A SunCam online continuing education course

Web-Based Programming For Engineers - Part 2

by

Kwabena Ofosu, Ph.D., P.E., PTOE

Abstract

The objective of this course series is to present web-based computer programming to engineers. Engineers generally learn a conventional computer programming language such as *FORTRAN*, *Pascal*, *C++*, etc. Since the advent of the internet and the World Wide Web, web browsers such as *Internet Explorer*, *Mozilla Firefox*, *Google Chrome*, etc. have built-in capabilities to interpret and implement programmed instructions written in a class of programming languages called scripting languages. Web-based programming involves writing codes, called scripts, in a scripting language. The scripts are embedded in the structure of web pages. Unlike conventional general purpose programming languages, web-based programming does not require any special software to be installed. The scripts are interpreted and implemented directly by the web browser. Web-based programming is an increasingly relevant and advantageous tool for engineers competing in the global marketplace in the age of the internet and the World Wide Web. Once uploaded to the World Wide Web, web-based applications are immediately exposed to a global audience.

This course is the second of a series on web-based programming. This class presents Hypertext Markup Language (HTML) forms and the *JavaScript* scripting language. This course uses screenshots and an easily readable click-by-click narrative that engages participants as they proceed through the topics. This course starts with an overview of the HTML language used to create web pages, followed by HTML forms and controls. The fundamentals of the *JavaScript* language are then presented in detail. Several examples and illustrations from many fields of specialty are presented to illustrate the creation and editing of interactive web pages using HTML codes and *JavaScript* scripts. At the end of this course, participants will be capable of building a web page from scratch, editing and updating existing web pages using HTML and *JavaScript* alone. Participants will be able to create fully functional interactive web pages and web applications that can be used to input and output data, as well as run complex algorithms. On completion of this course participants will be able to identify professional situations in which applying web-based programming will be of great benefit to them in their fields of specialty, and to their organizations.

TABLE OF CONTENTS

Abstract ii
List of Tables v
1. INTRODUCTION 1
1.2 The Internet 1
1.3 The World Wide Web 1
1.4 Web Pages 1
1.5 Scripting Languages 2
1.6 JavaScript 3
2. FORMS AND CONTROLS 4
2.1 HTML Documents 4
2.2 Objects and Events 10
2.3 Forms 10
2.4 Text Box 12
2.5 Password Text Box 14
2.6 Text Area 16
2.7 Combo Box 18
2.8 List Box 20
2.9 Button 21
2.10 Submit Button 22
2.11 Reset Button 23
2.12 Check Box 24
2.13 Radio Button 25
2.14 Practicum #1 27
3. *JAVASCRIPT* SOURCE CODE 44
3.1 *JavaScript* and HTML Tags 44
3.2 Commenting 48
4. VARIABLES 52
4.1 Definition 52
4.2 Variable Names 53

4.3 Data Types 54
4.3.1 Number Data Type 54
4.3.2 String Data Type 55
4.3.3 Boolean Data Type 55
4.3.4 Null Data Type 56
4.3.5 Undefined Data Type 56
4.3.6 Arrays 56
4.3.7 Objects 58
4.4 Operators 59
4.5 Order of Precedence 62
5. FUNCTIONS 63
5.1 Definition 63
5.2 Function Declaration 63
5.3 Function Names 64
5.4 Returning a Value 65
5.5 Scope of Variables 65
5.6 Built-in Functions 66
5.7 Practicum #2 68
6. DEPLOYING WEB-BASED APPLICATIONS 80
6.1 Web Hosting 80
6.2 Debugging HTML Files 80
6.3 Web Browser Compatibility 81
6.4 Getting Help 81
7. CONCLUSION 82
REFERENCES 83

List of Tables

Table 1: Arithmetic operators 60

Table 2: Assignment operators 61

Table 3: *JavaScript* built-in functions.. 66

1. INTRODUCTION

1.2 The Internet

The **internet** is a group of interconnected computers, computer networks, and groups of networks on a global scale. In order for a computer on the internet to communicate with other participating computers it must have special software that can interpret what other computers are "saying". This special software is called a **web browser** (or browser). Some of the common web browsers in use today include ***Internet Explorer***, ***Mozilla Firefox***, ***Google Chrome***, ***Opera***, and ***Safari***. All web browsers are free and come with the computer's operating system. For example, *Microsoft Windows* comes with the *Internet Explorer* browser.

1.3 The World Wide Web

The terms "internet" and "World Wide Web" have become synonymous and interchangeable in common usage. However, there is a clear difference. The internet is the physical network of all the networks of all the participating interconnected computers. The World Wide Web (or **the web**) is a network of virtual documents that reside on computers on this network (the internet). With a web browser these documents, called **web pages**, can be accessed, opened, and viewed from any participating computer. The web pages are accessed by the web browser using hyperlinks. The browser interprets the instructions (**source code** or **code**) in the document and displays the web page. Web pages are generally stored on server computers (servers) on the internet. Therefore, the web is a constituent part of the internet. A collection of related web pages is referred to as a **website**.

1.4 Web Pages

A web browser interprets the code in the web page document and then displays the contents to the user accordingly. However, web browsers are not designed to understand ordinary everyday language. **Hyper Text Markup Language** (**HTML**) is the language of the web. All web pages are HTML documents. The HTML code tells the web browser how to display the web page's words, numbers, images, etc., to the user.

HTML is free and there is no special or required software needed to develop HTML documents. It is pertinent to note that there are many commercial software products on the market for creating or editing web pages and websites. These products generally provide a visually pleasing interactive environment with enhanced graphics and other features which the user can apply to design web pages without any knowledge of HTML. These products cover a wide variety of capabilities, features, and prices. All of these products ultimately generate an HTML file(s) that is then uploaded to the web. It is important to note, however, that an HTML file can be created or edited using any text editor program. As with web browsers, text editors are free and come with the operating system of the computer. For example, *Microsoft Windows* comes with the text editors *WordPad* and *Notepad*.

The standards for the HTML language are provided by the **World Wide Web Consortium** (W3C). HTML is into its seventh version as of 2012, called *HTML5*. (World Wide Web Consortium, 2014)

1.5 Scripting Languages

HTML is a **markup language**. A markup language is a combination of keywords and symbols that provide instructions on how to display the contents of a document to the user. As a result, web pages written solely with markup language will only display information but will not have the capability to interact with the user. A markup language has no variables and is not used to perform calculations or run algorithms, etc., as one would with a programming language like *C++* or *Visual Basic*, etc.

To make web pages dynamic and interactive, code written in a **scripting language** has to be embedded in the HTML. A scripting language is technically a "lightweight" programming language that is designed to integrate and communicate with other programs such as web browsers. Scripting languages are interpreted directly and executed line-by-line by the web browser. In contrast, a programming language such as *C#* or *C++* is compiled, i.e., it is converted by special software called the **compiler**, into machine language which the computer can understand and then execute. Scripting languages are generally designed to be easier to learn and use than conventional programming languages and therefore facilitate rapid development of applications. Scripting languages generally have a looser structure and syntax rules than conventional programming languages. However, conventional programming languages are generally much faster than their scripting language counterparts.

There are many scripting languages in use today. Examples of currently popular scripting languages used to enhance and provide functionality to web pages include:

- *JavaScript*

- *VBScript*
- *Python*
- *Ruby*
- *Perl*
- *Tcl*
- *PHP*
- *Lua*
- *AppleScript*
- *C Shell*
- *Windows Powershell*

1.6 JavaScript

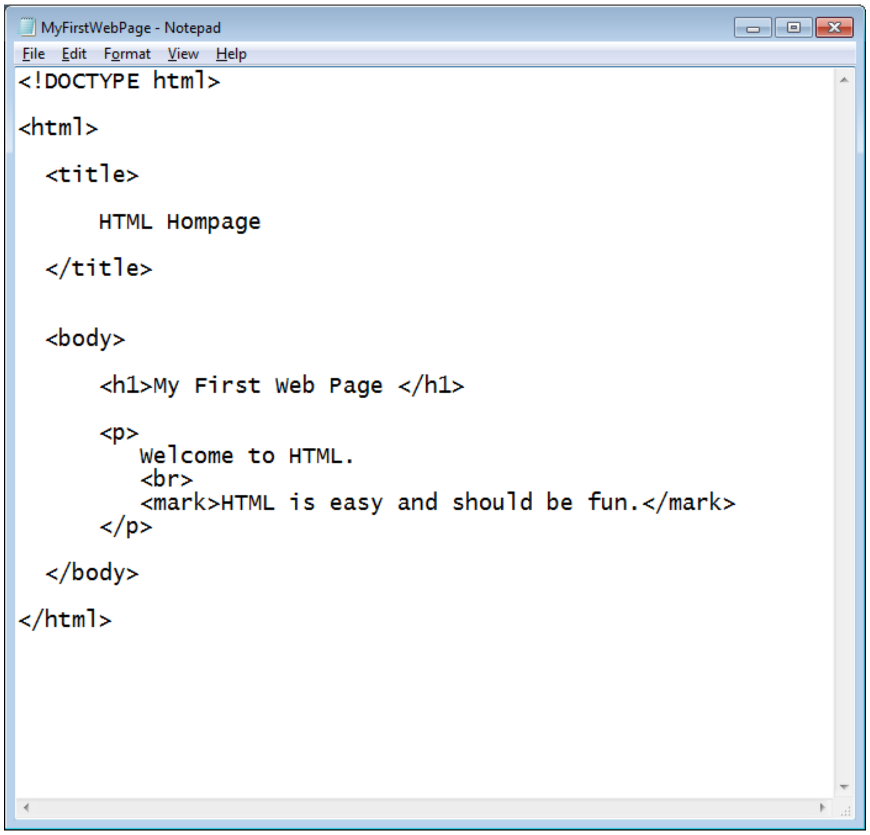
JavaScript is a scripting language originally developed by *Netscape*. It is commensurate in capability with the *Visual Basic* language. Although it shares part of its name as well as some fundamental ideas with *Java*, the two languages are otherwise not related. The syntax of *JavaScript* was developed from the *C* language. The standards for *JavaScript* are documented in *Standard ECMA – 262* which is maintained by ECMA International (ECMA International, 2011). *JavaScript* can be executed by all types of web browsers for PCs, laptops, cellphones, tablets, smartphones and more. In effect, *JavaScript* is the most widely used scripting language and hence the world's most widely used programming language. The most common use of *JavaScript* is to write functions that are embedded in HTML code to give interactive and dynamic capabilities to web pages.

2. FORMS AND CONTROLS

2.1 HTML Documents

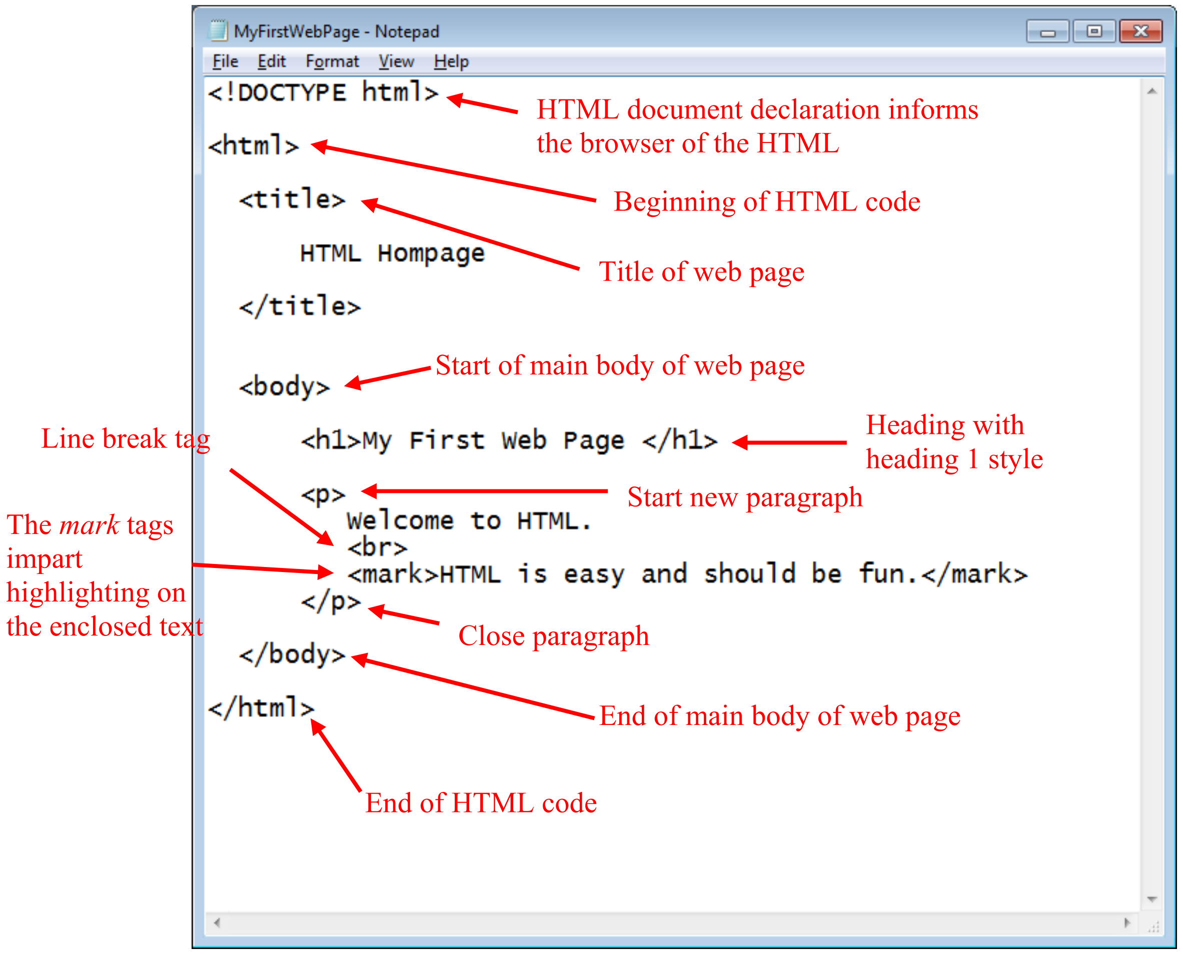
A web page is a text file saved with a "**.html**" or "**.htm**" suffix. The text file (html document) is created by typing the relevant code into a text editor such as *Notepad* or *Wordpad*. In this section a simple web page (HTML document) will be created to illustrate the basic structures of HTML.

Create a directory named *JavaScript_files* on your Desktop.
Upload your file ***MyFirstWebpage*** (from *SunCam*) to your *JavaScript_files* folder.
Right click on the file ***MyFirstWebpage***
Select *Open with*
Select *Notepad*.



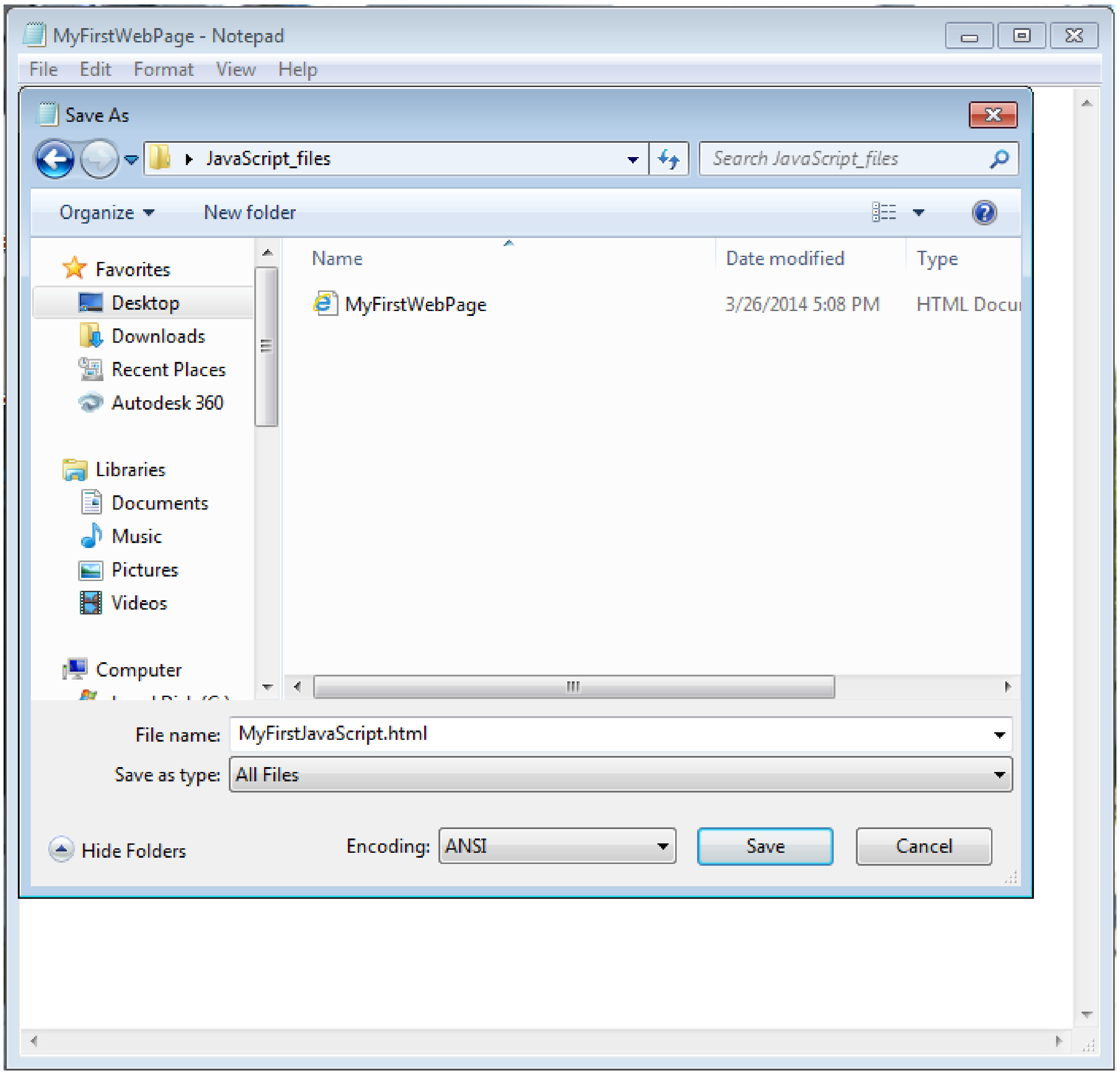
```
<!DOCTYPE html>

<html>

  <title>

      HTML Hompage

  </title>


  <body>

      <h1>My First Web Page </h1>

      <p>
         Welcome to HTML.
         <br>
         <mark>HTML is easy and should be fun.</mark>
      </p>

  </body>

</html>
```

