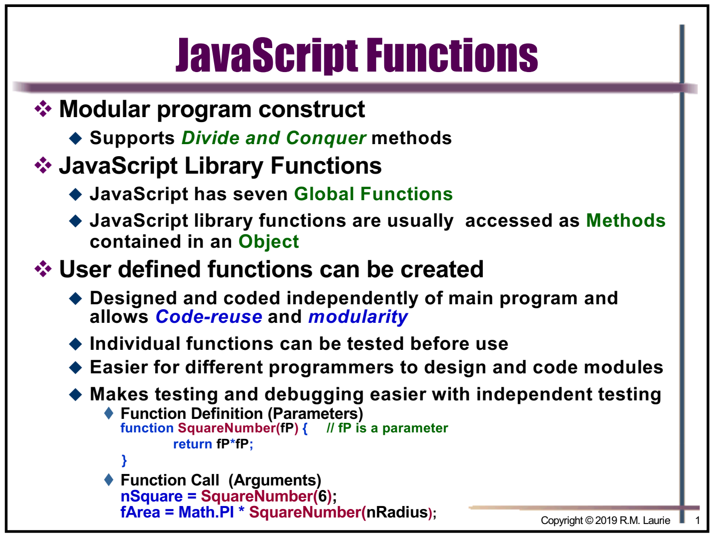


JavaScript Functions

- ❖ Modular program construct
  - ◆ Supports *Divide and Conquer* methods
- ❖ 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
  - ◆ Designed and coded independently of main program and allows *Code-reuse* and *modularity*
  - ◆ Individual functions can be tested before use
  - ◆ Easier for different programmers to design and code modules
  - ◆ Makes testing and debugging easier with independent testing
    - ♦ Function Definition (Parameters)

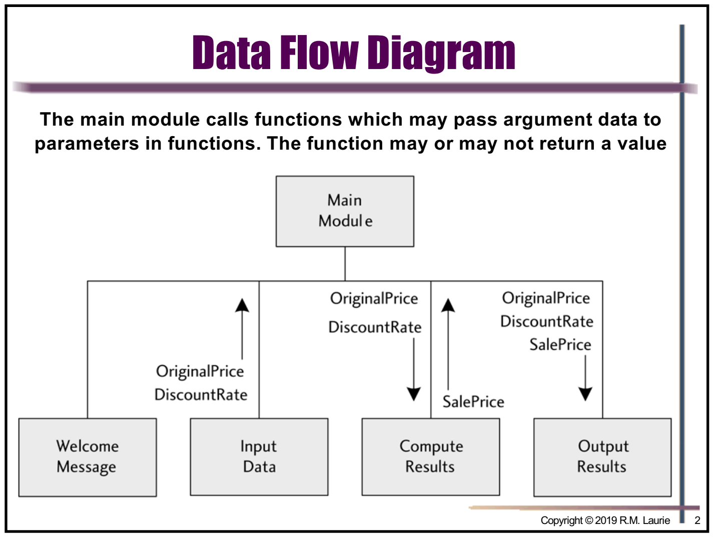
```
function SquareNumber(fP) {    // fP is a parameter
        return fP*fP;
}
```

    - ♦ Function Call (Arguments)

```
nSquare = SquareNumber(6);
fArea = Math.PI * SquareNumber(nRadius);
```

1

1

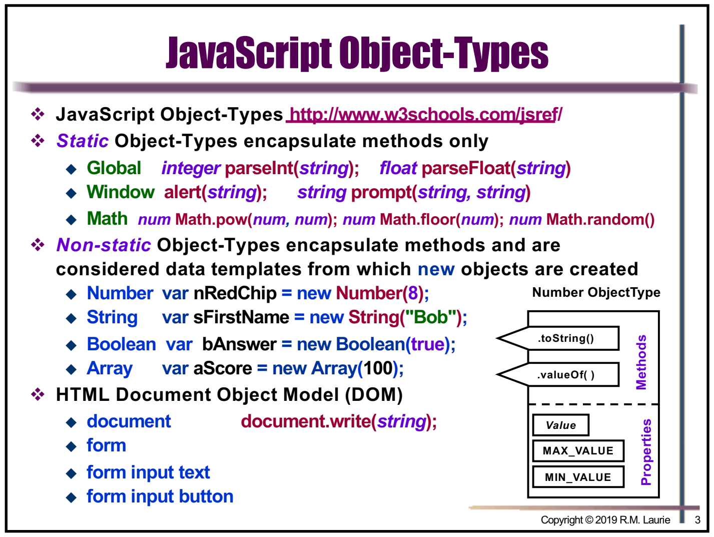


Data Flow Diagram

The main module calls functions which may pass argument data to parameters in functions. The function may or may not return a value

