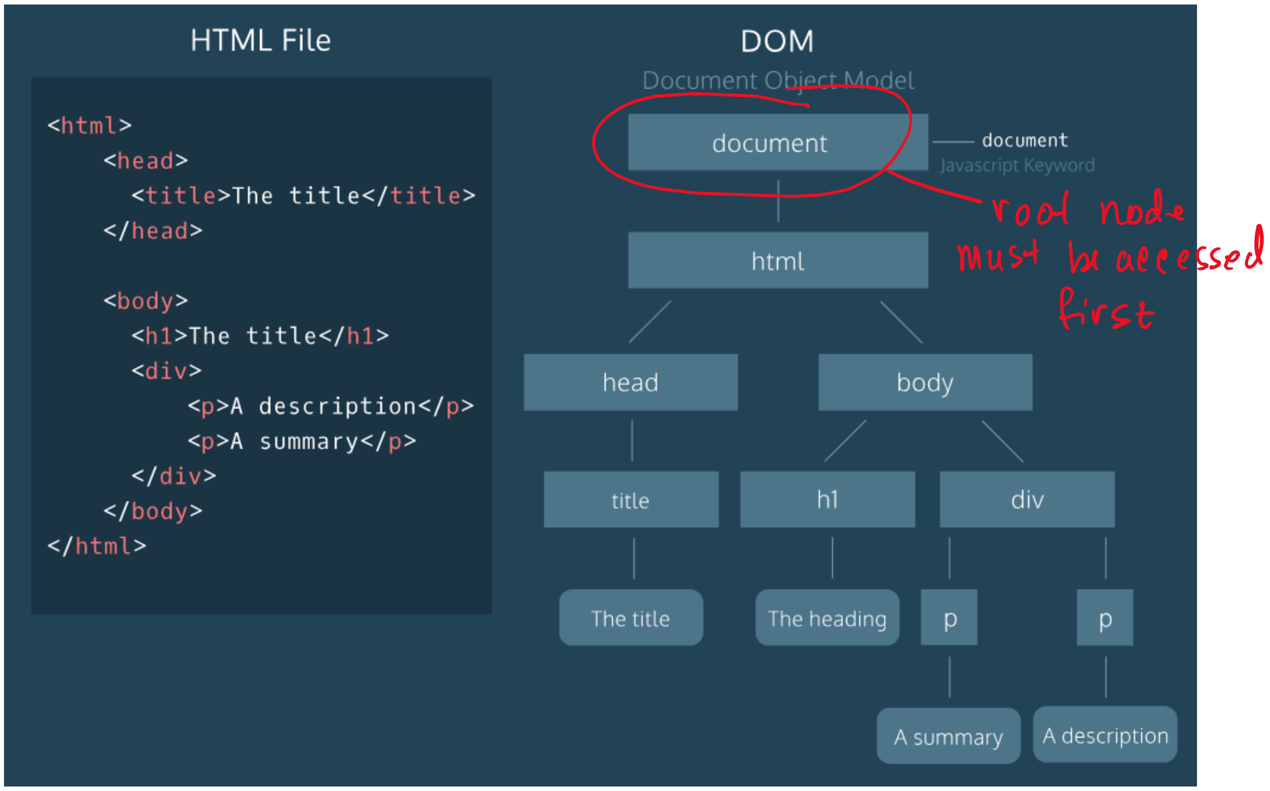
Set 19: Styling DOM Elements

Skill 19.01: Review how to select DOM elements
Skill 19.02: Write code to style text
Skill 19.03: Write code to style block elements
Skill 19.04: Write code to position elements

Skill 19.01: Review how to select DOM elements

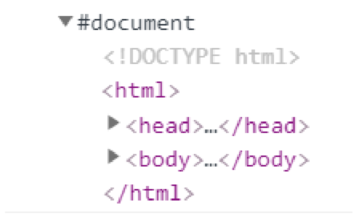
Skill 19.01 Concepts

Before we dive in to writing code to style DOM elements, let's first revisit how we can select DOM elements.

