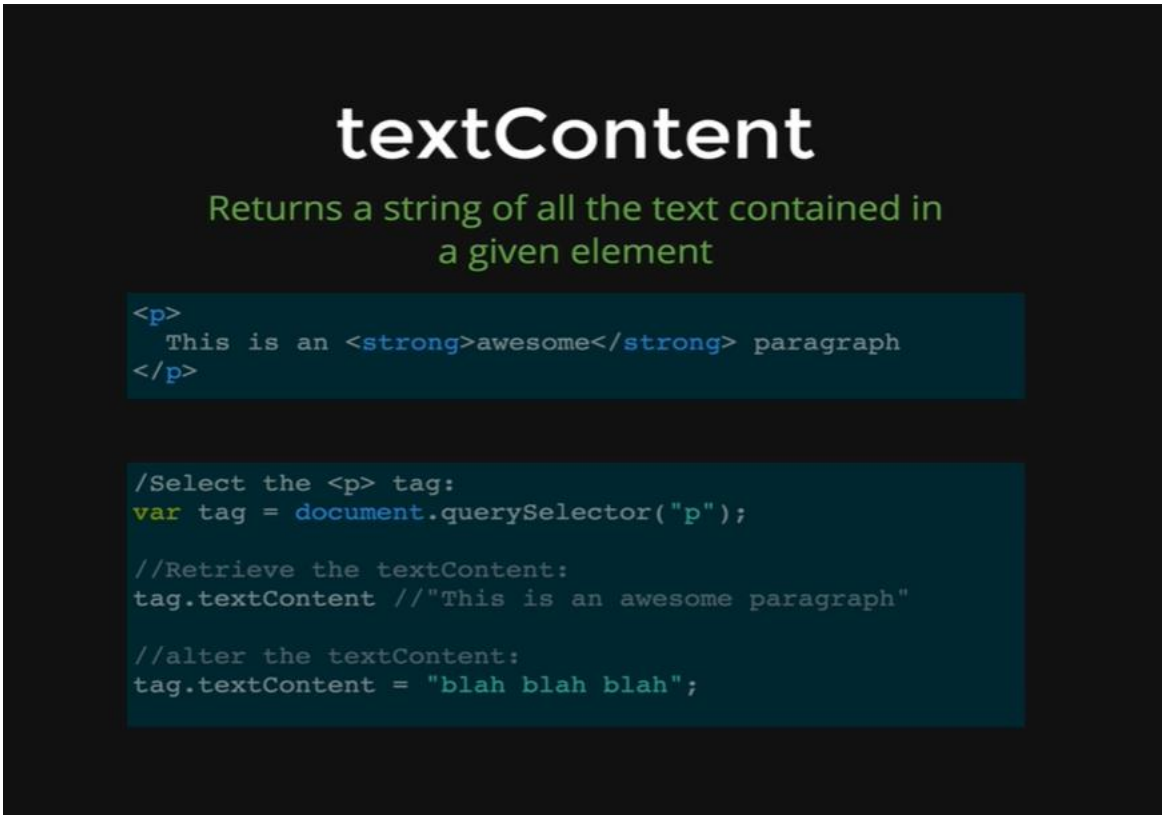
In this lesson we are going to talk about some other ways of Manipulating elements.



Remember, first we Select them and then we Manipulate them and so far, we have seen how we can change the color or any style properties and how we can use Class List to affect styling as well. In this lesson we are going to talk about two different properties both of which allow us to change the text or the HTML on the page.

If we want to change the content inside of an *h1* tag or inside of a s*trong* tag like we have on the picture or if we want to change an anchor tag and the text inside these tags. That is what we are going to see in this lesson.

As we can see on the picture, we have some texts in our paragraph tag and a *strong* tag that says *awesome* and then the rest of the paragraph texts. If we select this text like below,

var tag = document.querySelector("p");

We are selecting the whole *paragraph* text and we are storing it in *tag* variable, and we are using this variable to manipulate the text content property of this tag,

tag.textContent

This will retrieve the tags inside of that HTML element and text is defined as anything between HTML tags but not including the tags. That is why it returns,

This is an awesome paragraph

It does not contain the strong tags. It just extracts all the texts inside of that paragraph. However, many deep of layers it needs to go, it grabs all the texts and returns one big string. We can also set *textContent* like we did on the picture,

tag.textContent = "blah blah blah";

That will set the text content to be *blah blah blah* inside the paragraph tags.

```
<p>
    blah blah blah
</p>
```

We are going to go ahead and demonstrate how we can use the *textContent* property. Before that lets see the HTML page that we will be working with.