Review the **HTML tags**


MyFirstWebPage - Notepad
File Edit Format View Help
HTML document declaration informs the browser of the HTML
Beginning of HTML code
Title of web page
Start of main body of web page
Heading with heading 1 style
Start new paragraph
Close paragraph
End of main body of web page
End of HTML code
Line break tag
The *mark* tags impart highlighting on the enclosed text

```
<!DOCTYPE html>

<html>

  <title>

      HTML Hompage

  </title>


  <body>

      <h1>My First Web Page </h1>

      <p>
         Welcome to HTML.
         <br>
         <mark>HTML is easy and should be fun.</mark>
      </p>
  </body>

</html>
```

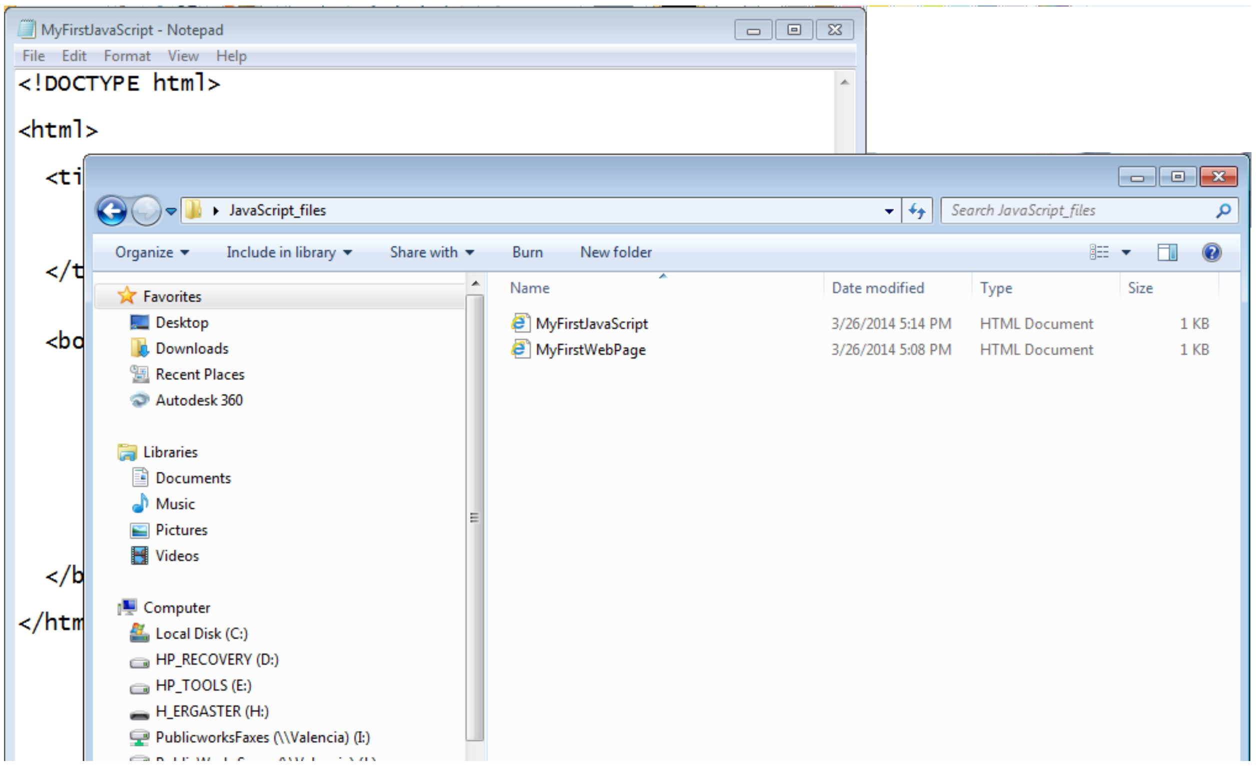
Select *File*
Select *Save As*
In *File name*, type ***MyFirstJavaScript.html***. (Note the "**.html**" suffix)
In the *Save as type*, select *All Files*.



Name	Date modified	Type
MyFirstWebPage	3/26/2014 5:08 PM	HTML Docu

File name: MyFirstJavaScript.html
Save as type: All Files
Encoding: ANSI

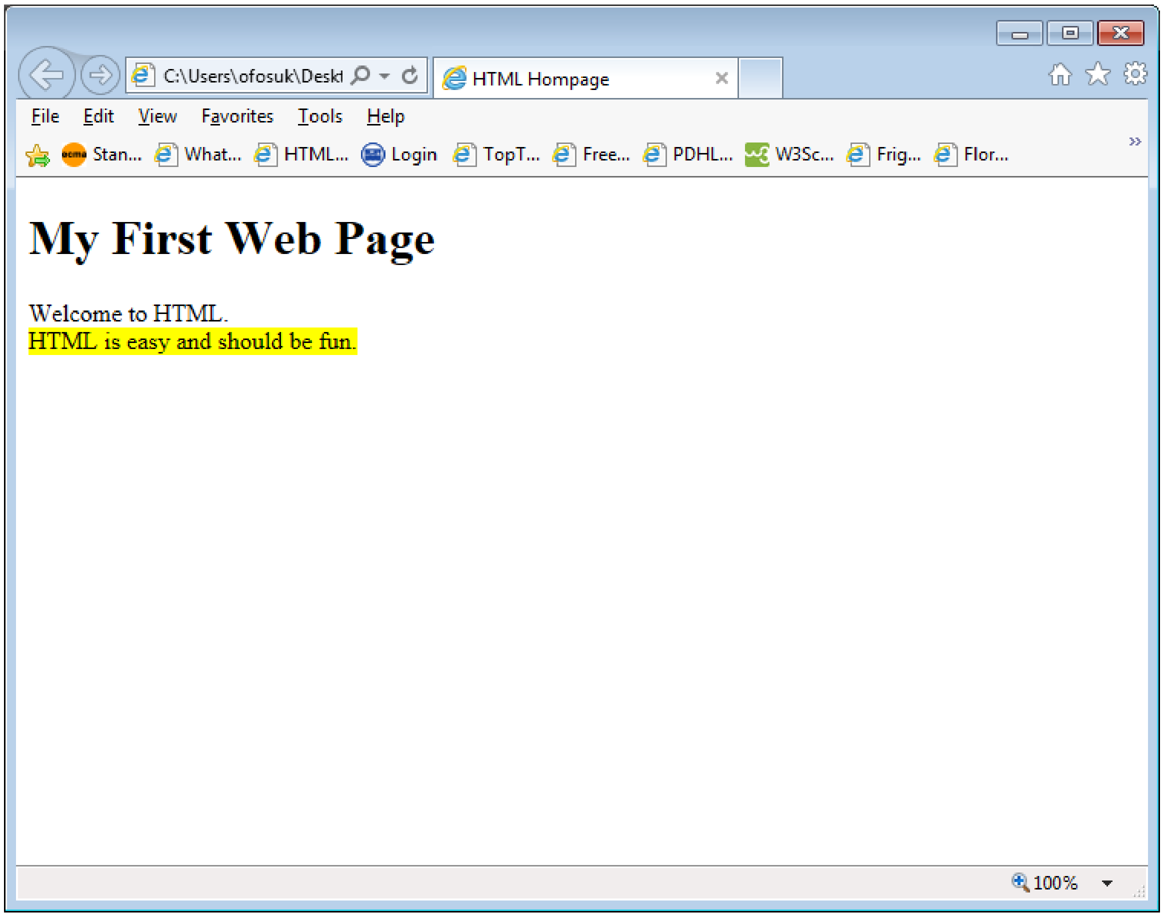
Click on *Save*.
Open your folder *JavaScript_files* to confirm your file has saved as intended.



```
<!DOCTYPE html>

<html>

  <ti

  </t

  <bo



  </b

</htm
```

Name	Date modified	Type	Size
MyFirstJavaScript	3/26/2014 5:14 PM	HTML Document	1 KB
MyFirstWebPage	3/26/2014 5:08 PM	HTML Document	1 KB

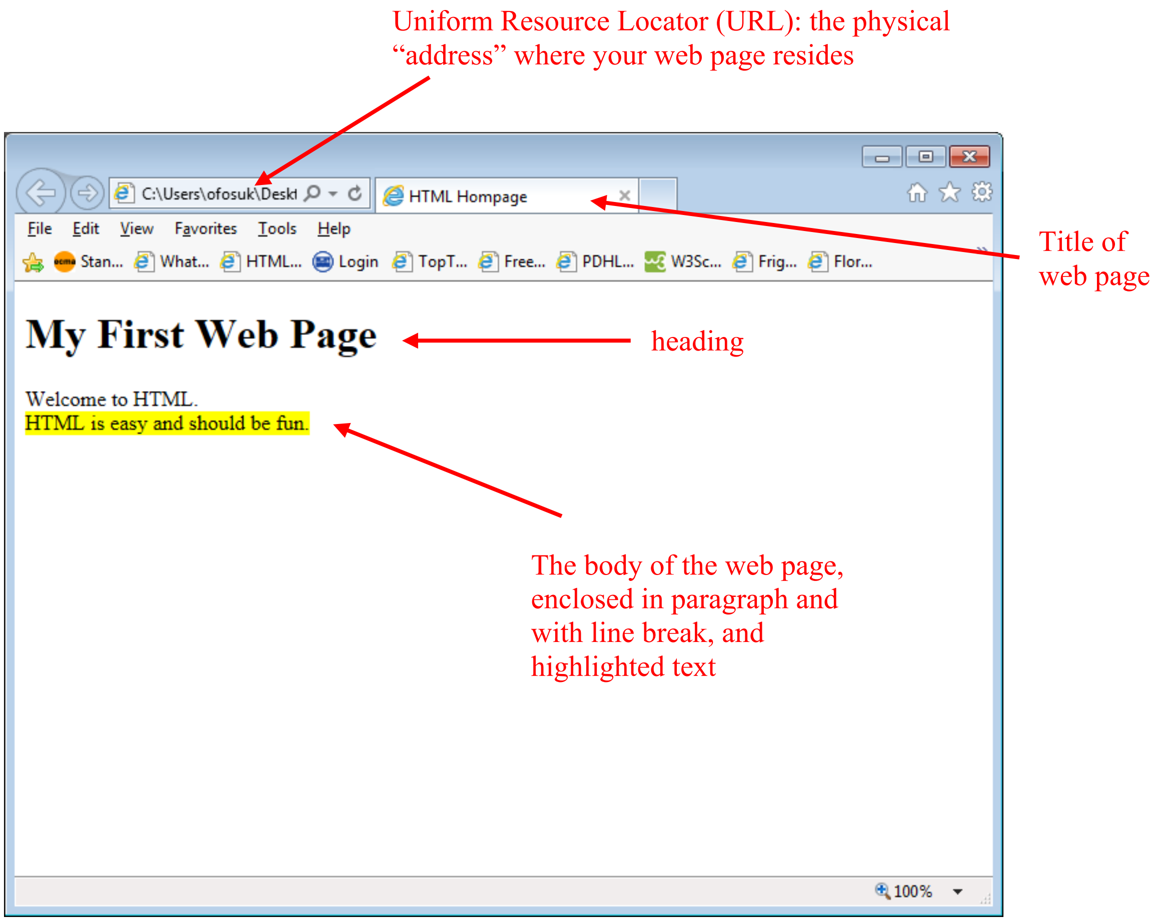
Double click on ***MyFirstJavaScript*** from the *JavaScript_files* folder to open this web page in your default web browser.


C:\Users\ofosuk\Deskt
HTML Hompage
File Edit View Favorites Tools Help
Stan... What... HTML... Login TopT... Free... PDHL... W3Sc... Frig... Flor...
100%

My First Web Page

Welcome to HTML.
HTML is easy and should be fun.

Review the web page elements and compare with your HTML code.


Uniform Resource Locator (URL): the physical "address" where your web page resides
C:\Users\ofosuk\Deskt
HTML Hompage
File Edit View Favorites Tools Help
Stan... What... HTML... Login TopT... Free... PDHL... W3Sc... Frig... Flor...
Title of web page
heading
The body of the web page, enclosed in paragraph and with line break, and highlighted text
100%

My First Web Page

Welcome to HTML.
HTML is easy and should be fun.

A list of HTML tags and references to comprehensive reference materials on HTML tags can be found in Part 1 of this course, on *SunCam*.

2.2 Objects and Events

The HTML document (web page) created in the previous section shall be capable of displaying to the user any relevant information typed in the main body and title. However, the web page does not have the capability to interact with the user.

In order for the web page to be interactive, **objects** called **form controls** (or **controls**) must be added to the web page in the HTML code. There are many types of controls. Examples of controls include click buttons, text boxes, drop down buttons (combo box), radio buttons, check boxes, and others.

Actions associated with manipulating a control are called **events**. For example, the event of *click* on a button, or the event of *change* a selection in a drop down field, etc. The developer must pre-plan and know what controls and events are relevant to achieve the functionality required by their project.

A control is created by invoking the relevant HTML tags for it. Controls have attributes that can be manipulated by assigning relevant values to them. Attributes are assigned values in the start tag of the control. Examples of attributes include color, size, name, font, alignment, background color, etc., etc.

2.3 Forms

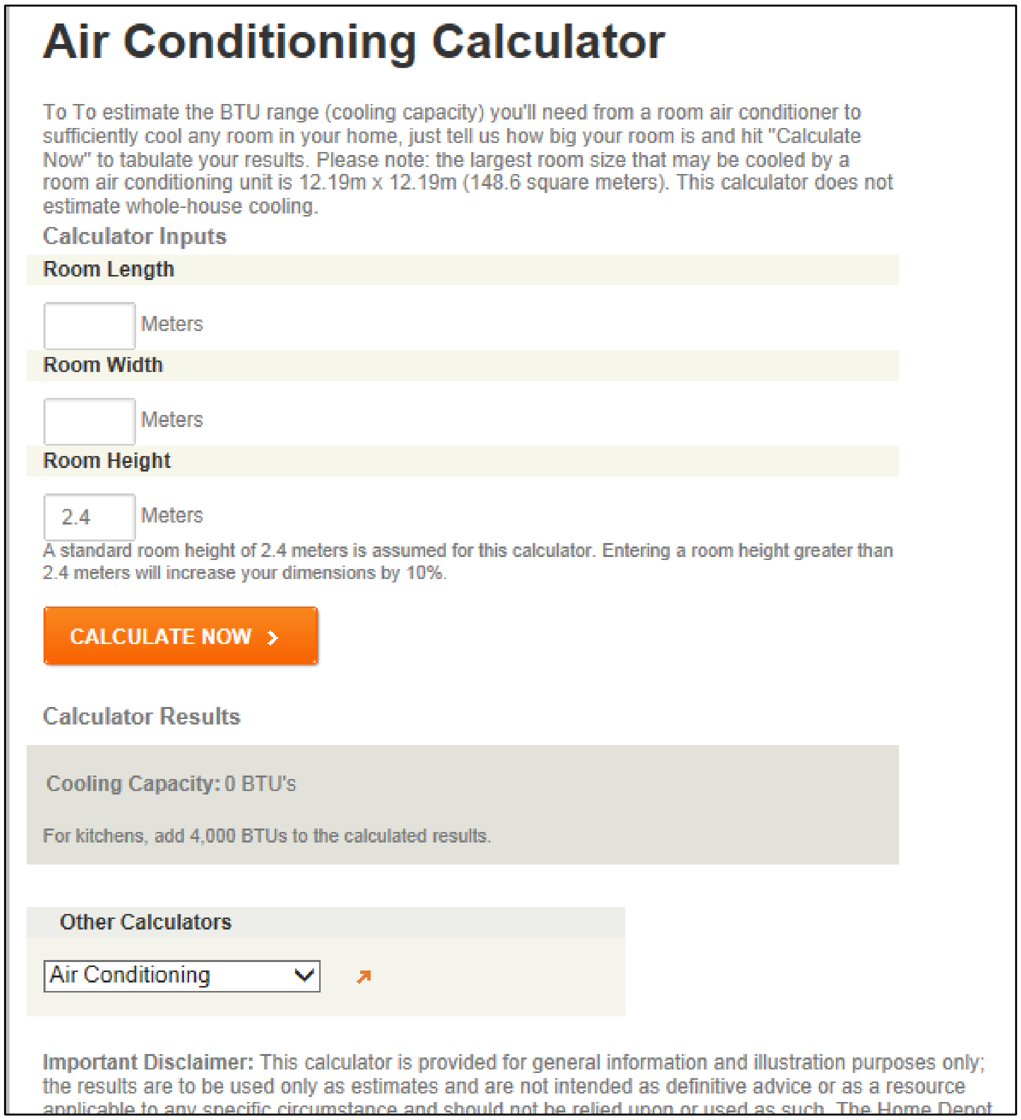
A form is a control that is used to gather together and manage other controls. The HTML tags to create a form on a web page are:

```
<form>
                The body of the form which will consists of other controls e.g. textboxes, push
              buttons, check boxes, radio buttons, etc., etc.…………………….
</form>
```

Note that the HTML tags are not case sensitive.

The framework for developing web-based applications is to add controls to a form(s), and then to write instructions that will be executed when some event occurs on/to a control. These instructions are written in the scripting language, in our case *JavaScript*. The *JavaScript* codes shall be embedded in the HTML code.

An example of a form with various controls is shown below.



Air Conditioning Calculator

To To estimate the BTU range (cooling capacity) you'll need from a room air conditioner to sufficiently cool any room in your home, just tell us how big your room is and hit "Calculate Now" to tabulate your results. Please note: the largest room size that may be cooled by a room air conditioning unit is 12.19m x 12.19m (148.6 square meters). This calculator does not estimate whole-house cooling.

