Functions and Events

Dr. Abdallah Mohamed

Key Points

1) Functions are used to group statements that perform a particular task so that they can be easily used.

2) Forms are used to input and receive output from the computer. A form consists of elements such as buttons, sliders, lists, and boxes.

3) Events are notifications that something occurs. Your program contains event handlers to indicate what to do when an event is detected.

Important: Programming Incrementally

NEVER write code in a *monolithic* fashion.

ALWAYS write code by adding only a few lines or features at a time and then testing.

Thus, coding is an ***incremental process***.

- ◆ Write some code.
- ◆ Test in browser. Fix errors.
- ◆ Repeat (until done).

Problem decomposition involves breaking down a large problem into subproblems which are easier to solve. Dividing problems into subproblems is called ***divide and conquer***.

Suggestion: Complete HTML document tags before writing complicated JavaScript code and event handling.

Functions and Procedures

A *procedure* (or *method*) is **a sequence of statements that have a specific task** that they perform.

- The statements in the procedure are mostly independent of other statements in the program.

A *function* is a procedure that returns a value after it is executed.

We use functions so that we **do not have to type the same code over and over**. We can also use functions that are built-in to the language or written by others.

★ *Defining and Calling Functions and Procedures*

Creating a function involves writing the statements and providing a *function declaration* with:

- a **name** (follows the same rules as identifiers)
- list of the inputs (called **parameters**) and their data types
- the output (**return value**) if any

Calling (or executing) a function involves:

- providing the name of the function
- providing the values for all parameters (inputs) if any
- providing space (variable name) to store the output (if any)

Defining a Function

Consider a method that converts a temperature in Celsius to Fahrenheit:

Function Declaration

```
function convertC2F(tempInC)
{
    return tempInC/5 * 9 + 32;
}
```

Function Keyword

Function Identifier (Name)

Parameter Identifier (Name)

Creating a Function

Question: This function is supposed to take two numbers as input and return their sum. What is wrong with it?

```
function addTwoNum(num1) {
    var result = num1 + num2;
}
```

A) The two numbers are not added together.

B) The result of the addition is not returned back.

C) Only one number to add is passed into the function.

D) The name of the function is not correct.

Creating a Function (2)

Question: We want to create a function that multiplies two numbers together. Which of these functions is correct?

A)
```
multTwoNum(num1, num2) {
    return num1 * num2;
}
```

B)
```
function multTwoNum(num1, num2, num3){
    return num1 * n2;
}
```

C)
```
function multTwoNum(num1, num2) {
    var result = num1 * num2;
}
```

D)
```
function multTwoNum(num1, num2) {
    return num1 * num2;
}
```

Example: Calling Convert Function

This is how to call our `convertC2F` function: **Order of Operations**

```
var myCTemp = 50;                              1
var myFTemp;                                   2


myFTemp = convertC2F(myCTemp);                 3
alert(myCTemp +"C is = "+myFTemp+"F");         6


function convertC2F( tempInC ){                4
    return tempInC / 5 * 9 + 32;               5
}
```

What happens if we move the function convertC2F to the top of the code?

Functions and Procedures Notes

- When declaring a function, you must put the parenthesis " `()` " after the name even if the function has no parameters.

- If a function returns nothing, you can just say " `return;` ".

- *Parameter* is the term used for input when viewing from inside the function (function's perspective). *Argument* is the term used for input when viewing from outside the function.

- Functions are declared only once, but can be called as many times as you want.

- Execution of the method halts at the return statement and any value in the statement is passed back to the caller.

- You may have multiple return statements in a method, but only one will ever be executed for a given execution.

