Front End Web Development

7/20/2022

can code communities

Updates

- Recipe Site projects have been returned for some and the rest are queued for the end of week

Responsive HTML

- Responsive web design is about creating web pages that look good on all devices.

- A responsive web design will automatically adjust for different screen sizes and viewports

- We will be using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones)

What is the Viewport?

- The viewport is the user's visible area of a web page.

- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

- Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

Setting the Viewport

- HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag

- To have a responsive page add the following <meta> tag to your HTML head

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Setting the Viewport

- This gives the browser instructions on how to control the page's dimensions and scaling.

- The width=device-width sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

- The initial-scale=1.0 sets the initial zoom level when the page is first loaded by the browser.

CSS Flexbox Layout

Before the Flexbox Layout module, there were four layout modes:

- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure.

Flexbox Layout

- The Flexbox Layout aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is dynamic.

- Flex layout gives the container the ability to alter its items width, height, and order to best fill the available space

- Flex layout is direction-agnostic (its free from any directional constraints) whereas block layout vertically biased and inline layout is horizontally biased

Flexbox

To start using the Flexbox model, you need to first define a flex container.

A flex container with three flex items:

```
14  <div class="flex-container">
15    <div>1</div>
16    <div>2</div>
17    <div>3</div>
18    <div>4</div>
19  </div>
```

Flexbox

The flex container becomes flexible by setting the display property to flex:

```
16  .flex-container {
17    display: flex;
18  }
```

Styling divs inside of divs

Use a “>” symbol between the class and div element to style divs inside of divs

```
8   .flex-container > div {
9     background-color: #f1f1f1;
10    margin: 10px;
11    padding: 20px;
12    font-size: 30px;
13  }
```

Exercise

1. Create 4 flex containers with 4 items inside of them
2. Style the flex containers and the divs inside of them

The flex-direction Property

The flex-direction property defines in which direction the container wants to stack the flex items.

The column value stacks the flex items vertically (from top to bottom):

```
.flex-container {
    display: flex;
    flex-direction: column;
    background-color: blue;
}
```

The flex-direction Property

The column-reverse value stacks the flex items vertically (but from bottom to top):

```
.flex-container {
    display: flex;
    flex-direction: column-reverse;
    background-color: blue;
}
```

The flex-direction Property

The row value stacks the flex items horizontally (from left to right):

```
1  .flex-container {
2      display: flex;
3      flex-direction: row;
4      background-color: blue;
5  }
```

The flex-direction Property

The row-reverse value stacks the flex items horizontally (but from right to left):

```
1  .flex-container {
2      display: flex;
3      flex-direction: row-reverse;
4      background-color: blue;
5  }
```

Exercise

1. Change the flex-direction of your flex containers
2. Use
    a. Row
    b. Row-reverse
    c. Column
    d. Column Reverse

The flex-wrap Property

The flex-wrap property specifies whether the flex items should wrap or not.

The wrap value specifies that the flex items will wrap if necessary:

```
.flex-container {
    display: flex;
    flex-wrap: wrap;
    background-color: blue;
}
```

The flex-wrap Property

The nowrap value specifies that the flex items will not wrap:

```
1  .flex-container {
2      display: flex;
3      flex-wrap: nowrap;
4      background-color: blue;
5    }
```

The flex-wrap Property

The wrap-reverse value specifies that the flexible items will wrap if necessary, in reverse order:

```
1  .flex-container {
2      display: flex;
3      flex-wrap: wrap-reverse;
4      background-color: blue;
5    }
```

Exercise

1. Create a new flex container with 10 divs inside
2. Style the flex container and the divs
3. Use:
    a. Wrap
    b. Nowrap
    c. Wrap-reverse

The flex-flow Property

The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

```
.flex-container {
    display: flex;
    flex-flow: row wrap;
    background-color: blue;
}
```

The justify-content Property

The justify-content property is used to align the flex items:

The center value aligns the flex items at the center of the container:

```
.flex-container {
    display: flex;
    justify-content: center;
    background-color: blue;
}
```

The justify-content Property

The flex-start value aligns the flex items at the beginning of the container:

```
1  .flex-container {
2      display: flex;
3      justify-content: flex-start;
4      background-color: blue;
5  }
```

The justify-content Property

The flex-end value aligns the flex items at the end of the container:

```
1  .flex-container {
2      display: flex;
3      justify-content: flex-end;
4      background-color: blue;
5  }
```

The justify-content Property

The space-around value displays the flex items with space before, between, and after the lines:

```
.flex-container {
    display: flex;
    justify-content: space-around;
    background-color: blue;
}
```

The justify-content Property

The space-between value displays the flex items with space between the lines:

```
.flex-container {
    display: flex;
    justify-content: space-between;
    background-color: blue;
}
```

Exercise

1. Create a new flex container with 10 divs inside
2. Style the flex container and the divs
3. Use:
    a. Justify-content
        i. Center
        ii. Flex-start
        iii. Flex-end
        iv. Space-around
        v. Space-between

The <form> Element

The HTML <form> element is used to create an HTML form for user input

The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the type attribute.

The one we will be using is the <input type="text"> which displays a single-line text input field

The <label> Element

The <label> tag defines a label for many form elements.

The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

Form Code

```
29  <form>
30    <label for="fname">First name:</label><br>
31    <input type="text" id="fname" name="fname"><br>
32    <label for="lname">Last name:</label><br>
33    <input type="text" id="lname" name="lname">
34  </form>
```

Exercise

- Add two forms to your page
  - One for First and Last Name
  - One for Job Title and Job Location

HTML <iframe> Tag

The <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

It is a good practice to always include a title attribute for the <iframe>. This is used by screen readers to read out what the content of the <iframe> is.

HTML YouTube Videos

Define an <iframe> element in your web page

Let the src attribute point to the video URL

Use the width and height attributes to specify the dimension of the player

```
46  <iframe src="https://www.youtube.com/embed/CIpOxa5hxOw"></iframe>
```

```
14  iframe {
15    width: 1000px;
16    height: 500px;
17  }
```

Exercise

- Add two Youtube videos to your page

Assignment

- I want you to create a personal portfolio project
- This will be a project that we add on to throughout the semester
- Think a page that represents you and your work
- Some ideas for content
    - A image of you
    - Links to social media, LinkedIn, GitHub(when we get there)
    - Examples of your work
    - Your resume