JavaScript and jQuery Course

Session 05 Functions and Events

Function

- Block of code
  - that we can reuse
  - Makes writing code cleaner
- Same naming rules as variables
- Executes when you call it
- Called when an event occurs
  - page load
  - click

Function – stored in a variable

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

sayHello();
```

Stored in a variable

Function Call

usually called from inside another function or when page loads

Function - named

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

sayHello();
```

Uses the function keyword

How the function is called - usually from another function

Anonymous Function – called from an event

```
window.onload = function() {
    // statements here
}
```

What we have used
Executes when the
page load event happens

5

Function Call Example

```
function showDate() {
  // code here
}

window.onload = function(){
    showDate();
}
```

Executes when the
page load event happens

6

Traditional JS Event Handling

Using an even as an **inline** HTML attribute


```
<h2 onclick="function">
<body onload="showDate()">
```

7

Function Call Example

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

function keyword
function has
parentheses

```
sayHello();
```

How the function
is called
- usually from
another function

8

Function Call Example

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

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

  btn.onclick = sayHello;
}
```

Function is called when event happens
No parentheses

9

Function

- Can have *parameters* which allow us to pass *arguments* to the function
- Can return a value

These terms are used interchangeably by some

10

Function that uses a Parameter

```
var name = "Teresa";

function sayHello(who)
{
  alert("Hello " + who);
}

window.onload = sayHello(name);
```

parameter

parameter

argument

11

Function that uses two Parameters

two parameters

```
<head>
  <script>
    function showMessage(what, who) {
      alert("I say " + what + " to you " + who);
    }

    window.onload = showMessage("Hello", "Teresa");
  </script>
</head>
```

12

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

13

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;
```

14

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)

15

Event Handler = The Function

- The function that is executed when an event occurs
- The function handles the event
- An event handler is attached to an event
- The function is attached to an event

16

Common Events

event name is preceded by 'on'

- click – onclick
- page load – onload / onunload
- mouse over – onmouseover / onmouseout
- mouse move – onmousein / onmouseleave
- mouse move – onmouseup / onmousedown
- entering an input field – onfocus / onblur
- submitting a form - onsubmit
- keystroke - onkeypress

17

Event Handler Syntax

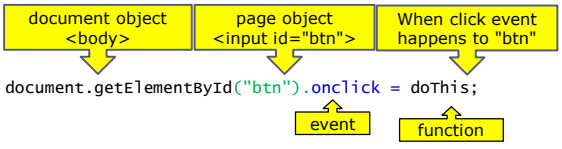
```
objectVariable.on_eventName = eventHandlerName;
```

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

btn.onclick = sayHello;
```

18

Event Handler Syntax




```
<form>
    <input type="button" id="btn">
</form>
```

19

Handling Many Events

```
function sayHello() {
  alert("Hello 1");
}

var sayHello2 = function() {
 alert("Hello 2");
}

window.onload = function() {
  var btn  = document.getElementById("btn");
  var btn2 = document.getElementById("btn2");

  btn.onclick  = sayHello;
  btn2.onclick = sayHello2;
}
```

20

DOM Event Listeners – not in text

- part of the JS event model
- attaches an event handler to the specified element by using the **addEventListener()** method
- can add many event handlers to one element
- remove an event listener by using the **removeEventListener()** method

21

DOM Event Listeners – not in text

```
element.addEventListener("mouseover", myFunction);

element.addEventListener("mouseover", myFunction, false);

element.addEventListener("click", mySecondFunction);

element.addEventListener("mouseout", myThirdFunction);
```

default

22

Check for required field

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

```
if (document.getElementById("first_name").value == ""){
   // give user error message
}
```

Check to see if it is empty

23

Check for valid number greater than 0

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

```
var theMilesDriven = document.getElementById("miles_driven").value;
if (isNaN(theMilesDriven) || theMilesDriven < = 0){
   // give user error message
}
```

24

Create a Function

```
<input type="button" id="btn_calc">
```

```
function calculate() {

var theMilesDriven = document.getElementById("miles_driven").value;

        if (isNaN(theMilesDriven) || theMilesDriven < = 0){
         // give user error message
        }

} // end function calculate
```

```
window.onload = function() {

         document.getElementById("btn_calc").onclick = calculate;

}
```

25

We do not need to check if it is empty

isNaN('') returns False – it is a number
isNaN(0) return False – it is a number

However…
isNaN('') returns 0

We are checking that it is not 0

```
if (isNaN(theMilesDriven) || theMilesDriven < = 0)
```

26