2

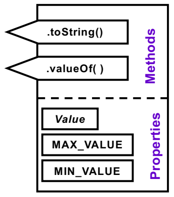
2



JavaScript Object-Types

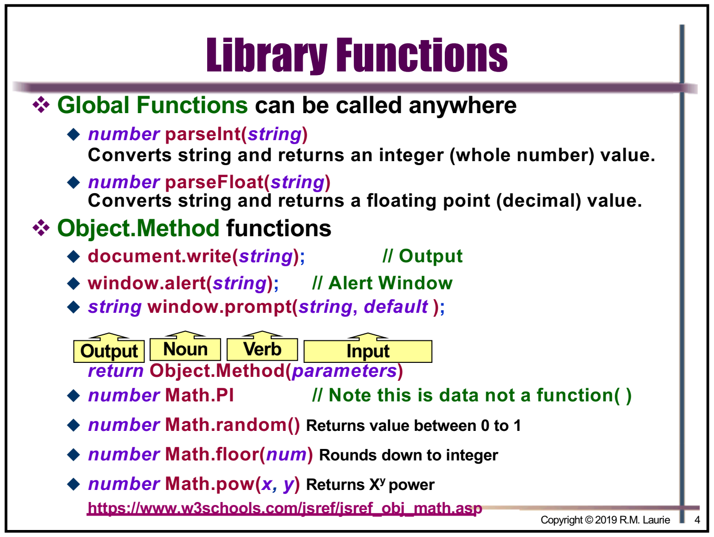
- ❖ JavaScript Object-Types http://www.w3schools.com/jsref/
- ❖ *Static* Object-Types encapsulate methods only
  - ◆ Global *integer* parseInt(*string*); *float* parseFloat(*string*)
  - ◆ Window alert(*string*); *string* prompt(*string, string*)
  - ◆ Math *num* Math.pow(*num, num*); *num* Math.floor(*num*); *num* Math.random()
- ❖ *Non-static* Object-Types encapsulate methods and are considered data templates from which new objects are 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



3

3



Library Functions

- ❖ Global Functions can be called anywhere
  - ◆ *number* parseInt(*string*)
    Converts string and returns an integer (whole number) value.
  - ◆ *number* parseFloat(*string*)
    Converts string and returns a floating point (decimal) value.
- ❖ Object.Method functions
  - ◆ document.write(*string*); // Output
  - ◆ window.alert(*string*); // Alert Window
  - ◆ *string* window.prompt(*string*, *default*);

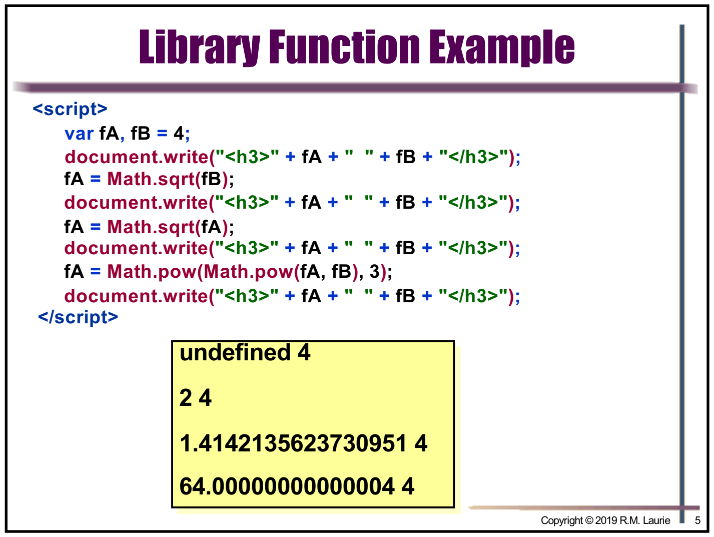
Output Noun Verb Input
return Object.Method(*parameters*)

