## Functions, Part 1 of 2



#### **Topics**

- Using Predefined Functions
- Programmer-Defined Functions
- Using Input Parameters
- Function Header Comments

**Review of Structured Programming** 



- Structured programming is a problem solving strategy and a programming methodology that includes the following guidelines:
  - The program uses only the sequence, selection, and repetition control structures.
  - The flow of control in the program should be as simple as possible.
  - The construction of a program embodies topdown design.

#### **Functions**



- When program control encounters a function name, the function is called (invoked).
  - Program control passes to the function.
  - The function is executed.
  - Control is passed back to the place where the function was called.

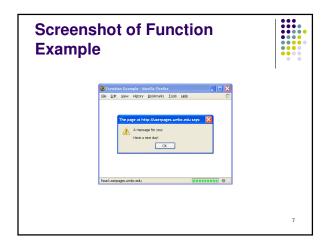
#### **Functions**

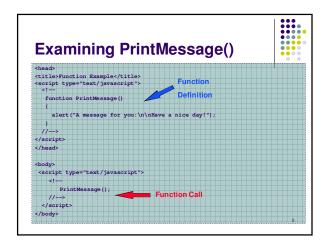


- We have used several predefined functions so far:
  - alert()
  - prompt()
  - document.write()
  - toFixed()
  - parseInt()
  - parseFloat()
- Programmers can write their own functions.
- Typically, each module in a program's design hierarchy chart is implemented as a function.

## **Sample Function Call** alert is the name of a predefined function in the JavaScript language alert("Hello World!"); this statement is is known as a function call this is a string we are passing as an argument (parameter) to the alert function

## **Sample Programmer-Defined Function** <title>Function Example</title> <script type="text/javascript"> function PrintMessage() alert ("A message for you: \n\nHave a nice day!"); </script> <script type="text/javascript"> <!--</pre> PrintMessage(): </script>





## 

### **The Function Definition**

 Control is passed to the function by the function call. The statements within the function body will then be executed.

```
function PrintMessage()
{
   alert("A message for you:\n\nHave a nice day!");
}
```

 After the statements in the function have completed, control is passed back to the place where the function was called.

10

# function FunctionName ( parameter<sub>1</sub>, . . . , parameter<sub>n</sub>) { variable declaration(s) statement(s)

**General Function Definition Syntax** 

 If there are no parameters, there should be nothing inside of the ()'s

function FunctionName() { ...

• There may be no variable declarations.

## **Using Input Parameters**



- Often it is the case that we would like to be able to share information with the function.
- It is possible to send input parameters into the function.
- We can pass information from the place where the function is called.
- The next slide illustrates sending a single parameter into a function.

12

```
<head>
<title>Function Parameter Example</title>
<script type="text/javascript">
<!--
function PrintMessage(counter)
{
  var i;
  for(i = 1; i <= counter; i = i + 1)
  {
    alert("Have a nice day!");
  }
}

//-->
</script>
</nead>
<hody>
<script type="text/javascript">
<!--
  var counter;
  counter;
  counter;
  counter = prompt("Enter a number:");
  PrintMessage(counter);
  //-->
  </script>
</sc
```

## **Good Programming Practice**



- You should include a function header comment before the definition of each function.
- This is a good practice and is required by the 104 Coding Standards.
- Your header comments should be neatly formatted and contain the following information:
  - function name
  - function description (what it does)
  - a list of any input parameters and their meanings
  - a list of any output parameters and their meanings
  - a description of any special conditions

14

## **Example of a Function Header Comment**



```
/***

** PrintMessage - prints a message a specified number of times

** Inputs: counter - the number of times the message will be
printed

** Outputs: None

***

** Outputs: None

**

function PrintMessage (counter)

{
    var i;
    for(i = 1; i <= counter; i = i + 1)
    {
        alert("Have a nice day!");
    }
}
```

15