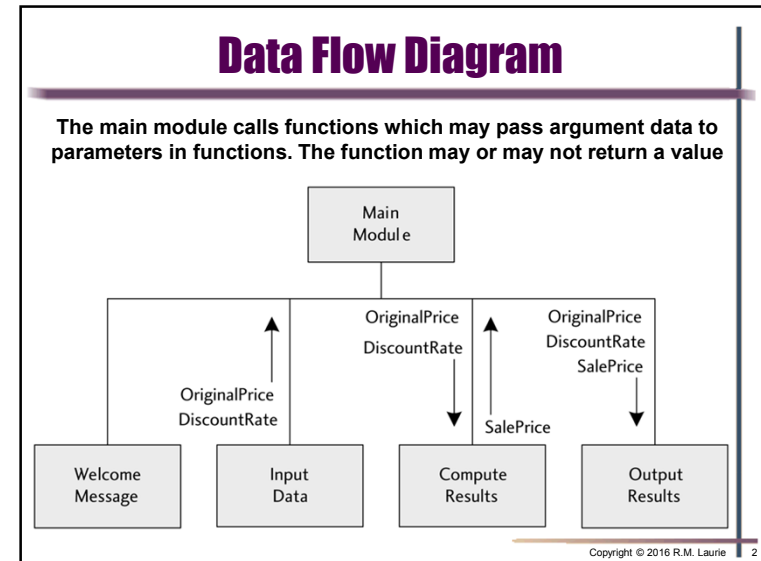


JavaScript Functions

- ❖ **Modular program construct**
 - ◆ Supports *Divide and Conquer* method
 - ◆ Individual functions tested before assembly
 - ◆ Code Reuse
- ❖ **JavaScript Library Functions**
 - ◆ JavaScript has seven **Global Functions**
 - ◆ JavaScript library functions are usually accessed as **Methods** contained in an **Object**
- ❖ **User defined functions can be created**

Copyright © 2016 R.M. Laurie 1



Why Use Functions (sub programs)?

- ❖ They can be designed and coded independently of the main program and allows *Code-reuse*
- ❖ Only the *structure* of the function is important; not the naming of its variables
- ❖ Makes it easier for different programmers to design and code different program modules
- ❖ Makes testing and debugging easier as modules can be tested independently of main program
- ❖ **Function Definition (Parameters)**
`function SquareNumber(fP) // A is a parameter`
`{`
`return fP*fP;`
`}`
- ❖ **Function Call (Arguments)**
`nSquare = SquareNumber(6);`
`fArea = Math.PI * SquareNumber(nRadius);`

Copyright © 2016 R.M. Laurie 3

Library Functions

- ❖ **Global Functions** can be called anywhere
 - ◆ `number parseInt(string)`
Converts the string and returns an integer (whole number) value.
 - ◆ `number parseFloat(string)`
Converts the string and returns a floating point (real number) value.
- ❖ **Object.Method** functions
 - ◆ `document.write(string);` // Output
 - ◆ `window.alert(string);` // Alert Window
 - ◆ `number Math.PI` // The Number 3.1415...
 - ◆ `string window.prompt(string, default);` // Prompt
`return Object.Method(parameters)`

Output
Noun
Verb
Input

Copyright © 2016 R.M. Laurie 4

Math Object Methods

- ❖ **number Math.PI** Returns 3.141592654558979
- ❖ **number Math.max(num1, num2)** Returns greater
- ❖ **number Math.min(num1, num2)** Returns lesser
- ❖ **number Math.pow(x, y)** Returns X^y power
- ❖ **number Math.floor(num)** Rounds down to integer
- ❖ **number Math.random()** Returns value between 0 to 1
- ❖ **number Math.sqrt(num)** Returns square root of num
- ❖ **number Math.sin(num)** Returns sine of num
- ❖ **number Math.asin(num)** Returns arc sine of num
- ❖ And many more methods...

