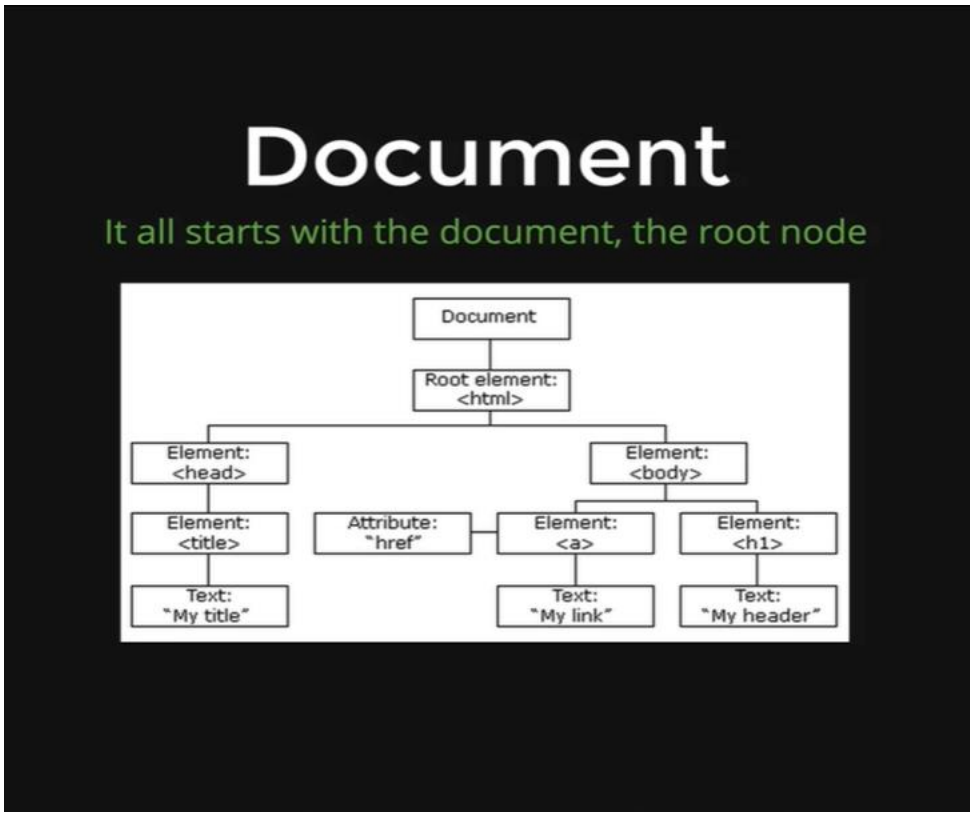
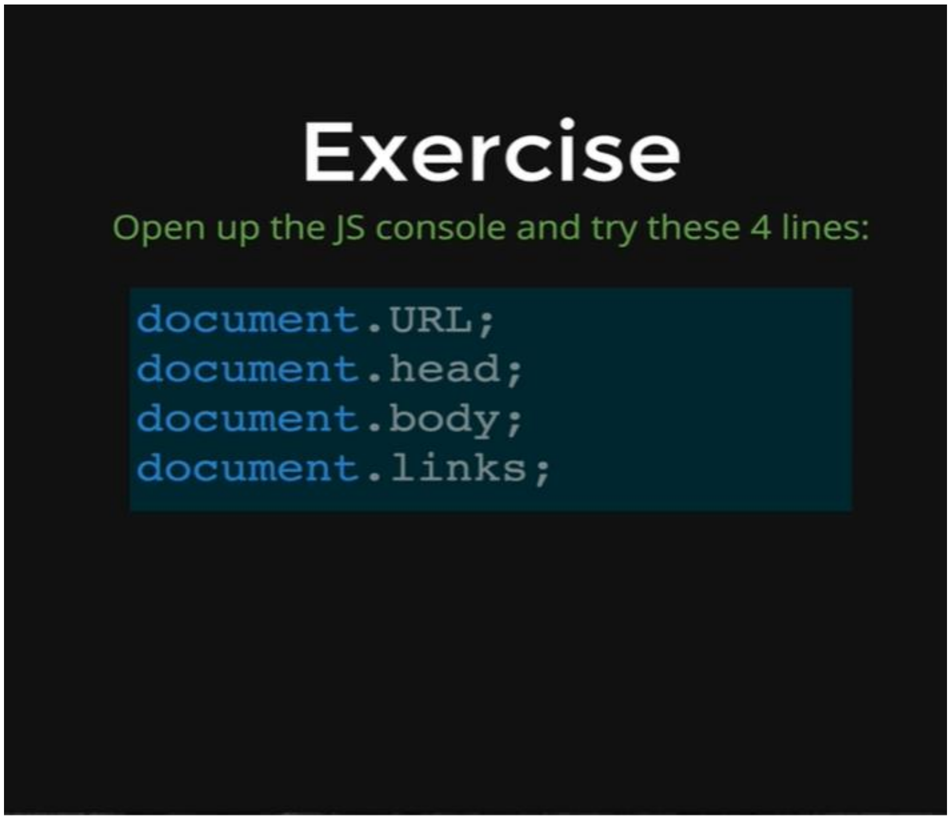
In this lesson we are going to talk about different ways of selecting elements by using JavaScript and the DOM. Before we talk about the actual selectors, we need to review the document one more time,



Remember that our entire DOM, all the objects, all the representation of our elements, all live inside this *Document Object.* It's the top-level object, or the root node.



Exercise

Open up the JS console and try these 4 lines:

```
document.URL;
document.head;
document.body;
document.links;
```

We need to type these four lines, one at a time on the console of our Google site and see what output it provides us.

document.URL

- ➔ *https://www.google.ca/*

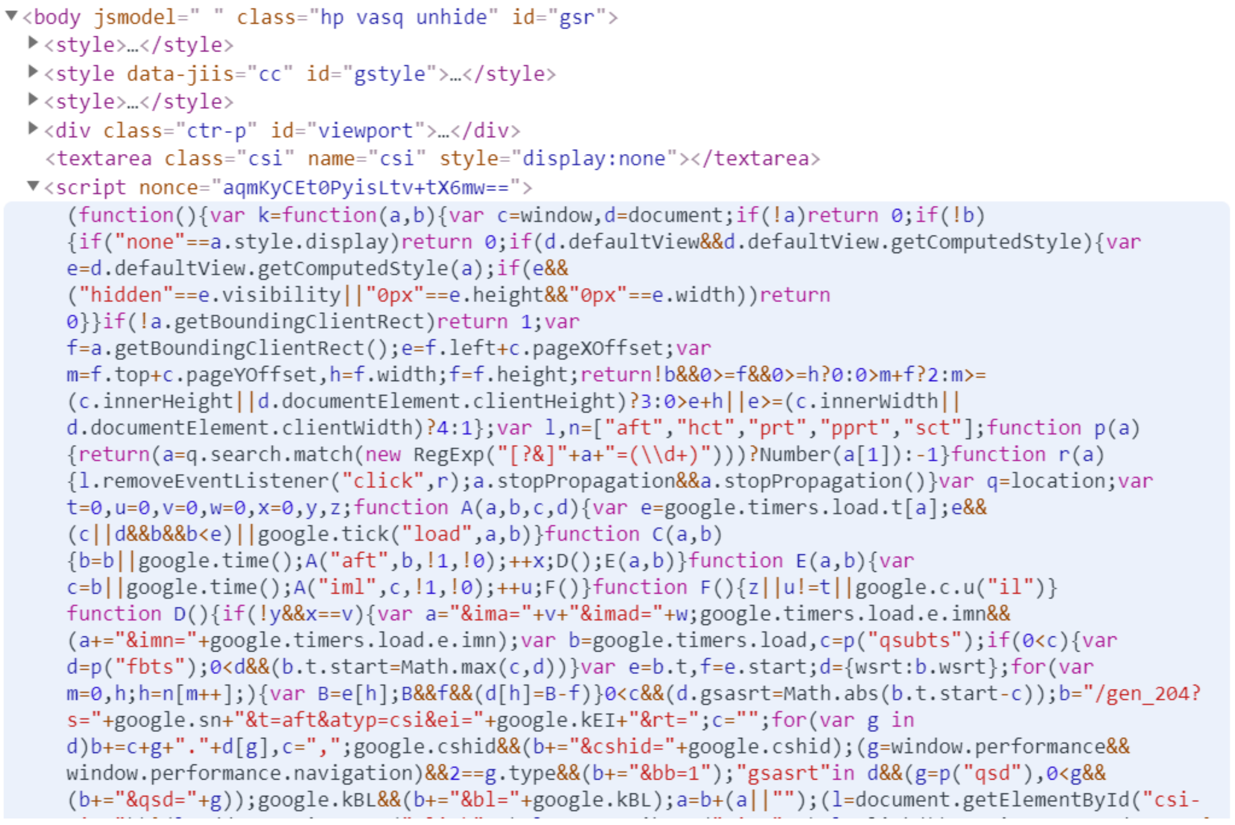
It shows us the URL of the page we are on.