Functions

Question: What is the output of this code?

```
var num=9;

var result = doubleNum(num);
document.write(result);

function doubleNum(n)
{   return n*2;  }
```

A) nothing

B) error

C) 9

D) 18

Functions (2)

Question: What is the output of this code?

```
function subtractNum(a, b) {
    return a-b; }

var x=5, y=8;

var result = subtractNum(x,y);
result = result + subtractNum(y,x);
document.write(result);
```

A) error

B) 3

C) -3

D) 0

Functions (3)

Question: What is the output of this code?

```
var num=9;

var result = doubleNum(doubleNum(num));
document.write(result);

function doubleNum(n)
{   return n*2;  }
```

A) 36

B) 18

C) 9

D) error

Functions (4)

Question: What is the output of this code?

```
function evenOrOdd(n)
{   if (n % 2 == 0) {
        return "even";
    }
    else {
        return "odd";
    }
}
var num = 10;
document.write(evenOrOdd(11));
document.write(evenOrOdd(num));
```

A) oddodd

B) oddeven

C) evenodd

D) eveneven

Built-In Functions

JavaScript has many built-in functions that you can use. These methods are grouped into objects.

◆An *object* is a related group of code and data.

(Some of the) pre-defined objects in JavaScript:

◆Array

◆Date

◆Math

⇨Functions: abs(x), floor(x), min(x,y), max(x,y), random(), sqrt(x)

◆Number

◆String

⇨Functions:

◆ substring(start, end) - start is first character index, end is last index (not inc.)

◆ charAt(index) - character at particular location in string (starting at 0)

◆ Others: toUpperCase(st), toLowerCase(st)

Built-in Function Example

```
var str = "hello, world!";
str = str.toUpperCase();
str = str.substring(0,5);
window.alert(str);
```

System realizes your data is a string and converts it to a String object automatically for you.

```
var num = 49;
window.alert(Math.sqrt(num));
```

Use functions in built-in Math object

```
// Create two random numbers between 0 and 1
window.alert("My random number: "+Math.random());
window.alert("My next random number: "+Math.random());
```

Advanced: Calling Object Methods

A method is called on an object by supplying an object instance and the name and arguments to the method.

Syntax:

> *objectInstance.methodName(arguments)*

Remember:

- ◆ Each object has its own methods that it can perform.
- ◆ Each object has its own area of memory storing its data.

Tricky: A String object may be created for us automatically, so we do not always have to create String objects. We use the already created Math object for math functions.

Practice Questions: Functions

1) Write a function that returns the sum of three numbers.

2) Write a function that returns the largest of two numbers.

3) Write a function that writes out the numbers from 1 to N where N is its input parameter.

4) Write a function that determines leap years:

```
function isLeapYear(year)
```

◆ A leap year is a year divisible by 4, except years divisible by 100 and not by 400. (i.e. 1900 is not a leap year, 2000 is.)

For each function, provide a sample function call.

Input Forms

A *form* is an input page that contains controls such as buttons, lists, and boxes that allow the user to input information.

Below is an example form that we will create using HTML:

HTML Forms

A *form* is created in HTML using a **form** tag and **input** tags (one for each input control).

Below is a login form with user id and password boxes:

```
<form name="MyForm" method="post" action="validateLogin.html">
User id: <input type="text" name="username" size="8"/><br/>
Password: <input type="password" name="password" size="8"/><br/>
<input class="submit" type="submit" name="Sub" value="Log In"/>
</form>
```

HTML Input Types

An ***input type*** is a control that allows the user to communicate information with the computer.

