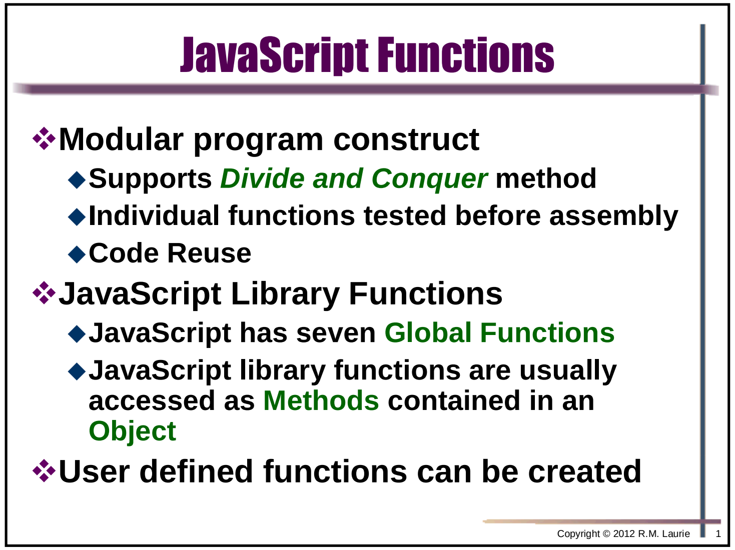
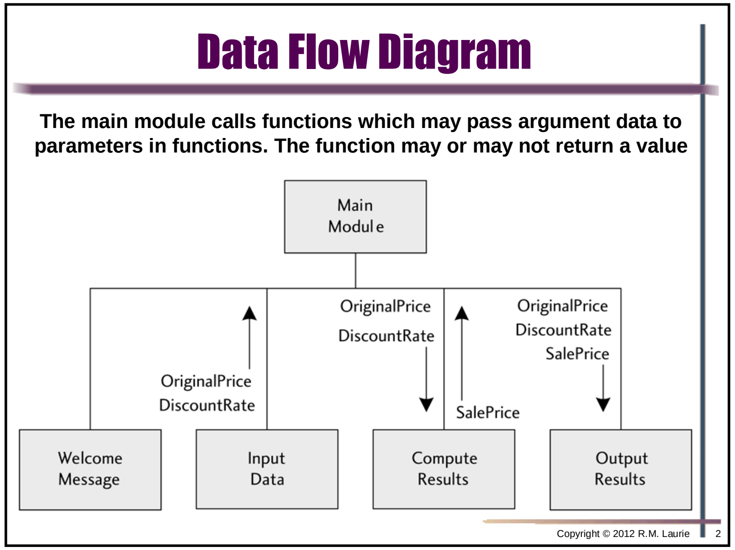