document.links

- ➔ *HTMLCollection(54) [a, a, a.gb_e, a.gb_e, a.gb_B.gb_pc, a#gb192.gb_d, a#gb1.gb_d, a#gb8.gb_d, a#gb36.gb_d, a#gb78.gb_d, a#gb5.gb_d, a#gb23.gb_d, a#gb53.gb_d, a#gb49.gb_d, a#gb24.gb_d, a#gb51.gb_d, a#gb31.gb_d, a#gb6.gb_d, a.gb_I.gb_9f, a#gb27.gb_d, a#gb25.gb_d, a#gb10.gb_d, a#gb30.gb_d, a#gb461.gb_d, a#gb300.gb_d, a#gb136.gb_d, a#gb357.gb_d, a#gb265.gb_d, a#gb429.gb_d, a#gb338.gb_d, a.gb_1f.gb_J, a.gb_B.gb_Da.gb_g, a.gb_db.gb_7a.gb_8a.gb_9f, a.gb_lb.gb_5f.gbp1.gb_Pe.gb_pb, a.gb_Kb.gb_Vb, a.gb_8b.gb_va.gb_Ob, a.gb_sb.gb_2f,* *a#gb_71.gb_zb.gb_6f.gb_eg.gb_Pe.gb_pb, a.gb_nb.gb_Db, a.gb_nb.gb_Cb, a#sbfblt.duf3.aciXEb, a, a.Fx4vi, a.Fx4vi, a#fsettl.Fx4vi, a, a, a, a, a, a#dk2qOd, a.Fx4vi, a.Fx4vi, a.Fx4vi, gb192: a#gb192.gb_d, gb1: a#gb1.gb_d, gb8: a#gb8.gb_d, gb36: a#gb36.gb_d, gb78: a#gb78.gb_d, …]*

And it gives us a whole list of every single link on the page, every single anchor tag.

document.body

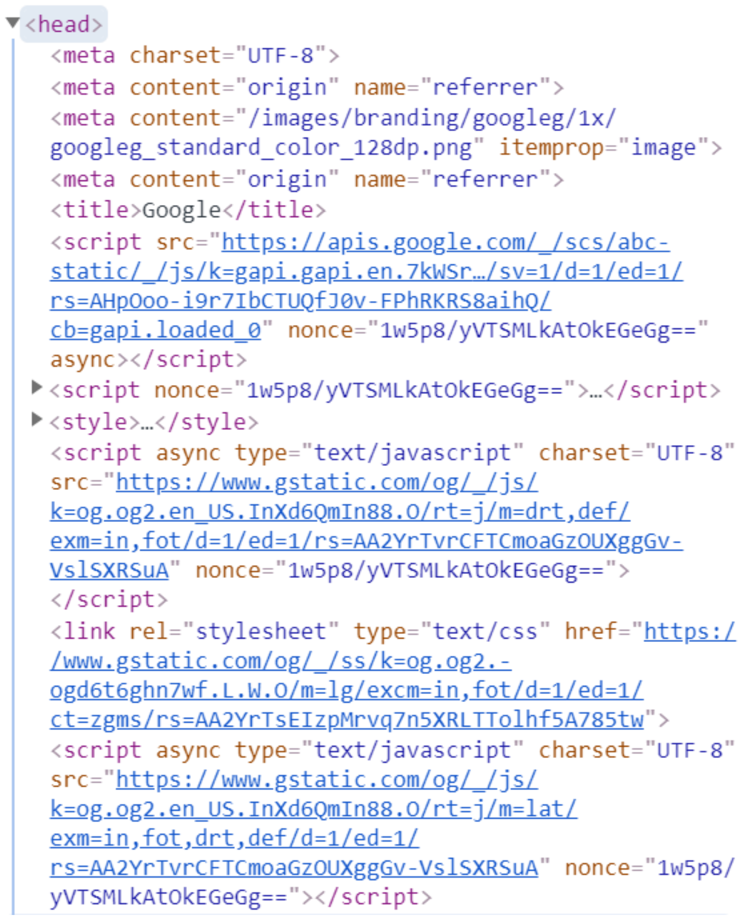


```
▼<body jsmodel=" " class="hp vasq unhide" id="gsr">
  ▶<style>…</style>
  ▶<style data-jiis="cc" id="gstyle">…</style>
  ▶<style>…</style>
  ▶<div class="ctr-p" id="viewport">…</div>
   <textarea class="csi" name="csi" style="display:none"></textarea>
  ▼<script nonce="aqmKyCEt0PyisLtv+tX6mw==">
     (function(){var k=function(a,b){var c=window,d=document;if(!a)return 0;if(!b)
     {if("none"==a.style.display)return 0;if(d.defaultView&&d.defaultView.getComputedStyle){var
     e=d.defaultView.getComputedStyle(a);if(e&&
     ("hidden"==e.visibility||"0px"==e.height&&"0px"==e.width))return
     0}}if(!a.getBoundingClientRect)return 1;var
     f=a.getBoundingClientRect();e=f.left+c.pageXOffset;var
     m=f.top+c.pageYOffset,h=f.width;f=f.height;return!b&&0>=f&&0>=h?0:0>m+f?2:m>=
     (c.innerHeight||d.documentElement.clientHeight)?3:0>e+h||e>=(c.innerWidth||
     d.documentElement.clientWidth)?4:1};var l,n=["aft","hct","prt","pprt","sct"];function p(a)
     {return(a=q.search.match(new RegExp("[?&]"+a+"=(\\d+)")))?Number(a[1]):-1}function r(a)
     {l.removeEventListener("click",r);a.stopPropagation&&a.stopPropagation()}var q=location;var
     t=0,u=0,v=0,w=0,x=0,y,z;function A(a,b,c,d){var e=google.timers.load.t[a];e&&
     (c||d&&b&&b<e)||google.tick("load",a,b)}function C(a,b)
     {b=b||google.time();A("aft",b,!1,!0);++x;D();E(a,b)}function E(a,b){var
     c=b||google.time();A("iml",c,!1,!0);++u;F()}function F(){z||u!=t||google.c.u("il")}
     function D(){if(!y&&x==v){var a="&ima="+v+"&imad="+w;google.timers.load.e.imn&&
     (a+="&imn="+google.timers.load.e.imn);var b=google.timers.load,c=p("qsubts");if(0<c){var
     d=p("fbts");0<d&&(b.t.start=Math.max(c,d))}var e=b.t,f=e.start;d={wsrt:b.wsrt};for(var
     m=0,h;h=n[m++];){var B=e[h];B&&f&&(d[h]=B-f)}0<c&&(d.gsasrt=Math.abs(b.t.start-c));b="/gen_204?
     s="+google.sn+"&t=aft&atyp=csi&ei="+google.kEI+"&rt=";c="";for(var g in
     d)b+=c+g+"."+d[g],c=",";google.cshid&&(b+="&cshid="+google.cshid);(g=window.performance&&
     window.performance.navigation)&&2==g.type&&(b+="&bb=1");"gsasrt"in d&&(g=p("qsd"),0<g&&
     (b+="&qsd="+g));google.kBL&&(b+="&bl="+google.kBL);a=b+(a||"");(l=document.getElementById("csi-
```

