JavaScript and jQuery Course

Session 06 Functions and Events Continued

Function

- Block of code - that we can reuse
- Executes when you call it
- Called when an event occurs

Function

- Can have *parameters* which allow us to pass *arguments* to the function
- Can return a value
- Same naming rules as variables

These terms are used interchangeably

Traditional JS Event Handling

An **inline** attribute

Writing the JavaScript directly inside the HTML element

```
<h2 onclick="function">
```

Anonymous Function – stored in a variable

```
var sayHello = function() {
  alert("Hello");
}

window.onload = function() {
  var btn = document.getElementById("btn");

  btn.onclick = sayHello;
}
```

Stored in a variable

Function is called when event happens
No parentheses

5

Named Function - created with "function" keyword

```
function sayHello() {
  alert("Hello");
}

window.onload = function() {
  var btn = document.getElementById("btn");

  btn.onclick = sayHello;
}
```

Function is called when event happens
No parentheses

6

Variable Scope

- **Local** – if variable is defined *inside* function
- **Global** – if variable is defined *outside* of function
- **Global** – if variable is defined *without using* var regardless of where it is (inside/outside of a function)

7

Named Function that returns a value

```
<script>
  function showSum() {
    var x = 2 + 3;
    return x;
  }

  var theSum = showSum();

  alert(theSum);
</script>
```

x only has a value inside the function

create new variable to store the value returned by the function

8

Named Function – returns value – 2 parameters

```
function calculateTax(theSubtotal, theTaxrate) {
  var tax = theSubtotal * theTaxrate;
      tax = parseFloat(tax.toFixed(2));
  return tax;
}

window.onload = function() {

  var subtotal = 85;
  var taxRate = .05;

  // calls function
  var salesTax = calculateTax(subtotal,  taxRate);
  alert(salesTax);
}
```

9

Textbook Example – Function stored in variable returns object reference to DOM Element

```
var $ = function(id) {

  return document.getElementById(id);
}
```

Stored in a variable - $

parameter

returns reference to the id

10

Textbook Example – Function stored in variable returns object reference to DOM Element

```
<input type="text" id="first_name">
```

First name:

Before:

```
var firstName=document.getElementById("first_name").value;
```

Now:

```
// $ function, shortcut to retrieve the object reference
 var $ = function(id) {
   return document.getElementById(id);
 }

 var firstName = $('first_name').value;
```

11

Using a Function for Repetitive Tasks

```
var theWidth = document.getElementById('width').value;

if (isNaN(theWidth) || theWidth <= 0) {

      // give error message if true

 }
```

```
function checkNumber(num) {
      if (isNaN(num) || num <= 0) {
                return true;
      }
 }
```

12

Using a Function for Repetitive Tasks

```
var theWidth = document.getElementById('width').value;

function checkNumber(num) {
      if (isNaN(num) || num <= 0) {
                return true;
      }
  }
```

```
if (checkNumber(theWidth) {

      // give error message if true

 }
```

13

Using a Function To Return Values

```
var theWidth = document.getElementById('width').value;

 function checkNumber(num) {
      if (isNaN(num) || num <= 0) {
          return false;
       }
       else {
           return num;
        }
}
```

```
if (!checkNumber(theWidth) {

      // give error message if true

 }
```

14

Using a Function To Return Values

```
var theWidth  = document.getElementById('width').value;
var theHeight = document.getElementById('height').value;

 function checkNumber(num) {
      if (isNaN(num) || num <= 0) {
          return false;
       }
       else {
           return num;
        }
}
```

```
 if (checkNumber(theWidth) && checkNumber(theHeight)) {

    // perform calculations and display results

  }
```

15

Strict Mode Directive

```
'use strict';
```

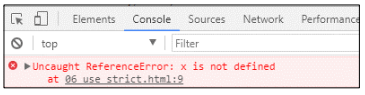
- Makes it easier to write "secure" JavaScript
- Changes previously accepted "bad syntax" into real errors
- Can be used globally or in a function

16

Strict Mode Directive

```
<script>
    'use strict';
     x = 3;  // This will cause an error without using - var

    alert(x);
</script>
```



17