Getting Started in Web Design

Learn by doing step by step exercises.

Includes downloadable class files that work on Mac & PC.

EDITION 6

noble desktop

Published by:
Noble Desktop
185 Madison Ave, 3rd Floor
New York, NY 10016
nobledesktop.com

Publish Date: 3-21-2022

Table of Contents

Before You Start, You'll Need to Get Some Files

These instructions tell you how to install the class files you'll need to go through the exercises in this workbook.

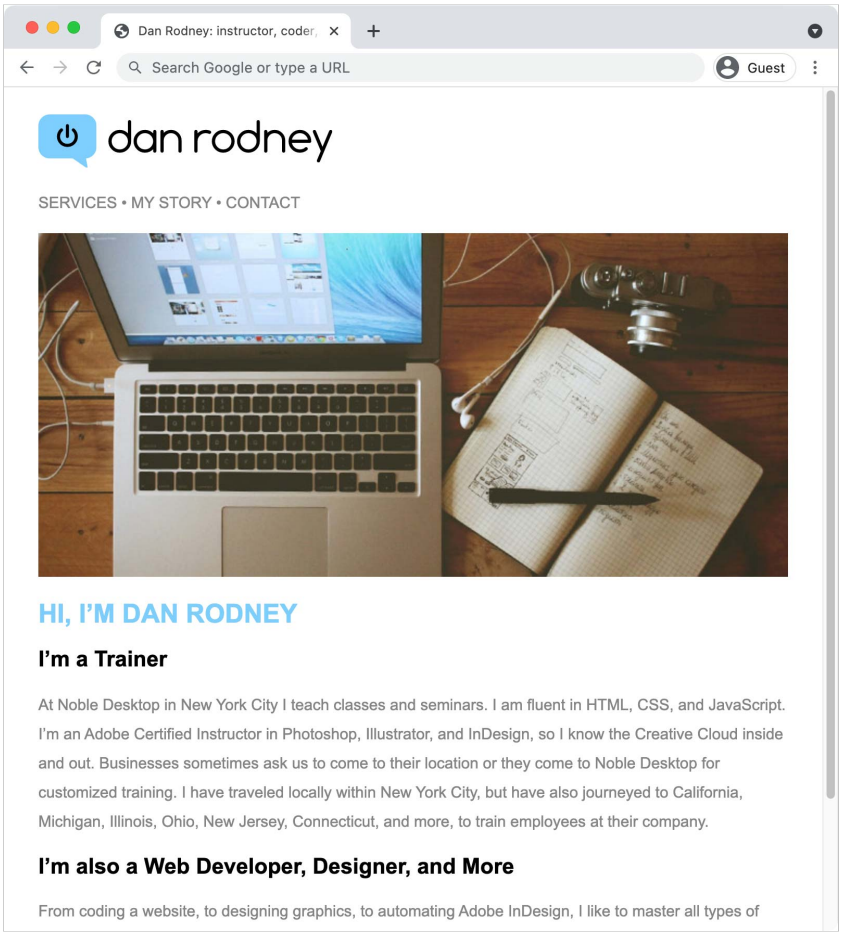
Downloading & Installing Class Files

1. Navigate to the **Desktop**.
2. Create a **new folder** called **Class Files** (this is where you'll put the files).
3. If you have the class files that were included when you downloaded this exercise (a **Getting Started in Web Design Class Files** folder) skip to the next numbered step.

   If you don't have the class files, download them as follows:

   - Go to **nobledesktop.com/download**
   - Enter the code **gswd-2006-30**
   - If you haven't already, click **Start Download**.
   - After the **.zip** file has finished downloading, unzip the file if it hasn't been done for you.
4. You should have a **Getting Started in Web Design Class Files** folder. Drag it into the **Class Files** folder you just made. These are the files you will use while going through the workbook.
5. If you still have the downloaded .zip file, you can delete that. That's it! Enjoy.

Exercise Preview



Exercise Overview

In this exercise you'll create a 2-page webpage with text and images, as you learn the foundations of how to code HTML and CSS.

Getting a Code Editor

You need a code editor to write your code in. If you prefer a specific code editor, feel free to use it. If you've never coded before, we recommend **Visual Studio Code**, which is a great free code editor for Mac and Windows.