➔

This gives us the entire body with every other element inside of it, that lives in the body.

document.head



```
▼<head>
   <meta charset="UTF-8">
   <meta content="origin" name="referrer">
   <meta content="/images/branding/googleg/1x/
   googleg_standard_color_128dp.png" itemprop="image">
   <meta content="origin" name="referrer">
   <title>Google</title>
   <script src="https://apis.google.com/_/scs/abc-
   static/_/js/k=gapi.gapi.en.7kWSr…/sv=1/d=1/ed=1/
   rs=AHpOoo-i9r7IbCTUQfJ0v-FPhRKRS8aihQ/
   cb=gapi.loaded_0" nonce="1w5p8/yVTSMLkAtOkEGeGg=="
   async></script>
  ▶<script nonce="1w5p8/yVTSMLkAtOkEGeGg==">…</script>
  ▶<style>…</style>
   <script async type="text/javascript" charset="UTF-8"
   src="https://www.gstatic.com/og/_/js/
   k=og.og2.en_US.InXd6QmIn88.O/rt=j/m=drt,def/
   exm=in,fot/d=1/ed=1/rs=AA2YrTvrCFTCmoaGzOUXggGv-
   VslSXRSuA" nonce="1w5p8/yVTSMLkAtOkEGeGg==">
   </script>
   <link rel="stylesheet" type="text/css" href="https:/
   /www.gstatic.com/og/_/ss/k=og.og2.-
   ogd6t6ghn7wf.L.W.O/m=lg/excm=in,fot/d=1/ed=1/
   ct=zgms/rs=AA2YrTsEIzpMrvq7n5XRLTTolhf5A785tw">
   <script async type="text/javascript" charset="UTF-8"
   src="https://www.gstatic.com/og/_/js/
   k=og.og2.en_US.InXd6QmIn88.O/rt=j/m=lat/
   exm=in,fot,drt,def/d=1/ed=1/
   rs=AA2YrTvrCFTCmoaGzOUXggGv-VslSXRSuA" nonce="1w5p8/
   yVTSMLkAtOkEGeGg=="></script>
```

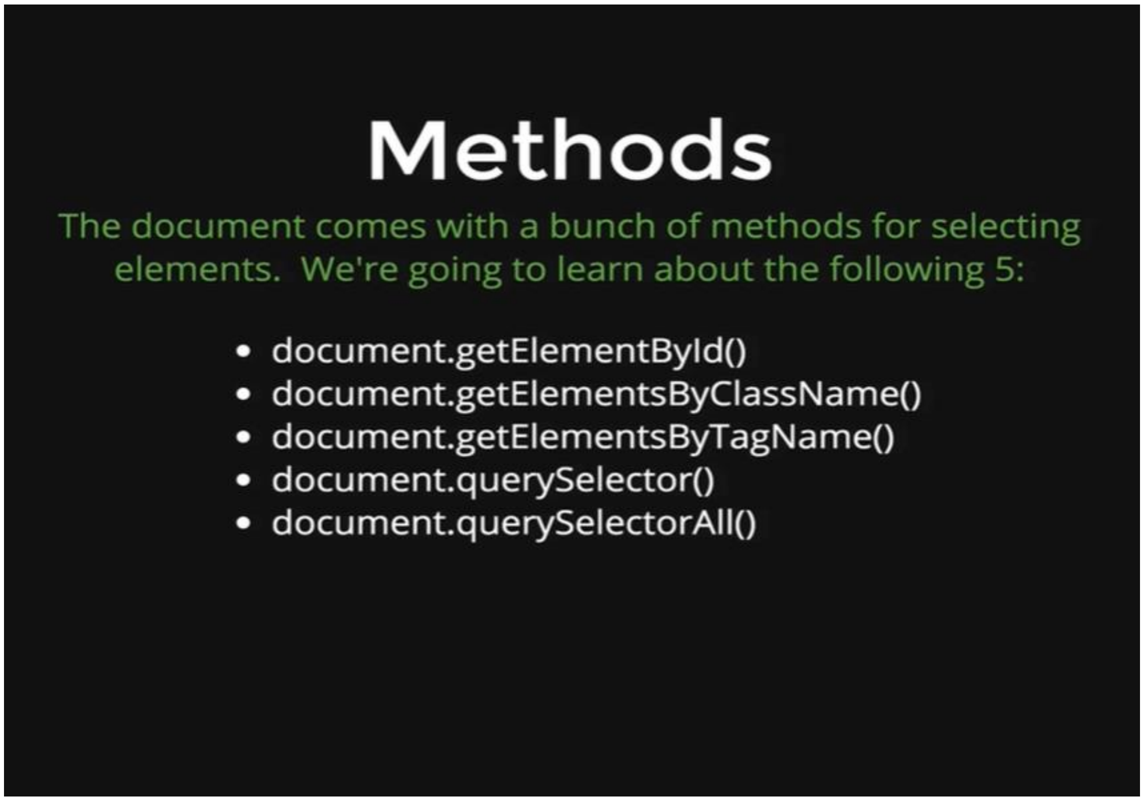
➔

We have already seen this in last couple of lessons. The reason we are visiting it again is because all the selectors, functions or methods that we are going to learn, also live inside the *Document.*

Anything we are going to learn in this lesson, is going to start with,

document.something

Let’s see what those *something*s are,



Methods

The document comes with a bunch of methods for selecting elements. We're going to learn about the following 5:

- document.getElementById()
- document.getElementsByClassName()
- document.getElementsByTagName()
- document.querySelector()
- document.querySelectorAll()