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Demo</title>
    <style type = "text/css">
        .big {
            font-size: 100px;
            color: orange;
```

```
                    border: 5px solid red
                }
        </style>
</head>
<body>

<h1>Welcome to My DOM</h1>

<p>Corgi mixes are <strong>super</strong> adorable</p>

<ul>
        <li>Orchids</li>
        <li>Succulents</li>
        <li>Tulips</li>
</ul>

<img src= http://barrelhorseworld.com/dogs/images/1145556d.jpg>
<img src=
http://api.ning.com/files/8Ni7RKwUY0n21uCgBw2JAxhPqmlBSklPmAiXbElaMFfa30mZFQKiE5gcWZP2iZJlSLIrtvQ3j4UH3Ezo5ZIc84ovRRC2J2ZT/TobySnow20.jpg>

</body>
</html>
```

Output:

Console coding:

var p = document.getElementsByTagName("p")[0];

p

➔ * ... *

We get our *p* document object and let's see what text it gives us when we run the *textContent* property on this tag name.

Console coding:

p.textContent

➔ *“Corgi mixes are super adorable”*

It gets all the texts from the paragraph tags and returns them to us through the *textContent property.*

Let’s do the same thing on our *ul* tags.

Console coding:

var ul = document.querySelector(“ul”);

ul

➔ *<ul>*
*
Orchids
*
*
Succulents
*
*
Tulips
*
*
*

ul.textContent

➔ *“ Orchids Succulents Tulips “*

We can see that it gets rid of the *li* tags and it returns us the pure text contents from inside of the *ul.*

Now, we can also update these contents. This is a little bit dangerous, because let’s say we want to update the paragraph’s text contents and we want to update it to say,

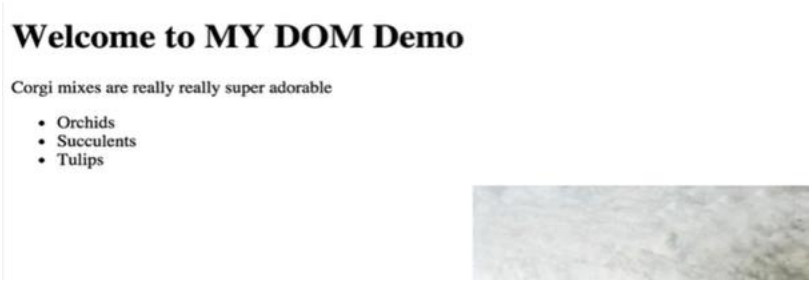
Corgi mixes are very super adorable.

If we make this update, we are going to get rid of the *strong* tags those were around the word *super*.

Console coding:

p.textContent = "Corgi mixes are very super adorable";

➔ *"Corgi mixes are very super adorable"*

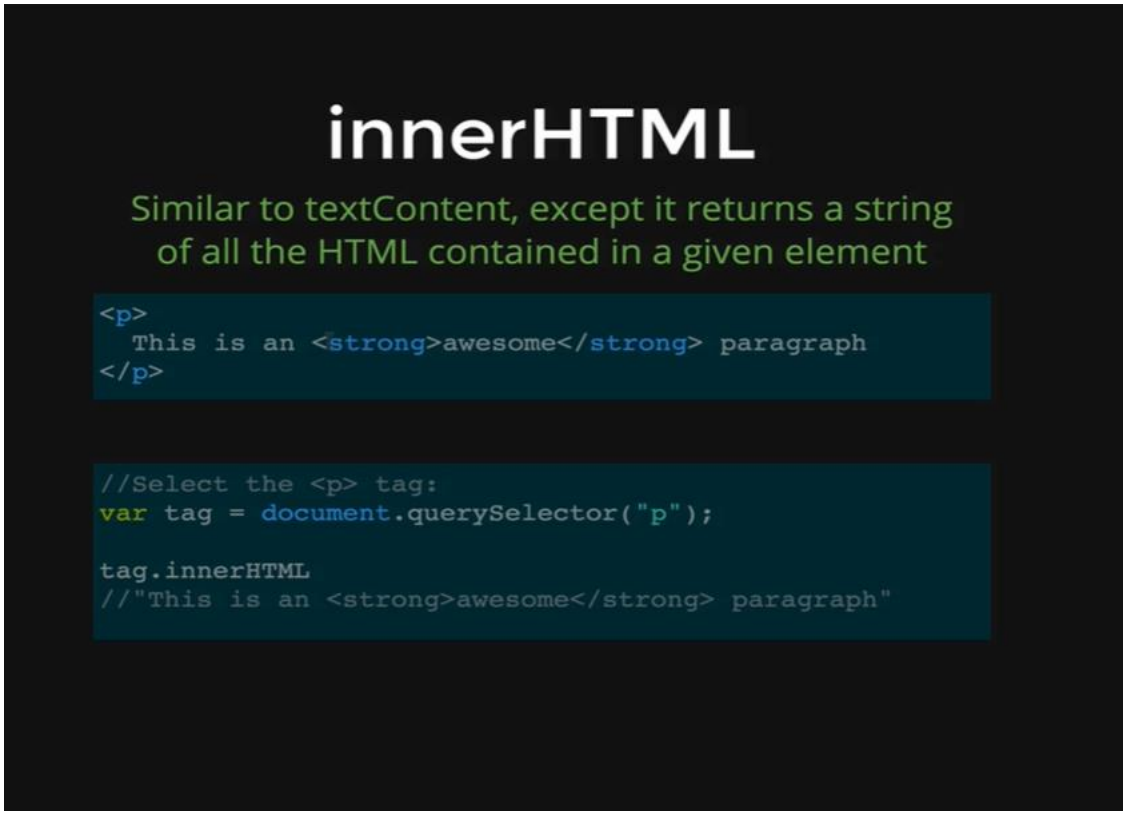


As we can see on the picture, the *textContent* property completely overrides the text by getting rid of the s*trong* tags those were inside the text contents.

That is why text content is a little bit dangerous in that sense, if we are using it to set something, its not bad but we just need to be careful about it. We need to know what is inside of our HTML elements, if there are HTML tags that we want to preserve, we cannot just automatically set the text content to be a new string, because that will erase everything.

There is another property that works along the same line as text content, except it keeps the HTML tags inside the contents intact. It does not just extracts the text; it maintains the structure with correct HTML tags as well as the texts while returning it.

That property is called *innerHTML*.