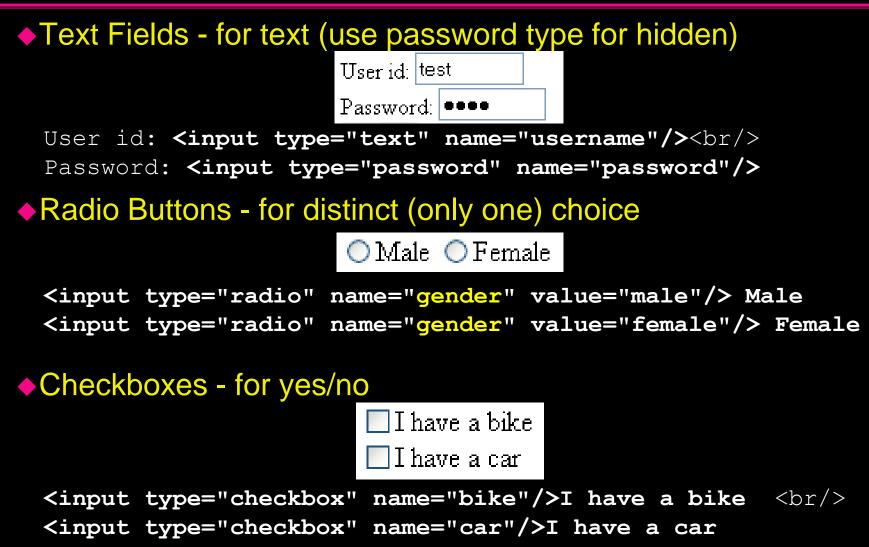An input control is specified using the `input` tag. The other key attributes are a `type` and a `name` for the control.

First name:	
Last name:	

```
<form name="fname" method="post" action="http://www.google.ca">
First name: <input type="text" name="firstname"/> <br/>
Last name:  <input type="text" name="lastname"/>
</form>
```

What type of control is it?

What is its name?
Use the name in code like a variable.

HTML Input Types

First name: [   ] } text fields
Last name: [   ]
○ Male ○ Female ← radio buttons
☐ I have a bike } checkboxes
☐ I have a car
My favorite color is: [blue ▾] ← **drop-down list box**
My hobbies are:
(Put your hobbies in here.) ← text area
[CLICK HERE!] [Submit]

action buttons

HTML Input Types (2)

◆ Text Fields - for text (use password type for hidden)

User id:	test
Password:	●●●●

```
User id: <input type="text" name="username"/><br/>
Password: <input type="password" name="password"/>
```

◆ Radio Buttons - for distinct (only one) choice

◯ Male   ◯ Female

```
<input type="radio" name="gender" value="male"/> Male
<input type="radio" name="gender" value="female"/> Female
```

◆ Checkboxes - for yes/no

☐ I have a bike
☐ I have a car

```
<input type="checkbox" name="bike"/>I have a bike  <br/>
<input type="checkbox" name="car"/>I have a car
```

HTML List

A drop-down list is created in HTML by using the **`select`** tag and an **`option`** tag for each item in the list.

- Each item in the list has a display text label and a value.

```
<select name="colors">
<option value="R">red</option>
<option value="B" selected="selected">blue</option>
<option value="Y">yellow</option>
<option value="G">green</option>
</select>
```

- Notes:
  - ⇨ `selected` attribute indicates the list item is selected.
  - ⇨ Depending on the list, multiple items may be selected at the same time.
  - ⇨ size attribute of `SELECT` tag indicates how many items are visible.

HTML Text Area

A text area is a text field that has a certain size in rows and columns. It is created in HTML by using the **textarea** tag.

```
(Put your hobbies in here.)
```

```
<textarea name="hobbies" rows="3" cols="30">
(Put your hobbies in here.)
</textarea>
```

◆Notes:
  ⇨Use `readonly` attribute to not allow the user to edit the field.

HTML Buttons

A button performs an *action* when clicked.

There are some default buttons with special names:

- `reset` - clears all form fields
- `submit` - sends all data in form fields to web server

how and where to send data
when submit button is clicked

```
<form name="myForm" action="doAction.html" method="get">
<input type="reset" value="Reset"/>
<input type="submit" value="Submit to Server"/>
</form>
```

Reset   Submit to Server

HTML User-Defined Buttons

User buttons can be created using either the **`button`** tag or **`input`** tag with type as button.

```
<input type="button" value="Click Button 1" name="button1"/>
```



```
<button name="button2">My Button 2</button>
```



```
<button name="button2"><img src="redball.gif"></button>
```


We must specify what to do when a user clicks on a button. A user click is one type of *event*.