JavaScript Functions

- ❖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



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

Main Modul e

OriginalPrice
DiscountRate

OriginalPrice
DiscountRate

OriginalPrice
DiscountRate
SalePrice

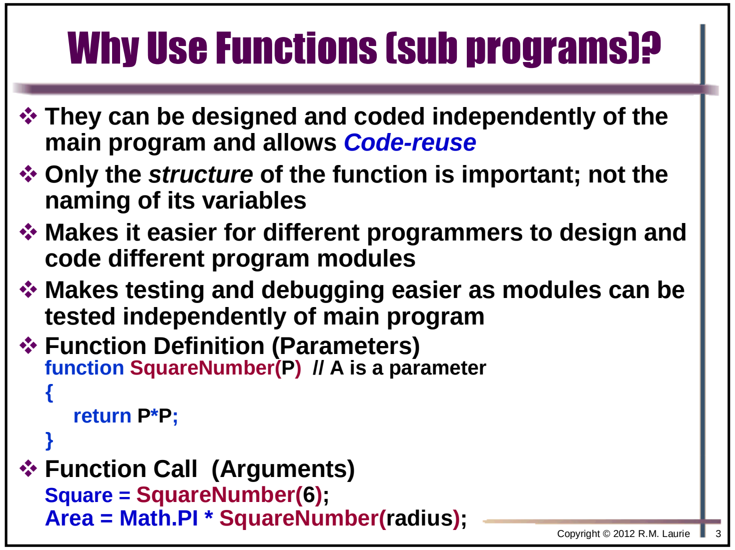
SalePrice

Welcome Message

Input Data

Compute Results

Output Results



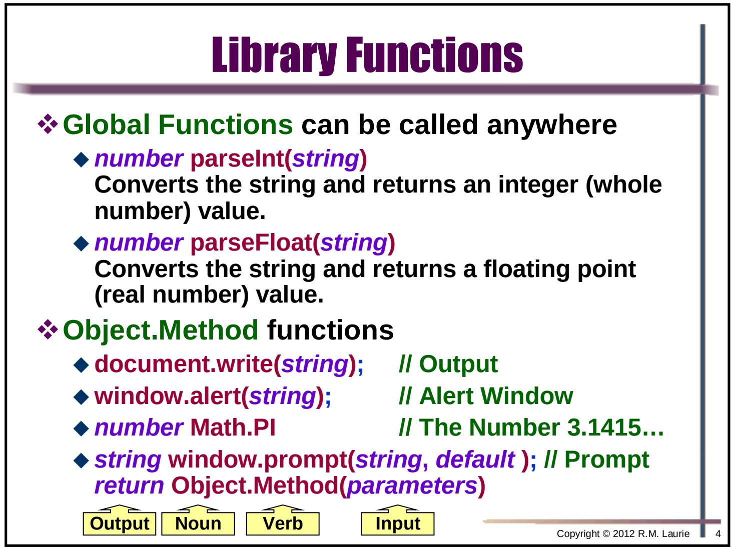
Why Use Functions (sub programs)?

- ❖ 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(P)  // A is a parameter
{
    return P*P;
}
```

- ❖ Function Call (Arguments)

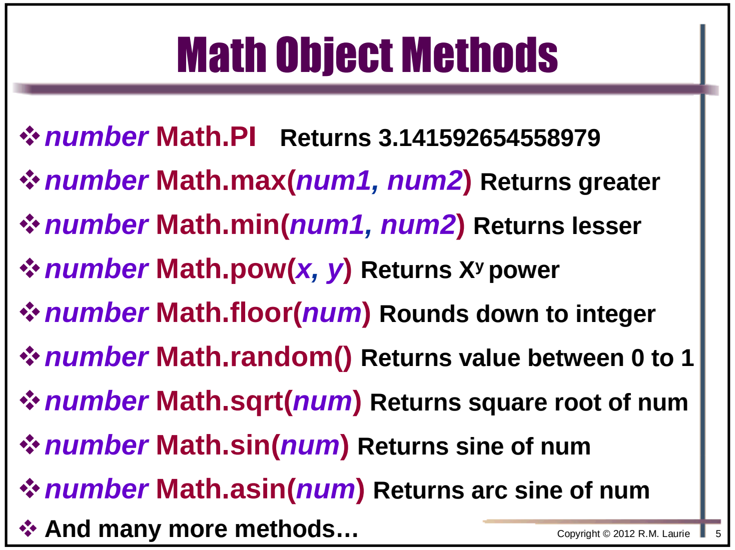
```
Square = SquareNumber(6);
Area = Math.PI * SquareNumber(radius);
```



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*)

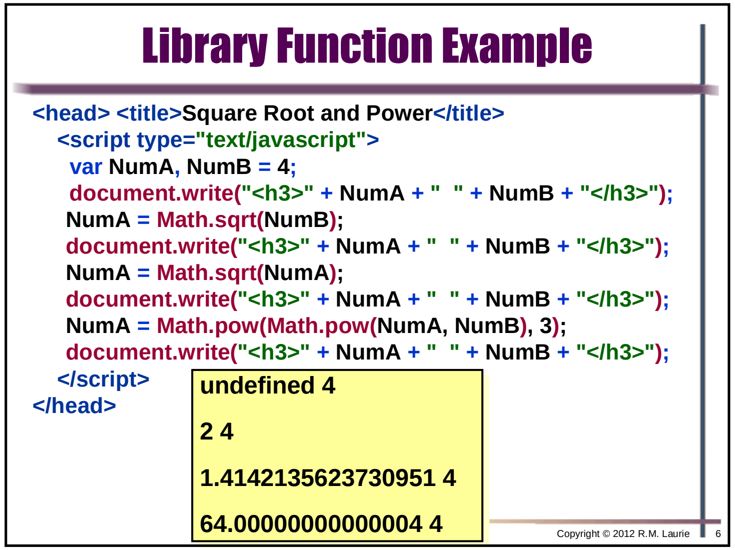
Output Noun Verb Input



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 © 2012 R.M. Laurie 5



Library Function Example

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

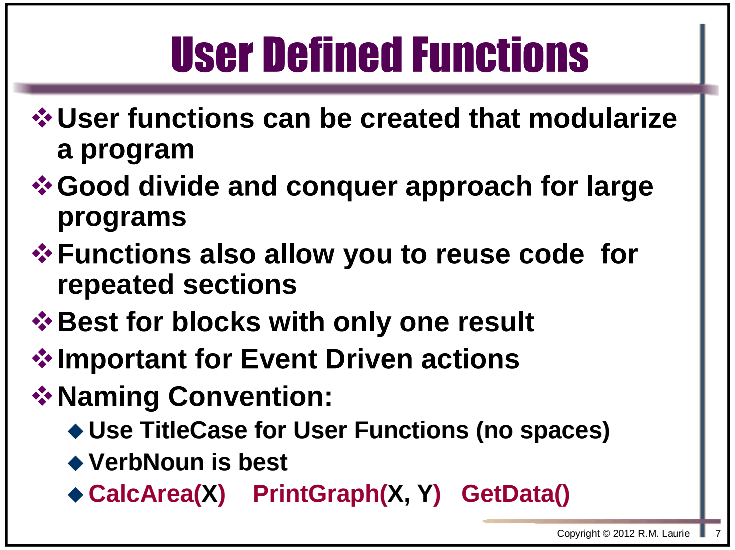
```
undefined 4

