W03 Rafting Project: About Us Page

Overview

Build the first page of the rafting website project, "**About Us**". This page provides contact information, history, and other promotions for the company. A wireframe for this page's required layout and content areas is provided.

Instructions

Step 1: Folder and File Setup

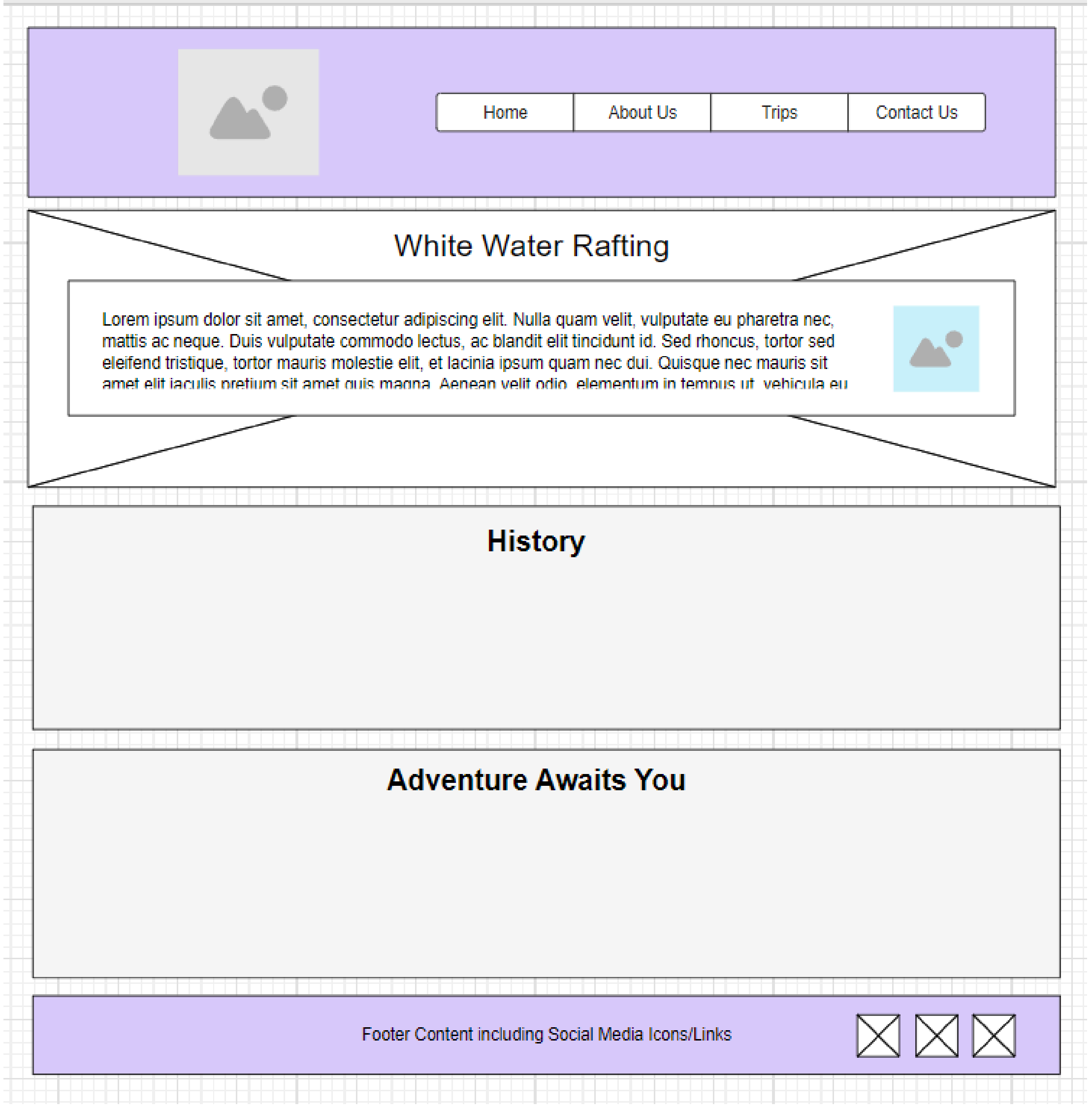
1. In VS Code, be sure you have your **wdd130** directory open and **add** a sub-folder named "**wwr**" (wwr stands for white water rafting). This folder will contain the rafting website project and you will add pages and assets to this folder for the rafting project throughout the course.
2. Add two folders *within* this **wwr** folder per course naming conventions to contain the images, "**images**", and the stylesheet, "**styles**".

   Do not be confused that you already have root level folders named *images* and *styles*. The new folders are for your rafting website project and are contained within the **wwr** project folder.

3. Add a file named "**about.html**" in the **wwr** directory.

Step 2: Design: Study the Wireframe

1. Refer to this wireframe sketch as you markup the basic structure of the **about.html** page.


Home
About Us
Trips
Contact Us
White Water Rafting
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu
History
Adventure Awaits You
Footer Content including Social Media Icons/Links

Wireframe Design for Rafting Project About Us Page

You do **not** need to complete the header and footer horizontal layouts (i.e., the navigation and social media icons) during this assignment.

Step 3: Develop the Page Structure: HTML

1. In your **about.html** document, begin your HTML page by writing the standards based basic HTML structure with the **`head`** with its standard, required content and an empty **`body`**.