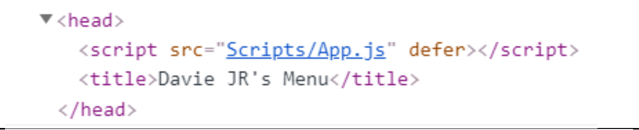
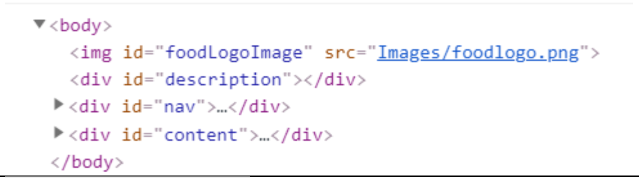
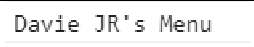


```
<html>
    <head>
      <title>The title</title>
    </head>

    <body>
      <h1>The title</h1>
      <div>
          <p>A description</p>
          <p>A summary</p>
      </div>
    </body>
</html>
```

Notice in the above example, that at the root of our model is the *document* object. In JavaScript, the *document* object is the door to the DOM structure. Before you can access a specific element within the document you must first access this root object. Consider the following code which logs the document object to the console. Within the document you can see the entire contents of the webpage.

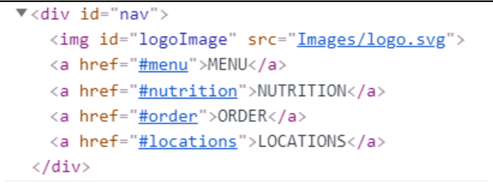
Code	Output
`console.log(document);`	 `▼#document` `<!DOCTYPE html>` `<html>` `▶<head>…</head>` `▶<body>…</body>` `</html>`

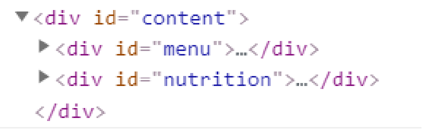
The elements within the document of the page can be selected using the *dot* notation. Below are some examples,

Code	Output
`console.log(document.head);`	 `▼<head>` `<script src="Scripts/App.js" defer></script>` `<title>Davie JR's Menu</title>` `</head>`
`console.log(document.body);`	 `▼<body>` `<img id="foodLogoImage" src="Images/foodlogo.png">` `<div id="description"></div>` `▶<div id="nav">…</div>` `▶<div id="content">…</div>` `</body>`
`console.log(document.title);`	 `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,

Code	Output

`console.log(document.getElementById("nav"));`	 ▼`<div id="nav">`
	`<img id="logoImage" src="Images/logo.svg">`
	`<a href="#menu">MENU</a>`
	`<a href="#nutrition">NUTRITION</a>`
	`<a href="#order">ORDER</a>`
	`<a href="#locations">LOCATIONS</a>`
	`</div>`

`console.log(document.getElementById("content"));`	 ▼`<div id="content">` ▶`<div id="menu">…</div>` ▶`<div id="nutrition">…</div>` `</div>`

```
<!DOCTYPE html>
<html>
  <head>
   <script src="App.js" defer></script>
   <title>Davie JR's Menu</title>
  </head>
  <body>
    <img id = "foodLogoImage" src="Images/foodlogo.png">
    <div id="description"></div>
    <div id = "nav">
      <img id = "logoImage" src="Images/logo.svg">
      <a href="#menu">MENU</a>
      <a href="#nutrition">NUTRITION</a>
      <a href="#order">ORDER</a>
      <a href="#locations">LOCATIONS</a>
    </div>
    <div id="content">
      <div id="menu">
         <img id = "burgerImage" src = "Images/burger.jpg">
         <h1>BBQ BACON BURGER</h1>
         <p>BBQ Bacon Burger</p>
         <a href="#"  id="button">ORDER NOW</a>
      </div>
      <div id="nutrition">
         <p>CALORIES</p>
         <p>678</p>
      </div>
    </div>
</body>
</html>
```

Skill 19.01 Exercise 1

Skill 19.02: Write code to style text

Skill 19.02 Concepts

Previously, we learned how to assign and change the text of an element using the *.innerHTML* property.

