

## Greeting Card

### Your Tasks (Mark these off as you go)

- ☐ Define key vocabulary
- ☐ Review how to add elements to an HTML page
- ☐ Review how modify element attributes
- ☐ Review how to select elements with JavaScript
- ☐ Write code to create shapes in JavaScript
- ☐ Review how to position elements
- ☐ Receive credit for this project guide

### ☐ Define key vocabulary

**HTML element**

**Attribute**

**Absolute (as it applies to positioning)**

**Relative (as it applies to positioning)**

### ☐ Review how to add elements to an HTML page

In a previous lesson, you learned how to create and add elements to an HTML page. For example, the following code (1) creates a new paragraph element and assigns the element to a variable called *myParagraph*, (2) adds text to the new element's *innerHTML*, and (3) appends it to the body of the document:

### App.js

```
var myParagraph = document.createElement('p');
myParagraph.innerHTML = "The text inside paragraph";
document.body.append(myParagraph);
```

### HTML

```
...<!DOCTYPE html> == $0
<html lang="en">
  ▶<head> ... </head>
  ▼<body>
    <p>The text inside paragraph</p>
  </body>
</html>
```

In addition to paragraphs, we can create other elements. Below are more examples of how we can create div elements or (dividers) to separate content.

### App.js

```
var t = document.createElement("h1");
t.innerHTML = "Happy Holidays!";
document.body.append(t);

var hat = document.createElement("div");
hat.innerHTML = "This is going to be the hat";
document.body.append(hat);

var head = document.createElement("div");
head.innerHTML = "This is going to be the head";
document.body.append(head);
```

### HTML

```
<!DOCTYPE html>
<html lang="en">
  ▶<head> ... </head>
  ... ▼<body> == $0
    <h1>Happy Holidays!</h1>
    <div>This is going to be the hat</div>
    <div>This is going to be the head</div>
  </body>
</html>
```

### Output

# Happy Holidays!

This is going to be the hat  
This is going to be the head

The code above can be styled to create a head and a hat – this lab guide will step you through the process. In the space below, create and append at least three more elements that could be used to create a character. For example, a body, or a nose, or an arm.

### □ Review how to modify element attributes

HTML attributes provide additional information about HTML elements. For example, an id attribute enables us to differentiate one element from another. Below illustrates how to add an id attribute to an html tag.

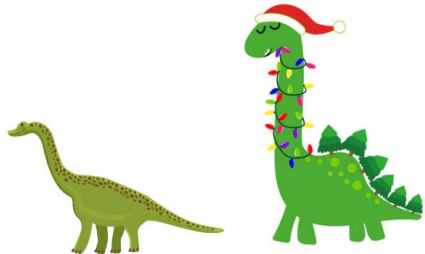
#### App.js

```
var p1 = document.createElement("p");
var p2 = document.createElement("p");
document.body.append(p1)
document.body.append(p2)
p1.id = "para1";
p2.id = "para2";
```

#### Index.html

```
<p id="para1"></p>
<p id="para2"></p>
```

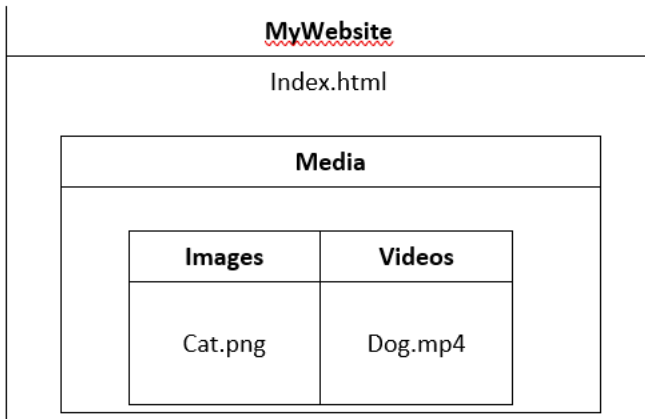
Some HTML elements require an attribute tag. For example, an image will not display if you do not properly specify the `src` attribute.

