An important part of styling a webpage with CSS is organizing boundaries and space.

**1.**

**1.**

**1.**

**1.**

**1.**

**1.**

**1.**

**2.**

**3.**

**1.**

**2.**

Every page element has boundary and space properties that can be controlled using CSS. The *CSS box model* illustrates each of these properties.

Instructions

Observe the CSS box model diagram to the right:

1. *content*: Includes text, images, or other media contained within an HTML element.
2. *padding*: The space between the content and the border. You can think of this like inner space.
3. *border*: The outline of an HTML page element. You can think of it like a picture frame that contains the element.
4. *margin*: The space between the HTML page element and the next nearest element(s).

After you have familiarized yourself with the Box Model, click Next to continue.

The *border* property can be used to visually define a page element's outer edge.

In CSS, the *border* property's value requires three parts:

1. *thickness*: Sets the thickness of the border, using pixels, ems, or rems.
2. *type*: Sets the border type. Common options are solid, dotted, anddashed. There are many others.
3. *color*: sets the border's color, using named colors, HEX, or RGB values.

The CSS below gives a paragraph element a solid black border that is 2 pixels thick:

p {

border: 2px solid black;

}

The web browser currently displays Tundra Gallery's homepage. Let's give each gallery item a border.

In **main.css** locate the .gallery-item class selector. Addborder: 5px solid #FFF;, like so:

.gallery-item {

border: 5px solid #FFF;

}

Click Run to see borders around each figure with the classgallery-item in the web browser.

The CSS *padding* property controls the empty space between the page element's content and its border. Increasing a page element's *padding* makes the space around the content more spacious, while decreasing it makes the space more compact.

p {

padding: 20px;

}

The CSS above would give paragraph elements a padding of 20px.

Instructions

In **main.css**, locate the .page-description class selector and give set its padding of 30px.

Click Run to see that the paragraph element's padding has increased on the webpage.

The CSS *margin* property controls the space between different HTML elements on a webpage. Use margin to bring page elements closer together or to move them further apart.

The CSS below ensures 2rems of space between elements with the classanswer and surrounding page elements.

.answer {

margin: 2rem;

}

Instructions

In **main.css**, locate the .gallery-item class selector. Currently, it has a margin of 2px. Change the value to 20px.

Click Run to see the space increase between each gallery item.

The margin property creates space on all sides of a page element. It's also possible to set separate margin spacings on each side of an element.

Additional margin properties:

1. *margin-top*: Sets the top margin.
2. *margin-bottom*: Sets the bottom margin.
3. *margin-left*: Sets the left margin.
4. *margin-right*: Sets the right margin.

**Note**: Below we will change margin properties for a div that encloses HTML*figure* elements. Figures are used to organize visuals, such as photos and diagrams.

Instructions

In **index.html**, figures with the class gallery-item are contained inside a div with a class gallery.

In **main.css**, locate the .gallery class selector. Set the margin-topproperty to 20px.

Click Run and notice the change in the web browser: the .gallerydiv now has more top-margin space.

Using borders, padding, and margins allows us to control boundaries and space for individual HTML elements.

But what CSS properties are available to move elements around on the page and create unique page layouts? The CSS *display* and *position* properties help accomplish this.

#### DISPLAY

Not all HTML elements are displayed on a page in the same way. *Display types*determine how HTML elements will be arranged in relation to each other.

The diagram to the right illustrates the *block* and *inline* display types.

Instructions

In the diagram, notice:

1. The two dotted rectangles represent webpages.
2. HTML heading, paragraph, and unordered list elements are block level: each appears on its own line on the webpage.
3. HTML image and anchor elements are displayed inline: they appear on the same line as their neighboring elements on the webpage.

Familiarize yourself with block and inline display types, then click Next to continue.

Display types can be overwritten in CSS by using the *display* property.

For example, we can make list items appear on the same line by setting display to inline:

li {

display: inline;

}

**Note**: Below, we will encounter an HTML *nav* element. Navs are used to organize site navigation menus on a webpage.

Instructions

On the Tundra Gallery homepage, notice the navigation bar (navbar) on the bottom left.