For example, the following code reassigns the inner HTML of the body element to the text 'The cat loves the dog':

```
document.body.innerHTML = "The cat hates the dog";
```

The *.innerHTML* property can also add any valid HTML, including properly formatted elements. The following example assigns an *h2* element inside the *<body>* element:

```
document.body.innerHTML = '<h2>This is a heading</h2>';
```

Below are more examples,

Code	Output
`var someText = "Yummy!";` `document.getElementById("description").innerHTML = someText;` `console.log(document.getElementById("description"));`	`<div id="description">Yummy!</div>`
`var someText = "Not available";` `document.getElementById("button").innerHTML = someText;` `console.log(document.getElementById("button"));`	`<a href="#" id="button">Not available</a>`

```
<!DOCTYPE html>
<html>
  <head>
   <script src="App.js" defer></script>
   <title>Davie JR's Menu</title>
  </head>
  <body>
    <img id = "foodLogoImage" src="Images/foodlogo.png">
    <div id="description"></div>
    <div id = "nav">
      <img id = "logoImage" src="Images/logo.svg">
      <a href="#menu">MENU</a>
      <a href="#nutrition">NUTRITION</a>
      <a href="#order">ORDER</a>
      <a href="#locations">LOCATIONS</a>
    </div>
    <div id="content">
      <div id="menu">
         <img id = "burgerImage" src = "Images/burger.jpg">
         <h1>BBQ BACON BURGER</h1>
         <p>BBQ Bacon Burger</p>
         <a href="#"  id="button">ORDER NOW</a>
      </div>
```

```
            <div id="nutrition">
               <p>CALORIES</p>
               <p>678</p>
            </div>
        </div>
</body>
</html>
```

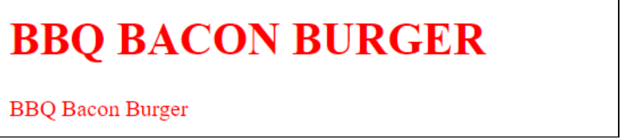
It is also possible to change how the text looks by modifying the *style* property of the element in which the text is displayed. The *.style* property of a DOM element provides access to the inline style of that HTML tag.

Font Color

To change the color of text we first indicate the *style* property, followed by the *.color* property. This is illustrated in the following example. Notice that we simply typed the color "red" and the browser new how to interpret it. Modern browsers support 140 named colors, which are listed here, https://htmlcolorcodes.com/color-names

```
<div id="menu">
      <img id = "burgerImage" src = "burger.jpg">
      <h1>BBQ BACON BURGER</h1>
      <p>BBQ Bacon Burger</p>
      <a href="#"  id="button">ORDER NOW</a>
</div>
```

Code	Output

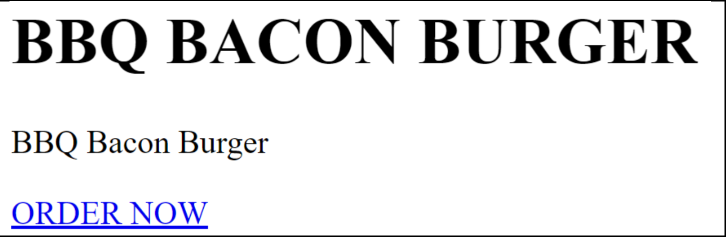
`var content = document.getElementById("menu");`
`content.style.color = "red";`
	 **BBQ BACON BURGER**
	BBQ Bacon Burger

Font Size

The size of font can be specified with the *.fontSize* property. Because different screens render text differently, it is best practices to specify the font size relative to the screen on which it is to be displayed. This is done using *em* units. For example, if you specify the *fontSize* to 2 em, the font will appear 2x as large. If you specify the *fontSize* to 1.5 em, the font will appear 1.5x as large, etc.

```
<div id="menu">
      <img id = "burgerImage" src = "burger.jpg">
      <h1>BBQ BACON BURGER</h1>
      <p>BBQ Bacon Burger</p>
      <a href="#"  id="button">ORDER NOW</a>
</div>
```

Code	Output

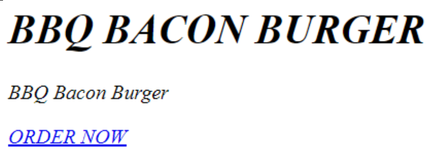
`var content =`
`document.getElementById("menu");`
`content.style.fontSize = "2em";`
	 **BBQ BACON BURGER**
	BBQ Bacon Burger
	ORDER NOW

Font Style

There are three basic font styles you can apply to text: normal, italic, or oblique. These can be applied with the *.fontStyle* property.

```html
<div id="menu">
      <img id = "burgerImage" src = "burger.jpg">
      <h1>BBQ BACON BURGER</h1>
      <p>BBQ Bacon Burger</p>
      <a href="#"  id="button">ORDER NOW</a>
</div>
```