1. Go to **code.visualstudio.com**
2. Download and install **Visual Studio Code**.

Getting Started

1. Launch **Visual Studio Code** (or your preferred code editor).
2. Hit **Cmd–O** (Mac) or **Ctrl–O** (Windows) to open a file.
3. Navigate to the **Desktop** and go into the **Class Files** folder, then **Getting Started in Web Design Class Files** folder, then **Dan Rodney Website**.
4. Double-click on **index.html** to open it.

   NOTE: **index.html** is a special filename reserved for the first page (the homepage) of a website. When you go to a website in a browser, the **index.html** is the first page that will be displayed.

5. We've typed out some text, but it doesn't have any HTML tags yet. Keep the following in mind as you start coding:

   - When you begin typing a tag, the code editor may display a list of suggested tags. For now, we recommend ignoring the code hints for opening tags.
   - When you finish typing an opening tag, some code editors will automatically create a closing tag. If it's not in the correct place, you can either delete it (and retype it where you want) or cut/paste it where you want it to be.
   - When you type **<** and **/** most code editors will automatically type the complete closing tag for you. This is a great time-saver but double-check your code to make sure that the tags are closed correctly.

   Add the following required tags (the code you need to type is highlighted in bold throughout this book):

```
<html>
<head>
   <title>Dan Rodney: instructor, coder, designer</title>
</head>
<body>
SERVICES • MY STORY • CONTACT
```

TEXT OMITTED TO SAVE SPACE

```
Make Book Jacket, Proper Fraction Pro, and Widow Fixer.
</body>
</html>
```

   NOTE: All HTML tags sit between a less-than sign (<) and a greater-than sign (>). Most tags "wrap" around the content they describe, which means there's a start tag and an end tag. The end tag has a slash character (/) preceding the tag name.

6. Let's break this code down. Notice that the **<html>** tag wraps around the entire document.

   Inside that tag, the content is split into two sections: the **<head>** and the **<body>**. The **<head>** contains information about the document, such as its title and other data that the visitor to the site will not interact with, whereas the **<body>** contains the document's actual content that a visitor will see and interact with.

   The **<title>** is meant to be an accurate and concise description of a page's content. Because it sits nested inside the **<head>** tag, it is indented. This helps to show the structure of content on the page. The **<title>** usually includes the company name and then something more specific about the document itself. Creating a concise, descriptive title is one of the most important steps of Search Engine Optimization (SEO), as the title plays a major role in most search engines' ranking scheme. Titles do not appear along with the content that shows up in the heart of the browser but they do at the top of the browser window in the "title bar." Titles also appear in most search engine results and in visitor's bookmarks.

7. Let's see what this page looks like in a browser. Note that when you preview this file, the text will be one big blob of text because we haven't formatted it yet. Before we preview, we need to save the file. Hit **Cmd–S** (Mac) or **Ctrl–S** (Windows).

8. Keep this file open in your code editor, but switch to your **Desktop** (called **Finder** on the Mac).

9. On the Desktop, go into the **Class Files** folder, then **Getting Started in Web Design Class Files** folder, then **Dan Rodney Website**.

10. **Ctrl–click** (Mac) or **Right–click** (Windows) on **index.html**, go to **Open with** and select your favorite browser (such as **Google Chrome**, **Safari**, **Firefox**, etc.).

11. Take a look at the page. The actual content of the page—which is what is wrapped inside the **<body>** tag—looks like one long paragraph of text. That's because line breaks in the code are ignored by web browsers.

    NOTE: Some browsers may display special characters wrong. For example, you see weird characters instead of the apostrophes (') and bullets (•). We'll fix them later.

12. Next, take a look at the title of the document in the browser's title bar. (It may appear at either the very top of the browser or on a tab.) It should say **Dan Rodney: instructor, coder, designer**.

13. Leave **index.html** open in the browser as you work, so you can reload the page to see the changes as you make them in the code.

14. Return to **index.html** in your code editor.

Headings, Subheadings, and Paragraphs

Even though there appear to be paragraphs in the provided text, the browser doesn't know where the paragraphs start and end and where to add space. We have to code that.

Some of the paragraphs are actually headings, which help organize the content semantically so visitors can quickly skim a page. Headings help search engines like Google to better understand what the page's content is about. There are six levels of headings from **h1** (the most important) to **h6** (the least important). Headings also make your website more accessible. Visually impaired visitors using a screen reader can hear the headings and jump between them using a keystroke.