Calculator Inputs

Room Length: Meters

Room Width: Meters

Room Height: 2.4 Meters

A standard room height of 2.4 meters is assumed for this calculator. Entering a room height greater than 2.4 meters will increase your dimensions by 10%.

Calculator Results

Cooling Capacity: 0 BTU's

For kitchens, add 4,000 BTUs to the calculated results.

Other Calculators: Air Conditioning

Important Disclaimer: This calculator is provided for general information and illustration purposes only; the results are to be used only as estimates and are not intended as definitive advice or as a resource applicable to any specific circumstance and should not be relied upon or used as such. The Home Depot

[Source: www.homedepot.com]

.

2.4 Text Box

A text box (or text field) is a rectangular control used to display data or to enter data. The tags to add a text box to an existing form are as follows

```
<form>
                    :
          <input type = “text”>
                    :
</form>
```

Note that an end tag is not used. Text boxes normally contain data which may be called by other controls on the form, or used for some computation. It is therefore recommended that each text box be given a name to establish a unique reference to that text box on the form. The name is set up by assigning a string value to the *name* attribute. It may also be relevant to uniquely identify the control (text box) to the web page. This can be done with the *id* attribute. For example,

```
<form>
                                :
          <input type = “text” name = “MonthlySalary” id = “SalaryAmount”>
                                :
</form>
```

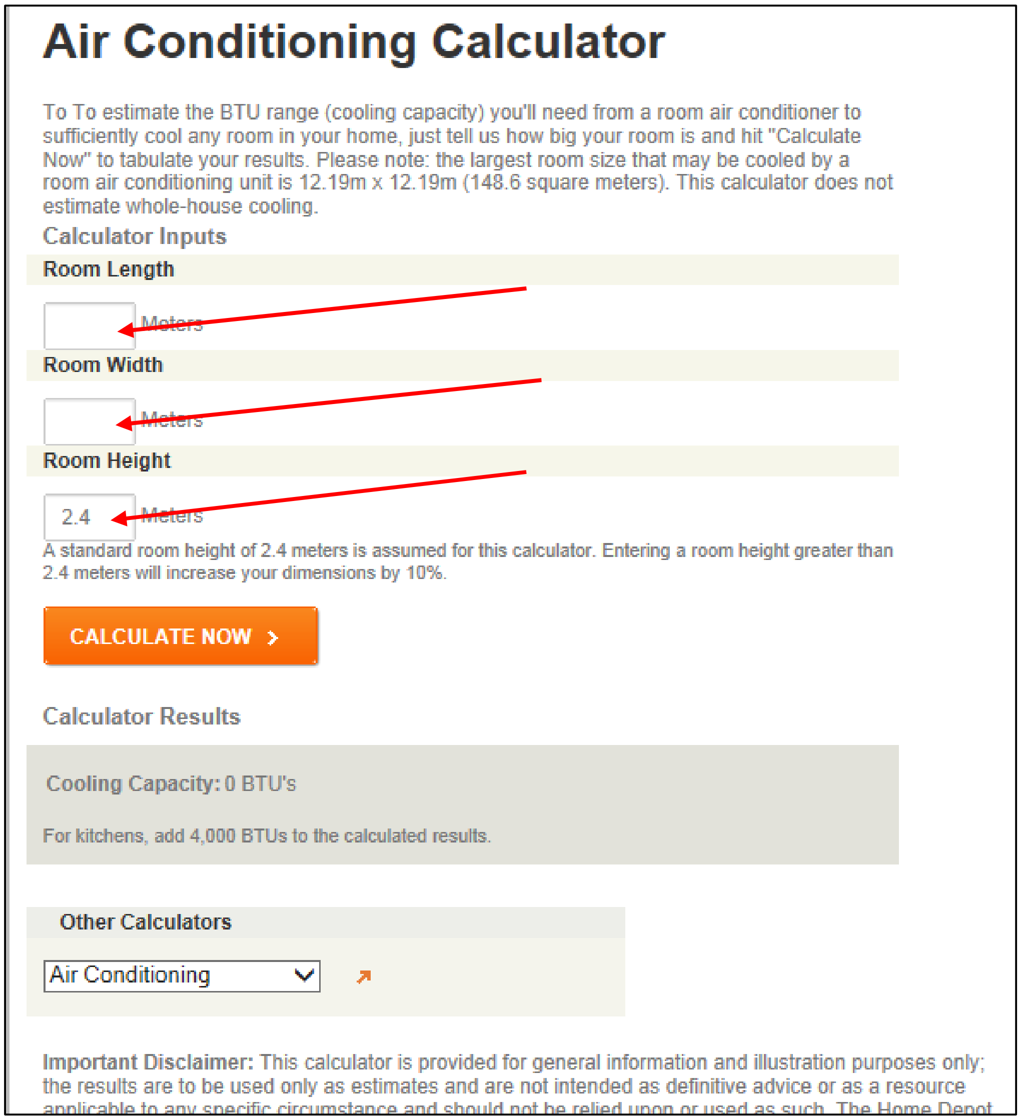
In this example, the *name* attribute uniquely identifies the text box to the form, whereas the *id* attribute uniquely identifies the textbox to the web page.

Examples of events associated with textboxes include:

onFocus: This is the event of the text box being active, or selected.
onChange: This is the event of the contents of the text box being changed

Instructions can be written in *JavaScript* for an event. The instructions will "**fire**" once the event happens to the text box.

An example of a form with text boxes is as follows



Air Conditioning Calculator

To To estimate the BTU range (cooling capacity) you'll need from a room air conditioner to sufficiently cool any room in your home, just tell us how big your room is and hit "Calculate Now" to tabulate your results. Please note: the largest room size that may be cooled by a room air conditioning unit is 12.19m x 12.19m (148.6 square meters). This calculator does not estimate whole-house cooling.

Calculator Inputs

Room Length
Meters

Room Width
Meters

Room Height
2.4 Meters

A standard room height of 2.4 meters is assumed for this calculator. Entering a room height greater than 2.4 meters will increase your dimensions by 10%.

Calculator Results

Cooling Capacity: 0 BTU's

For kitchens, add 4,000 BTUs to the calculated results.

Other Calculators

Air Conditioning

Important Disclaimer: This calculator is provided for general information and illustration purposes only; the results are to be used only as estimates and are not intended as definitive advice or as a resource applicable to any specific circumstance and should not be relied upon or used as such. The Home Depot

[Source: http://www.homedepot.com/]

2.5 Password Text Box

A password text box is a special text box that does not display the input typed in by the user but rather a series of periods or asterisks that mask the user input.

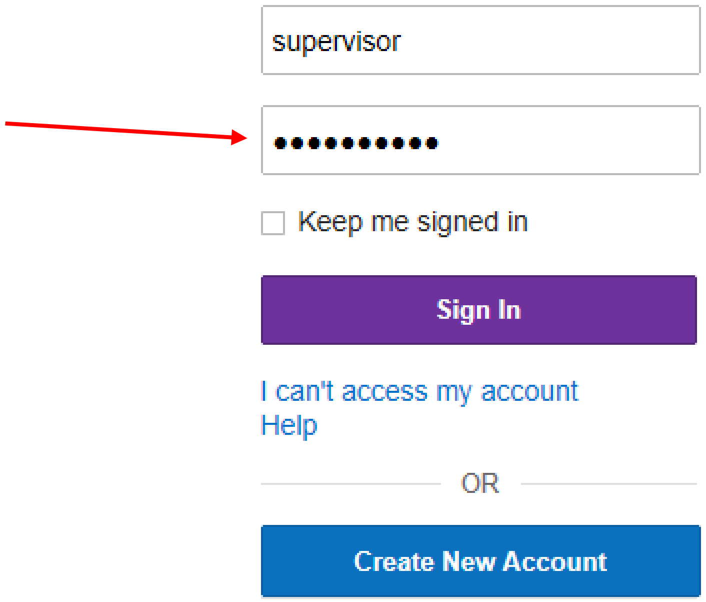
The tag used to create a password text box on a form is as follows

```
<form>
                              :
            <input  type = “password”>
                              :
</form>
```

The width of the control can be changed by manipulating the *size* attribute. The *size* attribute is measured by the number of characters. The value of the password will generally be called or transferred to another file or script, therefore, the control must be given a unique identifying name. The name of the control is established by assigning a string value to the *name* attribute, which uniquely identifies the control on the form. Alternately, or in addition, an *id* attribute may be assigned to uniquely identify the control to the web page. For example:

```
<form>
                              :
      <input  type = “password” name = “supervisorPassword” id = “pw1” size  = “8”>
                              :
</form>
```

An example of a password box is shown below



[Source: https://www.yahoo.com/]

2.6 Text Area

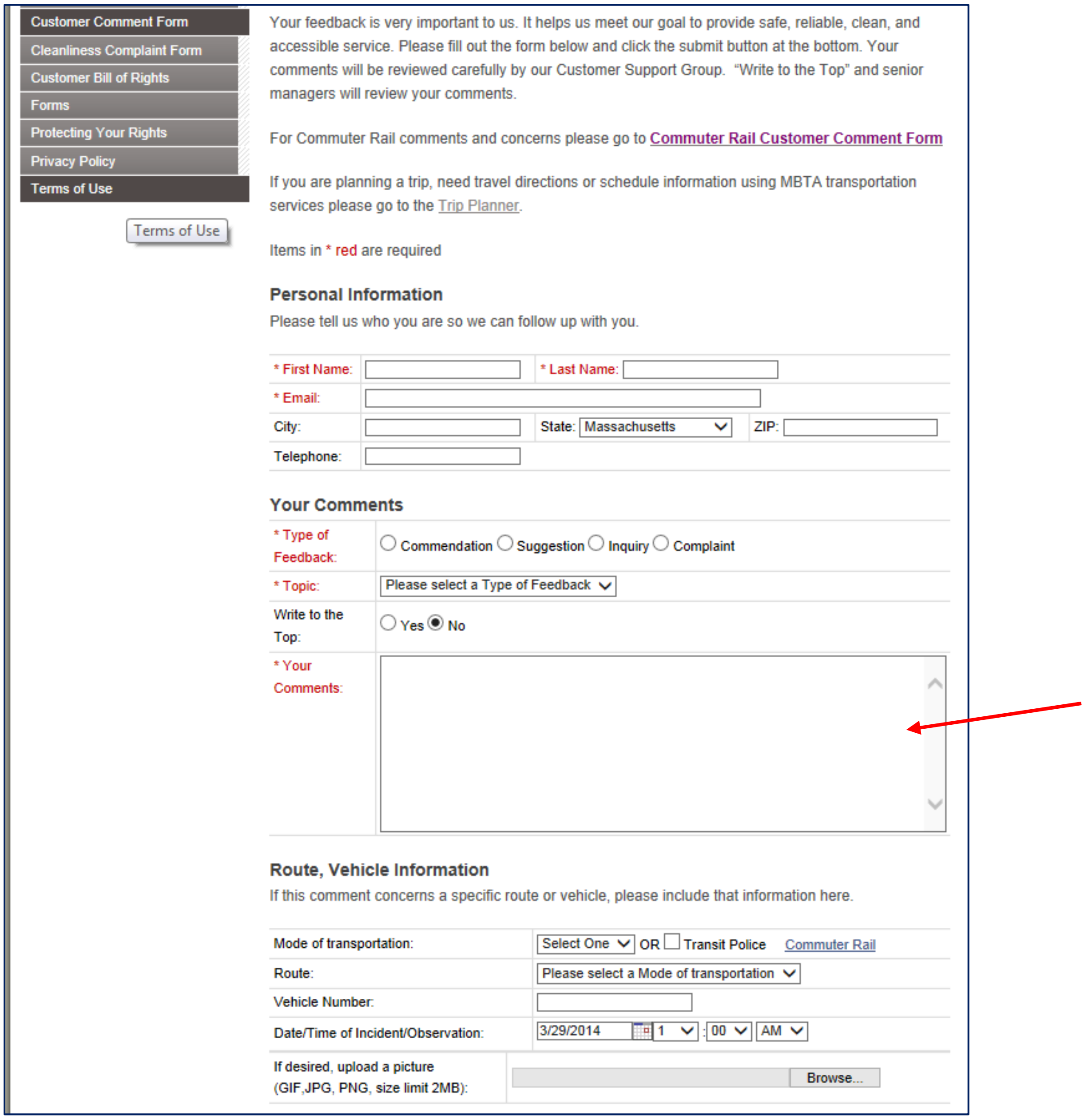
A text area can hold significantly more text than a text box. Text areas are used to enter and hold paragraphs of text. The tags to create a text area on a form are:

```
<form>
                              :
            <input type = "textarea" > Type your text here </textarea>
                              :
</form>
```

The dimensions of the text area may be changed by assigning the appropriate value to the *cols* (width) and *rows* (height) attributes. The height of the text area is measured by the number of visible lines of text, whereas the width is measured by the number of characters per line. If the contents of the text area will be transferred to or called by another file or script, the text area should be given a unique identifying name by assigning one to the *name* attribute which uniquely identifies the control to the form. Alternately, or in addition, the *id* attribute may be used to uniquely identify the control to the web page. All attribute values are assigned in the start tag. For example:

```
            <input type = "textarea" name = "txtaTextArea"  cols= "40" rows = "25">
                       Type your text here
            </textarea>
```

An example of a web form with a text area is as follows:



- Customer Comment Form
- Cleanliness Complaint Form
- Customer Bill of Rights
- Forms
- Protecting Your Rights
- Privacy Policy
- Terms of Use

Terms of Use

Your feedback is very important to us. It helps us meet our goal to provide safe, reliable, clean, and accessible service. Please fill out the form below and click the submit button at the bottom. Your comments will be reviewed carefully by our Customer Support Group. "Write to the Top" and senior managers will review your comments.

For Commuter Rail comments and concerns please go to Commuter Rail Customer Comment Form

If you are planning a trip, need travel directions or schedule information using MBTA transportation services please go to the Trip Planner.

Items in * red are required

Personal Information

Please tell us who you are so we can follow up with you.

* First Name:		* Last Name:			
* Email:					
City:		State:	Massachusetts	ZIP:	
Telephone:					

Your Comments

* Type of Feedback:	○ Commendation ○ Suggestion ○ Inquiry ○ Complaint
* Topic:	Please select a Type of Feedback
Write to the Top:	Yes [✓ No]
* Your Comments:	

Route, Vehicle Information

If this comment concerns a specific route or vehicle, please include that information here.

Mode of transportation:	Select One OR ☐ Transit Police Commuter Rail
Route:	Please select a Mode of transportation
Vehicle Number:	
Date/Time of Incident/Observation:	3/29/2014 1 : 00 AM
If desired, upload a picture (GIF,JPG, PNG, size limit 2MB):	Browse...

[Source: http://www.mbta.com/customer_support/feedback/]

2.7 Combo Box

The combo box enables a user to select an item from a list. The list appears when the “drop down” arrow to the right of the control is clicked on, and retracts once an item has been selected.

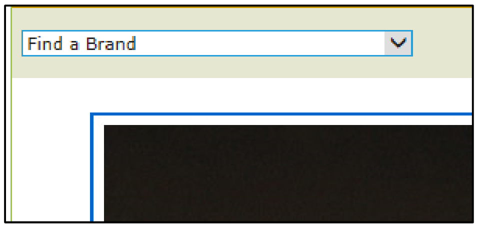
The tags to create a combo box on a form are as follows

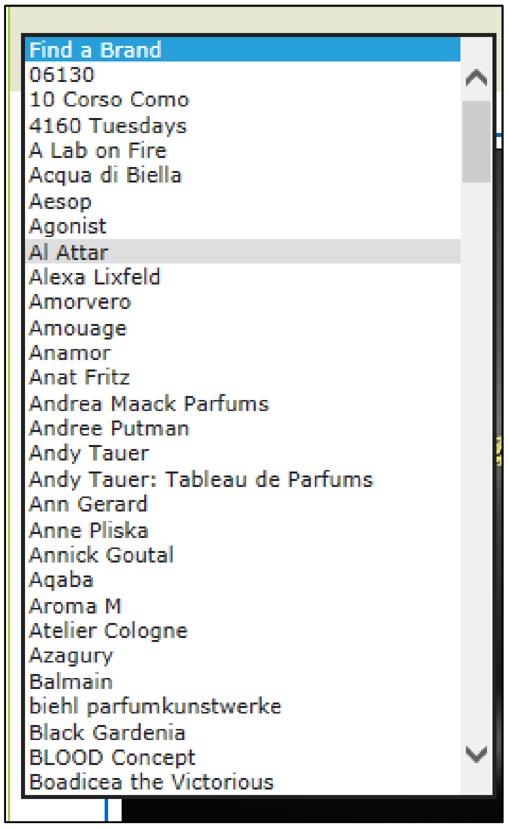
```
<form>
                    :
        <select >
                <option value = “Item1”>Item1</option>
                <option value = “Item2”>Item2</option>
                <option value = “Item3”>Item3</option>
                <option value = “Item4”>Item4</option>
                    :
        </select>

                    :
</form>
```

Attributes commonly applied to combo boxes include the *name*, the *id*, etc. When the form opens the first list item appears in the combo box by default. To change it to another item, the *selected* attribute is set to “true” in the *option* tag of that list item. For example

```
        <select >
                <option value = “florida”>Florida</option>
                <option value = “georgia” selected = “true”>Georgia</option>
                <option value = “alabama”>Alabama</option>
                <option value = “south carolina”>South Carolina</option>
                <option value = “north carolina”>North Carolina </option>
        </select>
```

The most commonly used event for combo boxes is *onChange*. In other words, once a new item is selected from the list, the code written for the *onChange* event fires. An example of a combo box is shown below.





Find a Brand
06130
10 Corso Como
4160 Tuesdays
A Lab on Fire
Acqua di Biella
Aesop
Agonist
Al Attar
Alexa Lixfeld
Amorvero
Amouage
Anamor
Anat Fritz
Andrea Maack Parfums
Andree Putman
Andy Tauer
Andy Tauer: Tableau de Parfums
Ann Gerard
Anne Pliska
Annick Goutal
Aqaba
Aroma M
Atelier Cologne
Azagury
Balmain
biehl parfumkunstwerke
Black Gardenia
BLOOD Concept
Boadicea the Victorious

[Source: http://www.luckyscent.com/]

2.8 List Box

Similar to a combo box, a list box enables the user to select an item from a list. However, the list box is taller than the combo box and displays its entire list at all times.