Code	**Output**

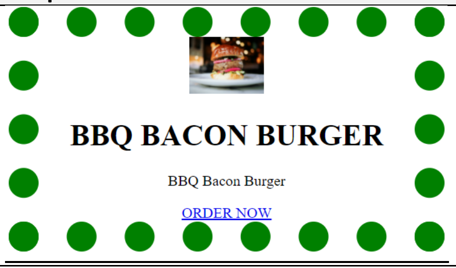
`var content = document.getElementById("menu");`
`content.style.fontStyle = "oblique";`
	 ***BBQ BACON BURGER***
	BBQ Bacon Burger
	ORDER NOW

Text Alignment

The text within an element can be aligned as left, right, or center using the *.textAlign* property.

```html
<div id="menu">
      <img id = "burgerImage" src = "burger.jpg">
      <h1>BBQ BACON BURGER</h1>
      <p>BBQ Bacon Burger</p>
      <a href="#"  id="button">ORDER NOW</a>
</div>
```

Code	**Output**

`var content = document.getElementById("menu");`
`content.style.textAlign = "center";`
`content.style.border = "dotted 2em green";`
	 **BBQ BACON BURGER**
	BBQ Bacon Burger
	ORDER NOW

Skill 19.02 Exercise 1

Skill 19.03 Write code to style block elements

Skill 19.03 Concepts

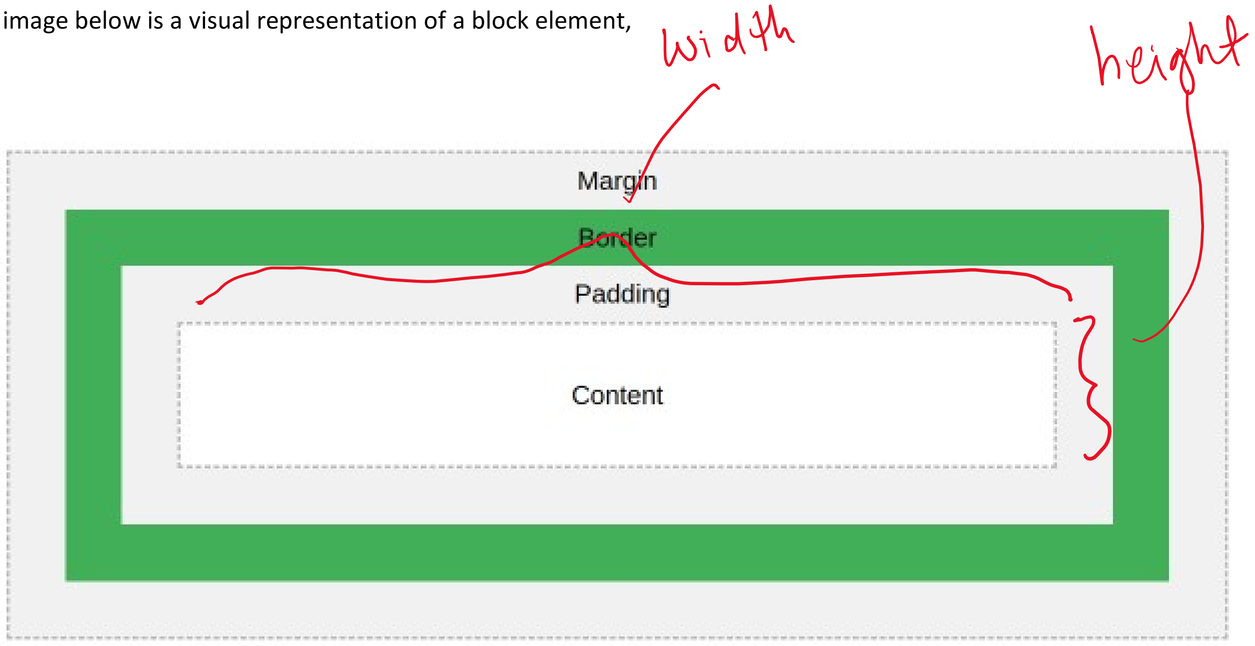
A block element is an element that takes up space on a web page. Below lists the basic characteristics of a block element,

- 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, whereas an inline element does not.