Label Tag

There is a `label` tag that will associate a label with an input tag and clicking on the label **gives that input focus.**
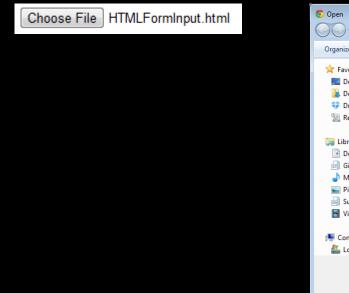
Example:

```
<label for="firstName">First Name:</label>
<input type="text" id="firstName" />
```
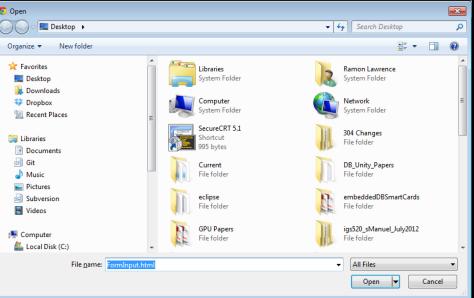
File Input (Browse for file button)

There is a `input` type called **file** that will create a text field and button to allow user to select a file on their computer.

Example:

```
<input type="file" id="fileselect" />
```

HTML Inputs

Question: A form has three radio buttons on it in a group (all have the same name). How many radio buttons can the user select of the three? (Select *one* of the correct answers.)

A) 0

B) 1

C) 2

D) 3

```html
<!DOCTYPE html>
<html>
<head><title>HTML Form Question</title></head>
<body style="background-color:white">
<form name="myForm" action="http://people.ok.ubc.ca" method="get">
Name: <input type="text" name="name"> <br>
Student id: <input type="password" name="studentid"/> <br>

Year:
<input type="radio" name="year" value="1"/>1
<input type="radio" name="year" value="2"/>2
<input type="radio" name="year" value="3"/>3
<input type="radio" name="year" value="4"/>4<br>

My major is:
<select name="major">
<option value="ARTS">Arts</option>
<option value="SCIENCE">Science</option>
<option value="COSC" selected="selected">Computer Science</option>
<option value="ENGL">English</option>
<option value="PSYC">Psychology</option>
</select><br>

My minors are:<br>
<input type="checkbox" name="Arts">Arts<br>
<input type="checkbox" name="English">English<br>
<input type="checkbox" name="Psychology">Psychology<br>
<input type="checkbox" name="Science">Science<br>

Other notes:<br/> <textarea rows="4" cols="30"></textarea><br>

<input type="button" value="Register">
<input type="reset" value="Reset">
<input type="submit" value="Submit">
</form>
</body></html>
```

Given this HTML code:

1) Draw what the page looks like.

2) Indicate what each button does.

GUI Programming Philosophy

In **graphical applications**, the programmer must **react** instead of **dictate** the events that occur in a program.

As a programmer, you design a graphical user interface with windows, buttons, and components that the user can interact with. You do not know the order or the sequence of events the user will generate, but you must be able to react to them.

Events and Event Handling Overview

An *event* is a notification to your program that something has occurred.

- For graphical events (mouse click, data entry), the browser notifies your program that an event occurred.
  - ⇨ There are different *kinds* of events such as keyboard events, mouse click events, mouse movement events, etc.

An *event handler* is part of your program that is responsible for "listening" for event notifications and handling them properly.

An *event source* is the user interface component that generated the event.

- A button, a window, and scrollbars are all event sources.
- The event source is **NOT** the user, the mouse, or the keyboard.

HTML Input Element Events

Form elements in HTML can be made to listen and perform actions when events occur. The type of event that can be detected depends on the input type.