A list box is created by adding the size attribute to the select tag (as in a combo box). The size attribute sets the number of items that will be displayed at all times. The syntax to add a list box to a form is of the structure:

```
<select size = “5”>
        <option>Jacksonville</option>
        <option>Miami</option>
        <option>Orlando</option>
        <option>Tallahassee</option>
        <option>Tampa</option>
</select>
```

Attributes commonly applied to list boxes include the *name*, the *id*, etc. A default selected item is set by adding the *selected* attribute to the relevant option tag. A list box can be configured to allow more than one selected item by the user, using the multiple attribute. For example:

```
<select size = “5” multiple>
        <option value = “jacksonville>Jacksonville</option>
        <option value = “miami”>Miami</option>
        <option value = “orlando” selected = “true”>Orlando</option>
        <option value = “tallahassee”>Tallahassee</option>
        <option value = “tampa”>Tampa</option>
</select>
```

The most commonly used event for list boxes is *onChange*.

2.9 Button

A button (push button, or click button) is a control that is clicked on or pressed. Based on what instructions have been written for the click event, various things may happen, e.g. a calculation may commence, an algorithm may be executed, data from the form may be copied to another form, etc., etc.

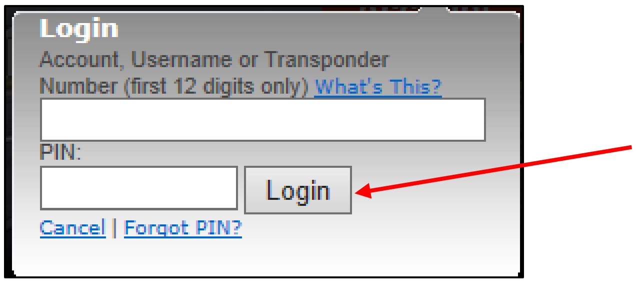
The tags to create a button on a form are as follows:

```
<form>
                    :
        <button type = "button">caption</button>
                    :
</form>
```

Attributes commonly applied to buttons include the *name*, the *id*, etc. The tag content holds the caption (the text "banner") that appears on the button. For example,

```
        <button type = "button" name = "calcSalary">RUN</button>
```

The *onclick* event is the most common event used for buttons. The code associated with this event will fire when the button is clicked on. An example of a button is shown below:


Login
Account, Username or Transponder Number (first 12 digits only) What's This?
PIN:
Login
Cancel | Forgot PIN?

[Source: https://www.sunpass.com/index]

2.10 Submit Button

The submit button is a special button that sends the data on the form to another file or script. The submit button has a built-in default value of "Submit" for the *value* attribute. In other words, the submit button has a default caption of "Submit". The default value may be changed by the programmer. The tag for a submit button is of the structure:

```
        <button type = "submit" name = "nameofButton">Your Caption</button>
```

The script of file to which the form contents will be copied is assigned to the *action* attribute in the form start tag. For example:

```
<form action = "mailto: Ofosuk@yahoo.com">
                        :
                        :
        <button type = "submit">Send Email</button>
                        :
                        :
</form>
```

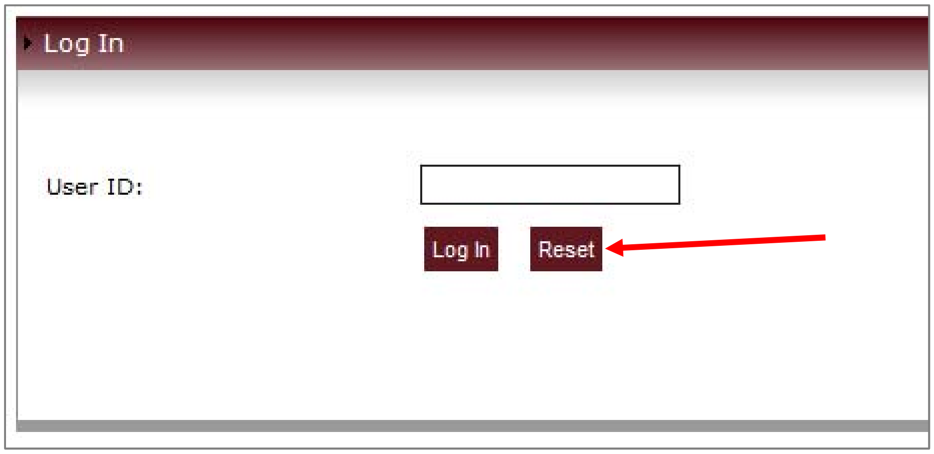
The most common event used for submit buttons is *onclick*.

2.11 Reset Button

The reset button is a special button which when clicked on clears all entries in all controls on the form. It is commonly used whereby a user wishes to clear the form and start all over. Like the submit button, the reset button comes with a default caption that may be changed by re-assigning the *value* attribute. The tag for a reset button is of the structure:

<button type = "reset" name = "nameofButton" >A Caption</button>

The most common event used for the reset button is *onclick*. An example of reset button is shown below.



[Source: https://fsucuebranch.org/ISuite5/Features/Auth/MFA/Default.aspx]

2.12 Check Box

A check box is a small box that enables a user to select or de-select an item. When the user clicks on the check box to select it, a check mark (or tick mark) appears in the box. A checked check box is unchecked by clicking on it to de-select it, at which time the box appears empty.

The tag to create a check box on a form is:

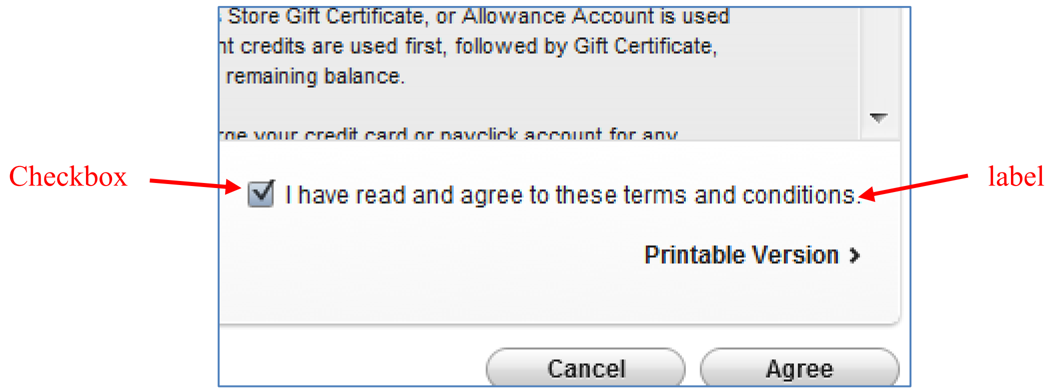
```
<form>
                    :
          <input type = “checkbox”>
                    :
</form>
```

By default the **state** of a check box is unchecked. This can be changed by setting the *checked* attribute to “true”. Check boxes are typically used to control the execution of scripts and also to control access to other scripts, forms and form controls. It is therefore important to give a check box a *name* and/or an *id*. A *value* attribute may be assigned, this will be the “value” of the check box when it is in the “checked” state. A check box does not have a caption, therefore, to distinguish it on a form, a label control or text label must be appended to it. For example:

```
<input type = “checkbox” checked = “true”  value = “1” name = “AllowToProceed”>label
```

The most commonly used event for check boxes is *onclick*, i.e., when the state of the control is changed the *onclick* code will fire.

An example of a check box in use is as follows:



Store Gift Certificate, or Allowance Account is used
nt credits are used first, followed by Gift Certificate,
remaining balance.

I have read and agree to these terms and conditions.

[Source: iTunes Store]

2.13 Radio Button

Also commonly called an **option button**, a radio button allows one item to be selected from a group. To create a radio button on a form use the tags:

```
<form>
                              :
<input type = "radio" name = "nameofcontrol" value = "value1">label1<br>
<input type = "radio" name = "nameofcontrol" value = "value2"> label2<br>
                              :
<input type = "radio" name = "nameofcontrol" value = "valuen">labeln

</form>
```

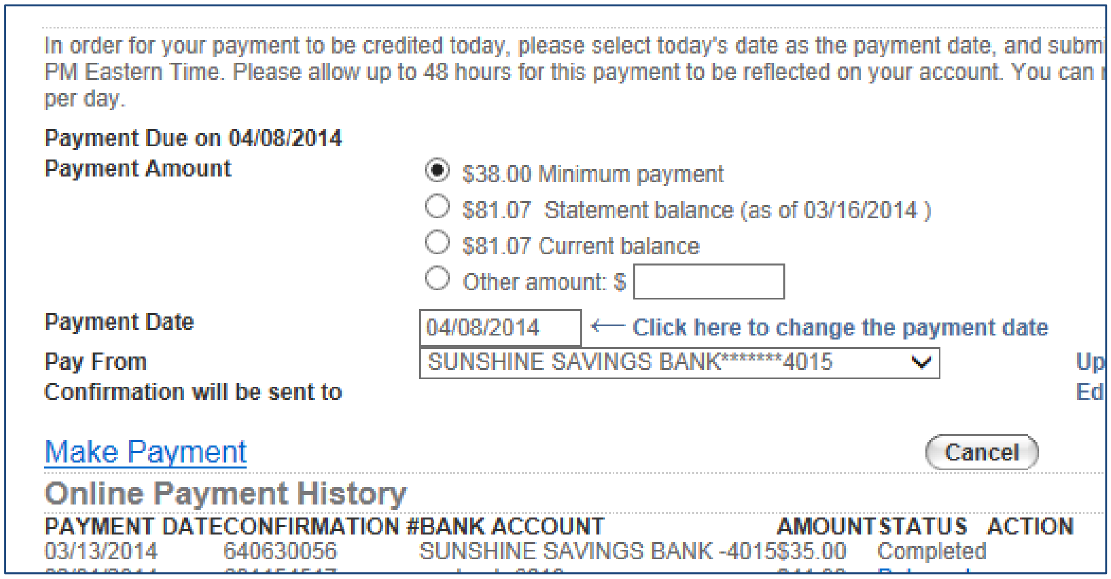
To select another option, the radio button for that item will be clicked on to select it, and all the other option buttons will be unselected.

Radio buttons are used in groups. All members of the group are given the same name. When one member is "checked", all of the other members will be "unchecked". One member of the group may be set to "checked" by default by assigning the "true" value to its *checked* attribute. For example:

```
Select your zip code
<input type = "radio" name="CustZip" value ="zip1">&nbsp; 32318<br>
<input type = "radio" name="CustZip" value ="zip2"checked = "true">&nbsp; 32319<br>
<input type = "radio" name="CustZip"  value ="zip3">&nbsp; 32320<br>
<input type = "radio" name="CustZip" value ="zip4">&nbsp; 32321<br>
<input type = "radio" name="CustZip" value ="zip5">&nbsp; 32322
```

Note that the special HTML string ***&nbsp;*** which provides a space, analogous to hitting the spacebar on the keyboard.

The most commonly used event for radio buttons is the *onclick* event. An example on the use of radio buttons is shown below.


In order for your payment to be credited today, please select today's date as the payment date, and subm
PM Eastern Time. Please allow up to 48 hours for this payment to be reflected on your account. You can
per day.

Payment Due on 04/08/2014

Payment Amount
● $38.00 Minimum payment
○ $81.07 Statement balance (as of 03/16/2014)
○ $81.07 Current balance
○ Other amount: $

Payment Date 04/08/2014 ← Click here to change the payment date

Pay From SUNSHINE SAVINGS BANK*******4015 Up

Confirmation will be sent to Ed

Online Payment History

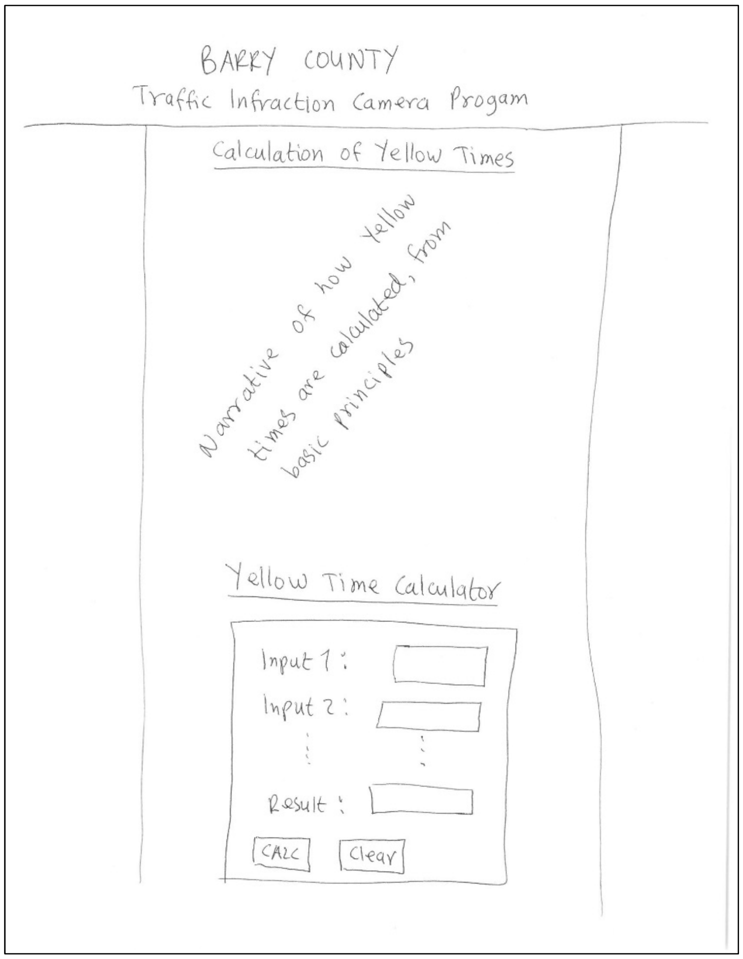
PAYMENT DATE	CONFIRMATION #	BANK ACCOUNT	AMOUNT	STATUS	ACTION
03/13/2014	640630056	SUNSHINE SAVINGS BANK -4015	$35.00	Completed	

[Source: https://www.onlinecreditcenter2.com/JCPenney/JCPCreditCenter.htm]

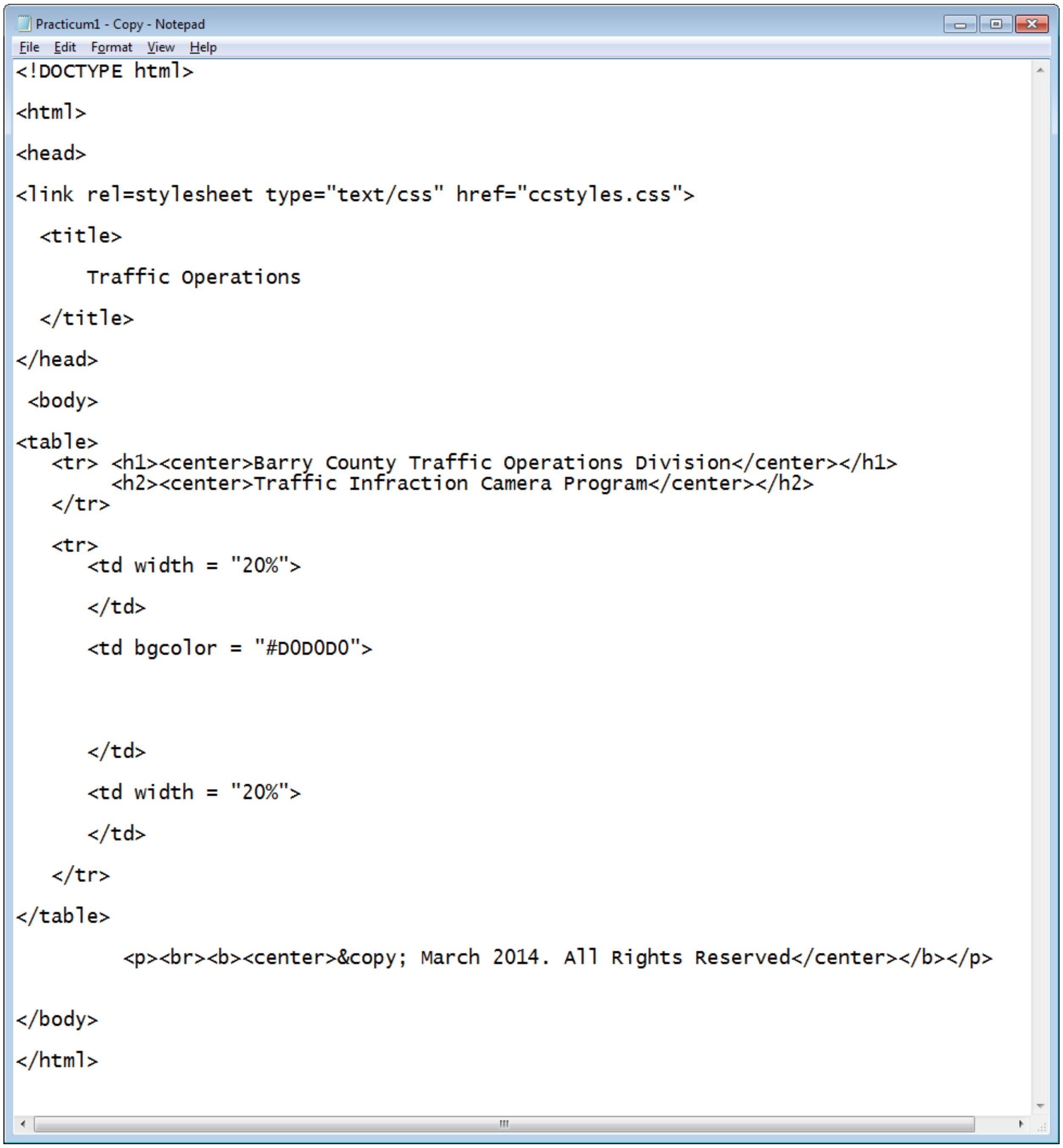
2.14 Practicum #1

Scenario: A local agency has installed traffic infraction cameras, commonly called red-light cameras, at strategic intersections under its jurisdiction as a response to traffic and law enforcement data showing continued and sustained increase in crashes and speeding by motorists. The public perception, fueled by media reports, however, is that the traffic signals' timing plans, particularly the yellow times, have been adjusted to catch more drivers and increase revenues from the traffic infraction camera program. As the traffic engineer you want to set up a web page to dispel such claims by explaining the fundamental principles of calculating the yellow time. The web page will also be interactive such that users can enter data and calculate baseline yellow times on their own and compare with their experiences.

