JS Functions Tutorial:

Functions

So far in our code we've had javaScripts that load and run when the web page loads in the browser. In order to make the javaScript run again, we have to reload the page in the browser. So, for example, last week we saw a web page with a script that used a prompt to ask the user, "Would you like to learn about an animal?" and if the user answered "yes", the script generated a random number and then used that random number to change the header to the name of the animal, the image to a picture of the animal, and the paragraph to a paragraph about the animal. If the user wanted to learn about a different animal, the user had to reload the entire web page. In another example, a script used a prompt to ask the user, "Heads or Tails?" and then, depending on what the user chose and the random number generated the script either wrote a paragraph, "Congratulations! You guessed correctly!" or "Sorry, you guessed heads and the computer generated tails." (or vice versa). Again, if the user wanted to play again, the user had to reload the web page in the browser. Wouldn't you prefer to create a web page with a button, and every time you clicked the button, the script happened again? Then if you wanted to run the javaScript again, you would just need to click on the button.

You can do this by adding a function and a button. A function is, essentially, naming your javaScript code. When you give your javaScript code a name, you can "call it", or make it run, by calling the name of the function. Every time you call the name of the function, the browser will run it, without having to reload the web page.

Defining a function

function *fname* ()

{

javaScript code that function does

}

Above is a basic template for a function. You must first declare that you are creating a function, so the very first word in the creation of a function is, well, *function*.

Fname is any name you choose to give your function. You can choose any name you wish, as long as it follows the javaScript naming conventions (i.e., no spaces, no special characters other than _, must start with a letter, and can't be a reserved javaScript word.

After the name of the function, you must have the (and). Later we may put something between those parentheses, but for now we can just include () after the function's name.

Then all the code that you want to belong to your function must go between an opening { and closing }

Example:

```
<!DOCTYPE html><html><head>      <meta charset= "utf-8" ></head>
<body>
   <p id = "firstp"> This is a paragraph</p>
   <script>
      function guessinggame()
      {      var x = Math.floor(Math.random()*6) + 1
             var y = parseInt(prompt("Enter a number between  1 and 6"))
```

```
            if (x == y)
            {       document.getElementById("firstp").innerHTML = "You guessed it!";
            }
            else
            {       document.getElementById("firstp").innerHTML = "You are wrong!";
            }
        }
    </script>
</body></html>
```

Above is an example of including a function in your javaScript code. The function includes all code between its opening { and its closing }.

We turned code into a function for a reason. By making javaScript code into a function and giving it a name, we can make the code happen whenever we call its name (e.g., by clicking a button on our web page, which we will learn how to add next). That means that the code inside of a function doesn't happen automatically. I'm going to repeat that. The code inside a javaScript function doesn't happen whenever you load the web page. So in order to make the code inside the function, we must call its name somehow. One way to do that is to add a button to our web page and make that button call the function's name.

Adding a button

The button is technically not javaScript code – it is HTML code. So it must go in the web page, but not inside your script. The button does, however, call the javaScript function's name, and, in doing so, makes the function happen.

To include a button in your web page, you'd include the following html code:

<input type = "button" value = "Click to play" onClick = "guessinggame()">

Breaking it down, this html code will create a button, with the text, "Click to play" on it. When you click it, it calls the function you named "guessinggame()", thus making the code inside the function be executed by the browser.

So, putting it all together, you'd get:

```
<!DOCTYPE html><html><head>        <meta charset= "utf-8" ></head>
    <body>
            <p id = "firstp"> This is a paragraph</p>
            <input type = "button" value = "Click to play" onClick = "guessinggame()">

     <script>
            function guessinggame()
            {       var x = Math.floor(Math.random()*6) + 1
                    var y = parseInt(prompt("Enter a number between  1 and 6"))
                    if (x == y)
                    {       document.getElementById("firstp").innerHTML = "Play the lottery!";
                    }
                    else
                    {       document.getElementById("firstp").innerHTML = "You are wrong!";
                    }
            }
    </script>
</body></html>
```

So, in the above example, the function guessinggame does not happen automatically . It is only executed when we click on the button. But we can click on the button again and again. Each time we click on it, the code inside the function will run. This is more efficient than reloading the entire web page in the browser.