1. In your code editor, at the top of the **body** section, add the following tags highlighted in bold:

```
<body>
<p>SERVICES • MY STORY • CONTACT</p>
```

   NOTE: Later we'll make these three items into links so they'll be a navigation bar for the site.

2. Save the file.
3. Return to the browser and click the **Reload** button to refresh the browser window with the new code.
4. Notice the text (at the top of the page) is now on its own line with space below it, forming a single-line paragraph.
5. Return to **index.html** in your code editor.
6. Next we have a main heading and a subheading. Add the following tags highlighted in bold:

```
<body>
<p>SERVICES • MY STORY • CONTACT</p>

<h1>Hi, I'm Dan Rodney</h1>

<h2>I'm a Trainer</h2>
```

7. Save the file.
8. Return to the browser and click the **Reload** button to refresh the browser window.
9. Notice that headings are rendered as bold, and more important headings are larger than less important ones. This appearance is only a default. Size, color, and more can be changed using Cascading Style Sheets (CSS) which you'll learn about later.
10. Return to **index.html** in your code editor.

11. For the first long paragraph, add the following opening and closing paragraph tags highlighted in bold. (Don't miss the closing tag at the end of the paragraph!)

```
<h2>I'm a Trainer</h2>
<p>At Noble Desktop in New York City I teach classes and seminars. I am fluent
in HTML, CSS, and JavaScript. I'm an Adobe Certified Instructor in Photoshop,
Illustrator, and InDesign, so I know the Creative Cloud inside and out.
Businesses sometimes ask us to come to their location or they come to Noble
Desktop for customized training. I have traveled locally within New York City,
but have also journeyed to California, Michigan, Illinois, Ohio, New Jersey,
Connecticut, and more, to train employees at their company.</p>
```

12. Below that paragraph we have another subheading, so add the following tags highlighted in bold:

```
<h2>I'm also a Web Developer, Designer, and More</h2>
```

13. Below that we have one remaining paragraph. Add the following tags highlighted in bold. (Don't miss the closing tag at the end of the paragraph!)

```
<p>From coding a website, to designing graphics, to automating Adobe InDesign,
I like to master all types of things. This diversity has been beneficial to me
and my clients in unexpected ways. For example, I learned JavaScript for
coding webpages, but was later able to use it to automate Adobe InDesign. I've
used it to save time and money for Noble Desktop, as well as create my own add-
ons such as Make Book Jacket, Proper Fraction Pro, and Widow Fixer.</p>
```

14. Save the file.

15. Return to the browser and **reload** the page to see the text with one main heading, two subheadings, and some paragraphs.

Adding a Doctype & Standard Meta Tags

1. Return to **index.html** in your code editor.

2. Place the cursor at the very beginning of the code—directly before the **html** tag—and hit **Return** (Mac) or **Enter** (Windows) to get a new line on top.

3. Now add the following code highlighted in bold:

```
<!DOCTYPE html>
<html>
```

This **document type definition** (DTD) ensures that browsers render the page according the latest HTML standards.

4. Next, add the following attribute (highlighted in bold) to the **html** tag:

```
<!DOCTYPE html>
<html lang="en">
```

Notice that **lang="en"** is written inside the **html** start tag, after the tag name but before the greater-than sign. The **lang** attribute declares the language of the page and **en** is the code for English. This helps make your content accessible, so screen readers can can properly speak the page's content to someone who's vision impaired.

> **Attributes**
>
> Many HTML tags can be enhanced with **attributes**, modifiers of HTML elements. Attributes are written inside the HTML element's start tag.
>
> Attributes have a name and a value. The name precedes the equal sign and the value is wrapped in double quotes.

5. There are two more tags that we must add for this page to work properly. Add the following two new lines of code highlighted in bold:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title>Dan Rodney: instructor, coder, designer</title>
</head>
```

Meta tags can do various things:

- The **charset** meta tag says that special characters are encoded as Unicode (UTF-8). This means that special characters (like •, ©, ™, etc.) can be typed into the code and they'll display properly across various platforms and devices.
- The **viewport** meta tag ensures that smaller devices such as mobile phones and tablets will display the page appropriately for their size (instead of rendering the page like a desktop and scaling it to fit their screen).

6. Notice that, unlike most of the other tags we have used so far, both the doctype and the meta tag are not closed. This is because these tags do not wrap around content. Instead, they have a predefined behavior of their own.

7. Save the file.

8. Return to the browser and **Reload** the page.

   Depending on your browser, you may not see much change. If your browser was previously showing weird characters (instead of bullets and apostrophes), you should now see the proper characters!

Adding Images

Images are inserted into the HTML document with a single piece of code. Web graphics must be saved as either JPEG, SVG, PNG, or GIF.

1. Return to **index.html** in your code editor.
2. To add a logo at the top of the page, add the following line of code highlighted in bold:

```
<body>
<img src="images/logo.svg" alt="Dan Rodney Logo">
<p>SERVICES • MY STORY • CONTACT</p>
```

3. Take a moment to review the code you just typed. The **<img>** tag requires the **src** attribute (an abbreviation of source) to call the appropriate image file. Images are typically stored in a subfolder of the website. The **Dan Rodney Website** has them in subfolder named **images**.

   The img tag's **alt** attribute is more commonly referred to as **alt text**. It provides "alternate" plain text content that describes the image. It's used by screen readers, search engines, and is displayed if the graphic does not load.

4. Save the file.
5. Return to the browser and reload the page. You should see a **Dan Rodney** logo at the top of the page.

   NOTE: Images are not actually embedded into an HTML page. The image tag is a placeholder for the linked source file and, therefore, images must be uploaded along with the HTML pages to your remote web server in order for visitors to see them.

   When uploading the website to make it live, some web servers might not display the SVG file until you configure them properly. That's beyond the scope of this exercise, but we wanted to mention it in case an SVG file works locally (on your computer) but not on the live website (web server).

6. Return to **index.html** in your code editor.

7. Insert another image by adding the following line of code highlighted in bold:

```
<body>
<img src="images/logo.svg" alt="Dan Rodney Logo">
<p>SERVICES • MY STORY • CONTACT</p>
<img src="images/home.jpg" alt="Laptop, Sketches, and Camera">

<h1>Hi, I'm Dan Rodney</h1>
```

8. Save the file.

9. Return to the browser and reload the page to see the photo.

   Congratulations, you've made your first webpage!

Styling Content with CSS

While HTML defines the type of content on a page (heading, paragraph, image, etc.), Cascading Style Sheets (CSS) tells the browser how that content should be formatted. Because CSS styles things behind the scenes, it is written in the **<head>** section of the document rather than in the **<body>**.

1. Switch back to **index.html** in your code editor.

2. Add the following bold code in the **head** section, just below the **title** tag:

```
   <title>Dan Rodney: instructor, coder, designer</title>
   <style>

   </style>
</head>
```

3. We tell the browser where to apply styling with a CSS selector. For example, a tag selector will find all instances of a particular HTML tag and assign some styling. Create an **h1** tag selector by adding the following bold code:

```
<style>
   h1 {

   }
</style>
```

   This rule will target all the h1's on the page.

4. Inside the h1 tag selector add the following bold code:

```
h1 {
   font-size: 25px;
   color: #7dcefc;
}
```

   The list of styling options in the curly braces {} are called **property declarations**. They tell a browser which predefined CSS properties we want to change (font-size, color, etc.) and what value we want to use (25px, #7dcefc, etc.). We often get those values from a design mocked up Figma, Adobe XD, Sketch, etc. You can copy the color value (a hex color code) from the design app's color picker.

5. Save the file.

6. Return to the browser and reload the page.

7. Notice that the heading **Hi, I'm Dan Rodney** is slightly smaller and is now blue.

8. Return to **index.html** in your code editor.

9. Make the heading all caps by adding new code that is highlighted below in bold:

```
h1 {
   font-size: 25px;
   color: #7dcefc;
   text-transform: uppercase;
}
```

10. Inside the **style** tag, add two new rules below the **h1** rule. Type only the new code highlighted below in bold:

```
   h1 {
      font-size: 25px;
      color: #7dcefc;
      text-transform: uppercase;
   }
   h2 {
      font-size: 21px;
   }
   p {
      font-size: 15px;
      color: #858585;
      line-height: 28px;
   }
</style>
```

    NOTE: The **line-height** property defines how tall lines are, and is called **leading** in print design applications.

11. Save the file.

12. Return to the browser and reload the page again to see the subheads and text are slightly smaller and the paragraphs are now gray.
13. Return to **index.html** in your code editor.
14. We want to change the font of all the text. Instead of adding a property to each of our three existing rules, we can style the element that contains all the text. Everything inside the container will inherit that setting! All the text is in the **body** tag, so we can style that. Add the new code highlighted below in bold:

```
<style>
   body {
      font-family: Arial, Helvetica, sans-serif;
   }
   h1 {
```

> **The Font-Family Property**
>
> The **font-family** properties is actually a list of fonts—a "wish list" of sorts. The first font will be applied if it's available on the user's computer. The second, third, or any following fonts will only be used if the preceding font is not available. This list is called the font "stack". If you wish to use custom fonts that are not typically pre-installed on computers, you would have to load them (which we won't be covering in this introduction).

15. Save the file.
16. Return to the browser and reload the page and notice the following:
    - All the text should now be a sans-serif font.
    - Resize the browser window smaller than the image to see the image remains the same size, forcing the page to scroll when the image is wider than the window. Ideally the image would scale down to fit on small screens such as tablets and phones.
17. Return to **index.html** in your code editor.

18. We can make a style to ensure all images never be larger than their container. Add the new code highlighted below in bold:

```
   p {
      font-size: 15px;
      color: #858585;
      line-height: 28px;
   }
   img {
      max-width: 100%;
   }
</style>
```

19. Save the file.

20. Return to the browser and reload the page.

21. Resize the browser window's width smaller and larger, and notice:

    - The image scales down to always fit inside the window. It does not not scale up to fill the window though, as that could lead to a pixelated low-resolution image on large screens.
    - When the window is wide, the text lines get very long and hard to read. It would look better is there was a limit on how wide they can get.

22. Return to **index.html** in your code editor.

23. Everything is in the body tag. If we limit the width of the body, all the content inside will wrap (or shrink) to fit. Add the new code highlighted below in bold:

```
body {
   font-family: Arial, Helvetica, sans-serif;
   max-width: 720px;
}
```

24. Save the file.

25. Return to the browser and reload the page.

26. Resize the window so it's wider than the photo.

27. Notice the text lines no longer fill the entire width of the window. They are limited to a width that matches the photo above.

    That's better, but all the leftover space is being added to the right. It would look better if the content was centered.

28. Return to **index.html** in your code editor.

29. Add the new code highlighted below in bold:

```
body {
   font-family: Arial, Helvetica, sans-serif;
   max-width: 720px;
   margin: auto;
}
```

30. Save the file.

31. Return to the browser and reload the page.

32. Resize the window from small to wide and notice the following:

    - When the window is wider than the photo and text, the content is now in a column that is centered!
    - When the window is narrow, the text will touch the edge of the window. That doesn't look very good and we'll fix that next.

    So what is margin and why does this work? Margin is space outside an element. Depending on the window's width, the space outside the body may be zero (on small screens) or a lot (on large screens). Setting margin to auto tells the browser to automatically figure out how much margin is needed, and set the left and right margins to be the same (effectively centering the element). It's not the most intuitive bit of code, but that's how we center containers!

33. Return to **index.html** in your code editor.

34. **Margin** is space **outside** an element. **Padding** is space **inside** an element. We've already added margins to the body tag, but we can still add padding! Add the new code highlighted below in bold:

```
body {
   font-family: Arial, Helvetica, sans-serif;
   max-width: 720px;
   margin: auto;
   padding: 25px;
}
```

35. Save the file.

36. Return to the browser and reload the page.

37. Resize the window wide and then narrow to see the page looks pretty good at any size!

38. We're done with this page for now. Close the browser window so you won't get confused moving forward.

Creating a Second Page

1. Return to **index.html** in your code editor.
2. Do a **File > Save As**.
3. If you're not already in the **Dan Rodney Website** folder, navigate to **Desktop > Class Files > Getting Started in Web Design Class Files > Dan Rodney Website**.
4. Name the file **contact.html** and save it into the **Dan Rodney Website** folder.
5. We have a different image for this page. Change the home.jpg image's **src** and **alt** text as shown below in bold:

```
<p>SERVICES • MY STORY • CONTACT</p>
<img src="images/contact.jpg" alt="Dan teaching a seminar">
```

6. Delete all the headings and paragraphs below that image, and replace it with a single paragraph as shown below in bold:

```
<body>
<img src="images/logo.svg" alt="Dan Rodney Logo">
<p>SERVICES • MY STORY • CONTACT</p>
<img src="images/contact.jpg" alt="Dan teaching a seminar">