A sketch of the web page is as follows:

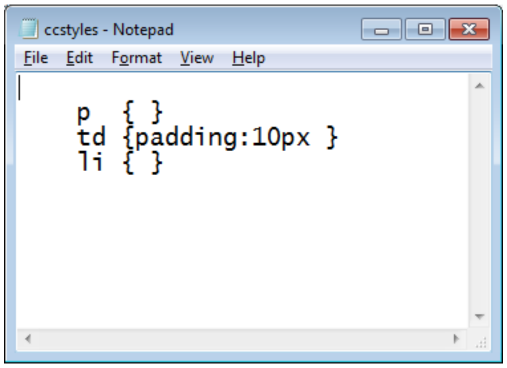

BARRY COUNTY
Traffic Infraction Camera Progam
Calculation of Yellow Times
Narrative of how Yellow times are calculated, from basic principles
Yellow Time Calculator
Input 1 :
Input 2 :
Result :
CALC
Clear

Solution:
Open a new session of *Notepad* (or your preferred text editor).
Type in the following to create the web page.

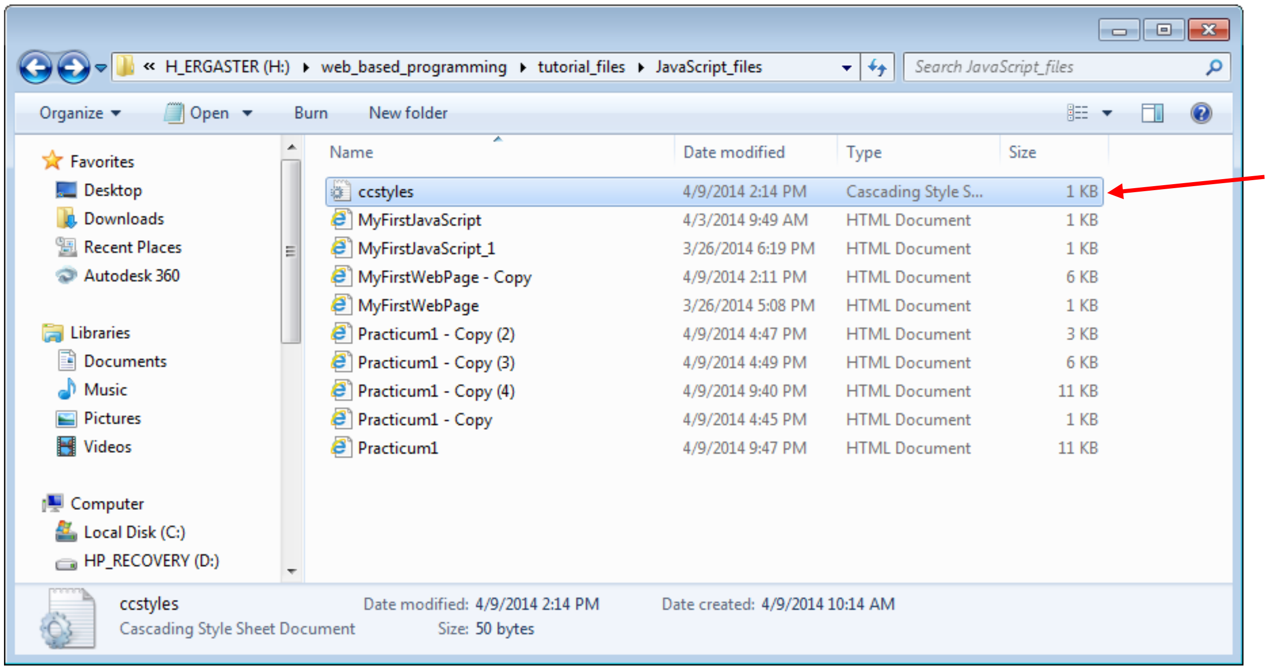


```
<!DOCTYPE html>

<html>

<head>

<link rel=stylesheet type="text/css" href="ccstyles.css">

  <title>

      Traffic Operations

  </title>

</head>

 <body>

<table>
   <tr> <h1><center>Barry County Traffic Operations Division</center></h1>
        <h2><center>Traffic Infraction Camera Program</center></h2>
   </tr>

   <tr>
      <td width = "20%">

      </td>

      <td bgcolor = "#D0D0D0">



      </td>

      <td width = "20%">

      </td>

   </tr>

</table>

         <p><br><b><center>&copy; March 2014. All Rights Reserved</center></b></p>


</body>

</html>
```

The layout of the web page consists of a table that has two rows. The first row consists of one column. The second row has three columns. The bulk of the narrative and controls will be placed in the second column of the second row. To distinguish it, a background color is provided. Note that no borders have been assigned at this time. Below the table, towards the bottom of the web page, add language concerning copyright privileges, disclaimers, etc. Note the copyright special character. The ***.css*** file is the cascading styles sheet document for this web page. Based on how it has been declared, it must be saved to the same folder as the web page HTML file you are developing. The ***.css*** file currently has minimal data. Currently the table cell padding is the only attribute specified. We shall upgrade and update the cascading style sheet as we proceed.



```
p  { }
td {padding:10px }
li { }
```

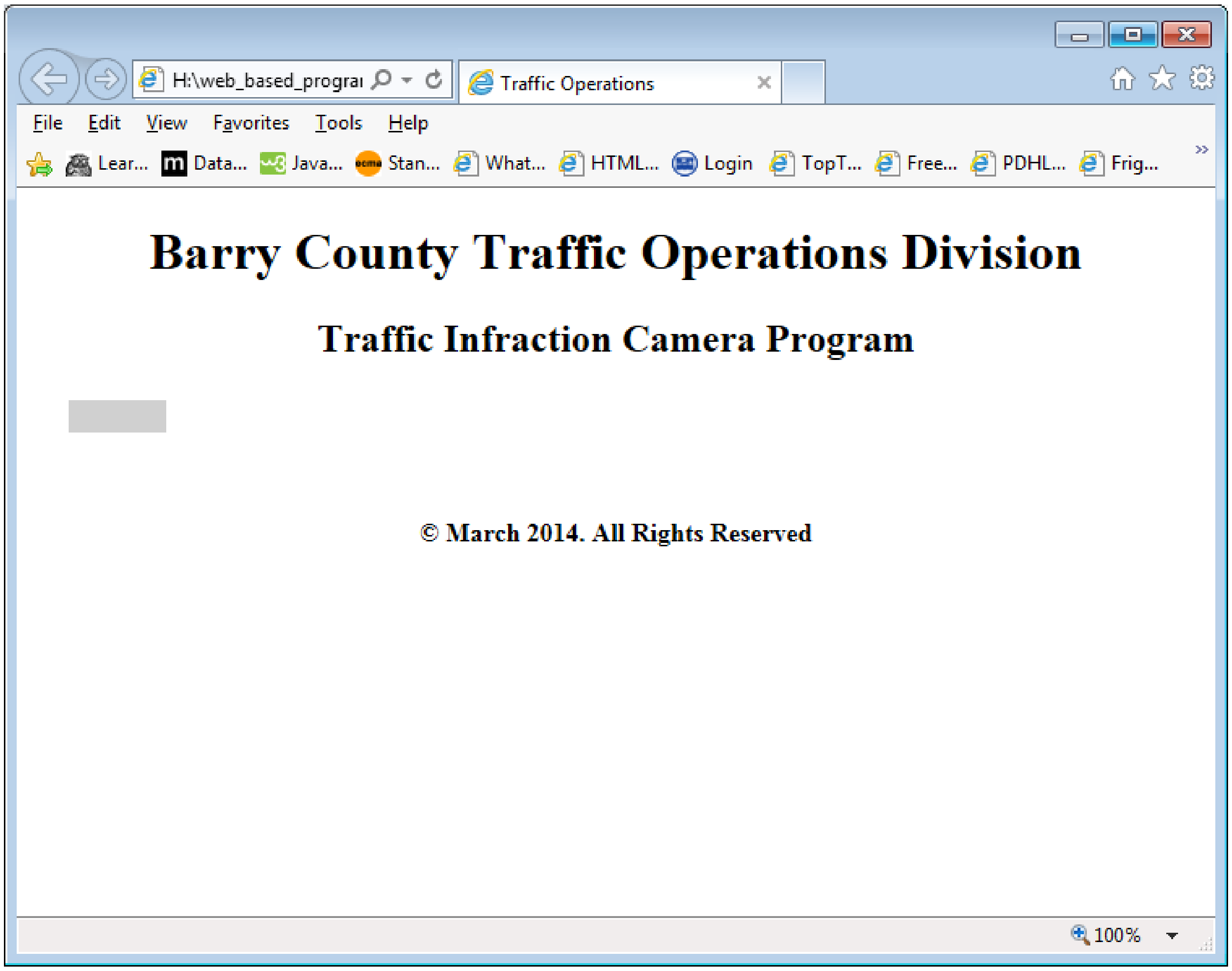


Name	Date modified	Type	Size
ccstyles	4/9/2014 2:14 PM	Cascading Style S...	1 KB
MyFirstJavaScript	4/3/2014 9:49 AM	HTML Document	1 KB
MyFirstJavaScript_1	3/26/2014 6:19 PM	HTML Document	1 KB
MyFirstWebPage - Copy	4/9/2014 2:11 PM	HTML Document	6 KB
MyFirstWebPage	3/26/2014 5:08 PM	HTML Document	1 KB
Practicum1 - Copy (2)	4/9/2014 4:47 PM	HTML Document	3 KB
Practicum1 - Copy (3)	4/9/2014 4:49 PM	HTML Document	6 KB
Practicum1 - Copy (4)	4/9/2014 9:40 PM	HTML Document	11 KB
Practicum1 - Copy	4/9/2014 4:45 PM	HTML Document	1 KB
Practicum1	4/9/2014 9:47 PM	HTML Document	11 KB

ccstyles   Date modified: 4/9/2014 2:14 PM   Date created: 4/9/2014 10:14 AM
Cascading Style Sheet Document   Size: 50 bytes

Save your text file as an ***.html*** file.
Select a file name of your choice and ensure that you save it to the same folder as your cascading styles sheet file.

Test your file by double clicking on it from the folder in which it resides. Confirm that the content and elements appear as intended. If not, adjust your code and repeat the review process.

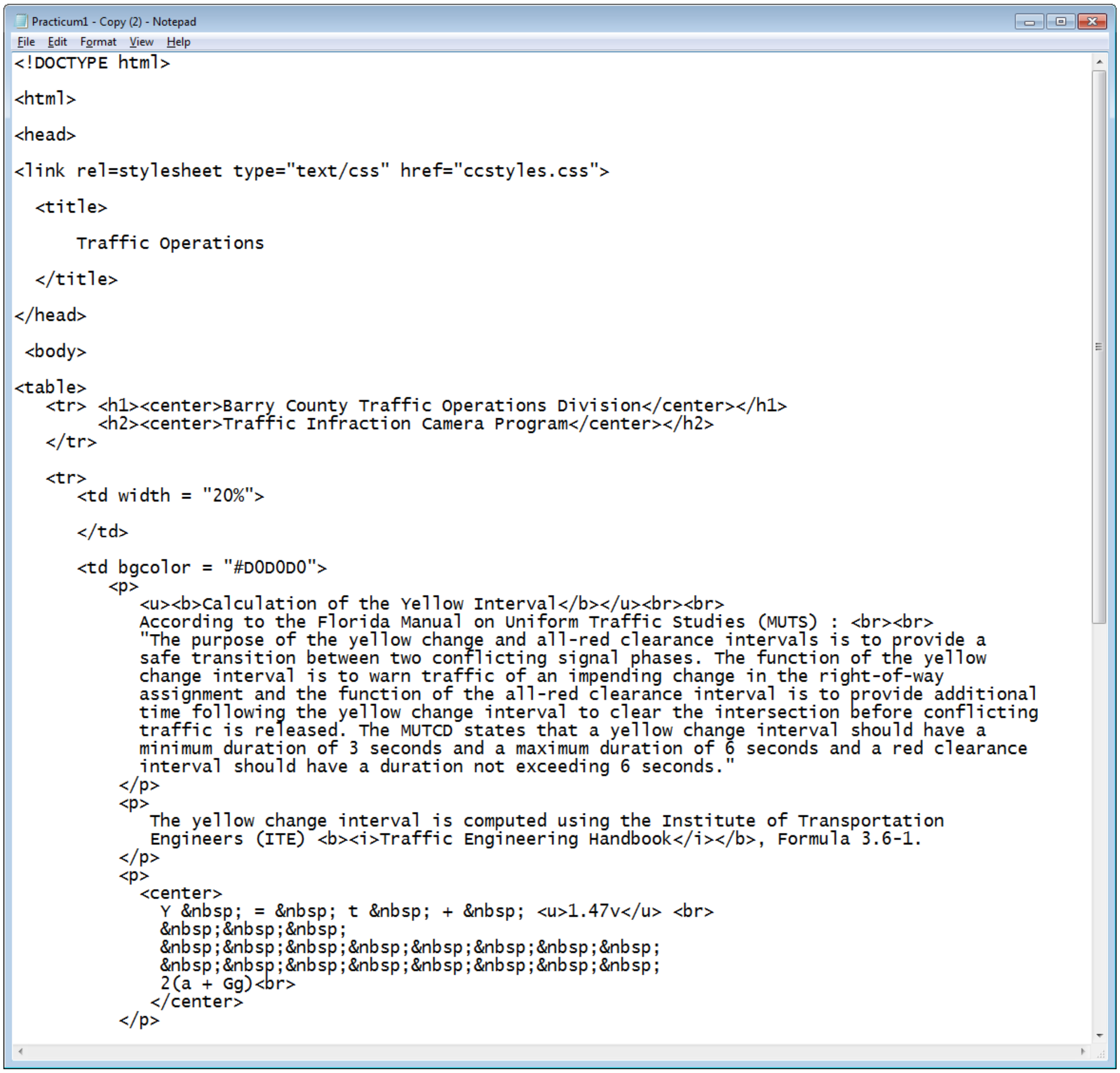

H:\web_based_prograı
Traffic Operations
File Edit View Favorites Tools Help
Lear... Data... Java... Stan... What... HTML... Login TopT... Free... PDHL... Frig...
100%

Barry County Traffic Operations Division

Traffic Infraction Camera Program

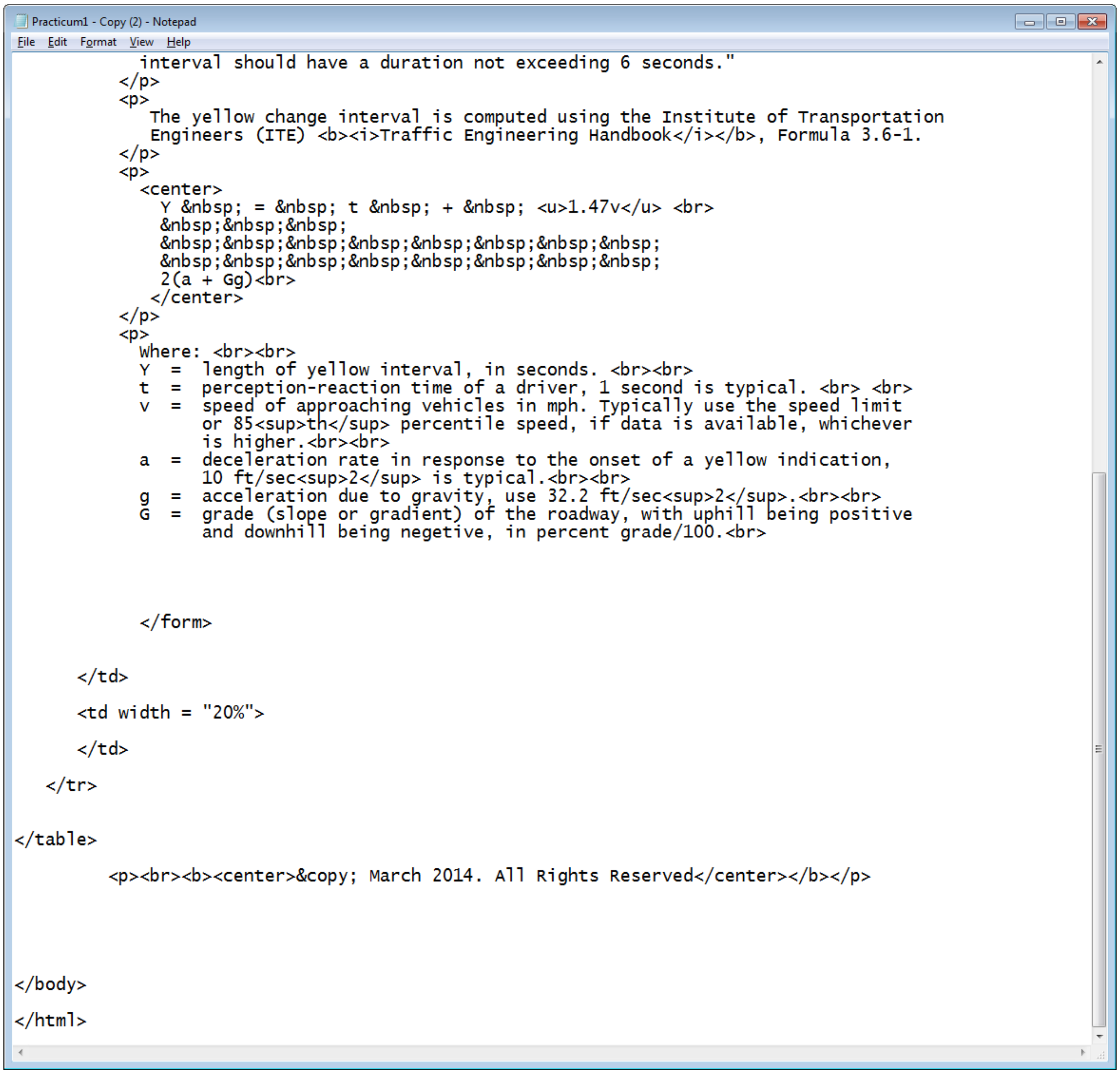
© March 2014. All Rights Reserved

Once confirmed that the web page displays as originally intended, type in the narrative for this web page. This narrative will be typed into the middle cell of the second row which has been given a background color to distinguish it.