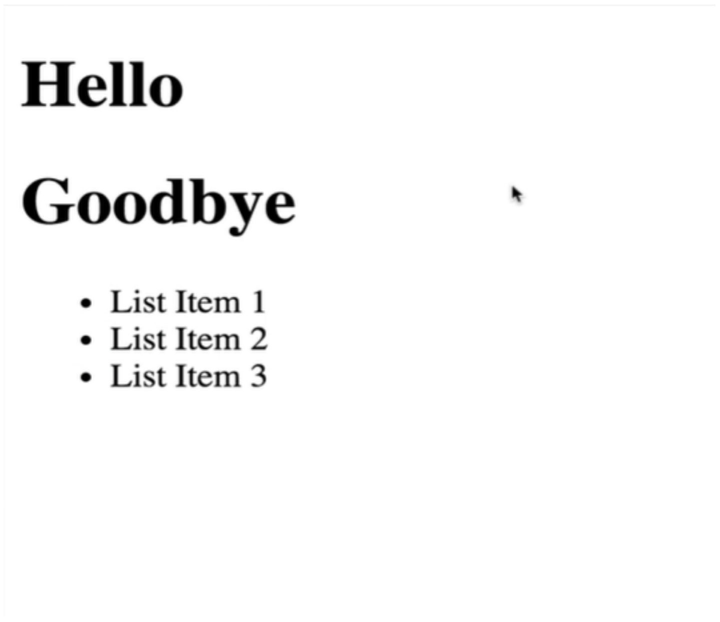
There are five main built-in methods that we are going to talk about in this lesson. They are all built-in to the document. We can see them on the picture. These five methods are added into the *Document Object.*

Before we move on, lets see the HTML code that we are going to use for the demonstration purposes.

```
<html>
    <head>
        <title>Selectors Intro</title>
    </head>
    <body>
        <h1>Hello</h1>
        <h1>Goodbye</h1>
        <ul>
```

```
                    <li id= “highlight”>List Item 1</li>
                    <li class= “bolded”>List Item 2</li>
                    <li class= “bolded”>List Item 3</li>
            </ul>
        </body>
</html>
```

Output:



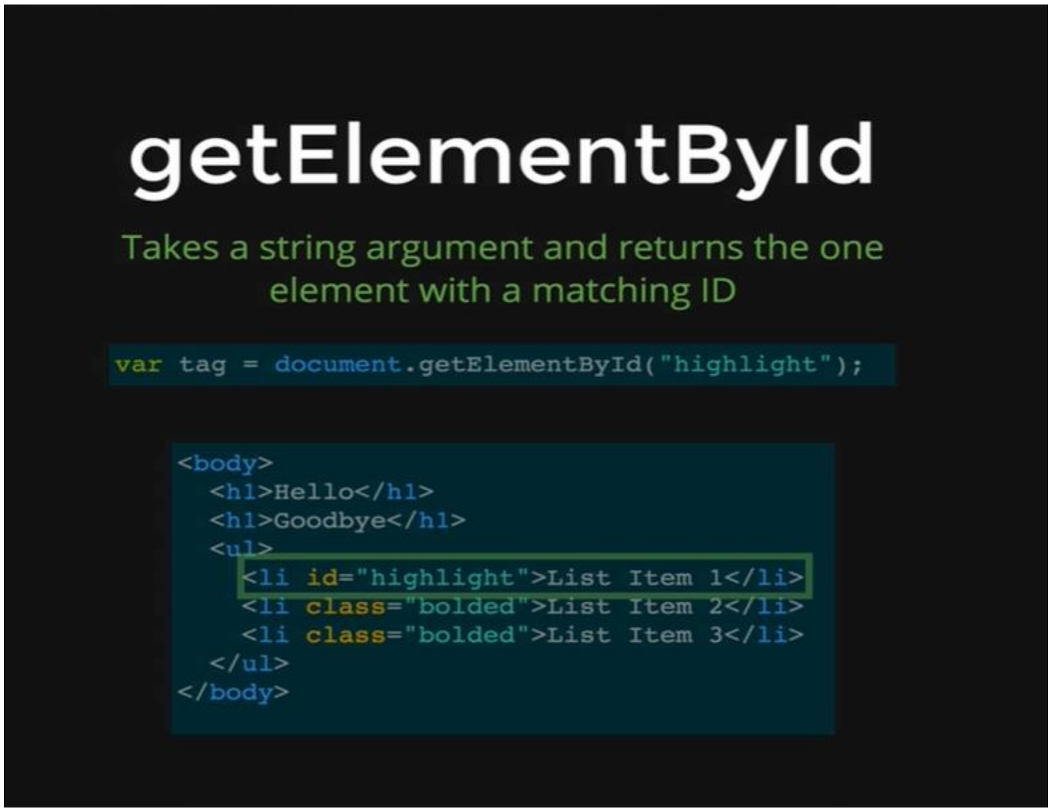
```
Hello

Goodbye

    • List Item 1
    • List Item 2
    • List Item 3
```

So, this is the HTML page that we will be using.

Let’s get started with the first method,



getElementById

Takes a string argument and returns the one element with a matching ID

```
var tag = document.getElementById("highlight");
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

1. *getElementById*

   Its name is a little bit self-explanatory. It takes in an ID name and it is going to return that one element that matches the ID. Remember that an ID can occur only one time at a page. We have an example on the picture.

   document.getElementById(“highlight”) is going to select the *li* with *id = highlight.*

   It selects the object and then it returns it to us. Let’s demonstrate this method by writing the method on our console.

   Console coding:

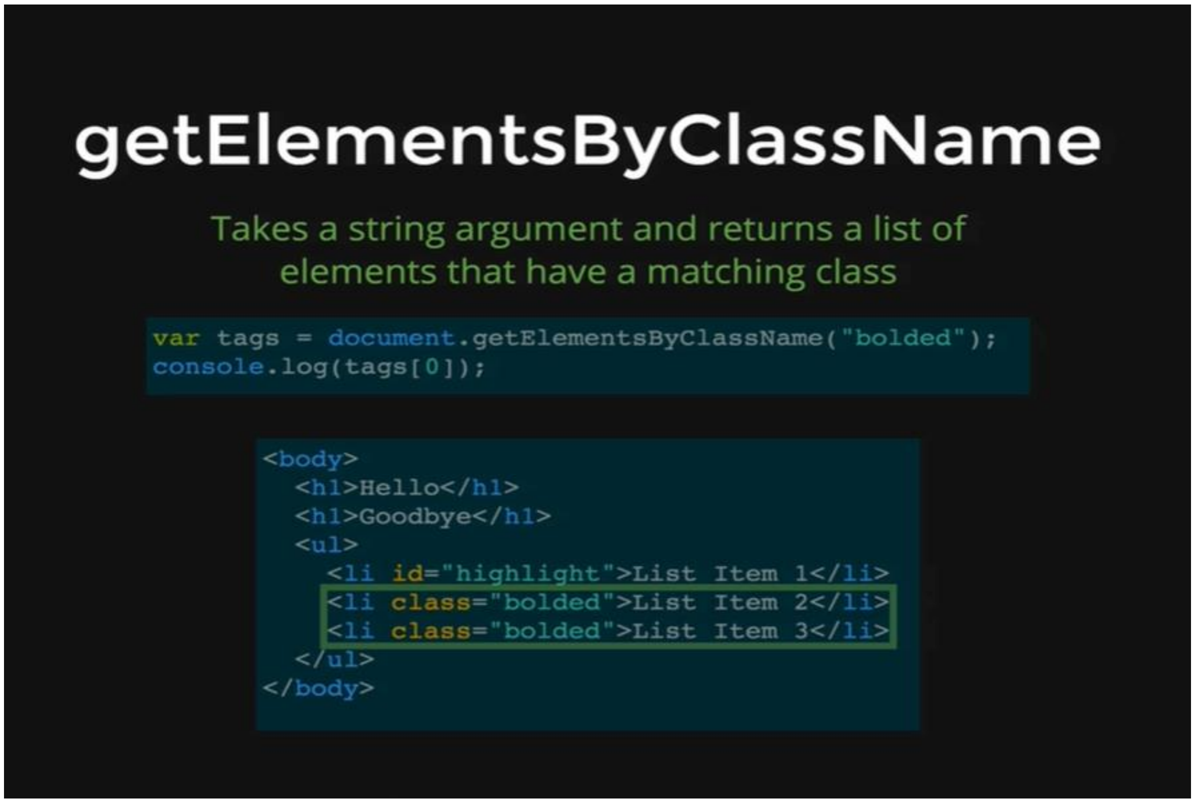
   var tag= document.getElementById(“highlight”);

tag

➔ *<li id= "highlight">List Item 1</li>*

We can see that it gives us the first *li* that has *id= highlight.* We selected something by an *id.* It is showing us as if it is an HTML, but the result of this is a JavaScript Object.

That is all there is to get element by ID. We call the method, we pass in a single string argument, the name of an ID that we want and then it goes and finds the matching element that has the same ID and it returns the object representation to us.



getElementsByClassName

Takes a string argument and returns a list of elements that have a matching class

