JavaScript: Video Tutorials1

(Welcome to the power!!!)

Note: these are tutorials that REALLY CLOSELY parallel week 8's videos/ppts. JavaScript is one of those things that's easier to learn if you play with it and learn the patterns. Thus your lab is to go through these tutorials and turn in the exercises at the end of each section.

HINTS:

You are learning to talk computerese (welcome to the geek world!! Don't worry – we're friendly, but a bit eccentric…). Computers are very unforgiving. While we can guess what another person was trying to say, computers demand precision. BE VERY CAREFUL ABOUT DETAILS.

- ***A " and a " are completely different things to the computer. Only the second will work.***
- ***If you opened it, you must close it***
  - ***That goes for ( ), {} , " " and ' '***
- ***Capital and small letters are completely different to the computer and bear no resemblance to each other whatsoever.***
  - ***Alert and alert are completely different words to the computer (only the second one works),***
  - ***as are Function and function (again, the second one works).***
- ***When you name something, do not put any spaces or special characters in the name. Equally, don't start with a number. So:***
  - ***this name is bad***
  - ***This*&^Name is bad***
  - ***2names is bad***
  - ***thisName1 is good!***

Contents

Terms:

HTML: a mark-up language used for web pages (no power)

CSS: a system for adding style to web pages – css allows developers to choose how to style the html elements in a web page

JavaScript: a programming language (like java, python, c++, etc)

- Developed to work with html code
- Much more powerful than html!
- Allows developers to create dynamic web pages

Dynamic Web pages: web pages that change based on user input (clicking on things, running your mouse over something, etc. This is different from static web pages, or plain basic html pages that don't change based on user actions.

JS Facts:

- JavaScript is built into every major browser
- JavaScript is free
- JavaScript is one of the three most sought-after programming languages employers are looking for
- JavaScript is (still!) one of the fastest growing programming languages

Adding javaScript to your html code:

There are 3 ways to include JavaScripts in your web page (a lot like CSS!)

Method 1: Include the javaScript in the body of the web page:

Step 1: Open your html template, and add the following script so the page looks like this:

```
<!DOCTYPE html>
<html lang = "en">
   <head>
       <meta charset = "utf-8">

       <title>JavaScript In the Body</title>
   </head>
       <body>
            <h1> JavaScript Test</h1>

            <script>
                 function yourfirstfunc()
                    {
                    alert("Hello World!")
                 }
            </script>

            <input type = "button" value = "click here to run the javaScript"
            onClick = "yourfirstfunc()">
       </body>
</html>
```

Try running this.

Your html page should have a button, and when you click on it, an alert box should pop up. If it doesn't work, check spellings closely – make sure:

a. You spelled script "script" and not "scirpt" (yep, that happens a lot!)
b. You opened and closed the script tag
c. You spelled function correctly
d. You had a small 'a' in alert
e. You had an opening { at the beginning of the function and a closing } at the end of the function
f. In the button, after onClick, the name of the function matches perfectly with the name of the function in the script. Caps and small letters matter.

Explanation of first JavaScript:

- **<script></script>**

  Anything between these two tags is in javaScript and not in html.

  Remember: every browser has javaScript built in, so it is able to read and interpret both html tags and javaScript code. However, we at least need to tell the browser that certain code should be read as javaScript.

- **function yourfirstfunc()**

  functions are a way of naming code. So this line tells the browser that all code between the { and the } will from now on be named yourfirstfunc(). Right now the only code between the { and the } is an alert box, but in the future we will add more code to functions, so this will be a lot more useful.

- **yourfirstfunc()**

  Like web pages, you can name functions just about anything. In this case I named the function yourfirstfunc(). That's a perfectly legitimate function name, as is func1(), alertfunc(), or even henry(). Yep, you could name a function henry(). (follow naming rules – no spaces, no special characters, can't start with a number)

- **{ … }**

  The code that belongs to yourfirstfunc() must all go between the opening { and the closing }. You are naming lines of code (in this case, one line, the alert("Hello world") line of code). In order to know what lines of code belong to a particular function, we indicate the beginning and the ending using the opening and closing { and }.

  That's the javaScript. It's lines of code that you gave a name to. However, if we want those lines of code to actually be executed, we must associate them with some part of your html code.