Copyright © 2016 R.M. Laurie 5

Library Function Example

```
<head> <title>Square Root and Power</title>
<script type="text/javascript">
  var fA, fB = 4;
  document.write("<h3>" + fA + " " + fB + "</h3>");
  fA = Math.sqrt(fB);
  document.write("<h3>" + fA + " " + fB + "</h3>");
  fA = Math.sqrt(fA);
  document.write("<h3>" + fA + " " + fB + "</h3>");
  fA = Math.pow(Math.pow(fA, fB), 3);
  document.write("<h3>" + fA + " " + fB + "</h3>");
</script>
</head>
```

```
undefined 4
2 4
1.4142135623730951 4
64.000000000000004 4
```

Copyright © 2016 R.M. Laurie 6

User Defined Functions

- ❖ User functions can be created that modularize a program
- ❖ Good divide and conquer approach for large programs
- ❖ Functions also allow you to reuse code for repeated sections
- ❖ Best for blocks with only one result
- ❖ Important for Event Driven actions
- ❖ Naming Convention:
 - ◆ Use TitleCase for User Functions (no spaces)
 - ◆ VerbNoun is best
 - ◆ CalcArea(fX) PrintGraph(fX, fY) GetData()

Copyright © 2016 R.M. Laurie 7

User Function Parts

- ❖ **Function Definition** is function code
 - ◆ Place in head after program code area
 - ◆ Parameter list
 - ◆ Inputs to the function from function calls
 - ◆ Parameters have *Local Scope (Visible in function only)*
 - ◆ Do Not use `var` to declare parameters variables
 - ◆ May return only one value or nothing
 - ◆ `return;` `return fArea;` `return fDiceRoll;`
 - ◆ Variables in function have *local scope*
- ❖ **Function Call** invoked in program or function
 - ◆ Arguments are values which are passed to function
 - ◆ Position and data type match required
 - ◆ If variables it passes contents of variable

Copyright © 2016 R.M. Laurie 8

```

<head>
<title>A Programmer-Defined square Function</title>
<script type="text/javascript">
  // MAIN PROGRAM
  document.write("<h3>Square numbers 1 to 9</h3>");
  for ( var nI = 1; nI <= 9; nI++)
    document.write("<b>The square of " + nI + " is "
      + SquareNumber(nI)+"</b><br>");

  //SQUARE FUNCTION DEFINITION
  function SquareNumber(nP)
  {
    return nP*nP;
  }
</script>
</head>
</body>
    
```

Calling function SquareNumber and passing it the value of nI.

Variable nP gets the value of variable nI.

Square numbers 1 to 9

The square of 1 is 1
 The square of 2 is 4
 The square of 3 is 9
 The square of 4 is 16
 The square of 5 is 25
 The square of 6 is 36
 The square of 7 is 49
 The square of 8 is 64
 The square of 9 is 81

The return statement passes the value of nP * nP back to the calling function.

Copyright © 2016 R.M. Laurie 11

```

<head> <title>Square Root and Power</title>
<script type="text/javascript">
  // MAIN PROGRAM
  var nA = 1;
  document.write("<h3>Start of Main Program<br />");
  PrintA(nA++);
  PrintB(++nA);
  document.write("End of Main Program</h3>");

  function PrintA( pA ) //FUNCTION DEFINITION
  {
    document.write("Function A: " + pA + "<br />");
    return;
  }
  function PrintB( pB ) //FUNCTION DEFINITION
  {
    document.write("Function B: " + pB + "<br />");
    return;
  }
</script>
</head> </body> </body>
    
```

← Function Calls

Main

PrintA(sA++)

PrintB(++sA)

Start of Main Program
 Function A: 1
 Function B: 3
 End of Main Program

Copyright © 2016 R.M. Laurie 10

```

<head>
<title>Nested function calls</title>
<script type="text/javascript">
  // MAIN PROGRAM
  var nA = 1;
  document.write("<h3>Start of Main"
    + " Program<br />");
  PrintA(++nA);
  document.write("End of Main Program</h3>");
  function PrintA( pA ) //FUNCTION DEFINITION
  {
    document.write("Function A: " + pA + "<br />");
    PrintB(7);
  }
  function PrintB( pB ) //FUNCTION DEFINITION
  {
    document.write("Function B: " + pB + "<br />");
  }
</script>
</head> </body> </body>
    
```

Main

PrintA(++sA)

PrintB(7)

Start of Main Program
 Function A: 2
 Function B: 7
 End of Main Program

← Function Call

← Function Call

Copyright © 2016 R.M. Laurie 11

```

<head> <title>Many Function Calls</title>
<script type="text/javascript">
  // MAIN PROGRAM
  document.write("<h3>Start of Main"
    + " Program<br />");
  PrintA(2);
  PrintB(4);
  PrintA(6);
  document.write("End of Main Program</h3>");
  function PrintA( pA ) //FUNCTION DEFINITION
  {
    document.write("Function A: " + pA + "<br />");
    PrintB("Nested in A");
  }
  function PrintB( pB ) //FUNCTION DEFINITION
  {
    document.write("Function B: " + pB + "<br />");
    return;
  }
</script></head> </body> </body>
    
```