  - ◆ *number* Math.PI // Note this data not a function( )
  - ◆ *number* Math.random() Returns value between 0 to 1
  - ◆ *number* Math.floor(*num*) Rounds down to integer
  - ◆ *number* Math.pow(*x*, *y*) Returns X^y power

https://www.w3schools.com/jsref/jsref_obj_math.asp

4

4



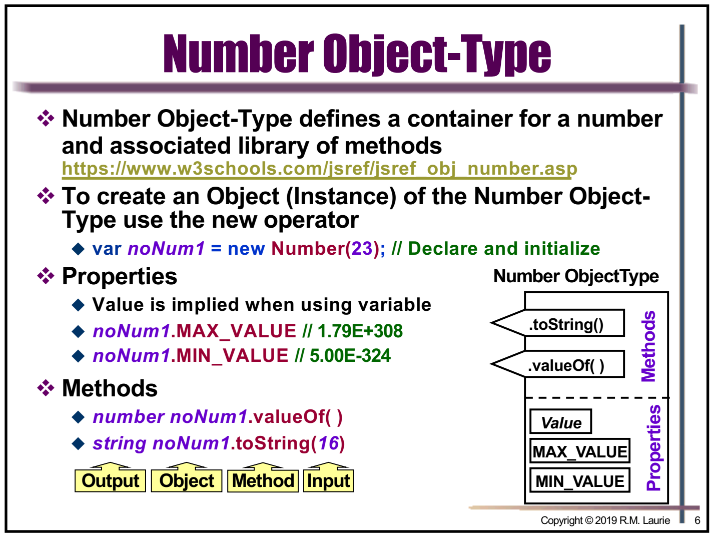
Library Function Example

```
<script>
    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>
```

```
undefined 4

2 4

1.4142135623730951 4

64.00000000000004 4
```

Copyright © 2019 R.M. Laurie 5

5

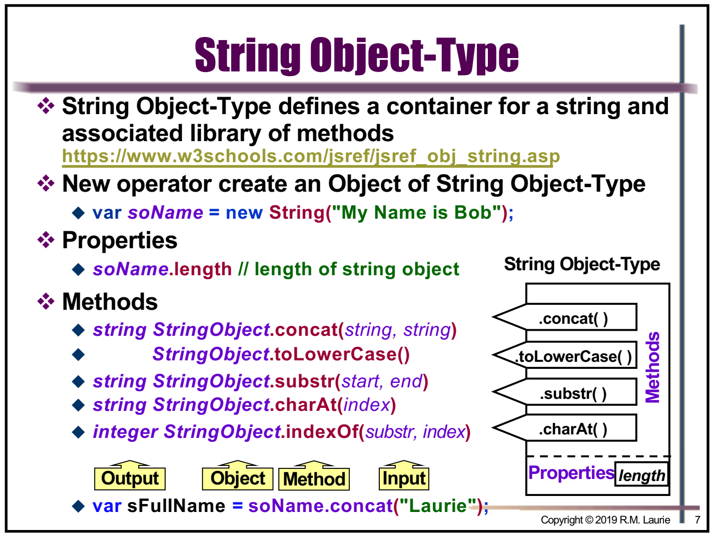


Number Object-Type

- ❖ Number Object-Type defines a container for a number and associated library of methods https://www.w3schools.com/jsref/jsref_obj_number.asp
- ❖ To create an Object (Instance) of the Number Object-Type use the new operator
  - ◆ var *noNum1* = new Number(23); // Declare and initialize
- ❖ Properties
  - ◆ Value is implied when using variable
  - ◆ *noNum1*.MAX_VALUE // 1.79E+308
  - ◆ *noNum1*.MIN_VALUE // 5.00E-324
- ❖ Methods
  - ◆ *number noNum1*.valueOf( )
  - ◆ *string noNum1*.toString(*16*)