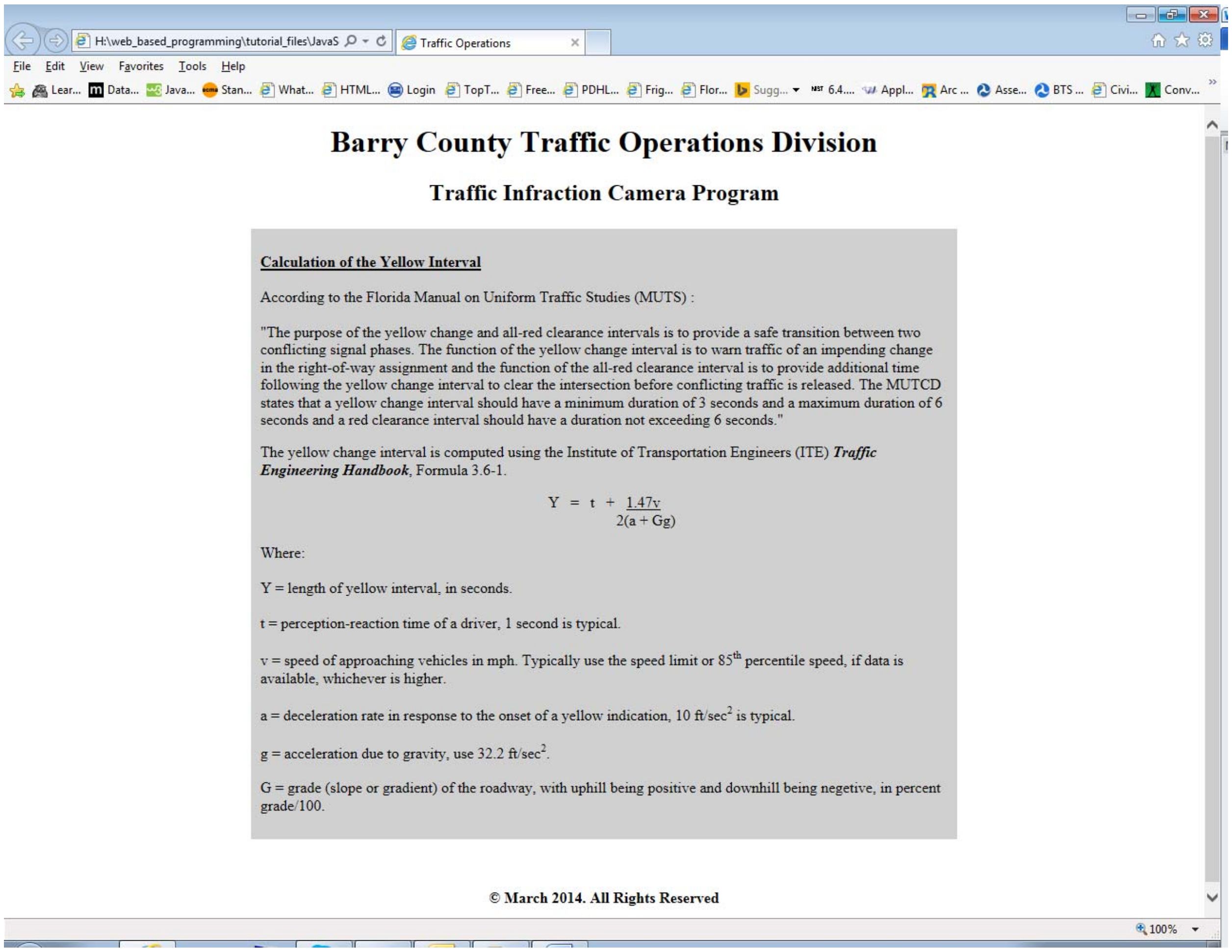
```
<!DOCTYPE html>

<html>

<head>

<link rel=stylesheet type="text/css" href="ccstyles.css">

  <title>

      Traffic Operations

  </title>

</head>

 <body>

<table>
   <tr> <h1><center>Barry County Traffic Operations Division</center></h1>
        <h2><center>Traffic Infraction Camera Program</center></h2>
   </tr>

   <tr>
      <td width = "20%">

      </td>

      <td bgcolor = "#D0D0D0">
         <p>
            <u><b>Calculation of the Yellow Interval</b></u><br><br>
            According to the Florida Manual on Uniform Traffic Studies (MUTS) : <br><br>
            "The purpose of the yellow change and all-red clearance intervals is to provide a
            safe transition between two conflicting signal phases. The function of the yellow
            change interval is to warn traffic of an impending change in the right-of-way
            assignment and the function of the all-red clearance interval is to provide additional
            time following the yellow change interval to clear the intersection before conflicting
            traffic is released. The MUTCD states that a yellow change interval should have a
            minimum duration of 3 seconds and a maximum duration of 6 seconds and a red clearance
            interval should have a duration not exceeding 6 seconds."
          </p>
          <p>
             The yellow change interval is computed using the Institute of Transportation
             Engineers (ITE) <b><i>Traffic Engineering Handbook</i></b>, Formula 3.6-1.
          </p>
          <p>
            <center>
              Y &nbsp; = &nbsp; t &nbsp; + &nbsp; <u>1.47v</u> <br>
              &nbsp;&nbsp;&nbsp;
              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
              2(a + Gg)<br>
             </center>
          </p>
```

and continuing,



```
            interval should have a duration not exceeding 6 seconds."
          </p>
          <p>
             The yellow change interval is computed using the Institute of Transportation
             Engineers (ITE) <b><i>Traffic Engineering Handbook</i></b>, Formula 3.6-1.
          </p>
          <p>
            <center>
              Y &nbsp; = &nbsp; t &nbsp; + &nbsp; <u>1.47v</u> <br>
              &nbsp;&nbsp;&nbsp;
              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
              2(a + Gg)<br>
             </center>
          </p>
          <p>
            Where: <br><br>
            Y  =  length of yellow interval, in seconds. <br><br>
            t  =  perception-reaction time of a driver, 1 second is typical. <br> <br>
            v  =  speed of approaching vehicles in mph. Typically use the speed limit
                  or 85<sup>th</sup> percentile speed, if data is available, whichever
                  is higher.<br><br>
            a  =  deceleration rate in response to the onset of a yellow indication,
                  10 ft/sec<sup>2</sup> is typical.<br><br>
            g  =  acceleration due to gravity, use 32.2 ft/sec<sup>2</sup>.<br><br>
            G  =  grade (slope or gradient) of the roadway, with uphill being positive
                  and downhill being negetive, in percent grade/100.<br>



            </form>


      </td>

      <td width = "20%">

      </td>

   </tr>


</table>

         <p><br><b><center>&copy; March 2014. All Rights Reserved</center></b></p>




</body>

</html>
```

Save your HTML file.
Test your file.



H:\web_based_programming\tutorial_files\JavaS
Traffic Operations
File Edit View Favorites Tools Help
100%

Barry County Traffic Operations Division

Traffic Infraction Camera Program

Calculation of the Yellow Interval

According to the Florida Manual on Uniform Traffic Studies (MUTS) :

"The purpose of the yellow change and all-red clearance intervals is to provide a safe transition between two conflicting signal phases. The function of the yellow change interval is to warn traffic of an impending change in the right-of-way assignment and the function of the all-red clearance interval is to provide additional time following the yellow change interval to clear the intersection before conflicting traffic is released. The MUTCD states that a yellow change interval should have a minimum duration of 3 seconds and a maximum duration of 6 seconds and a red clearance interval should have a duration not exceeding 6 seconds."

The yellow change interval is computed using the Institute of Transportation Engineers (ITE) ***Traffic Engineering Handbook***, Formula 3.6-1.

$$Y = t + \frac{1.47v}{2(a + Gg)}$$

Where:

Y = length of yellow interval, in seconds.

t = perception-reaction time of a driver, 1 second is typical.

v = speed of approaching vehicles in mph. Typically use the speed limit or 85th percentile speed, if data is available, whichever is higher.

a = deceleration rate in response to the onset of a yellow indication, 10 ft/sec^2 is typical.

g = acceleration due to gravity, use 32.2 ft/sec^2.

G = grade (slope or gradient) of the roadway, with uphill being positive and downhill being negetive, in percent grade/100.

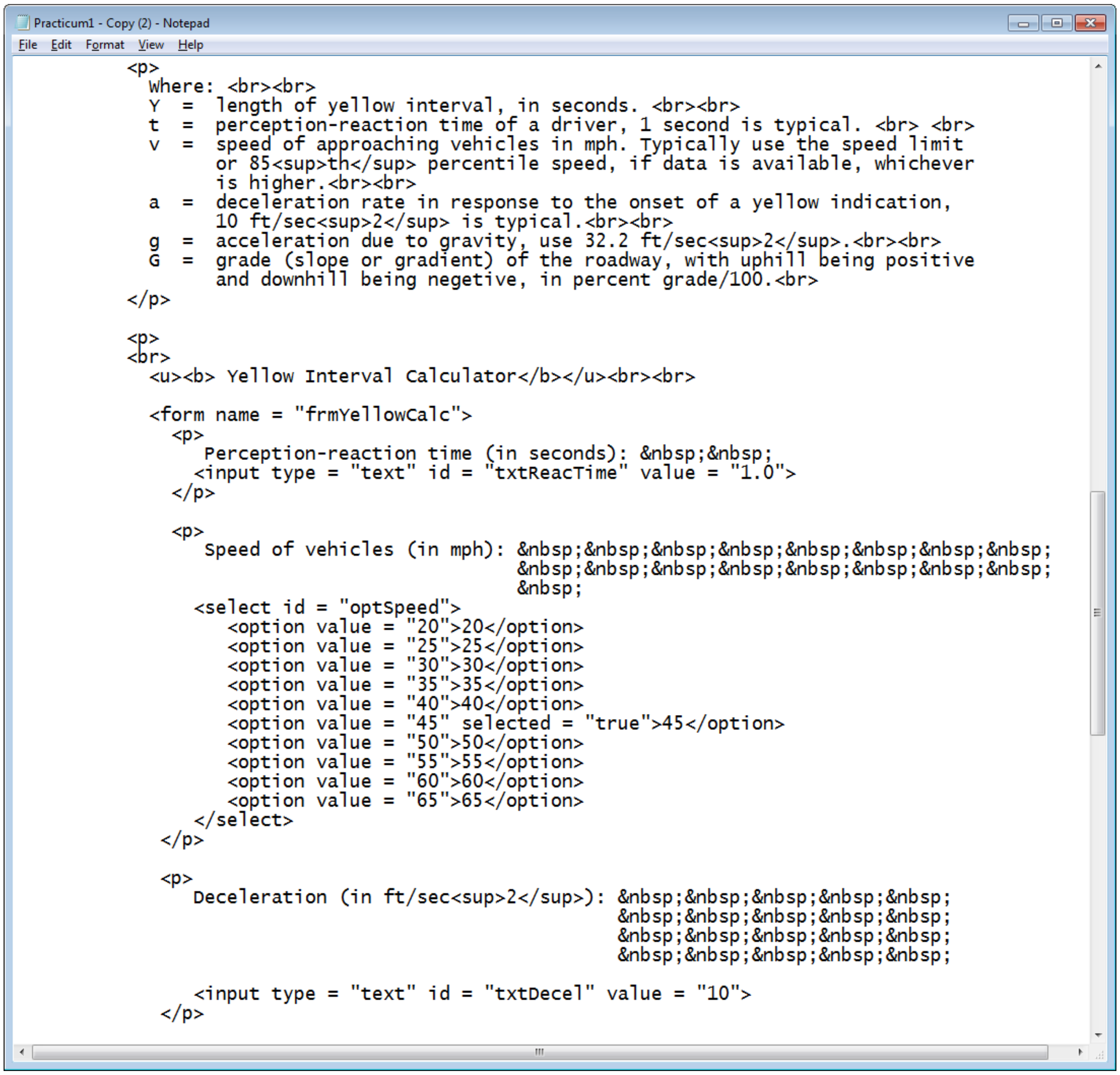
© March 2014. All Rights Reserved

Note the use of the superscript tags to display the mathematical notation. Also, in order to align the mathematical notation accordingly, the special space character ***&nbsp;*** was used multiple times as needed. Alternately, you can use the tab and line break special characters as needed. Another strategy is to create sub-tables and manipulate the cell alignment attributes to achieve the desired result.

If you prefer tables with borders and background colors in the other table cells, you may add the necessary attributes in your HTML code or in the ***.css*** file.

Next we shall add a form and form controls to create the interactive part of the web page where users can enter numbers and click buttons and calculate the yellow time and all-red time for a traffic signal. This may help dispel rumors that the traffic engineers have manipulated these settings to catch more drivers and boost revenues from the red light camera program.

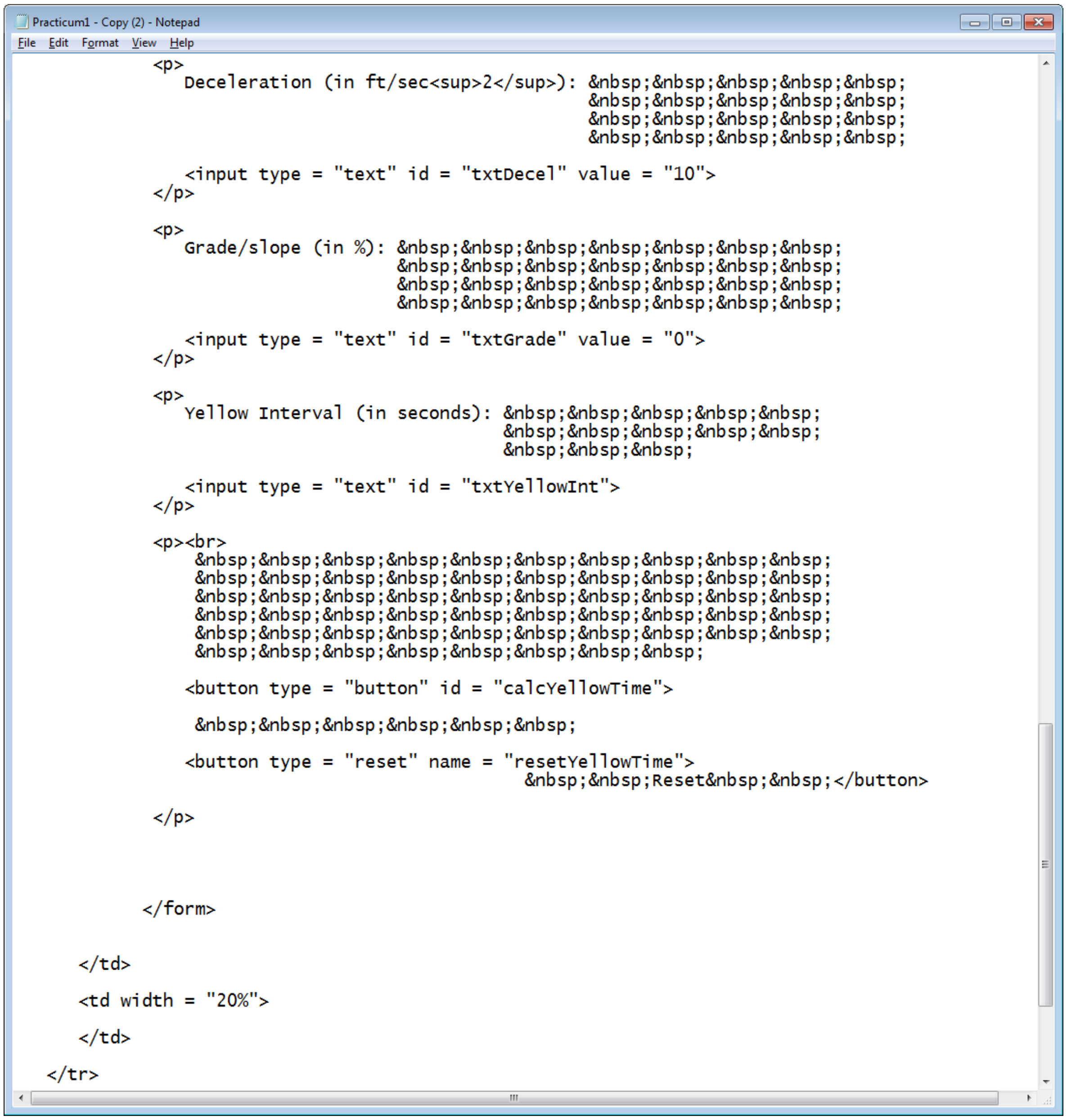
Add the following code for the forms and form controls.