← Function Calls

Start of Main Program
 Function A: 2
 Function B: Nested in A
 Function B: 4
 Function A: 6
 Function B: Nested in A
 End of Main Program

Main

PrintA(2)

PrintB(Nest)

PrintB(4)

PrintA(6)

PrintB(Nest)

← Function Call

Copyright © 2016 R.M. Laurie 12

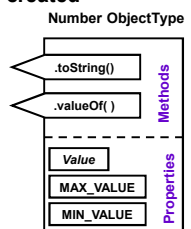
Program Objects and Classes

- ❖ **Object oriented design (OOD)** breaks problem into objects in a top-down process
 - ◆ Supports *Divide and Conquer* approach
 - ◆ Supports *Code Reuse*
- ❖ **Object-Type (Class in Java or C++)**
 - ◆ Definition of a type of object
 - ◆ Describes all properties and methods associate with objects of this type
- ❖ **An Object is a self contained instance of an object-type (Class) that contains**
 - ◆ **Properties** (data, attributes, member variable)
 - ◆ **Methods** (functions, operations, instructions)

Copyright © 2016 R.M. Laurie 13

JavaScript ObjectTypes

- ❖ JavaScript ObjectTypes <http://www.w3schools.com/jsref/>
- ❖ **Static** ObjectTypes encapsulate methods only
 - ◆ Global `integer parseInt(string); float parseFloat(string)`
 - ◆ Window `alert(string); string prompt(string, string)`
 - ◆ Math `num Math.pow(num, num); num Math.floor(num); num Math.random()`
- ❖ **Non-static** ObjectTypes encapsulate methods and are considered data templates from which new objects can be created
 - ◆ Number `var nRedChip = new Number(8);`
 - ◆ String `var sFirstName = new String("Bob");`
 - ◆ Boolean `var bAnswer = new Boolean(true);`
 - ◆ Array `var aScore = new Array(100);`
- ❖ **HTML Document Object Model (DOM)**
 - ◆ document `document.write(string);`
 - ◆ form
 - ◆ form input text
 - ◆ form input button

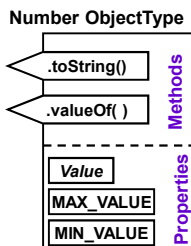


Number ObjectType

Copyright © 2016 R.M. Laurie 14

Number Object-Type

- ❖ Number Object-Type defines a container for a number and associated library of methods
- ❖ <http://www.javascriptkit.com/jsref/number.shtml>
- ❖ To create an Object (Instance) of the Number Object-Type use the new operator
 - ◆ `var NumberObject = new Number(value);`
- ❖ **Properties**
 - ◆ Value is implied when using variable
 - ◆ `NumberObject.MAX_VALUE // 1.79E+308`
 - ◆ `NumberObject.MIN_VALUE // 5.00E-324`
- ❖ **Methods**
 - ◆ `number NumberObject.valueOf()`
 - ◆ `string NumberObject.toString(radix)`



Number ObjectType

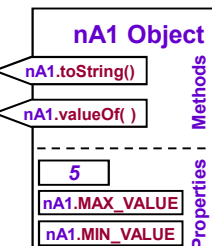
Output
Object
Method
Base

Copyright © 2016 R.M. Laurie 15

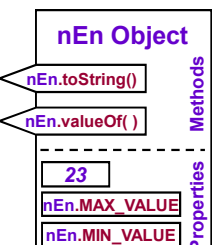
Creating new Number Objects

```
var nA1 = new Number(5);
```

```
var nA1 = new Number(5);
var nEn = new Number(23);
var nSum;
nSum = nEn + nA1;
```



nA1 Object



nEn Object

Copyright © 2016 R.M. Laurie 16

String Object-Type

- ❖ String Object-Type defines a container for a string and associated library of methods
- ❖ To create an Object (Instance) of the String Object-Type use the new operator
 - ◆ `var StringObject = new String("My Name is Bob");`
- ❖ Properties
 - ◆ `StringObject.length` // length of string object
- ❖ Methods
 - ◆ `string StringObject.concat(string, string,...)`
 - ◆ `StringObject.toLowerCase()`
 - ◆ `string StringObject.substr(start, end)`
 - ◆ `string StringObject.charAt(index)`
 - ◆ `integer StringObject.indexOf(substr, index)`