Copyright © 2019 R.M. Laurie 6

6

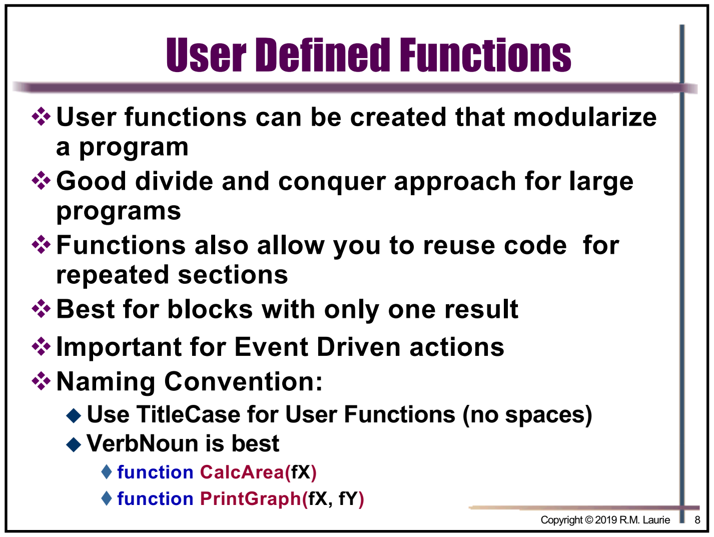


String Object-Type

- ❖ String Object-Type defines a container for a string and associated library of methods https://www.w3schools.com/jsref/jsref_obj_string.asp
- ❖ New operator create an Object of String Object-Type
  - ◆ var *soName* = new String("My Name is Bob");
- ❖ Properties
  - ◆ *soName*.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*)
  - ◆ var sFullName = soName.concat("Laurie");

Copyright © 2019 R.M. Laurie 7

7

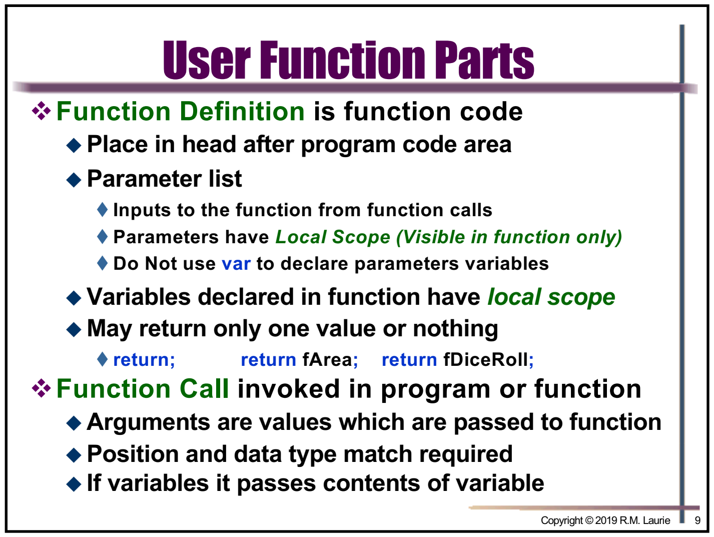


User Defined Functions

- ❖ 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
    - ♦ function CalcArea(fX)
    - ♦ function PrintGraph(fX, fY)

Copyright © 2019 R.M. Laurie 8

8

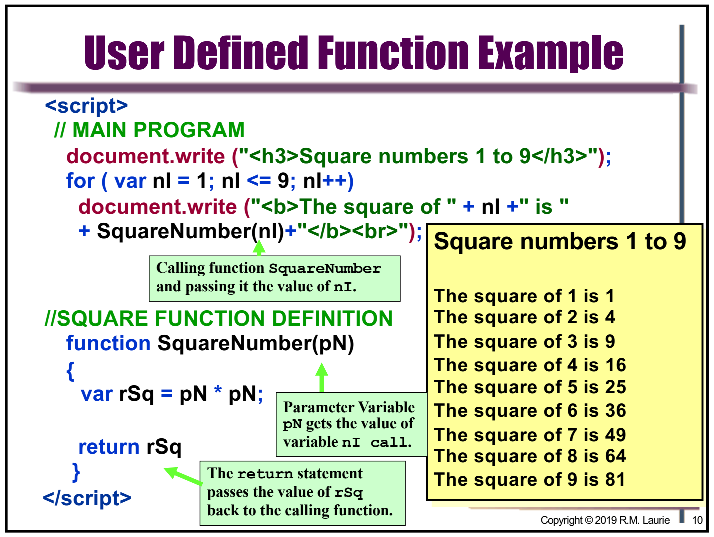


User Function Parts

- ❖ **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
  - ◆ Variables declared in function have *local scope*
  - ◆ May return only one value or nothing
    - ♦ return;   return fArea;   return fDiceRoll;
- ❖ **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 © 2019 R.M. Laurie 9