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,



We have already encountered many block elements. Below are some examples,

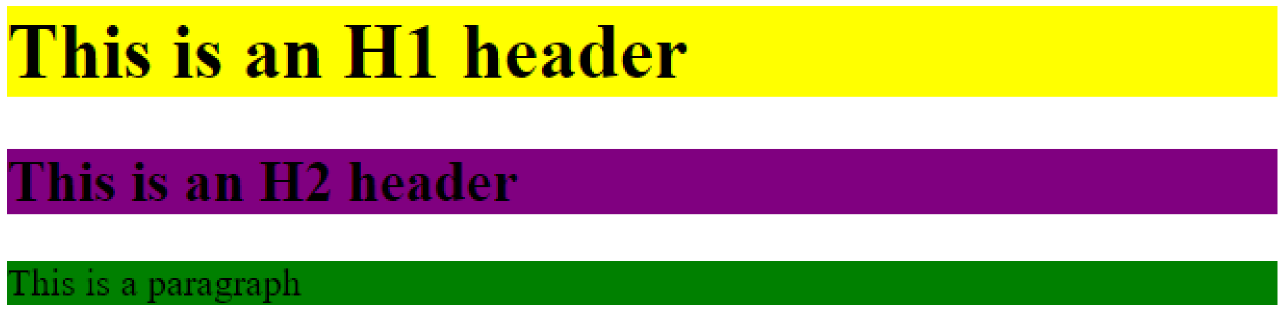
Index.html

```html
<h1 id="h1-head">This is an H1
header</h1>
<h2 id="h2-head">This is an H2
header</h2>
<p id="para">This is a paragraph</p>
```

App.js

```javascript
var h1 = document.getElementById("h1-head");
h1.style.backgroundColor = "yellow";
var h2 = document.getElementById("h2-head");
h2.style.backgroundColor = "purple";
var p = document.getElementById("para");
p.style.backgroundColor = "green";
```

Output

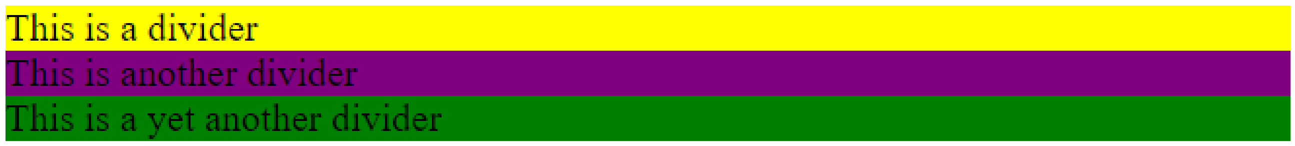


This is an H1 header

This is an H2 header

This is a paragraph

Another block element commonly used to organize content is the <div></div> tag. And, just as the tag implies, the <div> tag is used to *divide* content.

Index.html	**App.js**
`<div id="div1">This is a divider</div>`	`var div1 = document.getElementById("div1");` `div1.style.backgroundColor = "yellow";`
`<div id="div2">This is another divider</div>`	`var div2 = document.getElementById("div2");` `div2.style.backgroundColor = "purple";`
`<div id="div3">This is yet another divider</div>`	`var div3 = document.getElementById("div3");` `div3.style.backgroundColor = "green";`
Output	
 This is a divider This is another divider This is a yet another divider	

The above examples illustrate how to use the *backgroundColor* property to style the background of a block element. Below are examples of how we can style other properties associated with block elements.