When you declare the form element, you can also provide:

- the *events* that you are interested in

- the *actions* (code functions) that should be run when the event occurs

Example: A button that pops up a window when clicked:

Event source

```
<input type="button" value="Button 1" name="button1"
        onclick="window.alert('You clicked me!');"/>
```

Event

Event handler

HTML Input Element Events (2)

Buttons can be made to handle these events (and others):

- ◆`onclick` — occurs when button is clicked
- ◆`ondblclick` — button is double-clicked
- ◆`onmouseover` — mouse pointer passed over button

Common events for text fields and areas:

- ◆`onkeypress` — key has been pressed on the keyboard

For all form elements:

- ◆`onchange` — value of element has been changed
- ◆`onselect` — element has been selected by user
- ◆`onsubmit` — occurs before form data is submitted

HTML Event Example

Below is an example of a form that detects and alters its appearance using HTML events:

HTML Event Code

```html
<html>
<head><title>HTML Events</title></head>

<body style="background-color:white">
<form name="myForm" action="doAction.html" method="get"
                              onsubmit="formSubmit()">


First name: <input type="text" name="firstname"
            onkeypress="keyPress(event)"> <br>
Last name: <input type="text" name="lastname"
            onkeypress="return noNumbers(event)"> <br>


<input type="radio" name="sex" value="male"
            onclick="changeFontUp()">Male
<input type="radio" name="sex" value="female"
            onclick="changeFontDown()">Female<br>


<input type="checkbox" name="bike" onclick="noCar()">I have a bike<br>
<input type="checkbox" name="car" onclick="checkCar(this)">I have a
car<br>
```

Keypress events

OnClick events

HTML Event Code (2)

```
My favorite color is:
<select name="colors" onchange="changeColor(this)">    ← onChange event
<option value="red">red</option>
<option value="blue" selected="selected">blue</option>
<option value="yellow">yellow</option>
<option value="green">green</option>
</select><br>

My hobbies are:<br>
<textarea rows="5" cols="30" name="ta">
</textarea><br>

<input type="button" value="CLICK HERE!" onclick="count++;
myForm.ta.value=myForm.ta.value+'You clicked '+count+' times!\n';">
<input type="reset" value="Reset"
    onclick="return confirm('Are you sure?');"/>
<input type="submit" value="Submit" name="submitButton"
    onmouseover="overSubmit()" onmouseout="offSubmit()">
</form>
```

Mouse moves over and off events.

HTML Event Code (3)

```
<script type="text/javascript">

var count=0;                       // Global variable

function formSubmit()
{   window.alert("You are trying to submit the form!"); }

function getKey(ev)
{   // Handles differences between IE and other browsers
    var keyVal;
    if (window.event) // IE
        keyVal = ev.keyCode;
    else if (e.which) // Netscape/Firefox/Opera
        keyVal = ev.which;
    return keyVal;
}


function keyPress(ev)
{   window.alert("You pressed: "+String.fromCharCode(getKey(ev)));  }
```

HTML Event Code (4)

```
function noNumbers(ev)
{
    var keyVal = getKey(ev);
    if (keyVal >= 48 && keyVal <= 57) // Key pressed is a number
        return false;
}


function changeFontUp()
{   document.body.style.fontSize="120%"; }


function changeFontDown()
{   document.body.style.fontSize="80%"; }


function changeColor(el)
{   document.body.style.backgroundColor = el.value; }

function checkCar(el)
{   if (el.checked)
        window.alert('Get some exercise!  Ride a bike!');
    else
        window.alert("You didn't need a car anyways!");
}
```

Changes font size of all text in document.

Change background color of document.