9



User Defined Function Example

```
<script>
 // 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(pN)
  {
   var rSq = pN * pN;

   return rSq
   }
</script>
```

Calling function SquareNumber and passing it the value of nI.

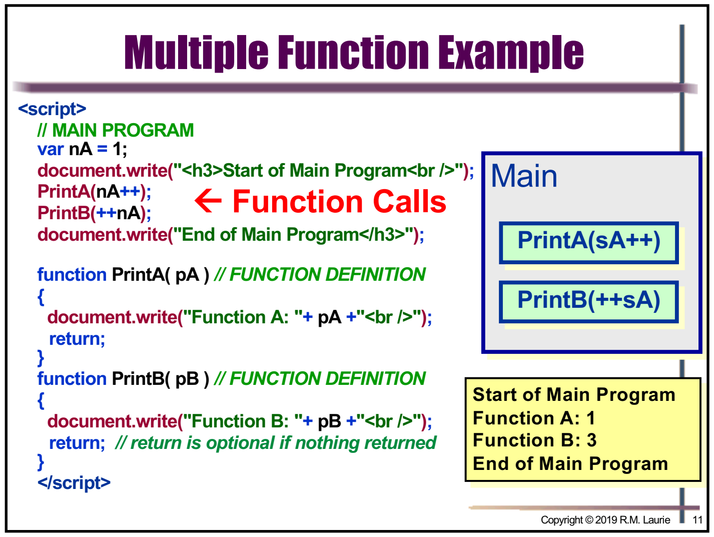
Parameter Variable pN gets the value of variable nI call.

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

```
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
```

Copyright © 2019 R.M. Laurie 10

10



Multiple Function Example

```
<script>
  // 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;  // return is optional if nothing returned
  }
  </script>
```

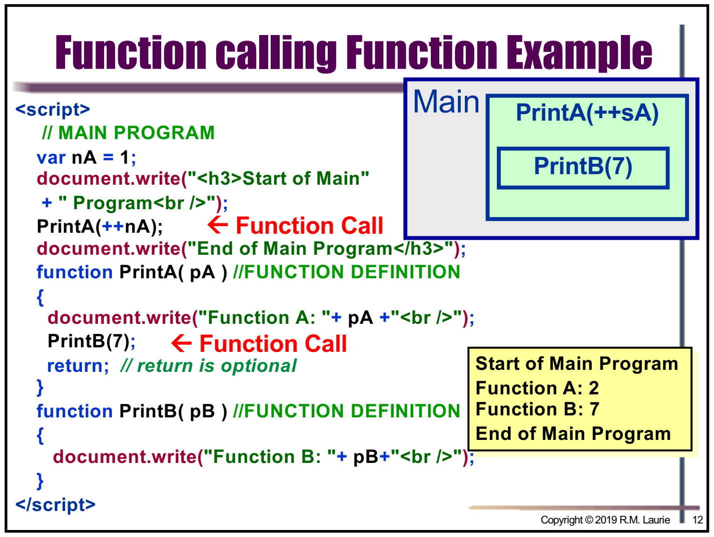
← Function Calls

Main
PrintA(sA++)
PrintB(++sA)

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

Copyright © 2019 R.M. Laurie 11

11



Function calling Function Example

```
<script>
   // 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);
     return;  // return is optional
   }
   function PrintB( pB ) //FUNCTION DEFINITION
   {
     document.write("Function B: "+ pB+"<br />");
   }
</script>
```