- HTML`<input type = "button" value = "click here to run the javaScript"`
  `onClick = "yourfirstfunc()">`
  Somewhere in your html code, you can take an action that will activate the function (aka call the function). The button below the script is html code. It is outside the script. Inside the button tag, you've got:

  - **onClick =**

  The onClick means that when you click on the button, something will happen.

  - **onClick = "yourfirstfunc()"**

  means that when you click on the button, you are activating yourfirstfunc(). The name of the function here (yourfirstfunc()) must match exactly the name of the function in the script.

Step 2: change the name of the function in both the script and in the button to be "puppies()" Try the code now. Does it work? (hint – it should!)

No, puppies isn't the most descriptive name for the function, and thus other programmers would not consider this to be in good form, but it does prove my point.

Method 2: You can add the javaScript to the head section of the html page.

Step 3: Move the script to the head section as follows:

```
<!DOCTYPE html>
<html lang = "en">
  <head>
     <meta charset = "utf-8">
     <title>JavaScript In the Body</title>

      <script>
              function puppies() {
                      alert("Now in the head section!")
              }
     </script>
  </head>
     <body>
              <h1> JavaScript Test</h1>
              <input type = "button" value = "click here to run the javaScript" onClick = "puppies()">
     </body>
</html>
```

Step 4: reload your web page in the browser and click on the button. It should still work.

Advantages of this method: you separate the javaScript from the web page – makes both easier to read – yet you still have the javaScript on the same page as the html code, so you can double-check that the function and the html code match.

Method 3: (Preferred Method) You can create a separate javaScript file, and put the javaScript code in the separate file, and then add a link to it in the head section of the html file.

Step 5: Create a new, blank file.

Step 6. Place the following javaScript code into the blank file:

```
function puppies2() {
        alert("In a separate file!")
}
```

Step 7. Save the file as jscode.js. If you have the option to save it as a javaScript file (which you do in notepad++), make sure you save it as a javaScript file

Note: the name jscode is just a name I picked. You can pick any name you like, as long as it doesn't have anything but letters in the name. However, it must have the .js extension. That tells the world and the browser it's a javaScript file.

Step 8. In your html file, create a link to the jscode.js file as follows:

```
<!DOCTYPE html>
<html lang = "en">
  <head>
                <meta charset = "utf-8">
                <title>JavaScript In the Body</title>

                <script src = "jscode.js"></script>
  </head>
    <body>
                <h1> JavaScript Test</h1>
                <input type = "button" value = "click here to run the javaScript" onClick = "puppies()">
    </body>
</html>
```

Step 9: reload your web page in the browser and click on the button. It should still work.

Advantages of this method*: you separate the javaScript from the web page – makes both easier to read – and you can reuse the code with other web pages. Also, as your javaScript code gets longer, you don't have to keep scrolling through the javaScript in order to see the html code.*

While I'm fond of method 2 for short scripts, as my code gets longer and more complex, I much prefer method 3. You should use either method 2 or method 3, but method 1 gets ugly quickly so after you've done this exercise, don't use that method again.

Multiple functions:

Step 1. In your script (I'm tired of typing javaScript, so I will refer to your javaScript code as your script from now on), either in the separate js file or in the script in the head section, create a second function below the first function. Give it a different name than the first function, and make the alert box say something different than in the first function. Mine looks like this:

```
function puppies() {
    alert("In a separate file!")
}
function func2() {
    alert("You clicked the second button!")
}
```

Step 2. Now add a second button to the html code. Make sure it calls your second function. The body of my html code now looks like this:

```
    <h1> JavaScript Test</h1>
    <input type = "button" value = "click here to run the javaScript" onClick = "puppies()">
```

<input type = "button" value = "click here to run the second function" onClick = "func2()">

Step 3. Save the code and load the html page into a browser. Test it by clicking on both buttons. A different alert box should pop up depending on the button you click on.

Using images as buttons:

Right now you're using the button form element as the button to be clicked on, but you can use anything, even an image!

Step 1. In your JavaScript code, add another function (in my case, func3()) to your script. Make the alert box say something different.

Step 2. Download a picture of a button (or anything you want!)

Step 3. Add the image you just downloaded to your **html** code.

Step 3. Modify the button so that when you click on it, it calls a new function. So my html code looks like this:

<h1> JavaScript Test</h1>
<input type = "button" value = "click here to run the javaScript" onClick = "puppies()">
<input type = "button" value = "click here to run the second function" onClick = "func2()">
<img src = "button.png" style = "height: 100px; width: 100px;" onClick = "func3()">

Step 5. Save the code and load the html page into your browser. Test it by clicking on your image. The alert box associated with func3() should show up.