Copyright © 2016 R.M. Laurie 17

Introduction to Arrays

- ❖ Grouping of similarly named variables, which are grouped sequentially in memory and accessed by their element (*index*) number
- ❖ Element numbering begins with 0 to one less than the total number of elements
- ❖ An Array element can hold numbers, strings, Boolean (true/false), and Objects
- ❖ There is Array Object-Type
- ❖ Declaring an array creates an Array object
 - ◆ `var nCounter = new Array(5);`
 - ◆ `Array.length` is a property
 - ◆ `Array.sort()` is a method

Counter[0]	30
Counter[1]	45
Counter[2]	53
Counter[3]	2
Counter[4]	879

Copyright © 2016 R.M. Laurie 18

Declaring Arrays

- ❖ Declaration:

Counter[0]	30
Counter[1]	45
Counter[2]	53
Counter[3]	2
Counter[4]	879

 - ◆ `var nCounter = new Array(5);`
 - ◆ Reserves Counter array memory
 - ◆ nCounter[0] to nCounter[4]
 - ◆ No values are stored in elements
 - ◆ May store assign values to elements individually
 - nCounter[0] = 30;
 - nCounter[1] = 45;
 - ...
 - ◆ `var nCounter = new Array(30, 45, 53, 2, 879);`
 - ◆ Reserves Counter array memory
 - ◆ nCounter[0] to nCounter[4] and initialized the first 5 elements to the the values shown

Copyright © 2016 R.M. Laurie 19

for Loop Array Initialization

- ❖ A for loop can be used to initialize a declared array
- ❖ Set all array elements to 0
 - ◆ `var nCounter = new Array(5);`
 - ◆ `for(var nK=0; nK < 5 ; nK++)`
 - ◆ `nCounter[nK] = 0;`
- ❖ This is very useful for large arrays such as:
 - ◆ `var nScore= new Array(100);`
 - ◆ `for(var nK=0; nK < 100 ; nK++)`
 - ◆ `nScore[nK] = 0;`

Counter[0]	0
Counter[1]	0
Counter[2]	0
Counter[3]	0
Counter[4]	0

Copyright © 2016 R.M. Laurie 20

Array Bounds Checking

- ❖ For JavaScript the array element quantity is optional. The following is acceptable syntax.
`var nCounter = new Array();`
- ❖ Elements can be added to an existing Array by assigning values to new array elements. The number of elements is increased to eight.
`var nCounter = new Array(5);`
`for(var nK = 0; nK < 8; nK++)`
`nCounter[nK] = 0;`
- ❖ The array length property specifies the total number of elements contained in an array.
`for(var nK=0; nK< nCounter.length; nK++)`
`nCounter[nK] = 0;`

Copyright © 2016 R.M. Laurie 21

Sentinel Controlled Array Processing

```
var Entry, Score = new Array();
for(var i = 0; i < 10000; i++)
{
    Entry = parseFloat(prompt("Enter Score (-1 to quit)","0"));
    if(Entry < 0)
        break;
    Score[i] = Entry;
}
for(var j = 0, Max = 0; j < Score.length; j++)
{
    document.write("Score " + (j+1) + " = "
        + Score[j] + "<br \>");
    if(Score[j] > Max) Max = Score[j];
}
document.write("Maximum Score = " + Max);
```

```
Score 1 = 68
Score 2 = 87
Score 3 = 96
Score 4 = 87
Score 5 = 93
Maximum Score = 96
```

Copyright © 2016 R.M. Laurie 22

Array and String Object Methods

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

String HTML Wrapper Methods

The HTML wrapper methods return the string wrapped inside the appropriate HTML tag.

Method	Description
anchor()	Creates an anchor
big()	Displays a string using a big font
blink()	Displays a string in bold
bold()	Displays a string in bold
fixed()	Displays a string using a fixed-pitch font
fontcolor()	Displays a string using a specified color
fontsize()	Displays a string using a specified size
italics()	Displays a string in italic
link()	Displays a string as a hyperlink
small()	Displays a string using a small font
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript text
sup()	Displays a string as superscript text

Copyright © 2016 R.M. Laurie 23

The screenshot shows a web browser window with the following content:

```
75 = 1001011 = 4B = K
How Many Cars?
HOW MANY CARS?
String Length = 14
How Many Cars?

Mazda,Volvo,BMW,Ford
Array Length = 4
BMW,Ford,Mazda,Volvo
BMW,Ford,Mazda,Volvo

BMW,Ford,VW,Volvo
Array Length = 4

BMW,Ford,VW,Volvo,Mazda
Array Length = 6

BMW,Ford,VW,Volvo,Honda,Mazda
Array Length = 6

BMW,Ford,VW,Volvo,Honda
Array Length = 5

BMW,Ford,VW,Volvo,Honda,KIA,Mini
Array Length = 7

The End
```