App.js
<pre> var dino1 = document.createElement("img") dino1.src = "images/brachiosaurus.png"  var dino2 = document.createElement("img") dino2.src = "images/brontosaurus.jpg"  document.body.append(dino1) document.body.append(dino2) </pre>
Output


Some attributes require the setAttribute function. For example, the width and height of an image,

App.js
<pre> var dino1 = document.createElement("img") dino1.src = "images/brachiosaurus.png" dino1.setAttribute("width", "200px")  var dino2 = document.createElement("img") dino2.src = "images/brontosaurus.jpg" dino2.setAttribute("width", "200px") </pre>
Index.html
<pre> &lt;img src="images/brachiosaurus.png" width="200px"&gt; &lt;img src="images/brontosaurus.jpg" width="200px"&gt; </pre>

Refer to the directory layout below. Then write code that could be used display the Cat.png image. Set the id attribute to “cat”, set the width attribute to “100px”.



## ❑ Review how to select HTML elements with JavaScript

All the elements within the document of an HTML page can be selected using the *dot* notation. Below are some examples,

Code	Output
<code>console.log(document.head);</code>	<pre>▼ &lt;head&gt;   &lt;script src="Scripts/App.js" defer&gt;&lt;/script&gt;   &lt;title&gt;Davie JR's Menu&lt;/title&gt; &lt;/head&gt;</pre>
<code>console.log(document.body);</code>	<pre>▼ &lt;body&gt;   &lt;img id="foodLogoImage" src="Images/foodlogo.png"&gt;   &lt;div id="description"&gt;&lt;/div&gt;   ► &lt;div id="nav"&gt;...&lt;/div&gt;   ► &lt;div id="content"&gt;...&lt;/div&gt; &lt;/body&gt;</pre>
<code>console.log(document.title);</code>	Davie JR's Menu

In addition to selecting elements by their name, we can also select elements by their id. Consider the output for the following code snippets which reference the HTML page below,

Index.html	
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;script src="App.js" defer&gt;&lt;/script&gt;     &lt;title&gt;Davie JR's Menu&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;img id = "foodLogoImage" src="Images/foodlogo.png"&gt;     &lt;div id="description"&gt;&lt;/div&gt;     &lt;div id = "nav"&gt;       &lt;img id = "logoImage" src="Images/logo.svg"&gt;       &lt;a href="#menu"&gt;MENU&lt;/a&gt;       &lt;a href="#nutrition"&gt;NUTRITION&lt;/a&gt;       &lt;a href="#order"&gt;ORDER&lt;/a&gt;       &lt;a href="#locations"&gt;LOCATIONS&lt;/a&gt;     &lt;/div&gt;     &lt;div id="content"&gt;       &lt;div id="menu"&gt;         &lt;img id = "burgerImage" src = "Images/burger.jpg"&gt;         &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;         &lt;p&gt;BBQ Bacon Burger&lt;/p&gt;         &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt;       &lt;/div&gt;       &lt;div id="nutrition"&gt;         &lt;p&gt;CALORIES&lt;/p&gt;         &lt;p&gt;678&lt;/p&gt;       &lt;/div&gt;     &lt;/div&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	
App.js	Output
<pre>console.log(document.getElementById("nav"));</pre>	<pre>▼ &lt;div id="nav"&gt;   &lt;img id="logoImage" src="Images/logo.svg"&gt;   &lt;a href="#menu"&gt;MENU&lt;/a&gt;   &lt;a href="#nutrition"&gt;NUTRITION&lt;/a&gt;   &lt;a href="#order"&gt;ORDER&lt;/a&gt;   &lt;a href="#locations"&gt;LOCATIONS&lt;/a&gt; &lt;/div&gt;</pre>
<pre>console.log(document.getElementById("content"));</pre>	<pre>▼ &lt;div id="content"&gt;   ► &lt;div id="menu"&gt;...&lt;/div&gt;   ► &lt;div id="nutrition"&gt;...&lt;/div&gt; &lt;/div&gt;</pre>

Refer to the HTML code block above. Then, indicate what is printed.	
Code	Output
<code>console.log(document.getElementById("nutrition"));</code>	
<code>var n = document.getElementById("description"); n.innerHTML = "Coding is fun!" console.log(n)</code>	
<code>var someElement = document.getElementById("nutrition"); var anotherElement = document.getElementById("content"); anotherElement = someElement; console.log(anotherElement);</code>	

## ❑ Write code to create shapes in JavaScript

A block element is an element that takes up space on a web page. The following HTML tags create block elements,

```
<p></p>
<h1></h1>
<div></div>
```

All block elements have the following characteristics,

- A block-level element always starts on a new line.
- A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
- A block level element has a top and a bottom margin.