```
var tags = document.getElementsByClassName("bolded");
console.log(tags[0]);
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

2. *getElementsByClassName*

   The name tells us exactly what it does, just like *getElementById.* Although this is slightly different, as it has *Elements* rather than *Element*. This is because a class can occur as many times as we want on a page. So, we pass in the string *"bolded"* and that is going to return us a list of all the elements in the page that has the matching class names *"bolded".*

In this case we have two elements with class name *"bolded",* but if there were 10,000 elements in a page that had that class name then we would get all 10,000 elements in a list.

Let's demonstrate this on our console.

Console coding:

```
var tags= document.getElementsByClassName("bolded");

tags

➔ [        <li class= "bolded">List Item2</li>,
           <li class= "bolded">List Item3</li>          ]
```

We can see the *tags* variable contains a list of two *li*s both with class names equal to *"bolded".* Technically, its not an Array, its something called a node list, which is array-like. Think of it as a lightweight array. It comes with some of the things that we would expect from an Array, but some of the advanced Array features are missing from this list.

For instance, we can access an element from this list using an index,

```
tags[0]
  ➔ <li class="bolded">List Item 2</li>
```

We can find out its length using the length property,

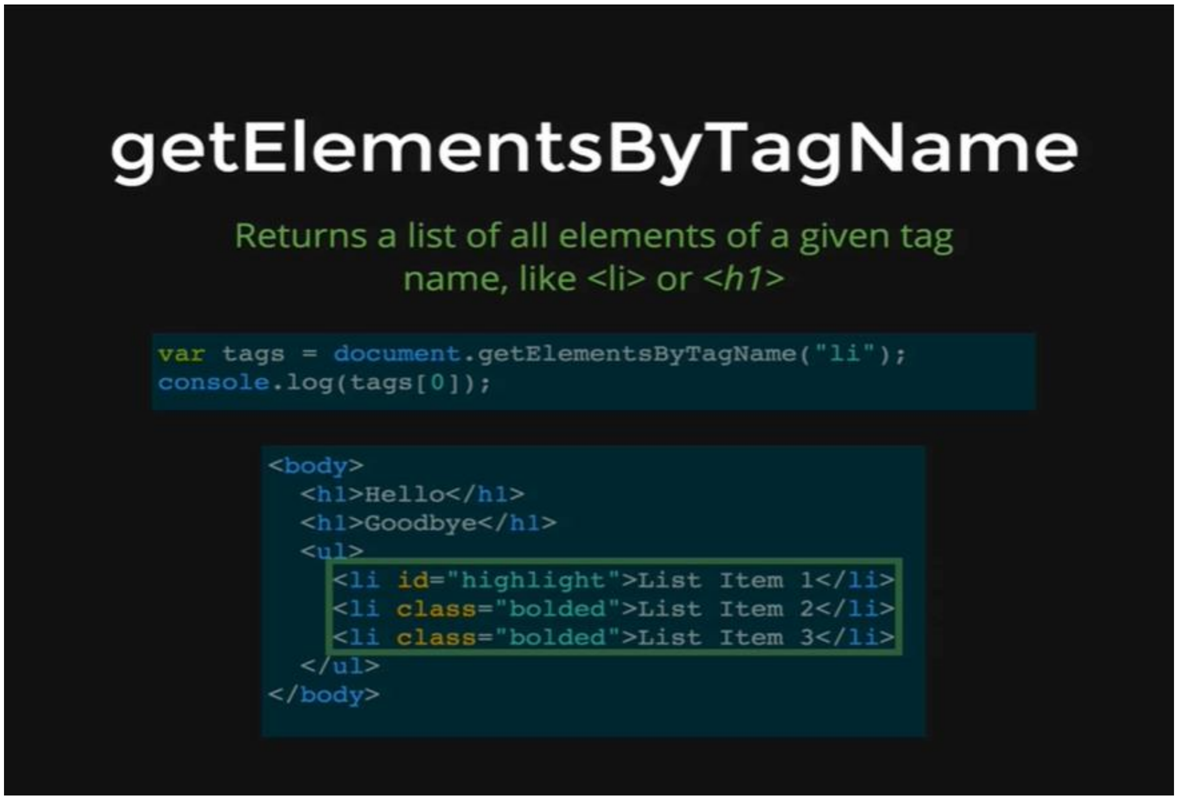
```
tags.length
  ➔ 2
```

But we cannot use a forEach property in it, *tags.forEach()*

➔ *ERROR*

The error tells us *tags.forEach* is not a function. That is because there is no forEach defined for these node lists. That is defined for Arrays and these are not Arrays.

Just like *getElementById, getElementsByClassName* shows us a nice string representation, but behind the scene it is an Object.



getElementsByTagName

Returns a list of all elements of a given tag name, like <li> or <h1>

```
var tags = document.getElementsByTagName("li");
console.log(tags[0]);
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

3. *getElementsByTagName*

This one works just like *getElementsById* or *getElementsByClassName*, except it returns a list of elements that match a given tag name like *li* or *h1*. In this case we are running,

document.getElementsByTagName("li");

This is going to return to us the list of 3 *li*s that exist on this page.

Let's demonstrate that on our console.

Console coding:

```
var tags= document.getElementsByTagName("li");

tags
➔ [      <li id= "highlight">List Item 1</li>,
         <li id= "bolded">List Item 2</li>,
         <li id= "bolded">List Item 3</li>   ]
```

We get this list, and again its not quite an Array, it's a node list that contains three *li*s. These are objects, they are not strings or HTMLs. They are JavaScript Objects with all the properties that we showed earlier.

We can do the same thing for *h1*s.