The browser's developer tools show the following JavaScript code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Number and String Objects</title>
  <script type="text/javascript">
    var Num1 = new Number(75);
    var Title = new String("How Many Cars?");
    var Cars = new Array("Mazda","Volvo","BMW", "Ford");
    document.write(Num1+" = "+Num1.toString(2)
    +" = "+Num1.toString(16).toUpperCase()
    +" = "+String.fromCharCode(Num1));
    document.write("<cp>"+Title+"<br>"
    +Title.toUpperCase()+"<br>"
    +"String Length = "+ Title.length+"<br>"
    +Title);
    document.write("<cp>"+Cars+"<br>"
    +"Array Length = "+ Cars.length+"<br>"
    +Cars.sort()+"<br>"+Cars+"</p>");
    Cars[2]="VW";
    document.write("<cp>"+Cars+"<br>"
    +"Array Length = "+ Cars.length+"</p>");
    Cars[5]="Mazda";
    document.write("<cp>"+Cars+"<br>"
    +"Array Length = "+ Cars.length+"</p>");
    Cars[4]="Honda";
    document.write("<cp>"+Cars+"<br>"
    +"Array Length = "+ Cars.length+"</p>");
    Cars.pop();
    document.write("<cp>"+Cars+"<br>"
    +"Array Length = "+ Cars.length+"</p>");
    Cars.push("KIA","Mini");
    document.write("<cp>"+Cars+"<br>"
    +"Array Length = "+ Cars.length+"</p>");
    Title = "The End";
    document.write("<h2>"
    +Title.fontcolor("#FF0000").blink()+"</h2>");
  </script>
</head><body></body></html>
```

Passing Array to Function

- ❖ **Pass-by-value** is used to pass the value of an argument in a function call to the function parameter.
 - ◆ Number, string, and Boolean values
 - ◆ Individual Array Elements
- ❖ **Pass-by-reference** is used to pass entire array to a function
 - ◆ Pass the memory location where array is stored not the values
 - ◆ Modifications to the array in function affect the array values in entire program

Copyright © 2016 R.M. Laurie | 25

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Card Suits</title>
  <script type="text/javascript">
var Suit = new Array("&spades;","&clubs;","&hearts;","&diams;");
var Rank = new Array("A","2","3","4","5","6","7","8","9","10","J","Q","K");
document.write("<h3>Your hand is:<br />");
DealHand(Suit, Rank);
document.write("<br />Opponent hand is:<br />");
DealHand(Suit, Rank);
document.write("<br />Good Luck</h3>");

function DealHand(A, B) {
  for(var i=1; i <=5; i++)
    DealCard(A, B);
  document.write("<br />");
}

function DealCard(S, R) {
  var i, j;
  i = Math.floor(Math.random( ) * S.length);
  j = Math.floor(Math.random( ) * R.length);
  document.write("&nbsp;&nbsp;&nbsp;&nbsp;" + R[j] + S[i]);

  </script>
</head><body></body></html>
    
```

Your hand is:
4♠ 8♠ 4♦ A♥ J♥

Opponent hand is:
K♥ 3♦ 9♥ 5♣ 3♣

Good Luck

Your hand is:
A♣ 5♥ 8♦ A♠ 3♣

Opponent hand is:
3♦ A♣ 4♦ J♥ 5♦

Good Luck

Copyright © 2016 R.M. Laurie | 26

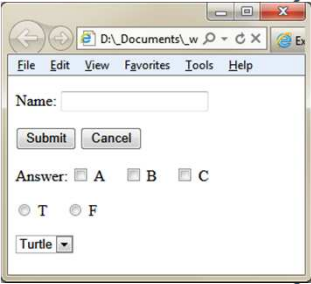
Event Driven Programming

- ❖ **Procedural Program Paradigm**
 - ◆ *Command line programming* is DOS style programming
 - ◆ Sequential processing modeled using flowcharts
 - ◆ Programs may include:
 - ◆ Sequential, selection, and repetition structures
 - ◆ Functions calls to user defined or library procedures
 - ◆ Arrays
- ❖ **Event Driven Program Paradigm**
 - ◆ Microsoft Windows and Mac OSX are operating system environments that designed around event driven concepts
 - ◆ Program execution is determined by user actions or **Events** (**onclick**, **onkeyup**, **onchange**) on a **Graphical User Interface**
 - ◆ Functions can read and write to **DOM Document Object Model**
 - ◆ Program divided into three sections:
 - ◆ **Graphical User Interface** = GUI created using HTML forms
 - ◆ **Events** triggered by user interacting with **GUI**
 - ◆ **Event handling** calls JavaScript functions