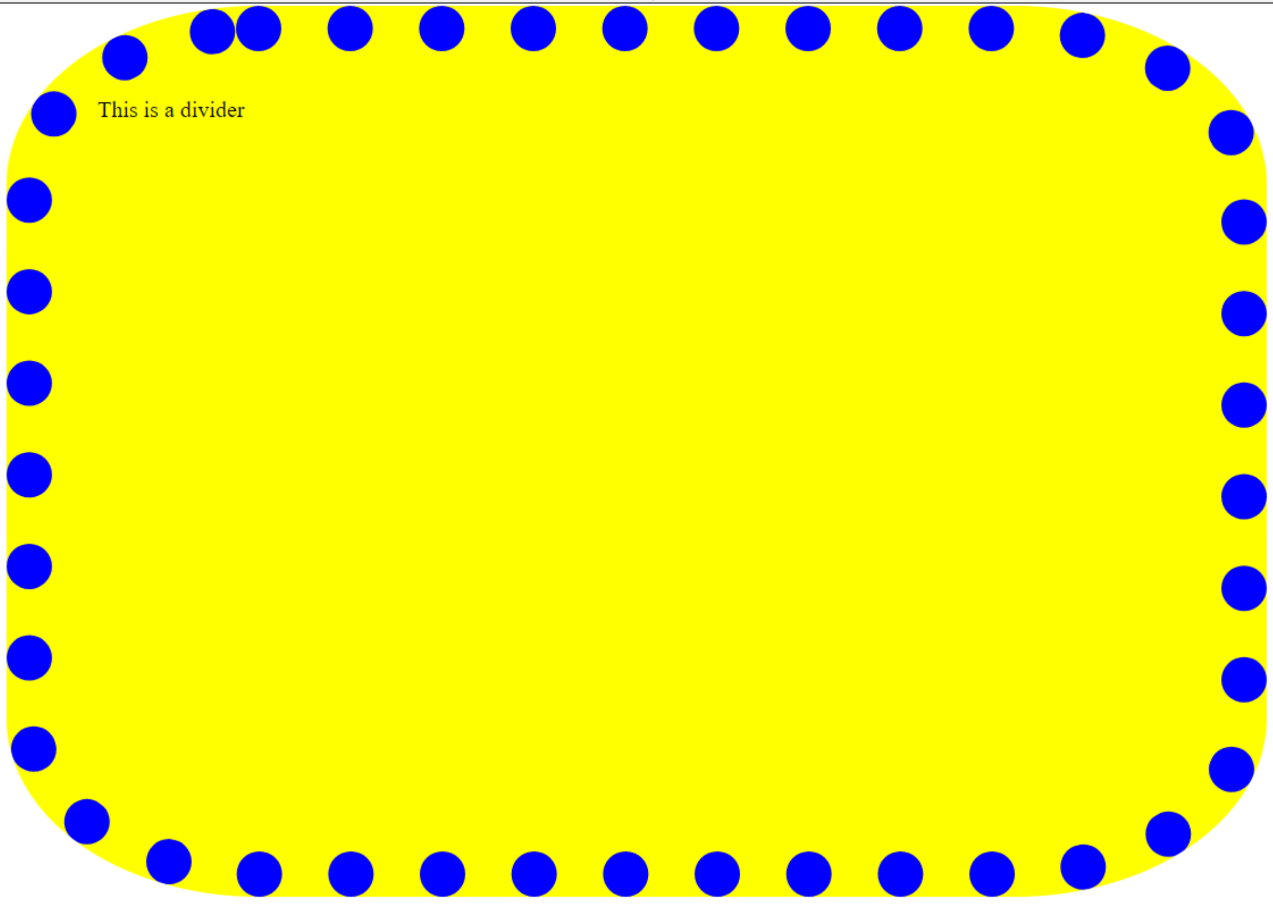
Index.html

```html
<div id="div1">This is a divider</div>
```

App.js

```javascript
var div1 =
document.getElementById("div1");
div1.style.backgroundColor = "yellow";
div1.style.width = "50%";
div1.style.height = "500px";
div1.style.padding = "2em";
div1.style.borderWidth = "2em";
div1.style.borderStyle = "dotted";
div1.style.borderColor = "blue";
div1.style.borderRadius = "20%";
div1.style.margin = "2em";
```

Output



This is a divider

Property	Description	Example
backgroundColor	The background color of the element	`var div1 = document.getElementById("div1");` `div1.style.backgroundColor = "yellow";`
width	Specifies the width of the element	`div1.style.width = "50%";`
height	Specifies the height of the element	`div1.style.height = "500px";`
padding	Specifies the amount of space between the content area and the border	`div1.style.padding = "2em";`
margin	specifies the amount of space between the border and the outside edge of the element.	`div1.style.margin = "2em";`
borderWidth	Specifies the thickness of a border	`div1.style.borderWidth = "2em";`
borderColor	Specifies the color a border	`div1.style.borderColor = "blue";`
borderStyle	Specifies the type of border: dotted, dashed, double, solid, etc*	`div1.style.borderStyle = "dotted";`
borderRadius	Specifies the curvature of the the border	`div1.style.borderRadius = "20%";`
*For more border styles, check this out: https://www.w3schools.com/css/css_border.asp		

Notice in units used in the above examples: em, px, and %. The units you choose depends on your use case. Below is a description of each unit,

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
px	A fixed size. pixels (1px = 1/96th of 1in)
%	Relative to the parent element

Skill 19.03 Exercises 1

Skill 19.04 Write code to position block elements

Skill 19.04 Concepts

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.

