

## Set 21: Creating DOM Elements

**Skill 21.01: Write code to add an element to an HTML page**

**Skill 21.02: Write code to remove or hide an HTML element**

**Skill 21.03: Reference media files**

**Skill 21.04: Write code to add an image**

**Skill 21.05: Write code to modify an attribute**

**Skill 21.01: Write code to add an element to an HTML page**

### Skill 21.01 Concepts

In our previous lesson about the DOM we learned how to select and modify existing elements on our webpage. But, just as the DOM allows scripts to modify existing elements, it also allows for the creation of new ones. The `.createElement(tagName)` method creates a new element based on the specified tag name. However, it does not append it to the document. It creates an empty element with no inner HTML.

To create an element and add it to the web page, you must assign it to an element that already exists on the DOM. We call this process appending. The `.append` method will add the element as the last child node.

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 our previous lessons we have learned how to control elements on the page by accessing their *id*'s. In the example above, the paragraph we created does not have an *id* associated with it. To specify an *id* we can use the *id* property as follows,

## App.js

```
myParagraph.id = "p1";
```

## HTML

```
...<!DOCTYPE html> == $0
<html lang="en">
  <head> ... </head>
  <body>
    <p id="p1">The text inside paragraph</p>
  </body>
</html>
```

Now that we have associated an *id* to our new element, we can modify it using the same techniques we learned before.

In the below example, we change the text to read, "Hello DOM!"

## App.js

```
var myParagraph = document.createElement('p');
myParagraph.innerHTML = "The text inside paragraph";
document.body.append(myParagraph);
myParagraph.id = "p1"
document.getElementById("p1").innerHTML = "Hello DOM!"
```

## HTML

```
...<!DOCTYPE html> == $0
<html lang="en">
  <head> ... </head>
  <body>
    <p id="p1">Hello DOM!</p>
  </body>
</html>
```

## Output

Hello DOM!

## [Skill 21.01 Exercise 1](#)

## Skill 21.02: Write code to remove or hide an HTML element

### Skill 21.02 Concepts

In addition to modifying or creating an element from scratch, the DOM also allows for the removal of an element. The `.remove` method removes a specified element.

#### App.js

```
var someElement = document.createElement("div");
someElement.innerHTML = "My Element!";
someElement.id = "e1";
document.body.append(someElement);

var e = document.getElementById("e1");
e.remove();
```

#### HTML

```
...<!DOCTYPE html> == $
<html lang="en">
  <head> ... </head>
  <body> </body>
</html>
```

Nothing is there because it has been removed



If you want to hide an element, the `.hidden` property allows you to hide it by assigning it as true or false. Hiding an element does not remove an element however. The element is still on the page, just now viewable by the user.

#### App.js

```
var someElement = document.createElement("div");
someElement.innerHTML = "My Element!";
someElement.id = "e1";
document.body.append(someElement);

var e = document.getElementById("e1");
e.hidden = true;
```

## HTML

```
...<!DOCTYPE html> == $0
<html lang="en">
  ▶ <head> ... </head>
  ▼ <body>
    <div id="e1" hidden>My Element!</div>
  </body>
</html>
```

## Output



To show a hidden element, simply set the hidden property to false,

## App.js

```
var someElement = document.createElement("div");
someElement.innerHTML = "My Element!";
someElement.id = "e1";
document.body.append(someElement);

var e = document.getElementById("e1");
e.hidden = false;
```

## HTML

```
...<!DOCTYPE html> == $0
<html lang="en">
  ▶ <head> ... </head>
  ▼ <body>
    <div id="e1" hidden>My Element!</div>
  </body>
</html>
```

## Output

My Element!  There it is!

### [Skill 21.02 Exercise 1](#)

## Skill 21.03: Reference media files

### Skill 21.03 Concepts

The images and videos you include with your page are separate files and will need to be referenced correctly to be displayed.

Images and videos are referenced using `src` attribute - short for source. This tells the tag the location of the image or video to load.

Consider the following file structure. *Media* is the name of the folder. And, inside the folder we have a file called *Index.html* and an image called *Frog.jpg*. The following code could be used to reference the *Frog.jpg* image from the *Index.html* page.

| Media      | src = "Frog.jpg" |
|------------|------------------|
| Index.html |                  |
| Frog.jpg   |                  |

Now consider an example where the *Frog.jpg* we are trying to reference is stored in a directory that is different than *Index.html*. In the file structure below, we have created a directory called *Images* inside the *Media* folder and placed the *Frog.jpg* image inside of it. The following code could be used to reference the *Frog.jpg* image from the *Index.html* page.

| Media      | src = "Images/Frog.jpg" |
|------------|-------------------------|
| Index.html |                         |
| Images     |                         |
| Frog.jpg   |                         |

Now consider the situation below. *Media* and *Images* are both separate directories in the *MyWebsite* directory. Inside the *Media* directory we have an *Index.html* page and inside the *Images* directory we have our image *Frog.jpg* we want to reference. To do this, we must first "backout" of the *Media* directory, then enter the *Images* directory. The `..` syntax is used to backout of a directory.

| MyWebsite  |          | src = "../Images/Frog.jpg" |
|------------|----------|----------------------------|
| Media      | Images   |                            |
| Index.html | Frog.jpg |                            |

### Skill 21.03 Exercise 1

#### Skill 21.04: Write code to add an image

##### Skill 21.04 Concepts

The *img* tag allows you to add images to your page. Just like the *br* tag we learned about previously, the *img* tag doesn't require a closing tag. All the information needed to display your image is contained within the tag itself.