   Be sure to include a proper **`meta`** description and **`meta`** author elements along with all the other required **`head`** content.

2. Using the typography choices you made in your rafting site plan, provide the proper **Google Font `link`** references to the fonts and font styles that you plan on using.

   ▶ Check Your Understanding

3. In the **`body`**, add the following three main elements: **`header`**, **`main`**, and **`footer`**.
4. The **`header`** contains two items:

1. A rafting site **logo** that you have designed and built or that you have selected from the choices provided below. A sample logo was given in the rafting site plan link.

   Here are additional logos you can choose from if needed: WWR Sample Logos

   Where should this logo and all images used on the site be stored?

   ▶ Check Your Understanding

2. A **nav** element with four child **<a>** tags and labeled as shown. The **href** attribute values, the page references, can be set now even though the pages do not all exist yet.
   - ✔ index.html (the future home page of wwr, not to be confused with the course home page.)
   - ✔ about.html
   - ✔ trips.html
   - ✔ contact.html

Do NOT worry about the horizontal layout shown in the wireframe at this point.

5. The **main** element should contain the following three(3) elements:
   1. A **div** with a class of **hero** that contains these items:
      - ✔ A hero image **img** that will be in the entire background of the **div** background.

        Image Resources
        Course Rafting Image Repository
        Web Frontend Resources – see Images and Graphics section.

        All images used on your site must be optimized which means not pixelated and less than or equal to 100 kB in size.

      - ✔ A **h1** element that has the title of the rafting company as the text content.
      - ✔ An **article** element that contains:
        - An smaller **img** element that portrays a happy client or happy, working employee.
        - A **p** element with the company purpose, mission, creed, motto, etc. It is Ok to use non-sense, placeholder language for now.
   2. A **section** element with the following contents:
      - ✔ A **h2** element with a section titled "**History**".
      - ✔ A **p** element with a brief history of the company. It is OK to use nonsense language.

3. A `section` element with the following contents.
   - ✔ A `h2` element with a section titled "**Adventure Awaits You!**".
   - ✔ A series of five (5) `img` elements representing white water rafting.

     > Do not worry about the layout of the images. Later you will use CSS to layout the images.

   ▶ Check Your Understanding

6. A `footer` element that contains:
   - ✔ A **p** paragraph element containing a copyright symbol, the year, the rafting company name, and your name.
   - ✔ Three (3) **img** elements that are encased by **a** anchor elements pointing to three different social media outlets. The links can be generic.

     ▶ Example

     > **Social Media Icon Resources**
     > [iconfinder.com]
     > [Google Icons]

Step 4: Write the CSS

1. Create a CSS file for your rafting site named "**rafting.css**" and store it in the **styles** folder within the **wwr** folder.

   > Remember to add a link to this stylesheet in the **<head>** of your **about.html** page.
   >
   > ▶ Check Your Understanding

2. Define the CSS variables in the document `:root` pseudo-class selector using the color scheme you selected and documented in this week's site plan/graphic identity assignment.
3. Begin by styling the headings and paragraph elements.
4. Add colors to the fonts.
5. Remove the underlines from the social media image links. Allowing the default underline on a hyperlink that is an image is not a good design and may lead to confusion of function.

   > To remove the default style of having underlines under **<a>** hyperlinks, use the CSS declaration
   >
   > ```
   > text-decoration: none;
   > ```

However, the CSS rule in which you use this declaration cannot just be all hyperlinks. Your future navigation links and other hyperlinks may still want to use underlines

```
/* The following CSS selector (rule) removes underlines from all hyperlin
a  {
  text-decoration: none;
}
```

Use a specific selector that only selects the hyperlinks within the social media series of image hyperlinks.

6. Consider limiting the entire width of the page to make it easier to design the content.

Step 5: Style the Hero Container using Position Absolute

1. The **`div`** with the class of **`hero`** is positioned **`relative`** in order the **`h1`** and the **`article`** elements to be positioned on top of the hero image using **`absolute`** positioning.
2. The hero image fits within the allocated space.

   Use a relative width CSS declaration for the hero image and make the image a block to fill the width of the container.

   ▶ Example

3. The **`h1`** heading is positioned absolutely.
4. The **`article`** is positioned absolutely.
5. The **`img`** inside the article floats to the right of the **`p`** paragraph.
6. Consider styling the background-color and opacity of the **`article`** to fit the design and make sure it is readable.

Click here to see a sample of what the page may look like at this point.
Your page will have different colors, fonts, content, and images than the example page, but at this point it should have a similar layout.

Step 6: Testing

1. Every page in this course will be expected to pass the development standards checklist.
2. **Validate** and correct any errors with your HTML and CSS using the Web Developer browser extension under Tools and Validate Local HTML and CSS.
3. Test your work continuously as you work through the steps by having your page loaded in your local browser using **Five Server**.

You do **not** need to be connected to the internet while using Live rServer to test your pages locally during development.

4. Commit your changes and sync them to your wdd130 GitHub Pages enabled repository.
5. Audit your page using the ✔ page evaluation tool to verify that you have the basic document content.
6. Make corrections as needed and be sure to recommit and sync your updated work.

Submission

1. Submit your GitHub pages URL (website address) in Canvas.
   Here is the sample URL. You will substitute your GitHub username for *githubusername*.

```
https://githubusername.github.io/wdd130/wwr/about.html
```