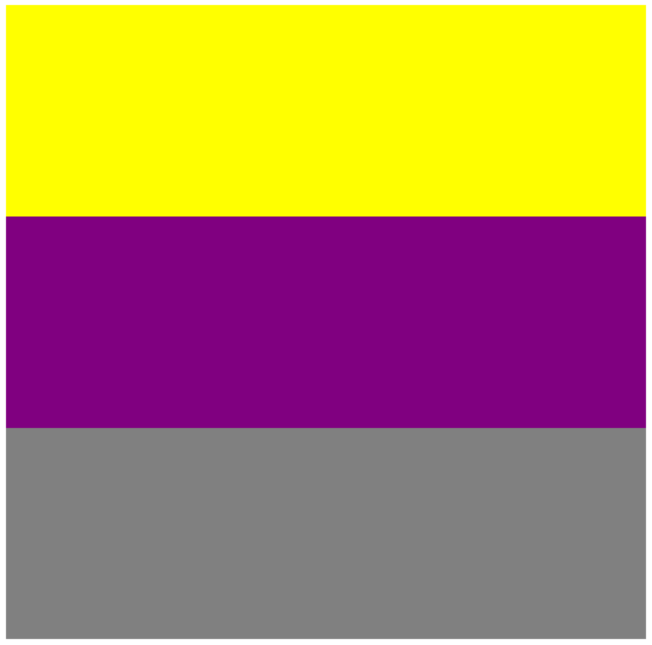
Index.html

```
<div id = "one"></div>
<div id = "two"></div>
<div id = "three"></div>
```

App.js

```
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";
```

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

Each type of positioning is described in more detail below.

Relative positioning

The *relative* value allows you to position an element *relative* to its default position on the web page.

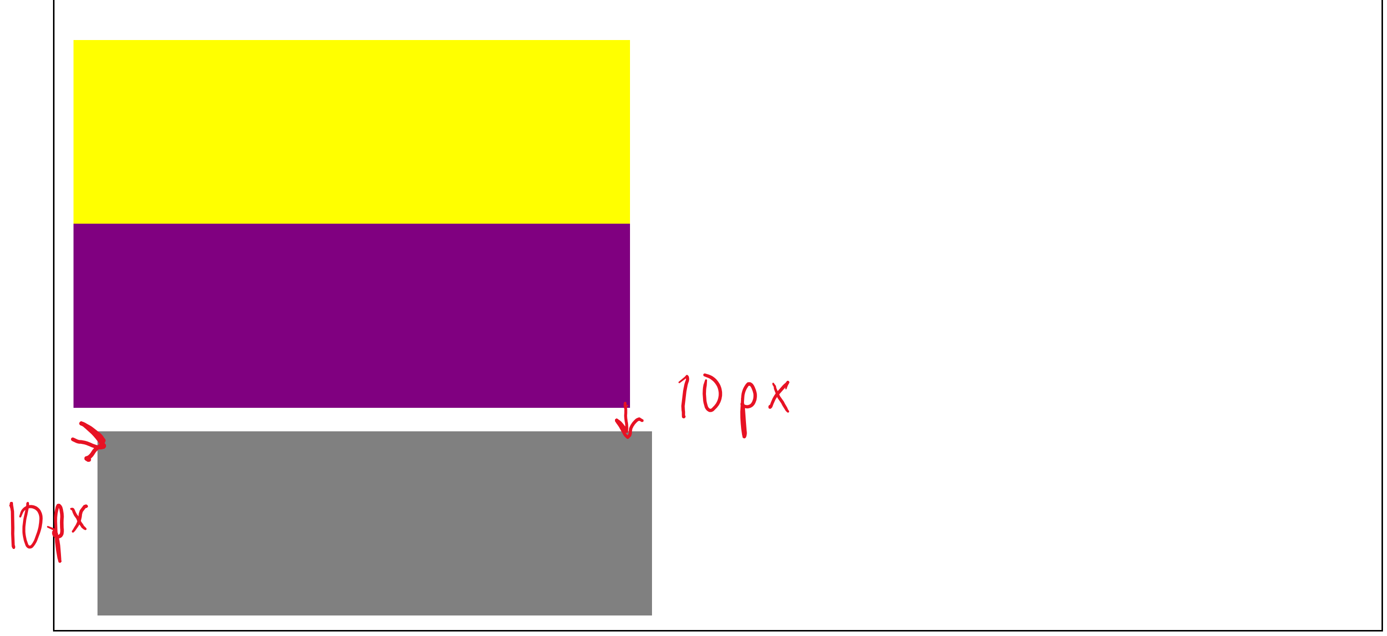
In the example below, div three is shifted 10px from the top and left border relative to its default location.

