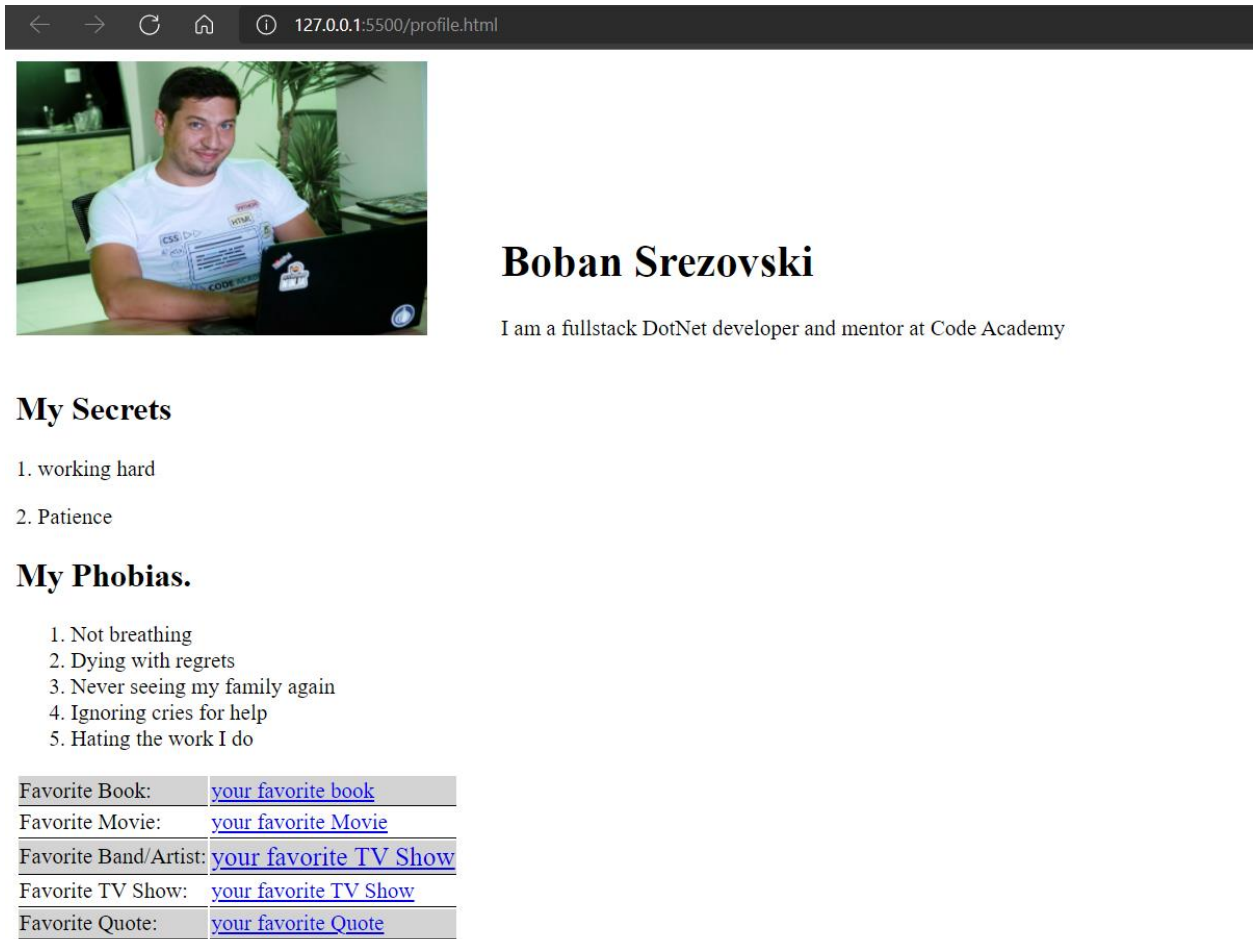
Your profile was pretty bomb after you updated it last, but it looked a little bare. Let's use CSS to style it up a bit!

Exercise

For this exercise, we'll provide suggestions but you can style your profile anyway you wish.

1. Create a file named, "profile.css."
2. Open "profile.html"
 - Change the <link> to your profile.css file.
3. Using a mix of HTML and CSS, consider making the following modifications:
 - Align your profile image to the left.
 - Force your profile image into a square shape.
 - Place your name heading and your "about me" paragraph into a single <div>.
 - Place this <div> right next to your profile image.
 - Add left margin to your <div> to give your profile image some room.
 - Add borders to your table leaving only a single dividing line between each entry.
 - Alternate the background color of each table row using an [nth-child selector](#)
 - Use the :hover [pseudo-class](#) to increase the size of each of your fears as the user mouses over them!

- Click the Save button to save your changes. Open the Profile.html you should see same result as the picture.



- Add, Commit and push the changes to GitHub.
- Have fun, play around, and make your profile spectacular!