Placing functions in the head section:

So far we've placed all of our javaScript code in the body of the web page. javaScript code, and in particular functions, can go in the body section of our html code or in the head section of the html code. With functions, we usually put them in the head section. The reason is that if there are images involved, by putting it in the head section, the images will preload. If we put the code in the body section, the images won't download until the function is called, making it run slow.

Plus, when we put the functions inside the body of our html code, it's just sloppier and harder to read through the html page. By separating the javascript and placing it in the head section, the web page is cleaner to read through.

We can also place javaScripts into separate files and link the separate file in the head section, like we did with the css style sheet. But for now, let's just place the javaScript code in the head section.

Below is an example of placing the javaScript code in the head section of your html page:

```
<!DOCTYPE html><html>
<head>
      <meta charset= "utf-8" >
      <script>
           function show_confirm()
           {     var r=confirm("Press a button");
                 if (r==true)
                 {    document.getElementById('p1').innerHTML = "You pressed OK!";
                 }
                 else
                 {    document.getElementById('p1').innerHTML = "You pressed Cancel!";
                 }
           }
       </script>
</head>
<body>
       <input type="button" value="Show confirm box" onclick="show_confirm()" >
       <p id = "p1"> Answer goes here </p>
</body>
</html>
```

In the following example, there are two functions in the head section, each with its own name. Every function name within a javaScript has to be unique.

```
<!DOCTYPE html><html><head><meta charset= "utf-8" >
        <script>
                function coffeeinfo()
                {
                        document.getElementById('p3').innerHTML = "<p>The word 'coffee' was at …. </p>"
                        document.getElementById('p3').style.color = "#CCBBAA"
                        document.getElementById('p3').style.backgroundColor = "#995500“                }

                function teainfo()
                {
                        document.getElementById('p3').innerHTML = "<p>The origin of tea can be traced..
</p>"
```

```
                    document.getElementById('p3').style.color = "#227700"
                    document.getElementById('p3').style.backgroundColor = "#BBCC22“
                    document.getElementById("p3").style.borderColor = "#114400"
                    document.getElementById("tea").style.borderColor = "#114400“
                    document.getElementById("tea").style.borderWidth = "10px"
                    document.getElementById("coffee").style.borderWidth = "0px"
                }
        </script>
        </head>
        <body>
                <table ><tr><td >
                             <img src = "Images/coffee.jpg" width = "300" height = "280"
                             alt = "pic of coffee" id = "coffee">
                        </td><td>
                             <img src = "Images/tea.jpg"  width = "360" height = "280" alt = "pic of tea" id = "tea">
                        </td></tr>
                        <tr><td>
                                <input type = "button" value = "Learn more about coffee" onClick =
"coffeeinfo()">
                        </td><td>
                                <input type = "button" value = "Learn more about tea" onClick = "teainfo()">
                        </td></tr>
                        <tr><td colspan = "2" id = "p3">
                        </td></tr>
                </table>
</body></html>
```

Methods for calling Functions (making them be executed by the browser)

There are a number of ways you can make a function happen in JavaScript. You've seen **onClick="*functionname*()"**

There's also:

- **onMouseOver() –** when you run your mouse over something
- **onMouseOut() –** when you take your mouse pointer off of something
- **onLoad() –** for when the web page loads

I use the above frequently, along with onKeyPress(below). But there are a number of other options we can include in the html code for calling javaScript functions, including:

- **onDblClick() – when you double-click on something**
- **onFocus() – when you put your cursor into a form element like a textbox**
- **onBlur() – when your cursor leaves a form element**
- **onKeyDown () – when you press a key down over something**
- **onKeyUp() – when you release a key over something**
- **onKeyPress()- when you press and release a key over something**
- **onMouseDown()- when you click the mouse over something (but don't release it)**
- **onMouseUp() – when you release the mouse over something**
- **onMouseMove()- moving the mouse while hovering over something**
- **onSubmit() – when submitting a form**
- **onUnload() – when you leave the current web page window you're in.**

OnMouseOver and onMouseOut:

Below is an example of using onMouseOver and onMouseOut to call the javaScript functions. In the example, there are two javaScript functions, ***changepara()*** and ***changethanks()***. In the html code in the body of the web page is a button. Moving your mouse over the button (you don't need to click on it – just run your mouse over it) will call the function, changepara() and the code inside it will be executed. When you move your mouse off of the button, the function changethanks() will be called and executed.

```
<!DOCTYPE html><html>
<head>
        <meta charset= "utf-8" >
        <script>
        function changepara()
        {
                document.getElementById('firstp').innerHTML = "GET YOUR MOUSE OFF THAT BUTTON!"
        }
        function changethanks()
        {
                document.getElementById('firstp').innerHTML = "Whew, that was close!"
        }
        </script>
</head>
<body>
        <p id = "firstp">This is a very important paragraph!!!</p>
        <input type = "button" value = "Don't click here"
                onMouseOver = "changepara()" onMouseOut = "changethanks()">
</body></html>
```

onMouseOver and onMouseout (and onClick, and almost every other method for calling functions) will work on any element with a tag on your web page. You don't have to create a button – you can place onMouseOver and onMouseOut in a paragraph, for example. In the code below, there's a paragraph with the id, "firstp". Inside the paragraph tag, there's a call to changepara() that happens in an onMouseOver event (i.e., when you run your mouse over the paragraph with the id "firstp"). Now, when you run your mouse over the paragraph, the function changepara() is called and executed. The function changepara() changes the innerHTML of the element with the id 'firstp' to "DON'T RUN YOUR MOUSE OVER THIS PARAGRAPH!". That means when you move your mouse over the paragraph, 'firstp', its text changes. Equally, when you move your mouse off the paragraph, the function changethanks() will be called and the code will be executed, meaning the innerHTML of the element with the id 'firstp' (the paragraph) will be changed to, "Thanks for taking your mouse off this paragraph".

So to summarize, when you move your mouse over the paragraph 'firstp', the text changes to, "DON'T RUN YOUR MOUSE OVER THIS PARAGRAPH!" and when you mover your mouse off of the paragraph, the text will change to, "Thank you for taking your mouse off this paragraph".

```
<!DOCTYPE html><html><head>  <meta charset= "utf-8" >
        <script>
        function changepara()
        {       document.getElementById('firstp').innerHTML = "DON'T RUN YOUR MOUSE OVER THIS
PARAGRAPH!"
        }
        function changethanks()
        {       document.getElementById('firstp').innerHTML = "Thank you for taking your mouse
off this paragraph"
        }
        </script>
        </head>
        <body>
                <p id = "firstp" onMouseOver = "changepara()" onMouseOut = "changethanks()">
                This is a very important paragraph!!!</p>
</body></html>
```

Equally, you can call functions using onMouseOver and onMouseOut on images on your web page. The functions called can change the images to new images. See if you can figure out what the following code does:

```
<!DOCTYPE html><html><head>  <meta charset= "utf-8" >
        <script>
        function changepic()
        {
                document.getElementById('pic1').src = "ghost.jpg"
        }
        function changeback()
        {
                document.getElementById('pic1').src = "woman.jpg"
        }
        </script>
        </head>
        <body>
                <p><img src = "woman.jpg" width = "300" height = "300" id = "pic1"
                onMouseOver = "changepic()" onMouseOut = "changeback()"> </p>
</body></html>
```

In the above example, when you move your mouse over the image on the web page with the id, 'pic1', the src of the image (the picture) changes from woman.jpg to ghost.jpg. When you move your mouse off of the image, the src picture changes back to that of a woman.