To tell the browser the location and name of the file to use, you need to use the *src* attribute. The *src* attribute is added to the image tag inside the brackets. An additional attribute, called *alt*, provides backup text in case your image doesn't download properly or for visually impaired users.

How the *img* tag can be used to display an image on your webpage is illustrated below,


|                       |  |
|-----------------------|--|
| Media                 |  |
| Index.html<br>dog.jpg |  |

1. Creates an image tag using the abbreviation *img*. This is considered a self closing tag, since it doesn't need to wrap text as many other tags do. The */*right before the ending *>* is optional, but helps remind us that this tag doesn't need a closing tag.

2. The *src* attribute is short for source. This tells the tag which image to load. In this case, the page will look for an image with the filename *dog.jpg* in the same directory as the page. Image file names include extensions that tell the computer which type of image they are working with. Common extensions are *.jpg*, *.jpeg*, *.gif*, *.png*. Make sure to put quotation marks around your image filename.

3. The *alt* attribute is short for alternative text. While you won't see this text on your web page, it provides a backup in case your image doesn't download properly or for visually impaired users. In this example, if your browser failed to load the image you would see 

How to use JavaScript to display an image on an html page is illustrated below,

| App.js  | HTML   |
|---|--|
| <pre>1 var i = document.createElement("img"); 2 i.src = "cat.webp"; 3 i.alt = "cat butt" 4 document.body.append(i);</pre>   | <pre>&lt;!DOCTYPE html&gt; &lt;html lang="en"&gt;   &lt;head&gt; ... &lt;/head&gt;   ... &lt;body&gt; == \$0     &lt;img src="cat.webp" alt="cat butt"&gt;   &lt;/body&gt; &lt;/html&gt;</pre> |
| Output  |  |
|    |  |
| Explanation   |  |
| <p>The code in lines 1 thru 4 create the img tag shown in the HTML.</p> <p>Line 1: Creates the img element and stores it in a variable called i</p> <p>Line 2: Sets the src of the image to cat.webp</p> <p>Line 3: Sets the alternate text in case the image does not load</p> <p>Line 4: appends the image to the body of the index.html page</p> |  |

#### [Skill 21.04 Exercise 1](#)

#### Skill 21.05: Write code to modify an attribute

##### Skill 21.05 Concepts

Attributes are values that contain additional information about HTML elements. They usually come in name/value pairs, and may be essential depending on the element. Consider the *img* tag below. The image tag below requires the *src* attribute to indicate the location of the image. Without the *src* attribute, the image would not display.

```
<img id= "myImage" src = "path/to/my/image.jpg">
```

Additionally, we can use the height and/or width attributes to specify the size of the image,

```
<img id= "myImage" width = "200px" src = "path/to/my/image.jpg">
```

To modify the height and/or width of an image with JavaScript is illustrated below,

Let's return to our previous example. In this example, the cat image we inserted on our webpage is huge – 1000 px by 743 px. We can change the size of the image with JavaScript by using the *setAttribute* function,

|   |
|---|
| <b>App.js</b>   |
| <pre>1 var i = document.createElement("img"); 2 i.src = "cat.webp"; 3 i.alt = "cat butt" 4 document.body.append(i); 5 i.setAttribute("width", "200px");</pre>   |
| <b>HTML</b>   |
| <pre>...&lt;!DOCTYPE html&gt; == \$0 &lt;html lang="en"&gt;   &lt;head&gt; ... &lt;/head&gt;   &lt;body&gt;     &lt;img src="cat.webp" alt="cat butt" width="200px"&gt;   &lt;/body&gt; &lt;/html&gt;</pre> |
| <b>Output</b>   |
|   |

Notice that in this example, we only specified one dimension. This is good practice because the browser will scale the other dimension automatically. If we tried to specify both dimensions, we could end up with a distorted image,

|  |
|--|
| <b>App.js</b>  |
| <pre>1 var i = document.createElement("img"); 2 i.src = "cat.webp"; 3 i.alt = "cat butt" 4 document.body.append(i); 5 i.setAttribute("width", "300px"); 6 i.setAttribute("height", "200px");</pre> |



## HTML

```
...<!DOCTYPE html> == $0
<html lang="en">
  <head> ... </head>
  <body>
    
  </body>
</html>
```

## Output



Not all attributes require the *setAttribute* function to modify them. It just depends. For example, we saw previously that setting the id, or src attributes can be done using the dot notation,

```
1 var i = document.createElement("img");
2 i.src = "cat.webp";
3 i.alt = "cat butt" | Setting the src and id attributes can be done using the dot
4 document.body.append(i);
5 i.setAttribute("width", "300px");
6 i.setAttribute("height", "200px"); | Setting the width and height attributes
                                     requires the setAttribute function
```

### [Skill 21.05 Exercise 1](#)