Index.html

```html
<div id = "one"></div>
<div id = "two"></div>
<div id = "three"></div>
```

App.js

```javascript
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";

three.style.position = "relative";
three.style.top = "10px";
three.style.left = "10px";
```

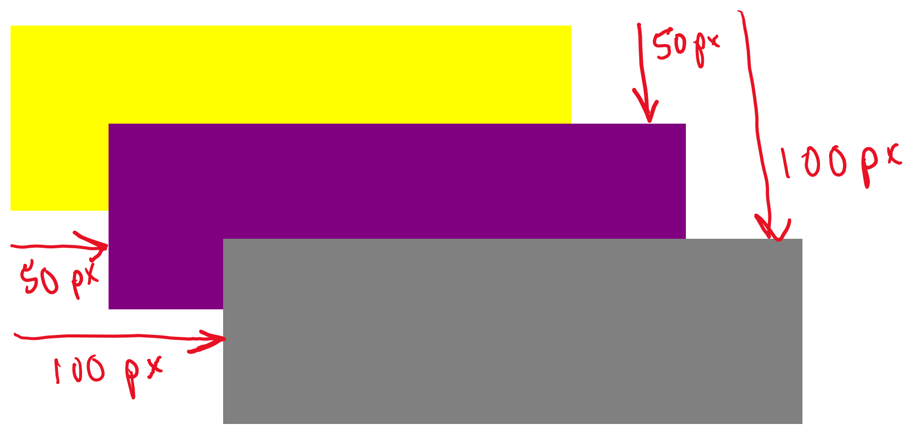
Output


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

`<div id = "one"></div>`
`<div id = "two"></div>`
`<div id = "three"></div>`
	`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";`

Output


Fixed positioning

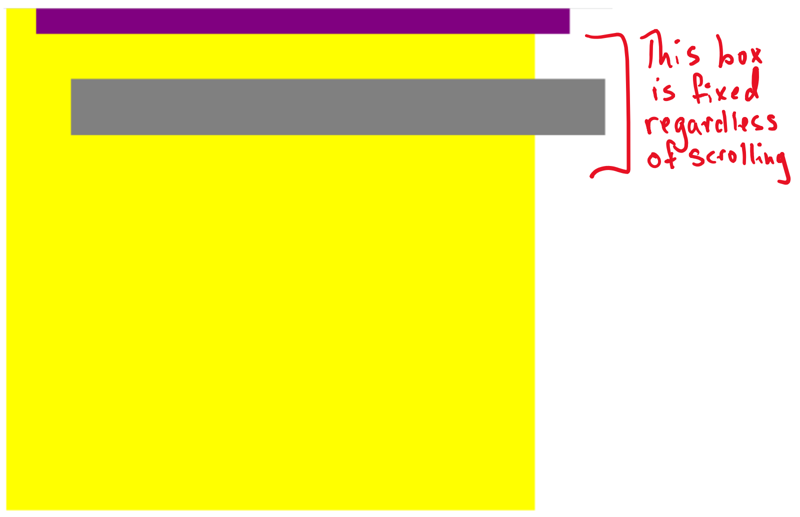
We can *fix* an element to a specific position on the page (regardless of user scrolling) by setting its position to *fixed*.

Index.html

```
<div id = "one"></div>
<div id = "two"></div>
<div id = "three"></div>
```

App.js

```
var one = document.getElementById("one");
var two = document.getElementById("two");
var three = document.getElementById("three");

one.style.height = "50em";
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 = "fixed";
three.style.top = "100px";
three.style.left = "100px";
```

Output


Skill 19.04 Exercise 1