Your turn: (6 pts)

HTML Part:

- Create a new html file. Inside the file create a table with 4 data cells – I don't care whether the 4 data cells are in one row with 4 data cells, or in two rows with two data cells each.
- Download 4 images of 4 different animals, or fruits, or vehicles or horror movie promo posters– your choice, just pick four distinctly different pictures. Place each picture in a separate data cell in your table.

JavaScript Part:

- Create a new, separate javaScript file using method 3 (above). Save it as something with the .js extension.
- Inside the javaScript file, create 4 separate functions. Each function should have its own name. The first function's alert box should say, "you clicked on…" with the name of what the first image is. So, for instance, if the first image was one of a kangaroo, you should have alert("You clicked on the kangaroo!")
- For each image you downloaded, make a subsequent function say what the image is.
- Make sure you save the javaScript file.

Back to HTML:

- Modify the html code so that each image calls the appropriate function when clicked on (using onClick inside the image tag)
- Link the .js file to your html code in the head section (using method 3).

Save the html file. Load it into a browser. Test it by clicking on each of the images. Does the correct alert box pop up?

(You're programming!!!)

Part 2: Parameters

Another way of thinking of programming is to think of it as a bunch of tools you need to learn, and once you've learned them you just have to put them all together to do what you want. I guess an analogy would be carpentry. First you have to learn how to use a hammer and drill and screwdriver and miter saw and circular saw, etc. The more of the tools you master, the more complex your woodworking projects can be. But you need to master the tools in order to be able to create awesome woodworking projects! That's what we're doing here- mastering tools.

Now that you know the first tool – how to add javaScript to your html code, we're going to add more tools. Your next tool is to understand and use **Parameters.**

Adding Parameters:

From the first tutorial, you may have a script that looks something like this:

```
function puppies() {
    alert("In a separate file!")
}
function func2() {
    alert("You clicked on a bug!")
}
function func3() {
    alert("You clicked on a pig!")
}
```

If you look at this, there are only 2 differences between each of these functions: the name of the function, and what the alert box prints out.

That's a bit tiring. What if we had 10 different images, or 50. Would we want 50 different functions, each of which only differed by what the alert box printed out? No! We want a better way!

So we're going to add a parameter to our functions. First I'll show you, and then I'll explain.

Step 1. Modify your script, so that there's only one function. The function will have a parameter. It should look like this:

```
function generalfunc(x) {
        alert(x)
}
```

Note that **x is the parameter** in this function. You'll soon see where this is going. x is a box in memory that can hold values, like sentences or numbers. So when you run the code, "alert(x)" what will be printed in the alert box is whatever value is inside of x

Step 2. Save the modified js file

Step 3. Now in your html, copy and paste the button a few times (so you now have 3 or 4 buttons below the old button.

Step 4: In the new buttons you just added**,** modify your html code. All 3 of the new buttons (leave the old button as it was) should now call the same function – the generalfunc – as follows:

```
<h1> JavaScript Parameter Test</h1>
<input type = "button" value = "click here to run the javaScript" onClick = "generalfunc('you clicked on 1st func')">
<input type = "button" value = "click here to run the second function" onClick = "generalfunc('you clicked on the 2nd button')">
<img src = "button.png" style="height: 100px; width: 100px;" onClick="generalfunc('you clicked on 3rd button')">
```

Here whenever you click on a button, what you put inside the (and) is what goes into **x**
In the first button, you've got:

```
onClick = "generalfunc('you clicked on 1st func')
```

when you click on the button, you're activating generalizedfunc, and in activating it, you're putting **'you clicked on 1st func'** into **x**
NOTE: It is when you call (activate) a function that you put a value into the parameter. In other words, **x is empty until you click on the button and then when you click on the button, 'you clicked on 1st func' goes into x** so x holds that sentence, but only after you've clicked on the function. That's why it changes when you click on another button.

Step 5. Save your html code and load it into a browser. Test it by clicking on each of the buttons. Does different text pop up in the alert box? It should.