← Function Call

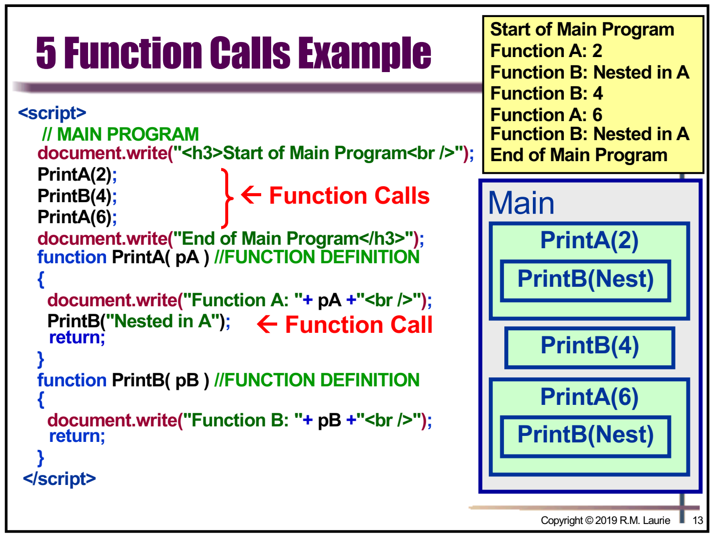
← Function Call

Main
PrintA(++sA)
PrintB(7)

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

Copyright © 2019 R.M. Laurie 12

12



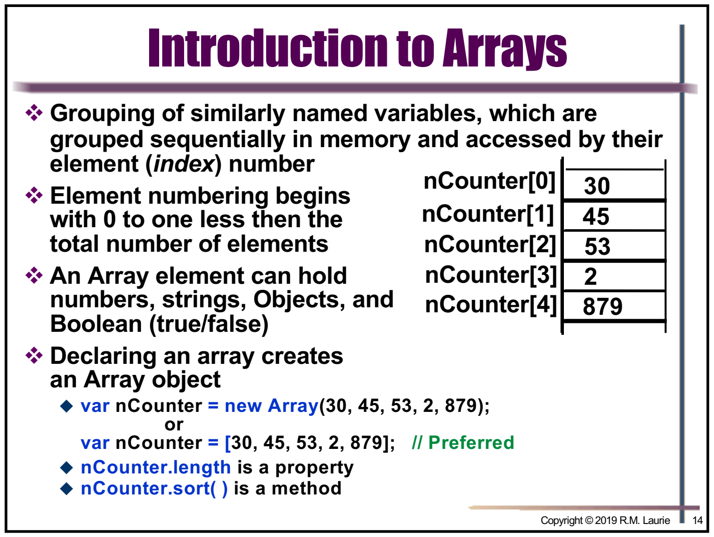
5 Function Calls Example

```
<script>
  // MAIN PROGRAM
 document.write("<h3>Start of Main Program<br />");
 PrintA(2);
 PrintB(4);          } ← Function Calls
 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 Call
   return;
 }
 function PrintB( pB ) //FUNCTION DEFINITION
 {
   document.write("Function B: "+ pB +"<br />");
   return;
 }
</script>
```

```
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
```

Copyright © 2019 R.M. Laurie 13

13



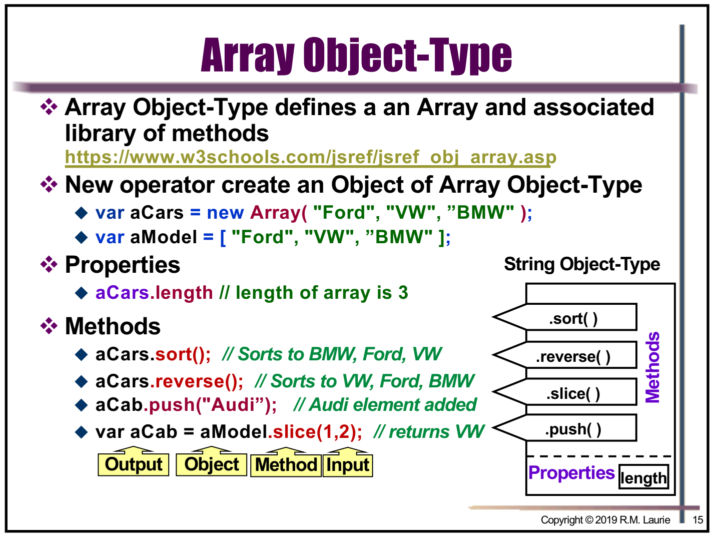
Introduction to Arrays

- 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 then the total number of elements
- An Array element can hold numbers, strings, Objects, and Boolean (true/false)
- Declaring an array creates an Array object
  - var nCounter = new Array(30, 45, 53, 2, 879);
    or
    var nCounter = [30, 45, 53, 2, 879]; // Preferred
  - nCounter.length is a property
  - nCounter.sort( ) is a method