HTML Event Code (5)

```
function overSubmit()
{   myForm.ta.value = "Time to submit the form, eh?";}

function offSubmit()
{   myForm.ta.value = "Changed your mind.   Chicken!";}

function noCar()
{   if (myForm.car.checked && myForm.bike.checked)
    {
        myForm.car.checked = false;
        myForm.ta.value = "Sold the car to get the bike.";
    }
}

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

A checkbox has a checked property.

myForm (form) has a text area called ta with a property value

Advanced: HTML Document Objects

Your JavaScript program has access to all parts (called *objects*) of your HTML document.

The entire document is represented by a `document` object.

- ◆ You can change foreground and background colors using it.

- ◆ When you name your document parts (such as forms and inputs), you can later refer to and change the properties of these elements using your JavaScript code.

A *property* is information about an object.

- ◆ Properties include `value`, `name`, and `type`. Other properties depend on the type of object.

- ◆ To change a property value using assignment, provide the name of the object then "." then the property name.

Watch for: The keyword "`this`" refers to the current object and its properties.

Advanced:
How are HTML pages created?

1) An HTML page can be created once (***statically***) and saved on a server. Every request for the page returns it exactly as it was originally created.

2) An HTML page can be produced ***dynamically*** by program code running on the server.

◆ The server-side code can access databases, run functions, or change the appearance or function of the page in response to user input and preferences.

Advanced: When is JavaScript code executed?

JavaScript code is executed:

◆ 1) While the page is being loaded

➪ A browser builds a page by reading through the HTML file, figuring out all tags and preparing to build page.

➪ Then, it removes JavaScript tags and all text between them, and does whatever the JavaScript tells it to do.

➪ Example: We have used `document.write()` to tell the browser to put text into the HTML document.

◆ 2) Interactively after the page is displayed (most common).

➪ Example: HTML elements (such as buttons) may have events associated with them that run JavaScript code.

Aside: New Windows

It is easy to open up a new browser window in JavaScript.

Use the `window.open()` method and provide the file URL.

```
<input type="button" value="Open a Window"
  onclick="window.open('HTMLCalculator.html', 'calcName', 'resizable=yes');">
```

File URL   **Window name**   **resizable?**

Events

Question: What is the event, event source, and event handler in this code?

```
<input type="button" value="Button 1" name="button1"
                     onclick="doButtonClick()">
```

A) event – `button`, event source – `button`, event handler - `onclick`

B) event – `onclick`, event source – `the mouse`, event handler - `onclick`

C) event – `onclick`, event source – `button`, event handler – `doButtonClick()`

D) event – `onclick`, event source – `button1`, event handler - `doButtonClick()`

Event Names

Question: Find the names of the three types of events below. Select the appropriate order of event names.

- ◆happen when the mouse is clicked
- ◆happen when a key is pressed
- ◆happen when the mouse passes over something

A) `onmouseclick, onkeypress, onmouseover`
B) `onclick, onkeyboardpress, onmouseout`
C) `onmouseclick, onkeyboardpress, onmouseover`
D) `onclick, onkeypress, onmouseover`

Events

Question: Philosophical challenge: If an event occurs but there is no code to handle it, did it actually happen?

◆ Example: You click on a button and see nothing change. Did something happen?

A) Yes, the event happened, but it was ignored by the operating system.

B) Yes, the event happened, but it was ignored by our program.

C) No, the event did not happen because our program was not listening for it.

Events and Case-Sensitivity

Question: TRUE or FALSE: Event names are case-sensitive.

A) TRUE
B) FALSE

Conclusion

Functions and *procedures* are used to group statements that perform a particular task so that they can be easily used.

- ◆ Functions must be declared before they can be called (used).

Forms are used to send input and receive output from the computer. A form consists of elements such as buttons, sliders, lists, and boxes.

- ◆ HTML forms use the `form` and `input` tags.

Events are notifications that something occurs. *Event handlers* are code statements that you write indicating to the computer what should be done when an event happens.

Objectives

- Define: function, procedure
- Explain the difference between creating and calling a function.
- Explain the difference between an argument and a parameter.
- Define: form. Be able to draw forms from code.
- List the different types of buttons. Define a button action.
- Define: event, event handler, event source