Copyright © 2016 R.M. Laurie | 27

HTML Forms and JavaScript Processing

- ❖ HTML Forms can be utilized to implement a (GUI) Graphical User Interface that interacts with JavaScript
 - ◆ Form element event triggers call to JavaScript function
 - ◆ JavaScript functions can read input data from form elements
 - ◆ JavaScript functions can write output data to form elements
 - ◆ Formatting of form elements can be done using CSS styles
- ❖ **Common form elements available in HTML**
 - ◆ Text Field
 - ◆ Buttons
 - ◆ Check boxes
 - ◆ Radio buttons
 - ◆ Select Menus
 - ◆ Text Area



Copyright © 2016 R.M. Laurie | 28

Form and Input Elements

- ❖ Form is a block level element


```
<form name="frmName" action="#"></form>
```

 - ◆ name attribute is identifier of the form for older browsers
 - ◆ id attribute is identifier of the form for newer browsers & DOM
 - ◆ action specifies the Server script on web server to process the sent data; for JavaScript "#" works well
 - ◆ Don't forget to close your form elements
- ❖ Text input element is for single line text input


```
<input type="text" name="txtFirstName" tabindex="1">
```


 - ◆ type="text" defines as a text box
 - ◆ name attribute is identifier of the form for older browsers
 - ◆ id attribute is identifier of the form for newer browsers & DOM
 - ◆ size attribute specifies character width of element
 - ◆ maxlength attribute specifies maximum number of characters entered
 - ◆ tabindex="1" is the first tab stop. Set to -1 to disallow tab
 - ◆ readonly="readonly" For results only not input
- ❖ Input button usually used to call function


```
<input type="button" name="btCalc" value="Calculate" onclick="calculate()">
```

Copyright © 2016 R.M. Laurie 29

DOM Access uses getElementById to access form objects

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>
  getElementById DOM Specifications
</title>
<script type="text/javascript">
  function NameSwap()
  {
    var First = document.getElementById("txtFirstName");
    var Last = document.getElementById("txtLastName");
    var Full = document.getElementById("txtFullName");
    Full.value = Last.value + ", " + First.value;
  }
</script>
</head>
<body>
<form id="frmName" action="#">
  <p>
    <label for="txtFirstName">First Name:</label>
    <input type="text" id="txtFirstName" tabindex="1">
  </p>
  <p>
    <label for="txtLastName">Last Name:</label>
    <input type="text" id="txtLastName" tabindex="2">
  </p>
  <p>
    <label for="txtFullName">Full Name:</label>
    <input type="text" id="txtFullName" tabindex="1">
  </p>
  <p>
    <input type="button" id="btnSwap" tabindex="2"
      value="Full Name" onclick="NameSwap();"
    </p>
</form>
</body>
</html>
```




Note that element id attribute is now the identifier. For old browser compatibility, sometimes name attributes included with id attributes

Copyright © 2016 R.M. Laurie 30

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>The Button Clicker</title>
<script>
  function AnsYes()
  {
    document.getElementById('Answer').innerHTML
      = "<b>I am glad you like programming</b>";
  }
  function AnsNo()
  {
    document.getElementById('Answer').innerHTML
      = "<b>You will like it if you study</b>";
  }
</script>
</head>
<body>
<h3>Button onclick Example program</h3>
<p>Do you like programming?</p>
<p><button onclick="AnsYes()">Yes</button> &nbsp;
  &nbsp;<button onclick="AnsNo()">No</button></p>
<p id="Answer">Click a button</p>
</body>
</html>
```

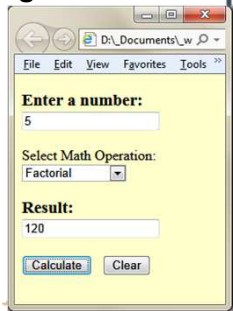
Note in this example a form is not utilized! Clicking a button calls a JavaScript function to change the inner text within an HTML element