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

Copyright © 2019 R.M. Laurie 14

14

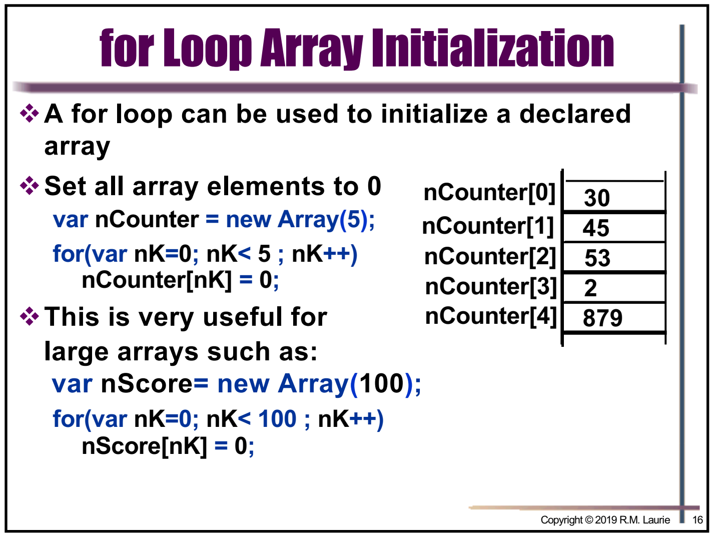


Array Object-Type

- Array Object-Type defines a an Array and associated library of methods
  https://www.w3schools.com/jsref/jsref_obj_array.asp
- New operator create an Object of Array Object-Type
  - var aCars = new Array("Ford", "VW", "BMW");
  - var aModel = ["Ford", "VW", "BMW"];
- Properties
  - aCars.length // length of array is 3
- Methods
  - aCars.sort(); // Sorts to BMW, Ford, VW
  - aCars.reverse(); // Sorts to VW, Ford, BMW
  - aCab.push("Audi"); // Audi element added
  - var aCab = aModel.slice(1,2); // returns VW