In **index.html**, this is represented by a nav element containing a ulwith three list items. Read through the HTML and locate this code.

In **main.css** locate the nav li selector. Give it a display property ofinline.

Click Run and notice the change in the web browser. List items, which are normally displayed as block-level elements, display inline within the navigation.

Nice work! The navbar is starting to come together nicely. It would be even better if we could get the Contact button to fill in the empty corner on the bottom right.

To achieve this, we can use the CSS *float* property. By using float, we have the option of floating elements left or right.

Consider the example code below. The class selector, .logo, floats left, and the id selector #search-bar floats right:

.logo {

float: left;

}

#search-bar {

float: right;

}

Let's arrange the Contact button using float.

Instructions

In **main.css**, locate the class selector .contact-btn and assign itfloat: right;.

Click Run and view the results in the web browser.

In the web browser, the gallery images that were arranged neatly in rows are now stacked on the left side of the webpage.

The CSS display value that arranged the images, *flex*, has been removed. In addition to other capabilities, flex can be used to easily align multiple page elements horizontally.

In the example code below, there is a div with class parent:

<div class="parent">

...

</div>

To make children of the div align horizontally on the webpage, in CSS we can use:

.parent {

display: flex;

}

Children elements of the div with class parent will now align horizontally. We can make sure no child element moves off the page by usingflex-wrap: wrap;:

.parent {

display: flex;

flex-wrap: wrap;

}

Finally, to center rows of children elements, we can usejustify-content: center;:

.parent {

display: flex;

flex-wrap: wrap;

justify-content: center;

}

Instructions

Before we start working with flex, expand the web browser view by clicking the arrow button in the top right corner. Take a moment to notice that the image gallery is arranged vertically and off-center.

Next, locate the .gallery class selector in **main.css**. Set the displayproperty to flex.

Click Run to see the gallery images align horizontally.

There's a problem. If you look to the right side of the webpage, some of the gallery images are cut off.

To solve this, below the display: flex; property, create aflex-wrap property and set it to wrap;.

Click Run to see the gallery images now wrapping to the next row.

Finally, we will center the gallery images. For the .gallery selector, add the property justify-content and set it to center.

Click Run to see the images centered on the webpage.

The CSS *position* property enables you to position HTML elements in exact locations on a webpage. One useful value for this property is *relative*. This value positions page elements on a webpage relative to where they would normally appear.

By first setting position: relative;, you can then use the CSS propertiestop, left, bottom, and right to shift an element away from where it would have normally appeared on the page.

The code snippet below moves a div with the class container 10px away from the up and 20px away from the left side of the page.

.container {

position: relative;

top: 10px;

left: 20px;

}

Ever click a button on a webpage that seemed to move down and then back up like a real-life button? We can implement this trick using the positionproperty.

Instructions

In **main.css**, locate .contact-btn a, which is the selector for anchor elements that have a parent with the "contact-btn" class.

Beneath the properties already listed for the .contact-btn a selector, add:

.contact-btn a {

position: relative;

}

Click Run.

Next, locate the .contact-btn a:active selector. :active is a*psuedo-class selector*, which we use to style an element only while it's being clicked.

Add the following CSS to the .contact-btn a:active selector:

top: 2px;

This will cause the Contact button to move down slightly while being clicked, simulating a real-life button being pushed.

Click Run then try clicking the Contact button in the web browser!

Nice work! You've learned a lot. Let's review the web and CSS concepts covered in this lesson.

#### WEB CONCEPTS

* *CSS Box Model*: illustrates the space and boundary properties of an HTML element that can be controlled using CSS.

#### CSS SKILLS

* *border*: sets the outline of an HTML page element, like a picture frame that contains the element.
* *padding*: sets the amount of space between an element's content and its border.
* *margin*: sets the amount of space between an HTML element and the next nearest element(s).
* *display*: property that determines how the selected element will be arranged in relation to other HTML elements on the page.
* *inline*: display value used to arrange HTML elements on the same line as neighboring elements.
* *flex*: display value that allows us to easily align multiple page elements vertically or horizontally.
* *float*: property used to float HTML elements left or right of neighboring elements.
* *position*: property used to position HTML elements in exact locations on a webpage.