```
var tags= document.getElementsByTagName("h1");
tags
  ➔ [ <h1>Hello</h1>, <h1>Goodbye</h1> ]
```

We get two *h1*s in a list. Its not just limited to tags that we see inside our HTML body, we can also do something like below,

```
var tags = document.getElementsByTagName("body");

tags
  ➔ [ <body> ... </body> ]
```

It returns to us a list with one element, with the body inside of it. We can also do it with the head.

var tags= document.getElementByTagName(“head”);

tags

- ➔ *[<head> ... </head>]*

We can also do it for the *HTML* element or the *title* or whatever we want, the important part is that it returns a list, even if there is only one element, it still returns a list.

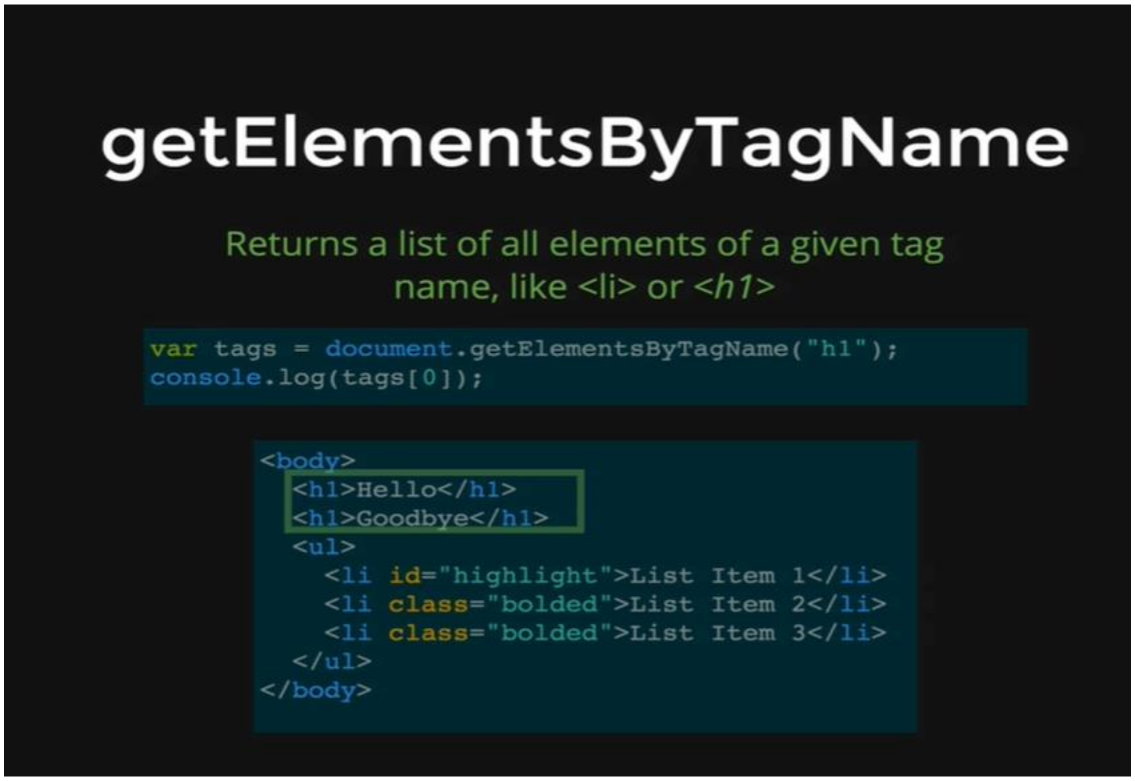
To select the body we can do like,

var body = document.getElementByTagName(“body”)[0]

body

- ➔ *<body> ... </body>*

Here in the *body* variable we get the giant object with the entire body. We used 0 in our square bracket because there is only one body tag in every HTML page, that is why there is only one item in the list and that is the body, which is going to be returned to us as the only element in the list with index 0.



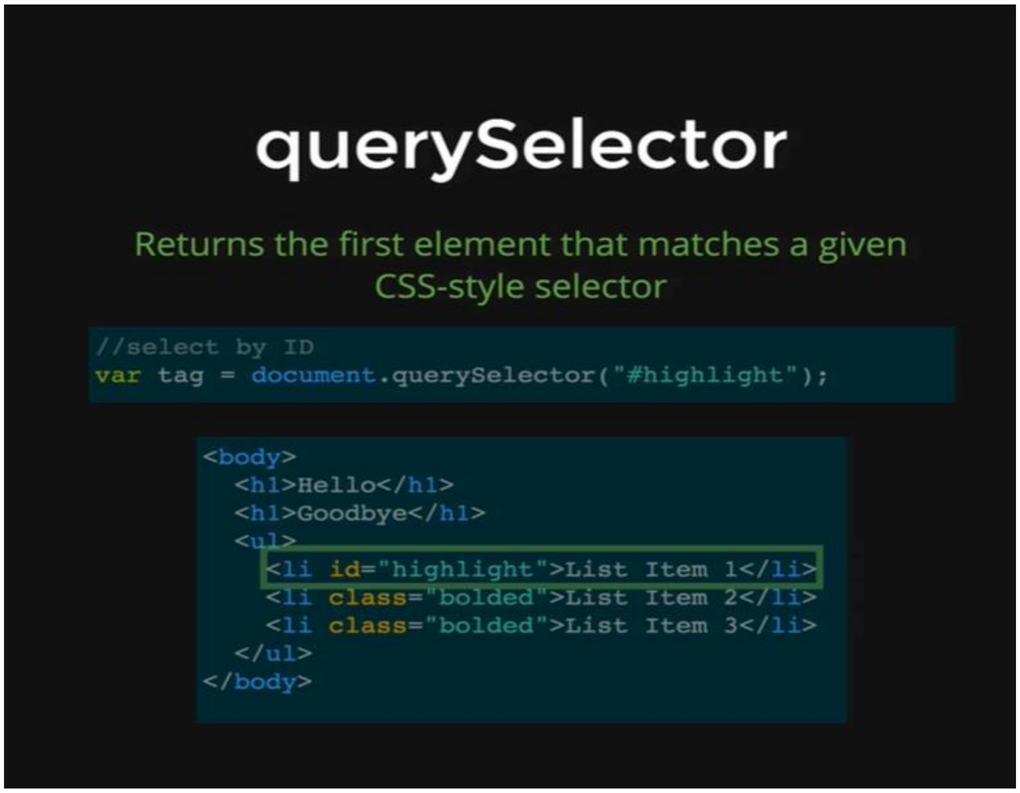
getElementsByTagName

Returns a list of all elements of a given tag name, like <li> or <h1>

```
var tags = document.getElementsByTagName("h1");
console.log(tags[0]);
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

Here we have another example selecting *h1*s, we can see that it selects the two and only *h1*s in the body.



querySelector

Returns the first element that matches a given CSS-style selector

```
//select by ID
var tag = document.querySelector("#highlight");
```