Explanation of Parameters (summary):

In the function above, x is the parameter. Parameters are empty boxs – we named the box x when we put x between the two parentheses. Yep, that's how you give a box a name (or at least one way). We will put values into parameter (or box) x by clicking on buttons. Then inside the function, x is the name of the box holding the sentence we sent into the function.

So when you click on the first button, you are calling the function generalfunc with the sentence, "you clicked on 1st func". When you click, that sentence goes into x, and thus has a temporary name of x inside the function. So every time you use x in the function, you are really using the sentence "you clicked on 1st func."

When you click on the second button, you are calling the function generalfunc again, but this time you're calling it with the sentence, "you clicked on the 2nd button". That is the sentence that goes into x this time, so when you use x in the alert box, you are really using the sentence, "

And when you click on the third button, the function generalfunc is called with “you clicked on 3rd button”, which goes into x, which is what the alert box displays.

Your turn: (6 pts)

Part a) (2) Add some more buttons. Then make the buttons call the generalfunc with different sentences. Test your code

Note: **make sure that the sentence is inside single quotes ‘** . The onClick uses double quotes “, and we can’t have both using double quotes, so the sentence itself has to be in single quotes.

Part b: (4) In your html file make a copy of your table with 4 images. Modify the copy so that all 4 images call only the one new function, each with the name of the image as one word. So, for instance, if you’ve got a picture of broccoli, the sentence would be ‘broccoli’.

Click on all the buttons to make sure everything is working

Part C: Variables (intro):

Variables are similar to parameters – they are boxes that hold values. They are temporary names we give to something else. Bear with me – the example I’m about to show you looks fairly pointless, but you will see how useful variables are shortly. Remember, we’re mastering tools! For now, here is how variables work:

Step 1. In your script, add a new function. It should have a parameter, but we won’t be using it right now. Inside the function, you’ll have a variable, and you’ll use it in the alert box. So your function should look like this:

```
function varfunc(x) {
        y = "hi there"
        alert(y)
}
```

Step 2. Now in your html code, make a button that calls varfunc when you click on it.

Step 3. Save your html and js files and load the html file into the browser. Test it by clicking on the button that calls the varfunc. Is it working? Does “hi there” pop up in the alert box?

Step 4. Change what the variable y holds. I don’t care what you change the sentence to, as long as it is no longer “hi there”. Save your js file and test your html code again by clicking on the button that calls the varfunc. Did the alert box’s sentence change to be what you changed the sentence y holds to?

Prompt box:

So now we will see more of a use for variables. They’re crucial for prompt boxes.

Step 1. Modify the var func to include a prompt box as follows:

```
function varfunc(x) {
    y = prompt("enter your favorite vegetable")
    alert(y)
}
```

Step 2. Now test it. Refresh your html code in the browser and click on the button that calls the function varfunc. Does a prompt box pop up asking you to enter your favorite vegetable?

The prompt box should pop up with whatever you put in the quotes (in this case, “enter your favorite vegetable”) in the box. The user should type in something. Whatever the user types in goes into the variable on the left side, in this case y. Once y holds what the user typed in, it can be used in the alert box.

Step 3. Try it again by clicking on the button again. Type in a different vegetable. The alert box should show the vegetable you just typed.

You can keep re-calling the function again and again, each time typing in a different vegetable. Each time the vegetable you type in should show up In the alert box.

You **MUST** have a variable on the left side of the prompt box. Otherwise there is nothing that will hold what the user typed in. Try it:

Step 4. Modify the function to be:

```
function varfunc(x) {
    prompt("enter your favorite vegetable")
```

```
    alert(y)
}
```

Nothing should happen after the prompt box, because there was no variable to hold what the user typed in. So whatever the user typed in went out into the universe, never to be found again. The user is now annoyed with you for making them type in their favorite vegetable uselessly.

(Note: this may not work at all because y doesn't have anything inside it. If that's the case, well, that still kinda proves my point that you need a variable on the left side of the prompt box.)

Step 5. Put the variable back into the function, and retest it in the browser. Make sure it works.

Your Turn: (6 pts)

1) (3)Create a button in your html that calls a function (make up a name) in your js file.

Then create a function with the name you gave it (in the html file, see the line above this) in the js file that creates a variable and, using an alert box, prints out that variable.

Save both files and make sure everything works.

2) (3)Create another button in your html file that calls a different function in your js file.
Then in the js file create a function with that name. In the function, use a prompt box to prompt the user for their favorite movie. Then, using the variable to the left of the prompt box, use the alert box to print out the movie the user typed in.