Copyright © 2016 R.M. Laurie 31

Select Menu

- ❖ Select menus use **select** and **option** elements
- ❖ Select menus work well with setting parameters
- ❖ Can be used to provide a Graphical User Interface (GUI) for JavaScript Programs
- ❖ This example utilizes a select menu to choose one of three functions:
 - ◆ Square
 - ◆ Square Root
 - ◆ Factorial
- ❖ Calculate button click calls **Calculate()** function
 - ◆ **onclick** is an event (Stay Tuned)



Copyright © 2016 R.M. Laurie 32

Select Option Menu and Text Area

- ❖ **Select Option Menu is drop down menu**

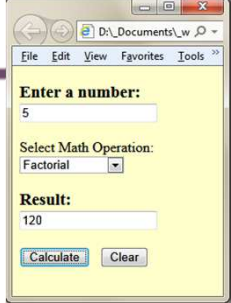
```
<select name="mnuMathOp" id="mnuMathOp">
  <option selected="selected">- Choose One -</option>
  <option>Square</option>
  <option>Square Root</option>
  <option>Factorial</option>
</select>
```
- ❖ **<select> element attributes**
 - ◆ **name** attribute is identifier of the form for older browsers
 - ◆ **id** attribute is identifier of the form for newer browsers & DOM
 - ◆ **size** attribute specifies options shown in menu
 - ◆ **disabled** can disable the menu
- ❖ **<option> element attributes**
 - ◆ **selected** is **true** or **false** if selected
- ❖ **Text area element is for multi-line text input**

```
<textarea rows="4" cols="30" id="txtAreaGreet"> Hello</textarea>
```

 - ◆ **rows** is the height
 - ◆ **cols** is the width
 - ◆ **id** is the identifier

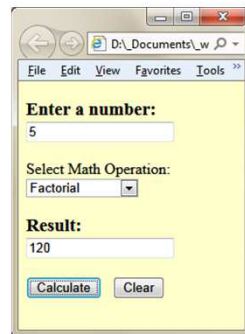
Copyright © 2016 R.M. Laurie 33

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Select Option Example</title>
    <script src="Calculator.js"></script>
  </head>
  <body style="background-color: #FFFFCC">
    <form name="frmCalc" action="#">
      <h3>Enter a number:<br />
        <input type="text" id="txtEntry" size="20"></h3>
      <p>Select Math Operation:<br />
        <select id="mnuMathOp">
          <option selected="selected">
            - Choose One -</option>
          <option id="opSq">Square</option>
          <option id="opRt">Square Root</option>
          <option id="opFc">Factorial</option>
        </select> </p>
      <h3>Result:<br />
        <input type="text" id="txtResult" size="20"
          readonly="readonly"></h3>
      <p><input type="button" name="btCalc" value="Calculate"
        onclick="Calculate()" &nbsp;&nbsp;&nbsp;&
        <input type="reset" name="btClear" value="Clear" /></p>
    </form>
  </body>
</html>
```



Copyright © 2016 R.M. Laurie 34

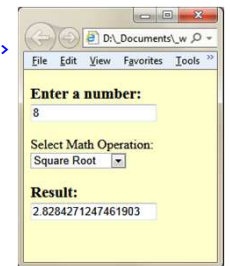
```
function Calculate()
{
  var Result, I, Selection;
  var Entry = document.getElementById("txtEntry");
  var Output = document.getElementById("txtResult");
  var OptSqur = document.getElementById("opSq");
  var OptRoot = document.getElementById("opRt");
  var OptFact = document.getElementById("opFc");
  Entry = parseFloat(Entry.value);
  if(OptSqr.selected)
    Result = Entry * Entry;
  else if(OptRoot.selected)
    Result = Math.sqrt(Entry);
  else if(OptFact.selected)
  {
    Result = 1;
    for(I = 1; I <= Entry; I++)
      Result = Result * I;
  }
  else
    window.alert("Select an Operation!");
  Output.value = Result;
  return;
}
```



Copyright © 2016 R.M. Laurie 35

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Select Option Example</title>
    <script src="Calculator.js"></script>
  </head>
  <body style="background-color: #FFFFCC">
    <form name="frmCalc" action="#">
      <h3>Enter a number:<br />
        <input type="text" id="txtEntry" size="20"
          onchange="Calculate()"></h3>
      <p>Select Math Operation:<br />
        <select id="mnuMathOp" onchange="Calculate()">
          <option selected="selected">
            - Choose One -</option>
          <option id="opSq">Square</option>
          <option id="opRt">Square Root</option>
          <option id="opFc">Factorial</option>
        </select> </p>
      <h3>Result:<br />
        <input type="text" id="txtResult" size="20"
          readonly="readonly"></h3>
    </form>
  </body>
</html>
```