We have our same paragraph tags containing the same HTML contents, with *strong* tags around the word *awesome.* Then we select the content using *document.querySelector,*

var tag = document.querySelector("p");

and then we run the *innerHTML* property on the tag variable.

tag.innerHTML

- ➔ *"This is an awesome paragraph"*

This gives us the HTML contents while keeping the strong tags inside the contents.

Let's demonstrate this on our website console.

Console coding:

var p = document.querySelector("p");

Now, lets review the *textContent* property first,

p.textContent

➔ *"Corgi mixes are super adorable"*

Now, lets test the *innerHTML* property.

p.innerHTML

➔ *"Corgi mixes are super adorable".*

We can see that we get the strong tag returned to us with the HTML contents.

Let's do the same thing for the *ul*s and *li*s those are inside of our HTML contents.

var ul = document.querySelector("ul");

ul.innerHTML

➔ *"
Orchids Succulent Tulips
"*

We get the *li* tags in our *ul* tags as well.

This is useful and we will see a few situations where we are going to take advantage of it with some of the projects that we build.

textContent is also useful if we are changing what's inside of a tag, so usually we do not set *innerHTML* to be something, because it is the same problem that we would have with setting the text content, it just overrides whatever is currently inside of *innerHTML*.

ul.innerHTML = "Plants are awesome!";

➔ *"Plants are awesome!"*

It just completely overrides it, just what it did with *textContent.*

That's why if we want to add new elements, or we want to interact with HTML, there are other ways of doing that, but if all we want to do is to change the title on our webpage,

Welcome to MY DOM Demo

Then we need to select our HTML tags, and we do not have any HTML tags inside of the *h1*. Its just an *h1* content. That is why we can use either *textContent* or *innerHTML*. It does not matter which one we use, but most people will use *textContent*, because its all there is, and we can reset it to be something else. We are not going to set it to a variable, so we get used to this type of coding.

Console coding:

document.querySelector("h1").textContent = "END OF THIS LESSON!"

➔ *"END OF THIS LESSON!"*

We can see that our changes got reflected up on our site.

To recap, *textContent* and *innerHTML* are two different ways of grabbing the content of a given element. We select the text first using the HTML tags it in, and then we run *innerHTML* or *textContent* on those selected content.

textContent will extract only the text and return a big string full of texts and *innerHTML* will give us everything as a string, including the HTML elements.