Part D: Branching (or if Conditions)

In English, we say, "if it's sunny, I'll play tennis with you. Otherwise I'm going to stay home"

There's something similar in programming. There's an if condition

An if condition is conditional: if this is true, do one thing, if it isn't, do something else.

So, for instance, you might have something like,

```
y=prompt("do you like chocolate?")
if (y === "yes") {
        alert("me too!")
}
else {
        alert("there's something very wrong with you.")
}
```

Step 1. Try this. Create a new function to hold the above chocolate code in your js file. Then save the file

Step 2. Add a button that calls this function in your Html code. Save the html file.

Test it by clicking on the button, and then when the prompt box pops up, type "yes"

Step 3. Test it again, this time when the prompt box pops up, type "no".

A few notes: with this if condition, there are only 2 possibilities – either the user typed in yes, or the user didn't. If the user typed in yes, the code runs the first condition and the alert box says, "me too!".

If the user typed in anything at all other than "yes" , the code runs the second condition and the alert box says, "there's something very wrong with you.")

=== the 3 equals signs are how we test to see if what is on the left side is the same as what is on the right side. So

If (y === "yes")

is testing to see whether what y holds (what the user typed in at the prompt box) is the same as "yes". There are only 2 possible answers to this question: yep, they're the same, or nope, they're not. If the answer is yep, they're the same we do the first condition. Otherwise we do the else condition.

If that doesn't make sense, just remember that in an if condition, you use 3 equals signs to see if 2 things are equivalent.

Else if:

Right now there are only 2 possibilities. But what if you have something with more than two possibilities? You can use an else if condition.

Step 4: change the function so that it holds:

```
y=prompt("what is your age?")
if (y >65) {
        alert("You get the senior discount")
}
else if (y < 18) {
        alert("You can order from the children's menu")
}
else {
        alert("you pay full price")
}
```

Step 5. Load the function and test it. Test with 3 different numbers – one greater than 65, one less than 18, and once between 18 and 65

If there are more than 3 possibilities, you can have multiple else if conditions.

If you have an else, it must be the last thing! And it does not have a condition, in other words, you can't have:

else (y===18)

Your turn: (12 pts)

1) (4)Make sure you've got the above age function working.
2) *(8)On the internet I found a chart about what your favorite candy says about you. It goes as follows:*
   - *Snickers -> you have a multi-faceted personality. You don't fit into a box. You like to travel and read as much as TV and sports.*
   - *100 Grand -> you're wise beyond your years. You were always the kind in class teachers were most in pressed with. You question things in a thoughtful way*
   - *MilkyWay-> you're smooth! You know what to say and when to say it. You love a minimal, clean look. You enjoy cooking more than going out*
   - *3 Musketeers -> You're a hero. You're super humble and unassuming. You read long books*
   - *Butterfingers->you're a little clumsy. You tend to lose stuff. Your favorite show is the Simpsons and you identify with Bart.*
   - *Candy Corn - >you probably need braces. You like cats.*
   - *Reese's Peanut Butter Cups -> You're a free spirit. You have a strong sense of self, and enjoy being around people who challenge you.*
   - *Smarties-> You're the quiet, loving type. You treat everything with respect. You're an introvert and you enjoy cooking with close friends.*
   - *Nerds -> You're a total goofball. You might not always be funny, but everyone knows they'll have a good time when you're around. You're weird in a good way. You're probably a morning person.*
   - *Starbursts -> you're the creative type. You like taking risks and stretching yourself in new, artistic fields. You like living near nature.*
   - *M&Ms ->you're a model citizen. You always recycle. You're always up for an adventure.*
   - *Kit Kat -> you're cool, and super laid back. You go with the flow. You love traveling*
   - *Nestle crunch -> You're a dreamer. You always have a positive attitude. You're not too hard on yourself.*
   - *None of the above -> You're picky. You know what you want and you are willing to wait to get it. Fine, but you're missing out on a lot of good candy!*

Write a javaScript function that prompts the user, "what kind of candy do you like?" and then, based on what the user types in, alerts the user with a paragraph showing about their personality.

Note that no one is going to type in None of the above. That should be your final, else condition.

****

Congratulations! You are writing code! You are officially a geek!!! Make sure you understand this – code builds on previous concepts, so if you don't get something go back over it and ask questions until you do.