Practicum1 - Copy (2) - Notepad

```
           <p>
             Where: <br><br>
             Y  =  length of yellow interval, in seconds. <br><br>
             t  =  perception-reaction time of a driver, 1 second is typical. <br> <br>
             v  =  speed of approaching vehicles in mph. Typically use the speed limit
                   or 85<sup>th</sup> percentile speed, if data is available, whichever
                   is higher.<br><br>
             a  =  deceleration rate in response to the onset of a yellow indication,
                   10 ft/sec<sup>2</sup> is typical.<br><br>
             g  =  acceleration due to gravity, use 32.2 ft/sec<sup>2</sup>.<br><br>
             G  =  grade (slope or gradient) of the roadway, with uphill being positive
                   and downhill being negetive, in percent grade/100.<br>
           </p>

           <p>
           <br>
             <u><b> Yellow Interval Calculator</b></u><br><br>

             <form name = "frmYellowCalc">
               <p>
                  Perception-reaction time (in seconds): &nbsp;&nbsp;
                 <input type = "text" id = "txtReacTime" value = "1.0">
               </p>

               <p>
                  Speed of vehicles (in mph): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                              &nbsp;
                 <select id = "optSpeed">
                    <option value = "20">20</option>
                    <option value = "25">25</option>
                    <option value = "30">30</option>
                    <option value = "35">35</option>
                    <option value = "40">40</option>
                    <option value = "45" selected = "true">45</option>
                    <option value = "50">50</option>
                    <option value = "55">55</option>
                    <option value = "60">60</option>
                    <option value = "65">65</option>
                 </select>
              </p>

              <p>
                 Deceleration (in ft/sec<sup>2</sup>): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                       &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                       &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                       &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                 <input type = "text" id = "txtDecel" value = "10">
              </p>
```

and continuing,

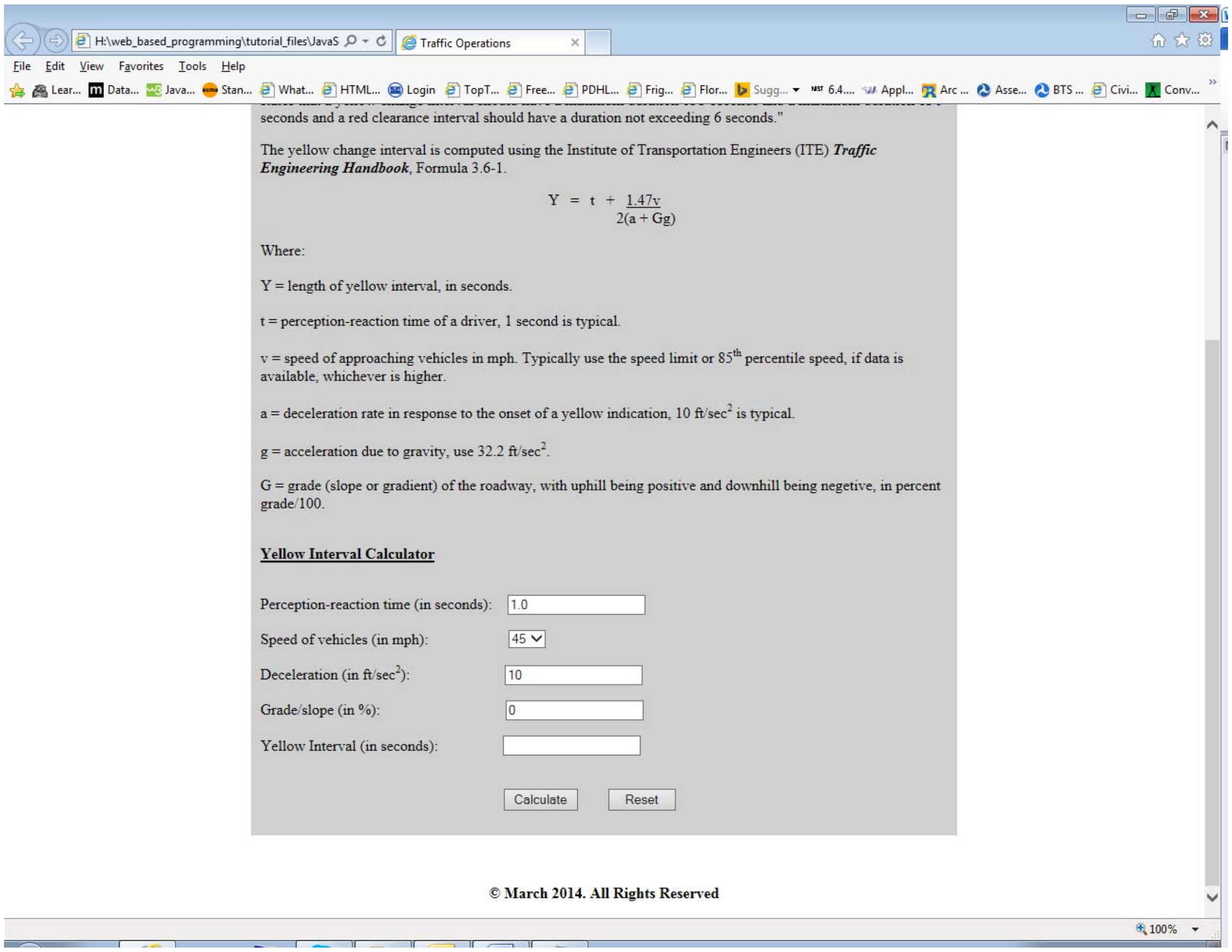


```
            <p>
               Deceleration (in ft/sec<sup>2</sup>): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                     &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                     &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                     &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

               <input type = "text" id = "txtDecel" value = "10">
            </p>

            <p>
               Grade/slope (in %): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

               <input type = "text" id = "txtGrade" value = "0">
            </p>

            <p>
               Yellow Interval (in seconds): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                             &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                             &nbsp;&nbsp;&nbsp;

               <input type = "text" id = "txtYellowInt">
            </p>

            <p><br>
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

               <button type = "button" id = "calcYellowTime">

                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

               <button type = "reset" name = "resetYellowTime">
                                              &nbsp;&nbsp;Reset&nbsp;&nbsp;</button>

            </p>



           </form>

      </td>

      <td width = "20%">

      </td>

   </tr>
```

Note that the special space character ***&nbsp;*** was used multiple times to align the controls as needed. Explore other innovative and create ways of using special characters and/or embedded sub-tables to achieve similar results.

Save your document.
Test your document in the web browser.

Note the default values in some of the controls. Review your code to see where these were set.


H:\web_based_programming\tutorial_files\JavaS
Traffic Operations
File Edit View Favorites Tools Help
100%

seconds and a red clearance interval should have a duration not exceeding 6 seconds."

The yellow change interval is computed using the Institute of Transportation Engineers (ITE) ***Traffic Engineering Handbook***, Formula 3.6-1.

$$Y = t + \frac{1.47v}{2(a + Gg)}$$

Where:

Y = length of yellow interval, in seconds.

t = perception-reaction time of a driver, 1 second is typical.

v = speed of approaching vehicles in mph. Typically use the speed limit or 85th percentile speed, if data is available, whichever is higher.

a = deceleration rate in response to the onset of a yellow indication, 10 ft/sec^2 is typical.

g = acceleration due to gravity, use 32.2 ft/sec^2.

G = grade (slope or gradient) of the roadway, with uphill being positive and downhill being negetive, in percent grade/100.

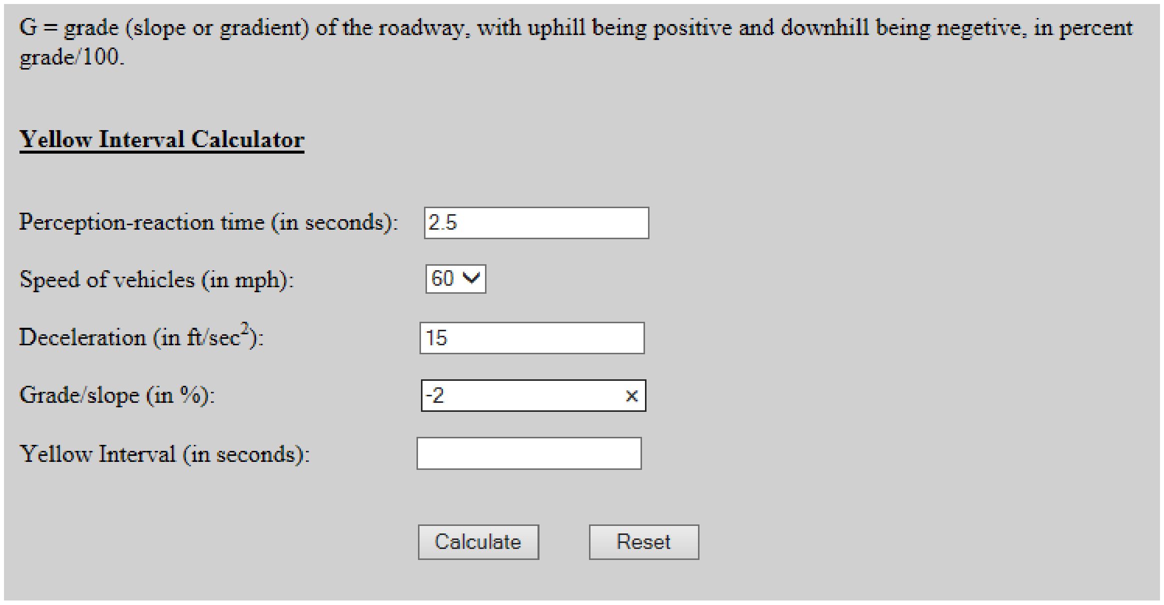
Yellow Interval Calculator

Perception-reaction time (in seconds):	1.0
Speed of vehicles (in mph):	45
Deceleration (in ft/sec^2):	10
Grade/slope (in %):	0
Yellow Interval (in seconds):	

Calculate Reset

© March 2014. All Rights Reserved

Change the values in some of the controls.

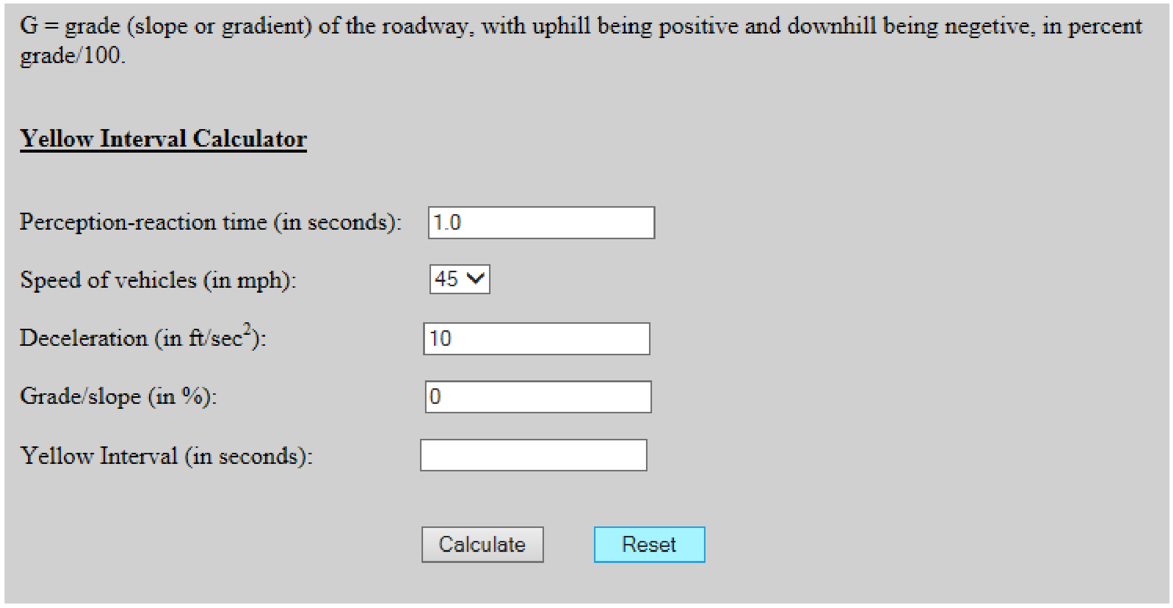


G = grade (slope or gradient) of the roadway, with uphill being positive and downhill being negetive, in percent grade/100.

Yellow Interval Calculator

Perception-reaction time (in seconds):	2.5
Speed of vehicles (in mph):	60
Deceleration (in ft/sec^2):	15
Grade/slope (in %):	-2
Yellow Interval (in seconds):	

Click the *Reset* button. The form "clears" and reverts to its original settings.



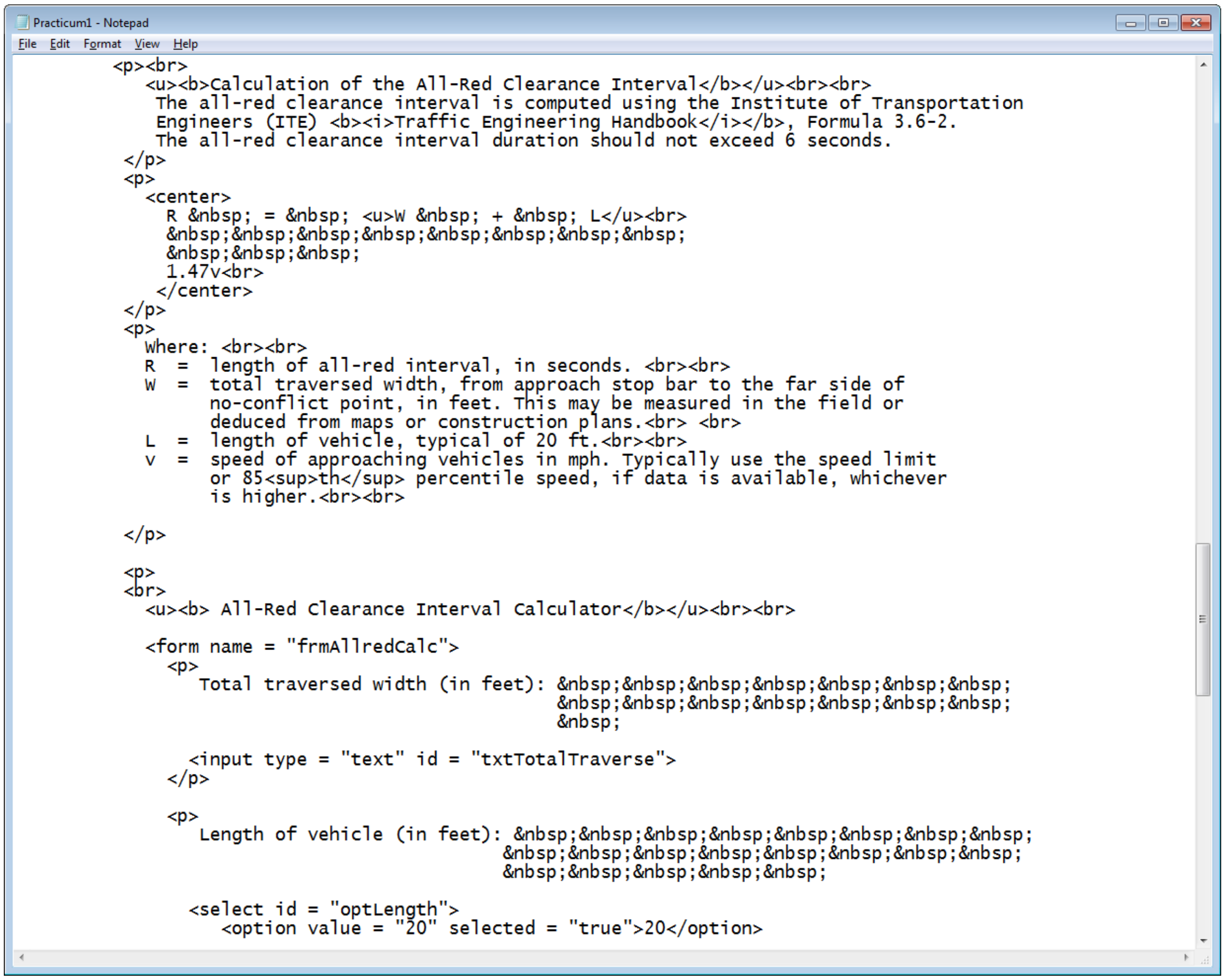
G = grade (slope or gradient) of the roadway, with uphill being positive and downhill being negetive, in percent grade/100.

Yellow Interval Calculator

Perception-reaction time (in seconds):	1.0
Speed of vehicles (in mph):	45
Deceleration (in ft/sec^2):	10
Grade/slope (in %):	0
Yellow Interval (in seconds):	

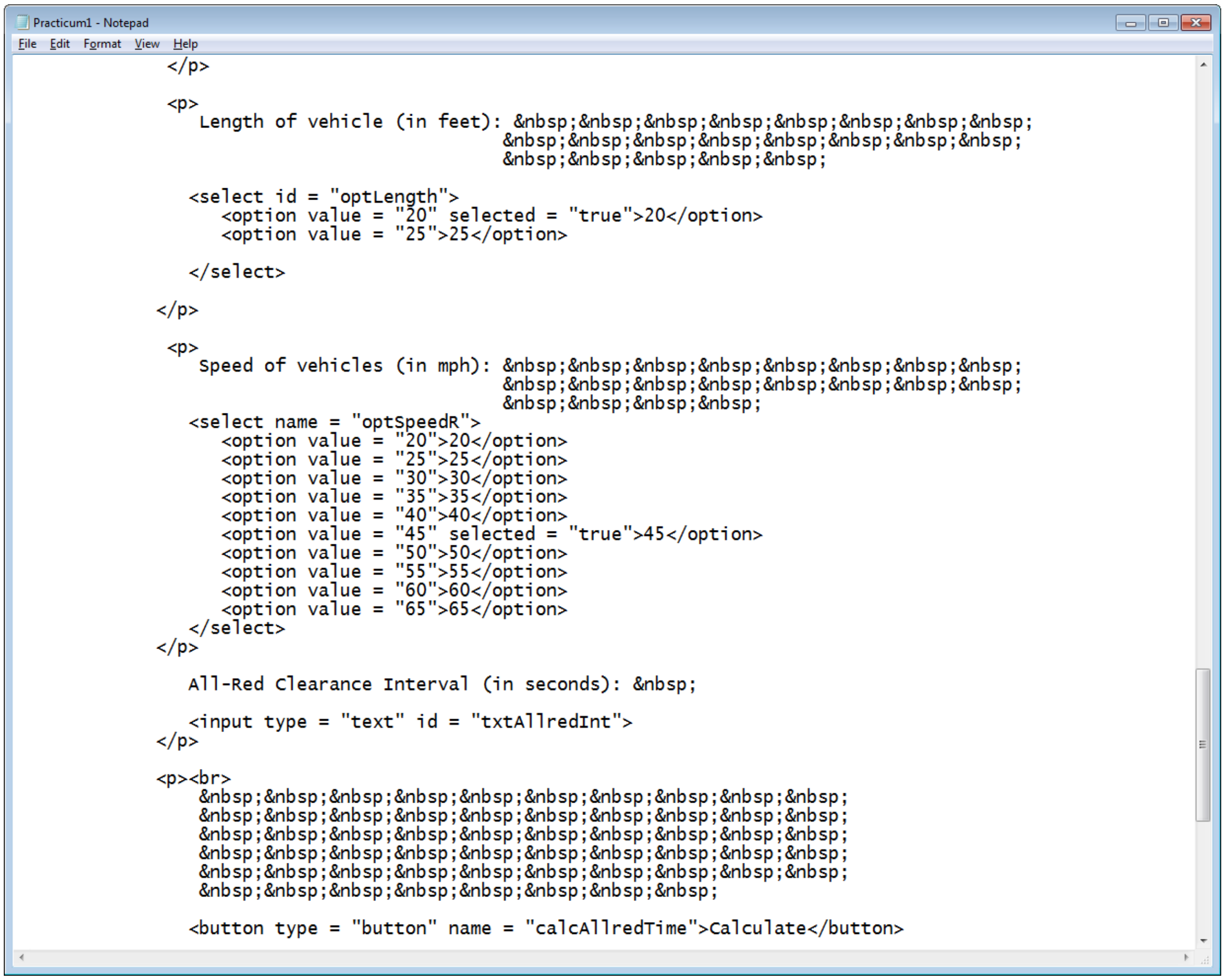
The *Calculate* button currently produces no result. Later on we shall come back and write code for it in *JavaScript*.
Save your file.