All block elements have the following properties which can be defined using the elements *style* property.

**Width and height** — specifies the width and height of the content area.

**Padding** — specifies the amount of space between the content area and the border.

**Border** — specifies the thickness and style of the border surrounding the content area and padding.

**Margin** — specifies the amount of space between the border and the outside edge of the element.

The image below is a visual representation of a block element,



In the following sections you will learn how to style these components to create different shapes and effects with JavaScript.

### Rectangle

By default, all block elements are rectangles. But, you will not see their “rectangle” shape unless you style the border, height, and width properties. These can be styled using the elements style property.

App.js	Output
<pre> 1  var rectangle = document.createElement("div"); 2  rectangle.style.width = "50%"; 3  rectangle.style.height = "100px"; 4  rectangle.style.borderWidth = "2em"; 5  rectangle.style.borderStyle = "dotted"; 6  rectangle.style.borderColor = "green"; 7  document.body.append(rectangle); </pre>	
Explanation	
<p>Line 1: creates the block element, div</p> <p>Line 2: styles the width to be 50% the width of the screen</p> <p>Line 3: styles the height to be 100px</p> <p>Line 4: styles the border width 2x the default text size</p> <p>Line 5: styles the border to be dotted. Other styles include solid and dashed</p> <p>Line 6: styles the border color to be green. You can also use hexadecimal colors here. For example, #0000ff</p> <p>Line 7: appends the rectangle variable to the body of the page</p>	



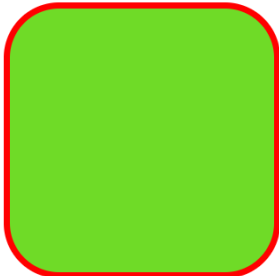
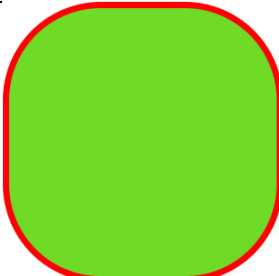
In the space below, write code to that could be used to create the square below. Note, the width and height are both 200px.

Happy Holidays!

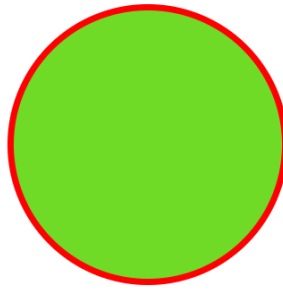


### Circle

A circle is nothing more than a rectangle with rounded edges. You can round the edges of any block element by setting the `borderRadius` property. Consider the following examples,

Code	Output
<pre>var box = document.createElement("div"); box.style.width = "200px"; box.style.height = "200px"; box.style.border = "red solid 5px"; box.style.backgroundColor = "#6fdb27" box.style.borderRadius = "20%" document.body.append(box)</pre>	
<pre>var box = document.createElement("div"); box.style.width = "200px"; box.style.height = "200px"; box.style.border = "red solid 5px"; box.style.backgroundColor = "#6fdb27" box.style.borderRadius = "35%" document.body.append(box)</pre>	

```
var box = document.createElement("div");
box.style.width = "200px";
box.style.height = "200px";
box.style.border = "red solid 5px";
box.style.backgroundColor = "#6fdb27";
box.style.borderRadius = "50%";
document.body.append(box)
```