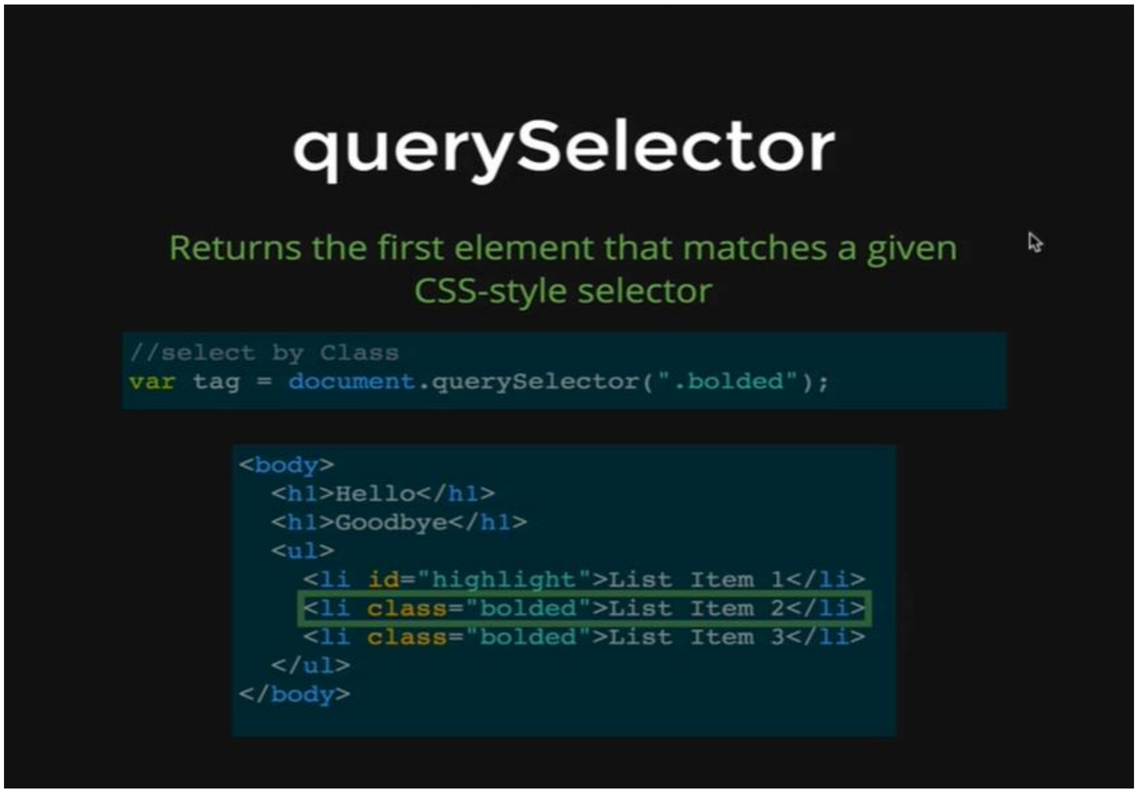
```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

4. *querySelector*

Now we change our gears a little bit, we no longer have a *getElement* or *getElementsBy* something syntax, this is called *querySelector.* It is a newer method that has not existed for all that long and it makes our lives a lot easier. We can use this method to do everything like the previous methods, like *getElementById*, *getElementsByClassName* or *getElementsByTagName.* We can replicate all of them with *querySelector*.

This method takes a CSS style selector. A CSS style selector means any of the selectors we would use inside of a CSS document. In this example we are selecting something based off the *id* equal to *highlight.* We cannot just write the word *highlight*, like we did with *getElementById.* What we need to do is use the CSS syntax with the hash (#) symbol and then write *highlight.* This is how we select something with a query selector if we want an Id, just like we would do with CSS. That is why we get the first *li* item with *id* equal to *highlight.*

Before we demonstrate this on the console, we want to show that we can do the same thing with class name.



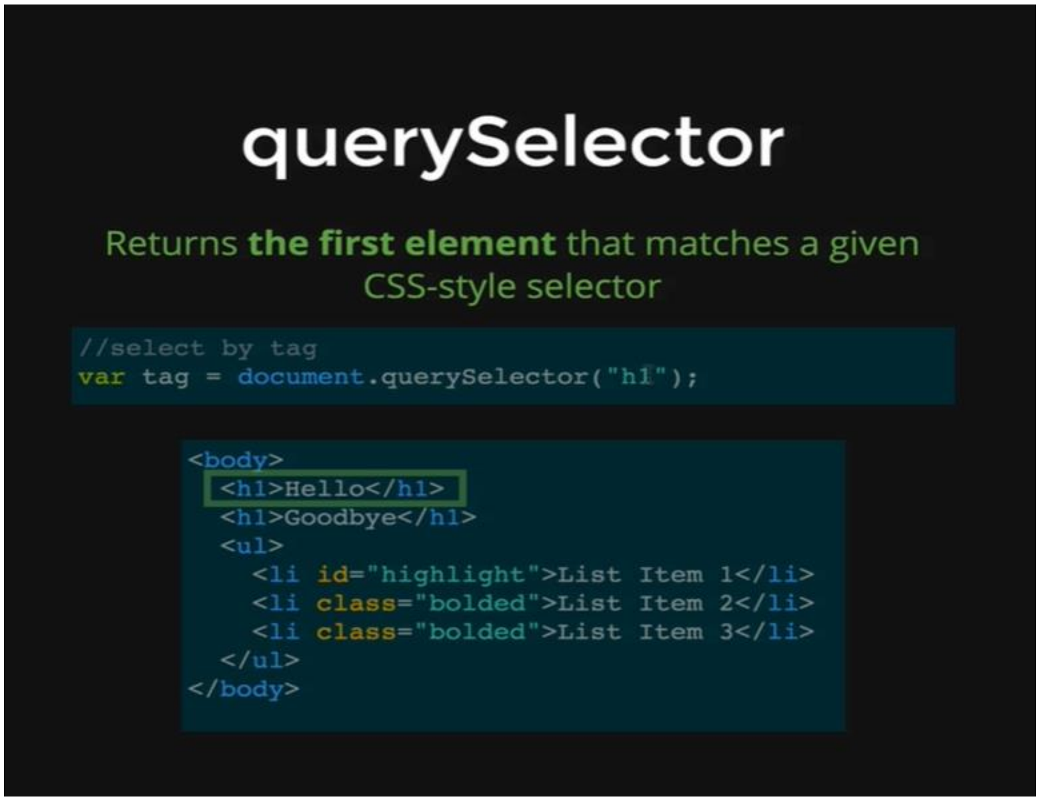
querySelector

Returns the first element that matches a given CSS-style selector

```
//select by Class
var tag = document.querySelector(".bolded");
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

As we can see on the picture, we passed the *bolded* class, by using the dot before the class name *bolded.* That is how we would select the *bolded* class using CSS and what is important, is that it only gives us the very first match. Even though there are two elements that has *bolded* as their class names, it only gives us the first one, and that is the point of *querySelector*, it always only returns one element. There is another method that we will see shortly that will return all the elements that match.