2 4

1.4142135623730951 4

64.00000000000004 4
```

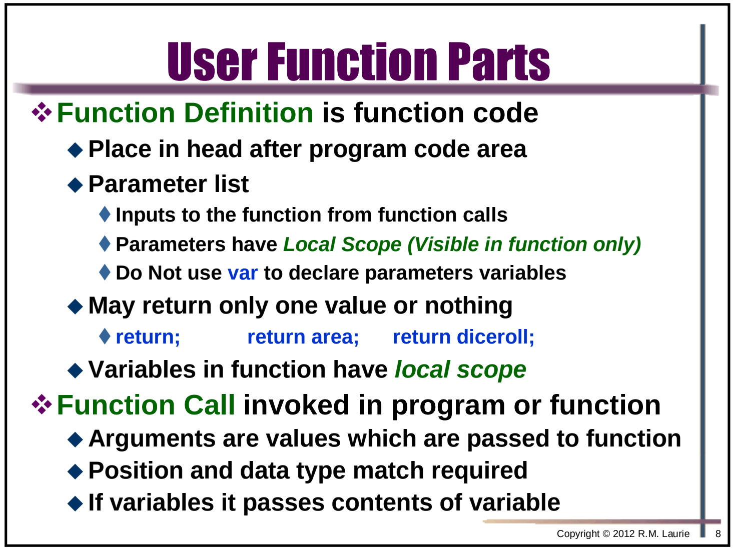
Copyright © 2012 R.M. Laurie 6



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(X) PrintGraph(X, Y) GetData()

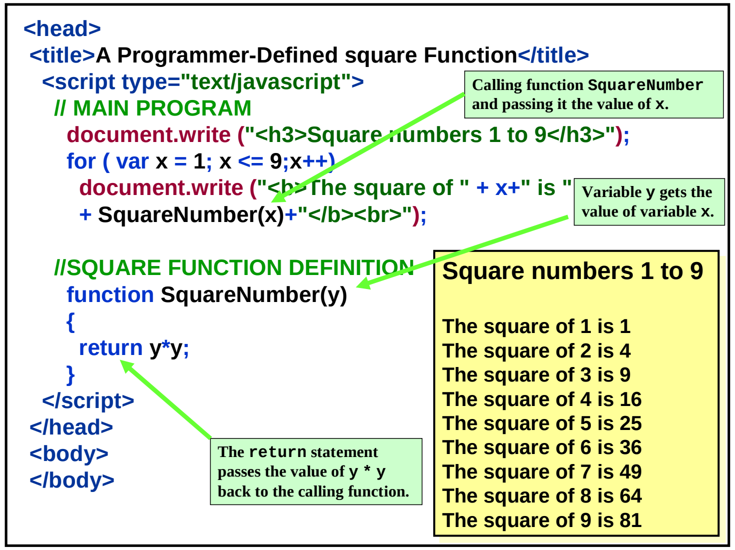
Copyright © 2012 R.M. Laurie 7



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 area; return diceroll;
  - 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 © 2012 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 x = 1; x <= 9;x++)
     document.write ("<b>The square of " + x+" is "
     + SquareNumber(x)+"</b><br>");

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

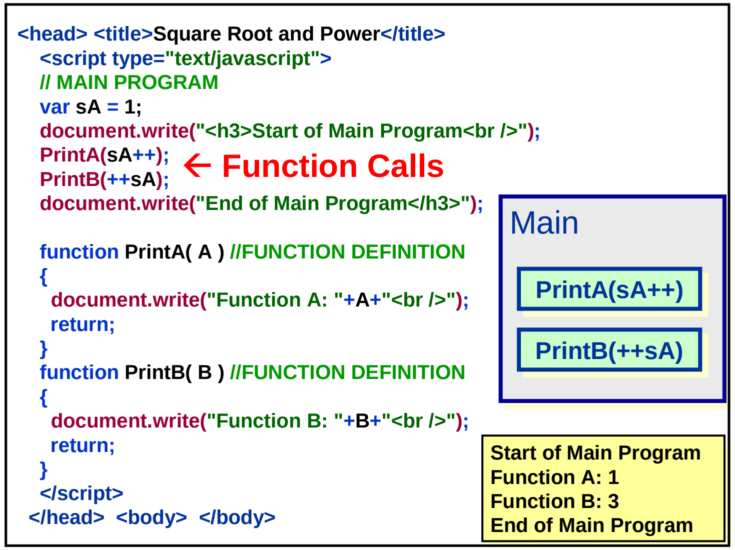
Calling function `SquareNumber` and passing it the value of `x`.

Variable `y` gets the value of variable `x`.

The `return` statement passes the value of `y * y` 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
```