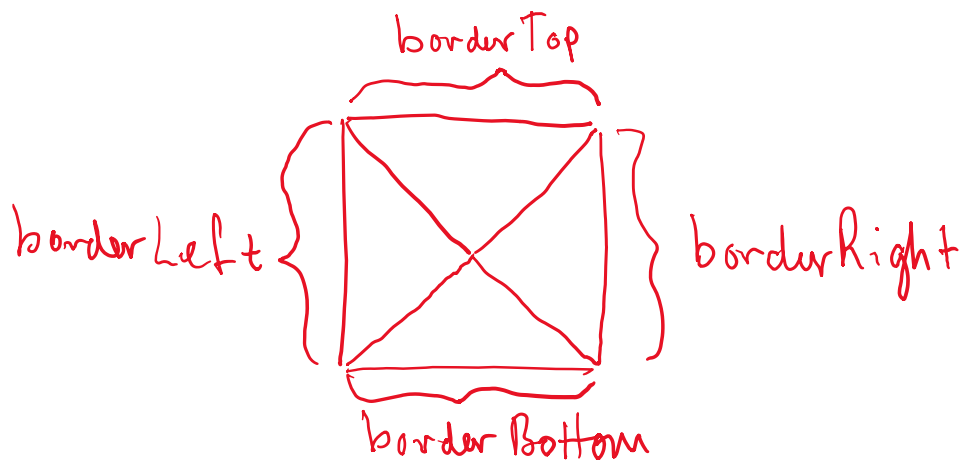


In a previous example you created an element that could be styled as a head for a character. Style this element to have a width and height, rounded corners, and a backgroundColor.

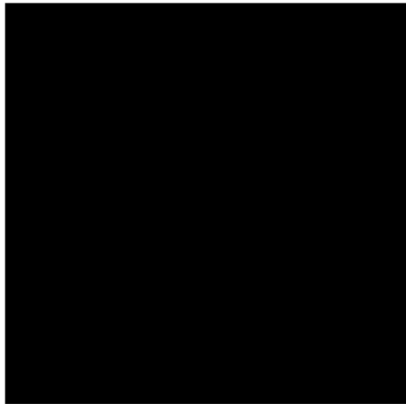
### Triangle

In addition to rectangles and circles it is also possible to make triangles from a basic block element. This is a littler trickier and requires some explanation.


Recall that when you create a *div* element we can specify the width of the border. So, imagine a *div* element with a width and height of zero, but with a really thick border.




The code below would create such an element,

Code	Output
<pre>var triangle = document.createElement("div"); triangle.style.border = "black solid 100px"; triangle.style.width = "0px"; triangle.style.height = "0px"; document.body.append(triangle);</pre>	

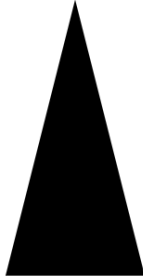
Now let's consider what happens when we set the transparency of different sides. In the code below we have created two triangles by setting the left and right borders to transparent.

Code	Output
<pre>var triangle = document.createElement("div"); triangle.style.border = "black solid 100px"; triangle.style.width = "0px"; triangle.style.height = "0px"; triangle.style.borderLeftColor = "transparent"; triangle.style.borderRightColor = "transparent"; document.body.append(triangle);</pre>	

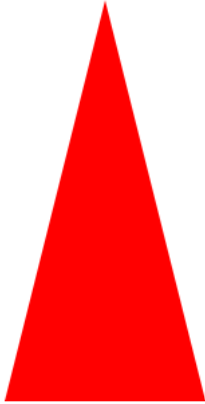
Now, if we set the top or bottom border to transparent we will have a single triangle,

Code	Output
<pre>var triangle = document.createElement("div"); triangle.style.border = "black solid 100px"; triangle.style.width = "0px"; triangle.style.height = "0px"; triangle.style.borderLeftColor = "transparent"; triangle.style.borderRightColor = "transparent"; triangle.style.borderTopColor = "transparent"; document.body.append(triangle);</pre>	

In addition to making equilateral triangles, you can set the width of the specific borders to create different types of triangles. In the example, below, the bottom border is 4 times as big as the other borders.

Code	Output
<pre> var triangle = document.createElement("div"); triangle.style.border = "black solid 50px"; triangle.style.width = "0px"; triangle.style.height = "0px"; triangle.style.borderLeftColor = "transparent"; triangle.style.borderRightColor = "transparent"; triangle.style.borderTopColor = "transparent"; triangle.style.borderBottomWidth = "200px"; document.body.append(triangle); </pre>	

In a previous example you created an element that could be styled as a hat for a character. Style this element to look as follows.



## □ Review how to position elements

Recall that the default position for all block elements is the far-left side of the screen. And, unless specified otherwise, they take up the full width of the page. Notice in the example below, each block element is positioned to the far-left and they are displayed vertically relative to one another.