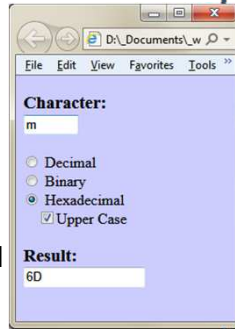
Same JavaScript function is called but this code uses the onchange event for either select menu or entry text box



Copyright © 2016 R.M. Laurie 36

GUI Using Radio Buttons and Check Box

- ❖ Radio buttons and check boxes can enhance a GUI Form
- ❖ In this example you can type in a single ASCII character and convert it to the specified number system
- ❖ Note that the checkbox is enabled only when hexadecimal is selected
- ❖ The display changes when the character is changed, any radio button is clicked, or the check box is clicked (when enabled)



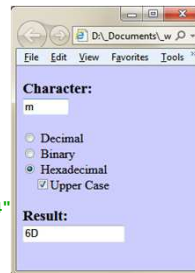
Copyright © 2016 R.M. Laurie 37

Check Box and Radio Buttons

- ❖ Checkboxes use input element
 - <input type="checkbox" id="chkUpper" onclick="Convert()" disabled="disabled">
 - ◆ type="checkbox" defines as a check box
 - ◆ name attribute is identifier of the form for older browsers
 - ◆ id attribute is identifier of the form for newer browsers & DOM
 - ◆ tabindex="1" is the first tab stop. Set to -1 to disallow tab
 - ◆ checked="checked" initializes to checked
 - ◆ disabled="disabled" disallows changing
- ❖ Radio buttons use input element and has same name to interlink
 - <input type="radio" name="radConv" id="radDec" onclick="Convert()">
 - ◆ type="radio" defines as a radio button
 - ◆ name attribute is required if link buttons to allow only one selection
 - ◆ id attribute must be unique for the page
 - ◆ tabindex="1" is the first tab stop. Set to -1 to disallow tab
 - ◆ checked="checked" initializes to checked
 - ◆ disabled="disabled" disallows changing

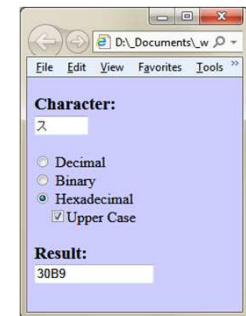
Copyright © 2016 R.M. Laurie 38

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Convert Character</title>
    <script src="baseConverter.js"></script>
  </head>
  <body style="background-color: #CCCCFF">
    <form id="frmConvert" name="frmConvert" action="#">
      <h3>Character:<br>
      <input type="text" id="txtEntry" value="" size="4"
        maxlength="1" onkeyup="Convert()"></h3>
      <p><input type="radio" name="radConv" id="radDec"
        onclick="Convert()"> Decimal<br>
        <input type="radio" name="radConv" id="radBin"
        onclick="Convert()"> Binary<br />
        <input type="radio" name="radConv" id="radHex"
        onclick="Convert()"> Hexadecimal<br>
        &nbsp;&nbsp;&nbsp;&nbsp;<input type="checkbox" id="chkUpper"
        onclick="Convert()" disabled="disabled">Upper Case</p>
      <h3>Result:<br>
      <input type="text" id="txtResult" size="16"
        maxlength="10"></h3>
    </form>
  </body>
</html>
```



Copyright © 2016 R.M. Laurie 39

```
function Convert()
{
  var Result="", KeyCode;
  var Entry = document.getElementById("txtEntry");
  var Dec = document.getElementById("radDec");
  var Bin = document.getElementById("radBin");
  var Hex = document.getElementById("radHex");
  var Upper = document.getElementById("chkUpper");
  var Output = document.getElementById("txtResult");
  KeyCode = Entry.value.charCodeAt(0); // Unicode
  Upper.disabled=true;
  if(Dec.checked)
    Result = KeyCode.toString(10);
  else if(Bin.checked)
    Result = KeyCode.toString(2);
  else if(Hex.checked)
  {
    Upper.disabled=false;
    Result = KeyCode.toString(16);
    if(Upper.checked)
      Result = Result.toUpperCase();
    else
      Result = Result.toLowerCase();
  }
  Output.value = Result;
  return;
}
```



Copyright © 2016 R.M. Laurie 40