Copyright © 2019 R.M. Laurie 15

15



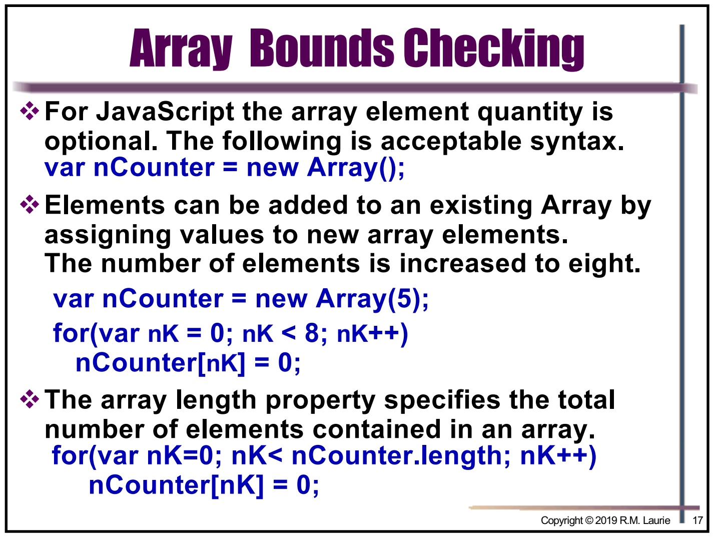
for Loop Array Initialization

- 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;
  ```

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

Copyright © 2019 R.M. Laurie 16

16



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.

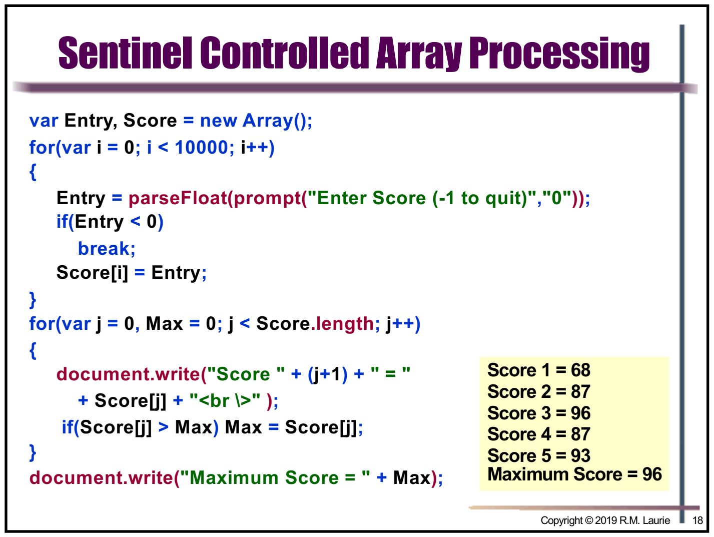
```
var nCounter = new Array(5);
for(var nK = 0; nK < 8; nK++)
  nCounter[nK] = 0;
```

- 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;
```

17

17



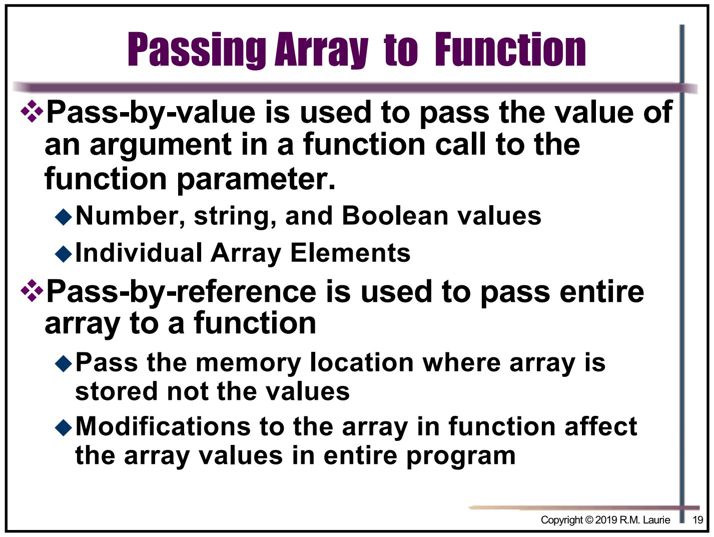
Sentinel Controlled Array Processing

```
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
```

18

18

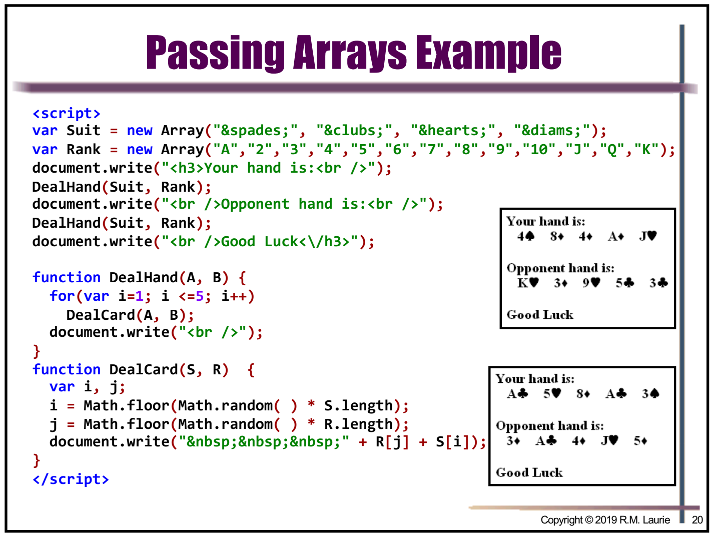


Passing Array to Function

- 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

19

19



Passing Arrays Example

```
<script>
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;" + R[j] + S[i]);
}
</script>
```

```
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
```

20

20