Index.html	App.js
<pre> &lt;div id = "one"&gt;&lt;/div&gt; &lt;div id = "two"&gt;&lt;/div&gt; &lt;div id = "three"&gt;&lt;/div&gt; </pre>	<pre> var one = document.getElementById("one"); var two = document.getElementById("two"); var three = document.getElementById("three");  one.style.height = "5em"; two.style.height = "5em"; three.style.height = "5em";  one.style.width = "50%"; two.style.width = "50%"; three.style.width = "50%";  one.style.backgroundColor = "yellow"; two.style.backgroundColor = "purple"; three.style.backgroundColor = "gray"; </pre>
Output	
	

The default position of an element can be changed by setting its *position* property. The *position* property can take one of four values:

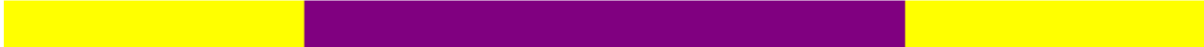
1. static- the default value (it does not need to be specified)
2. relative
3. absolute
4. fixed

For the purposes of this, class relative and absolute positioning are the most useful. Both are reviewed below,

### Relative positioning

The *relative* value allows you to position an element *relative* to its default position on the web page. Relative position is really handy for positioning elements inside other elements.

In the example below, div2 is centered in div1

App.js	
<pre> var div1 = document.createElement("div"); div1.style.backgroundColor = "yellow"; div1.style.height = "50px"; var div2 = document.createElement("div"); div2.style.backgroundColor = "purple"; div2.style.height = "50px";  div2.style.width = "50%"; div2.style.position = "relative"; div2.style.margin = "0 auto"  document.body.append(div1); div1.append(div2); </pre>	<p>← div2 is centered in div1 by setting the position to <i>relative</i> and the margin to <i>0 auto</i></p> <p>← Here we append div2 to div1</p>
Output	
	

### Absolute positioning

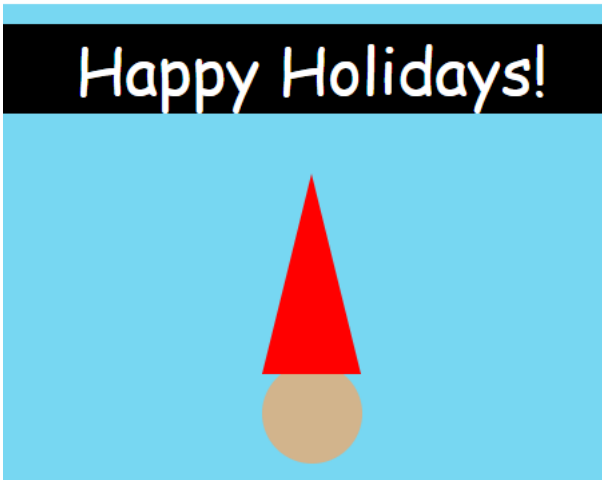
When an element's position is set to *absolute* you can place the element exactly where you want relative to the page. All elements on the page will ignore an element that is positioned with the *absolute* designation.

Index.html	App.js
<pre> &lt;div id = "one"&gt;&lt;/div&gt; &lt;div id = "two"&gt;&lt;/div&gt; &lt;div id = "three"&gt;&lt;/div&gt; </pre>	<pre> var one = document.getElementById("one"); var two = document.getElementById("two"); var three = document.getElementById("three");  one.style.height = "5em"; two.style.height = "5em"; three.style.height = "5em";  one.style.width = "50%"; two.style.width = "50%"; three.style.width = "50%";  one.style.backgroundColor = "yellow"; two.style.backgroundColor = "purple"; three.style.backgroundColor = "gray";  two.style.position = "absolute"; two.style.top = "50px"; two.style.left = "50px";  three.style.position = "absolute"; three.style.top = "100px"; three.style.left = "100px"; </pre>

## Output



The code below creates several HTML elements and appends them to the screen. Style the elements to appear like the screenshot below. Note that the element holding the text, the hat, and the head are in the background.



```
var background = document.createElement("div");
var hat = document.createElement("div");
var head = document.createElement("div");
var text = document.createElement("div");
background.style.width = "98%";//stretches the div but keeps it in bounds on the
screen
background.style.backgroundColor = "#77d7f2";//Sets the background to a shade of
blue
background.style.padding = "10px";//creates space between the elements and the
outer edge
document.body.append(background);

background.append(text);
background.append(hat);
background.append(head);
```

☐ **Receive credit for this project guide**

Submit this portion of the lab to Pluska to receive credit for the project guide. The project guide will be weighted 10% of your final project grade.