querySelector

Returns **the first element** that matches a given CSS-style selector

```
//select by tag
var tag = document.querySelector("h1");
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

querySelector can also take in a tag name like we have on the picture. We are passing the *h1* tag name and it returns us the very first *h1*. We can take CSS selectors that we write all the time on CSS and we can use that syntax to select elements with *querySelectors*.

Let's demonstrate that on the console.

Console coding:

var h1 = document.querySelector("h1");

h1

- ➔ *<h1>Hello</h1>*

var li = document.querySelector("#highlight");

li

- ➔ *<li id= "highlight">List Item 1</li>*

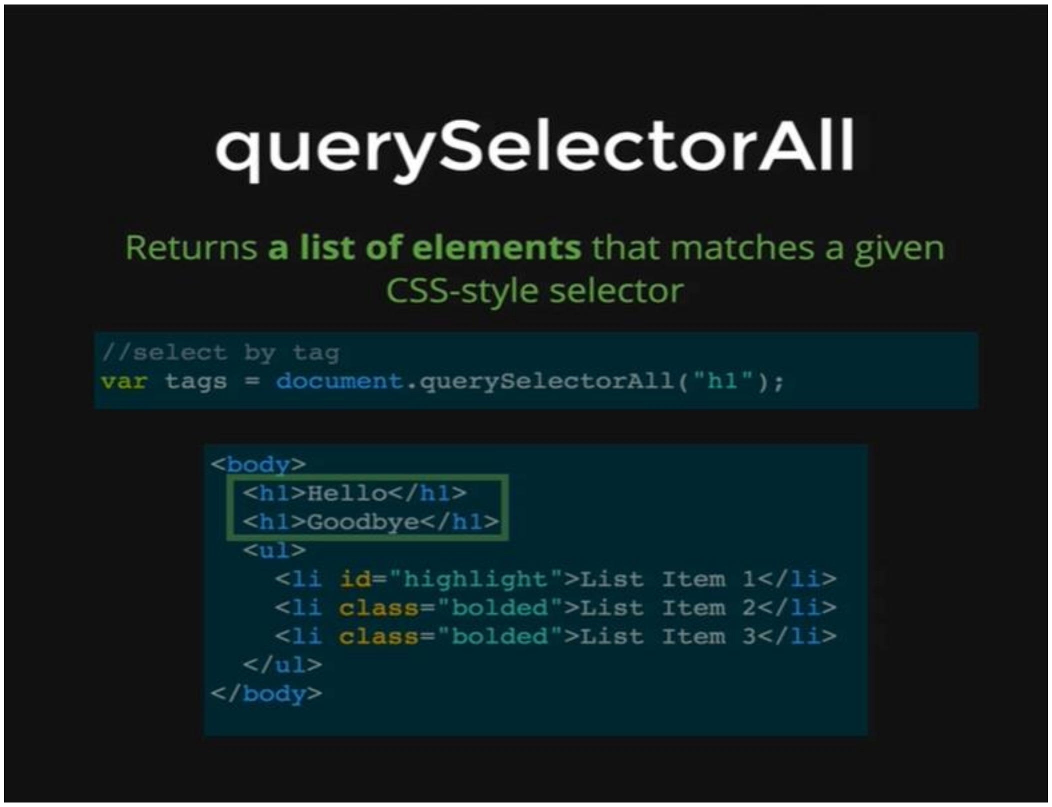
var li = document.querySelector(".bolded");

li

- ➔ *<li class= "bolded">List Item 2</li>*

As we mentioned *querySelector* returns the first match and that is it, sometimes it is useful, for instance if we wanted to select the body all we need to do is write *querySelector("body")* and that gives us the body; unlike when we did *document.getElementsByTagName("body")* that gave us a list and then we had to ask the first element. If we use the *querySelector* we would just get the body right away.

The alternative to this is *querySelectorAll.*



querySelectorAll

Returns **a list of elements** that matches a given CSS-style selector

```
//select by tag
var tags = document.querySelectorAll("h1");
```

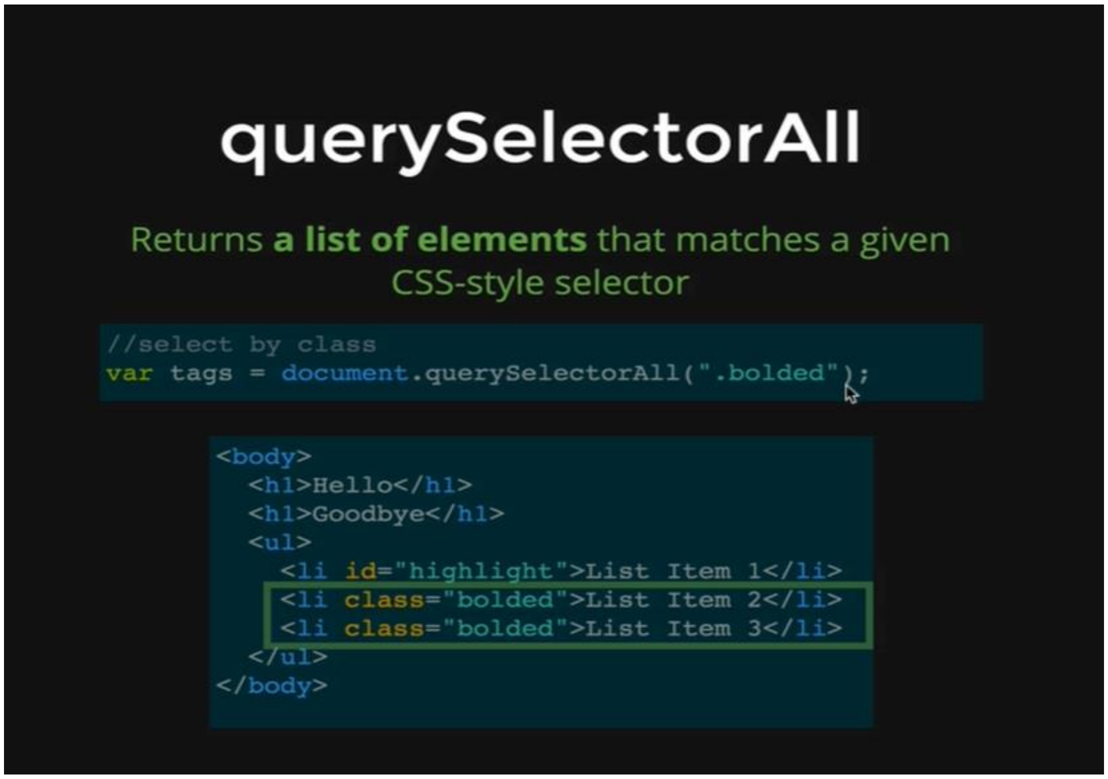
```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

5. *querySelectorAll*

   It works the same way; it takes a CSS selector, but it returns all matching elements. So, in this case we are using,

   documents.querySelectorAll("h1")

   And it returns us the two and only *h1*s in our HTML page.



querySelectorAll

Returns **a list of elements** that matches a given CSS-style selector

```
//select by class
var tags = document.querySelectorAll(".bolded");
```

```
<body>
  <h1>Hello</h1>
  <h1>Goodbye</h1>
  <ul>
    <li id="highlight">List Item 1</li>
    <li class="bolded">List Item 2</li>
    <li class="bolded">List Item 3</li>
  </ul>
</body>
```

In this case we are using the class *bolded*, so according to the CSS style we need to pass *".bolded"* and that gives us both elements with the classes equal to *"bolded"*.

Let's demonstrate this on the console.

Console coding:

var lis = document.querySelectorAll("li");

lis

➔ *[<li id= "highlight">List Item 1</li>,*
<li id= "bolded">List Item 2</li>,
<li id= "bolded">List Item 3</li>]

As we can see the *lis* variable has return us the list of all our *li*s in our HTML page. Again, this is a node list containing objects. They are not HTMLs, they are JavaScript Objects those are constructed from the HTMLs in our HTML page.

If we do the same thing that we did with *querySelector,*

var lis = document.querySelector("li");

lis

- ➔ *<li id= "highlight">List Item 1</li>*

We can see that we get only one item and not a list of all items.

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

bolded

- ➔ *<li class= "bolded">List Item 2</li>*

var bolded = document.querySelectorAll(".bolded");

bolded

```
➔ [     <li class= "bolded">List Item 2</li>
        <li class= "bolded">List Item 3</li>     ]
```

We can still select *querySelectorAll* if we are looking for one item.

var li = document.querySelectorAll("#highlight")

li

```
➔ [     <li id= "highlight">List Item 1</li>     ]
```

It’s a list with only one item, as we had only one *id* in our HTML page. Its rare that we will do that, but it will still work if there is only one match.