<p>Please email me. I am lonely!</p>
</body>
```

7. Lastly, the title needs to be updated for this page. In the **head** section at the top, edit the title as shown below in bold:

```
<title>Contact Dan Rodney</title>
```

8. Save the file.
9. Keep this file open in your code editor. To preview the contact page in a browser:
   - Switch to your **Desktop** (called **Finder** on the Mac).
   - On the Desktop, go into the **Class Files** folder, then **Getting Started in Web Design Class Files** folder, then **Dan Rodney Website**.
   - **Ctrl–click** (Mac) or **Right–click** (Windows) on **contact.html**, go to **Open with** and select your favorite browser (such as **Google Chrome**, **Safari**, **Firefox**, etc.).

   This page should look link the homepage, but with a different photo and a single paragraph below it.
10. We're done with this page for now. Close the browser window so you won't get confused.

Linking Between Pages

Now that we have a second page, we need to give users a way to get to it. We'll add a link in the navigation bar at the top of the page.

1. Switch back to your code editor.
2. We need to re-open **index.html**. Hit **Cmd–O** (Mac) or **Ctrl–O** (Windows).
3. Navigate to **Desktop > Class Files > Getting Started in Web Design Class Files > Dan Rodney Website** and double-click on **index.html** to open it.
4. In the first paragraph, add the following code (highlighted in bold) around **CONTACT**:

   ```
   <p>SERVICES • MY STORY • <a href="contact.html">CONTACT</a></p>
   ```

   The markup tag for links is the **<a>** tag, which stands for anchor. An anchor tag doesn't do anything on its own but is quite powerful when paired with its **href** attribute. Href stands for hyperlink reference. Its value is a website address (such as https://www.google.com) or a file path to the page you'd like to link.

   This type of link is relative to the page that it is sitting in. In this case, the link will look for a page called **contact.html** in the same folder as our file (**index.html**).
5. Save the file.
6. Preview **index.html** in your browser. If you don't remember how to preview, do the following:
   - Switch to your **Desktop** (called **Finder** on the Mac).
   - On the Desktop, go into the **Class Files** folder, then **Getting Started in Web Design Class Files** folder, then **Dan Rodney Website**.
   - **Ctrl–click** (Mac) or **Right–click** (Windows) on **index.html**, go to **Open with** and select your favorite browser (such as **Google Chrome**, **Safari**, **Firefox**, etc.).
7. Near the top of the page, notice that CONTACT now has an underline and is a different color than the other paragraph text.
8. Click on **CONTACT** and you should go to the contact page!

   While currently the back button would take us back to the homepage, if someone lands on this page (for example from a search engine) we need a way for them to get to the homepage. Normally the website's logo links to the homepage, so let's create that link next.
9. Keep this contact page open in the browser so we can come back to it in a moment.
10. Go back to your code editor.
11. Switch to **contact.html** in your code editor.

12. Find the first image (the logo). Add the following bold code around that image:

```
<body>
<a href="index.html"><img src="images/logo.svg" alt="Dan Rodney Logo"></a>
<p>SERVICES • MY STORY • CONTACT</p>
```

13. Save the file.
14. Return to the browser and reload the contact page.
15. Click the **Dan Rodney** logo and you should be taken to the homepage.
16. Click the **Contact** link in the navigation bar to go to the contact page.
17. Click the browser's back button to go back to the homepage.

    We don't like the default appearance of the contact link, so let's change it.

Styling Links

1. Switch back to your code editor.
2. Switch to **index.html** in your code editor.
3. Below the other CSS styles, add a new rule for anchor tags (the tag we used to make a link). Add the following bold code:

```
   img {
      max-width: 100%;
   }
   a {
      color: #f5721a;
      text-decoration: none;
   }
</style>
```

4. Save the file.
5. Return to the browser and reload the homepage.
6. The contact link should be orange and no longer have an underline.

   Congratulations on finishing your first multi-page website!

Noble Desktop Training

Learn live online or in person in NYC

Front-End Web Development
Full-Stack Web Development
JavaScript
Python
Software Engineering
Data Science & Data Analytics
SQL
WordPress

Motion Graphics & Video Editing
Adobe Premiere Pro
Adobe After Effects
InDesign, Illustrator, & Photoshop
Web, UX, & UI Design
Figma, Adobe XD, & Sketch
Digital & Social Media Marketing
and much more...

---

nobledesktop.com