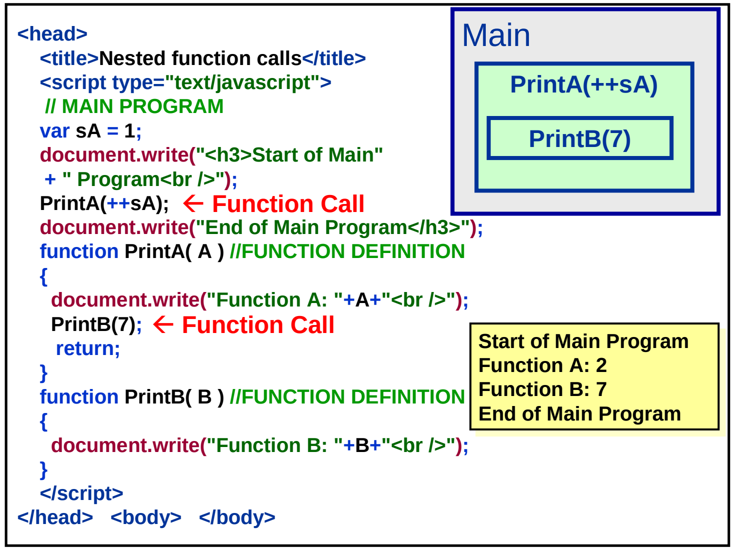


```
<head> <title>Square Root and Power</title>
   <script type="text/javascript">
   // MAIN PROGRAM
   var sA = 1;
   document.write("<h3>Start of Main Program<br />");
   PrintA(sA++);
   PrintB(++sA);
   document.write("End of Main Program</h3>");

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

← Function Calls

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

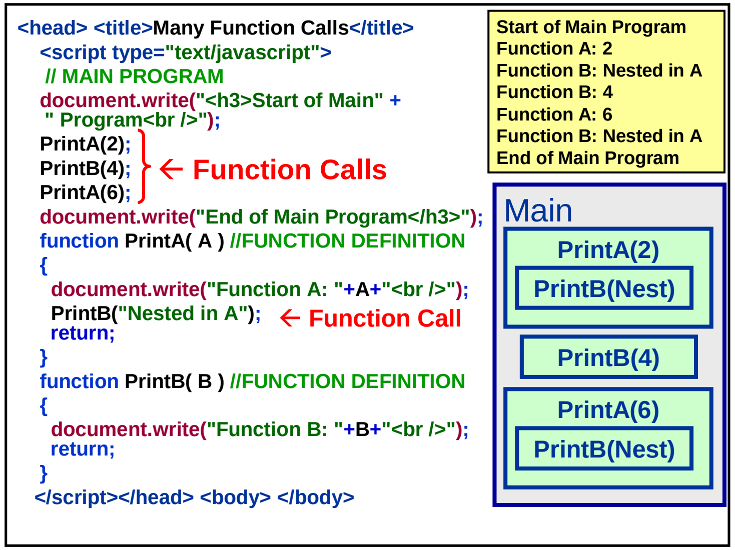


```
<head>
   <title>Nested function calls</title>
   <script type="text/javascript">
    // MAIN PROGRAM
   var sA = 1;
   document.write("<h3>Start of Main"
    + " Program<br />");
   PrintA(++sA);
   document.write("End of Main Program</h3>");
   function PrintA( A ) //FUNCTION DEFINITION
   {
 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

← Function Call

← Function Call

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



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

} ← Function Calls

← Function Call

```
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
```