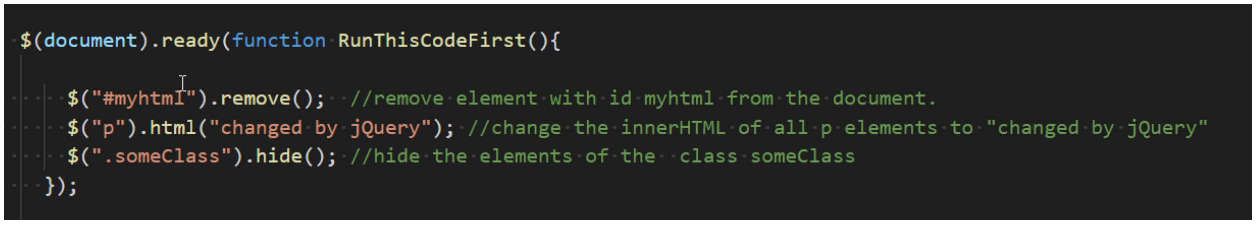
$(document).ready and window.onload

Lesson Time: 30 Minutes

In previous JS lessons, we've placed a <script> tag at the body of the body. One of the main reasons for this approach is to ensure the <script> only runs after all other elements of the HTML and CSS are loaded.

JQuery gives us a helpful way of making sure scripts are not run until the document has finished loading through the $(document).ready method.



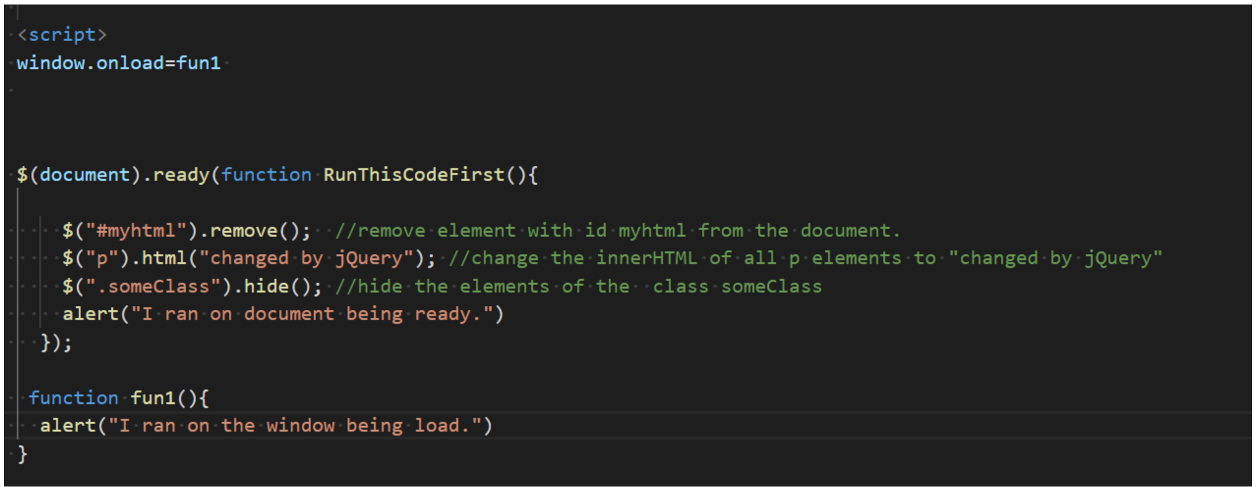
```
$(document).ready(function RunThisCodeFirst(){

    $("#myhtml").remove();  //remove element with id myhtml from the document.
    $("p").html("changed by jQuery"); //change the innerHTML of all p elements to "changed by jQuery"
    $(".someClass").hide(); //hide the elements of the  class someClass
  });
```

The statements inside of the document.ready will run as soon as the page finishes loading. We can use this to safely remove our <script> tag from the bottom of our HTML document and just link to in the html <head> now.

A few small details on this function.

- $(document).ready() code will run once the page is loaded, but BEFORE images are loaded.
- $(document).ready() will run AFTER the pure JS window.onload event.

Below is an example of both of these in action.



```
<script>
window.onload=fun1



$(document).ready(function RunThisCodeFirst(){

    $("#myhtml").remove();  //remove element with id myhtml from the document.
    $("p").html("changed by jQuery"); //change the innerHTML of all p elements to "changed by jQuery"
    $(".someClass").hide(); //hide the elements of the  class someClass
    alert("I ran on document being ready.")
  });

 function fun1(){
  alert("I ran on the window being load.")
}
```

When the page is loaded, the fun1 function will execute first due to being called by window.onload, and RunThisCodeFirst will be executed second due to being called by $(document).ready()

Together, these are both options for running JS code before anything else can happen, and are good ways to get your <script> tag out of the <body> of html and linked inside the <head>

Review the 11_2example.html to see this in action.

Callbacks

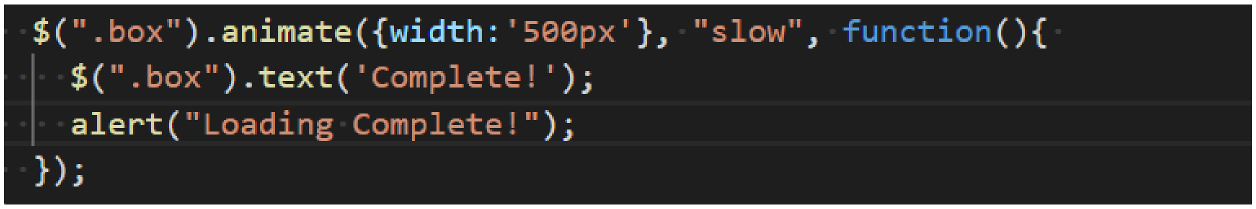
Lesson Time: 30 Minutes

JS has a powerful feature called callbacks. We didn't talk about them directly during the JS lesson, but they are available in vanilla JS, and we actually used them. JQuery is a good tool to illustrate what a callback is.

When we use a callback, we basically are saying that when some code has finished running, do something else *as soon as it's finished.* A good way to memorize this is to think of the code saying to you "Hey, I'm done, and now I'm calling you back!" You told me you wanted me to give you a call back when I finished!"

Our AJAX onreadystatechange we saw in the previous lesson is an example of a callback. In the readystatechange code, we wrote a callback to tell AJAX what to do after the HTTP communication was finished.

In JQuery, we have some cool effect and animation tricks we can use, and we can put a callback after the effect is finished. Here's a basic example.



```
$(".box").animate({width:'500px'}, "slow", function(){
  $(".box").text('Complete!');
  alert("Loading Complete!");
});
```

In the code above, we are animating an element by slowly increasing it's width. When the width has been resized, a callback occurs. The callback runs a function to pop an alert and change the text of the element after the main job is finished.

Key Terms	
Lesson Files	11_2example.html
Additional Resources	https://learn.jquery.com/
Further Learning	