Add the following additional information and form calculator for the All-Red Interval.



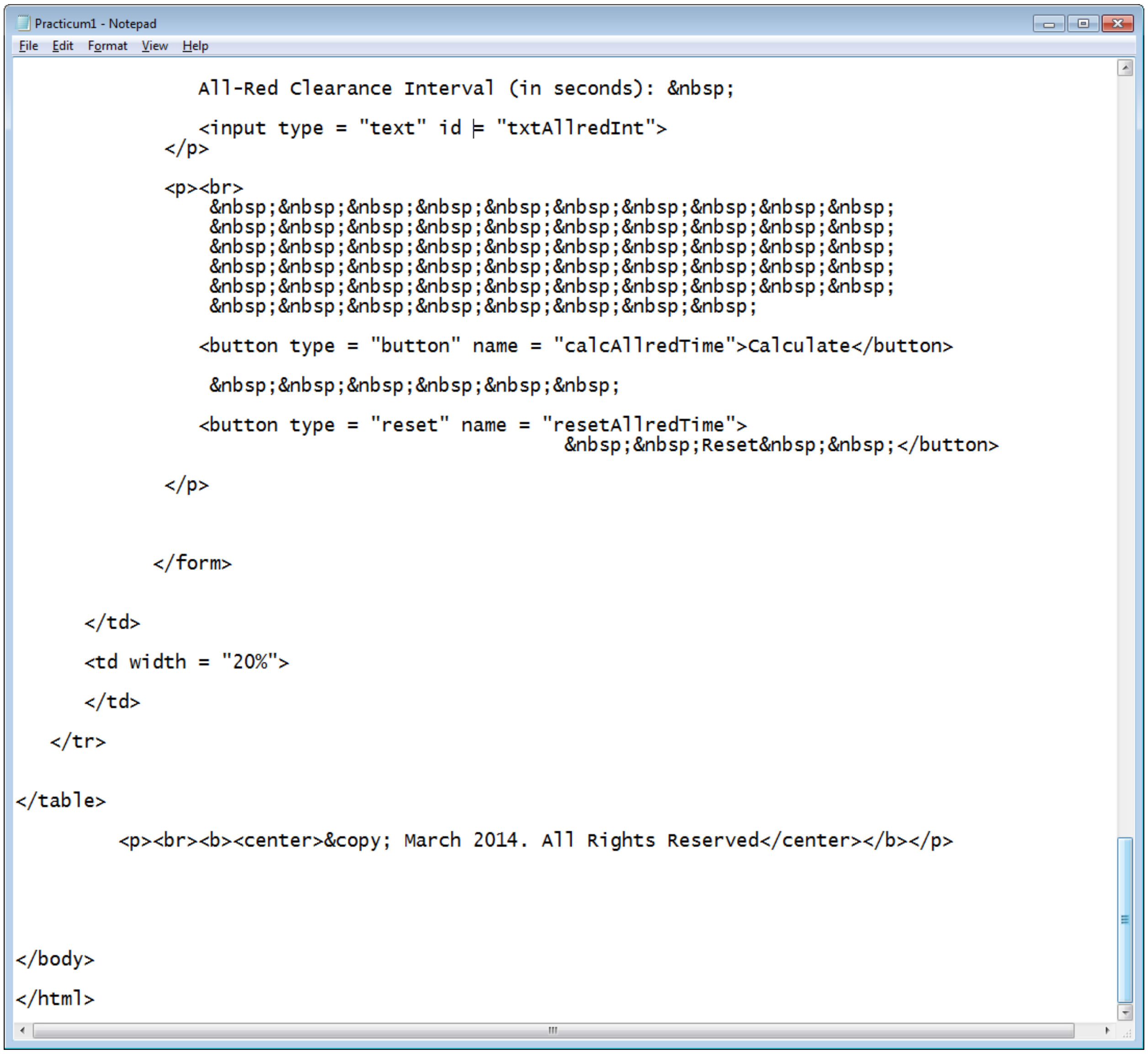
```
          <p><br>
             <u><b>Calculation of the All-Red Clearance Interval</b></u><br><br>
              The all-red clearance interval is computed using the Institute of Transportation
              Engineers (ITE) <b><i>Traffic Engineering Handbook</i></b>, Formula 3.6-2.
              The all-red clearance interval duration should not exceed 6 seconds.
           </p>
           <p>
             <center>
               R &nbsp; = &nbsp; <u>W &nbsp; + &nbsp; L</u><br>
               &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
               &nbsp;&nbsp;&nbsp;
               1.47v<br>
              </center>
           </p>
           <p>
             Where: <br><br>
             R  =  length of all-red interval, in seconds. <br><br>
             W  =  total traversed width, from approach stop bar to the far side of
                   no-conflict point, in feet. This may be measured in the field or
                   deduced from maps or construction plans.<br> <br>
             L  =  length of vehicle, typical of 20 ft.<br><br>
             v  =  speed of approaching vehicles in mph. Typically use the speed limit
                   or 85<sup>th</sup> percentile speed, if data is available, whichever
                   is higher.<br><br>

           </p>

           <p>
           <br>
             <u><b> All-Red Clearance Interval Calculator</b></u><br><br>

             <form name = "frmAllredCalc">
               <p>
                  Total traversed width (in feet): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                                   &nbsp;

                 <input type = "text" id = "txtTotalTraverse">
               </p>

               <p>
                  Length of vehicle (in feet): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                 <select id = "optLength">
                    <option value = "20" selected = "true">20</option>
```

and continuing,



```
                </p>

                <p>
                   Length of vehicle (in feet): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                               &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                               &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                  <select id = "optLength">
                     <option value = "20" selected = "true">20</option>
                     <option value = "25">25</option>

                  </select>

               </p>

                <p>
                   Speed of vehicles (in mph): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                               &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                               &nbsp;&nbsp;&nbsp;&nbsp;
                  <select name = "optSpeedR">
                     <option value = "20">20</option>
                     <option value = "25">25</option>
                     <option value = "30">30</option>
                     <option value = "35">35</option>
                     <option value = "40">40</option>
                     <option value = "45" selected = "true">45</option>
                     <option value = "50">50</option>
                     <option value = "55">55</option>
                     <option value = "60">60</option>
                     <option value = "65">65</option>
                  </select>
               </p>

                  All-Red Clearance Interval (in seconds): &nbsp;

                  <input type = "text" id = "txtAllredInt">
               </p>

               <p><br>
                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                  <button type = "button" name = "calcAllredTime">Calculate</button>
```

and finally,

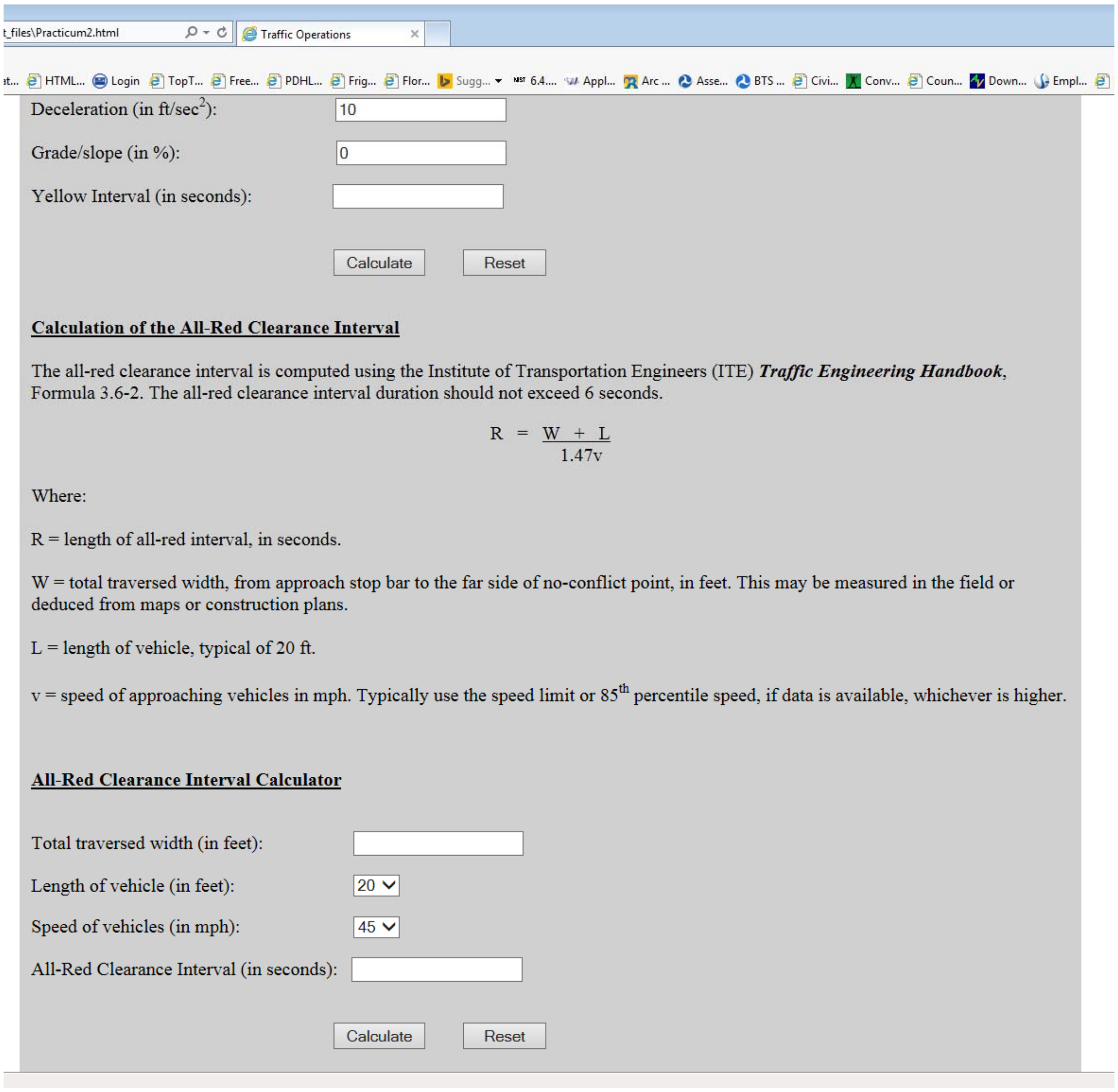


Practicum1 - Notepad

File Edit Format View Help

```
                All-Red Clearance Interval (in seconds): &nbsp;

                <input type = "text" id = "txtAllredInt">
             </p>

             <p><br>
                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                <button type = "button" name = "calcAllredTime">Calculate</button>

                 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                <button type = "reset" name = "resetAllredTime">
                                                &nbsp;&nbsp;Reset&nbsp;&nbsp;</button>

             </p>


            </form>

      </td>

      <td width = "20%">

      </td>

   </tr>

</table>

         <p><br><b><center>&copy; March 2014. All Rights Reserved</center></b></p>



</body>

</html>
```

Test your file in the web browser.


t_files\Practicum2.html
Traffic Operations
HTML... Login TopT... Free... PDHL... Frig... Flor... Sugg... 6.4.... Appl... Arc ... Asse... BTS ... Civi... Conv... Coun... Down... Empl...
Calculate
Reset

Deceleration (in ft/sec^2):	10
Grade/slope (in %):	0
Yellow Interval (in seconds):	

Calculation of the All-Red Clearance Interval

The all-red clearance interval is computed using the Institute of Transportation Engineers (ITE) ***Traffic Engineering Handbook***, Formula 3.6-2. The all-red clearance interval duration should not exceed 6 seconds.

$$R = \frac{W + L}{1.47v}$$

Where:

R = length of all-red interval, in seconds.

W = total traversed width, from approach stop bar to the far side of no-conflict point, in feet. This may be measured in the field or deduced from maps or construction plans.

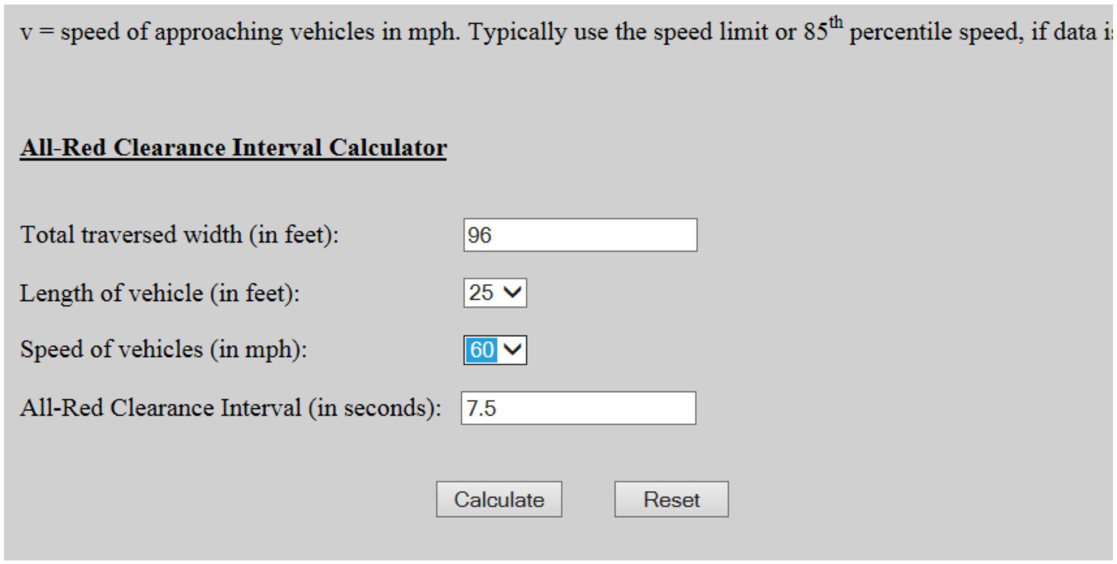
L = length of vehicle, typical of 20 ft.

v = speed of approaching vehicles in mph. Typically use the speed limit or 85th percentile speed, if data is available, whichever is higher.

All-Red Clearance Interval Calculator

Total traversed width (in feet):	
Length of vehicle (in feet):	20
Speed of vehicles (in mph):	45
All-Red Clearance Interval (in seconds):	

Test the form controls.
Change values of the controls.

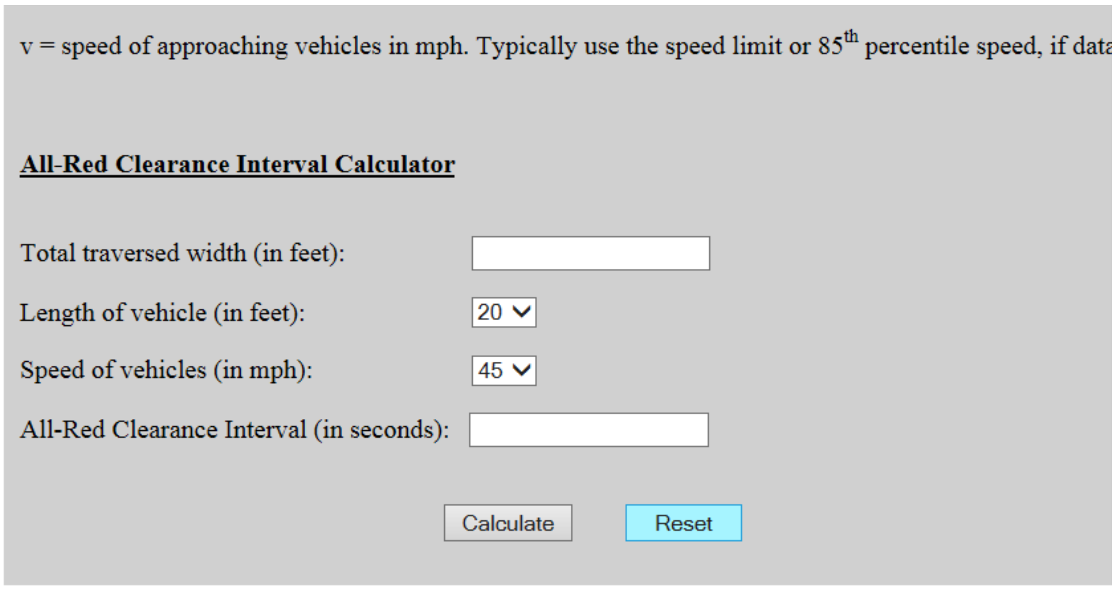


v = speed of approaching vehicles in mph. Typically use the speed limit or 85th percentile speed, if data i

All-Red Clearance Interval Calculator

Total traversed width (in feet):	96
Length of vehicle (in feet):	25
Speed of vehicles (in mph):	60
All-Red Clearance Interval (in seconds):	7.5

Click on the reset button. The forms "clears" and reverts to its original settings.



v = speed of approaching vehicles in mph. Typically use the speed limit or 85th percentile speed, if data

All-Red Clearance Interval Calculator

Total traversed width (in feet):	
Length of vehicle (in feet):	20
Speed of vehicles (in mph):	45
All-Red Clearance Interval (in seconds):	

Code in *JavaScript* shall be added to give functionality to the *Calculate* button later in this course.

You may change formatting settings, styles, etc., in a creative manner to make an even more eye-catching and visually appealing web page. You may add pictures for example. In the white space to the left and right of our main content you may add links to other web pages, reports, videos, slideshows, presentations, weather forecasts, relevant reference materials, etc., etc.

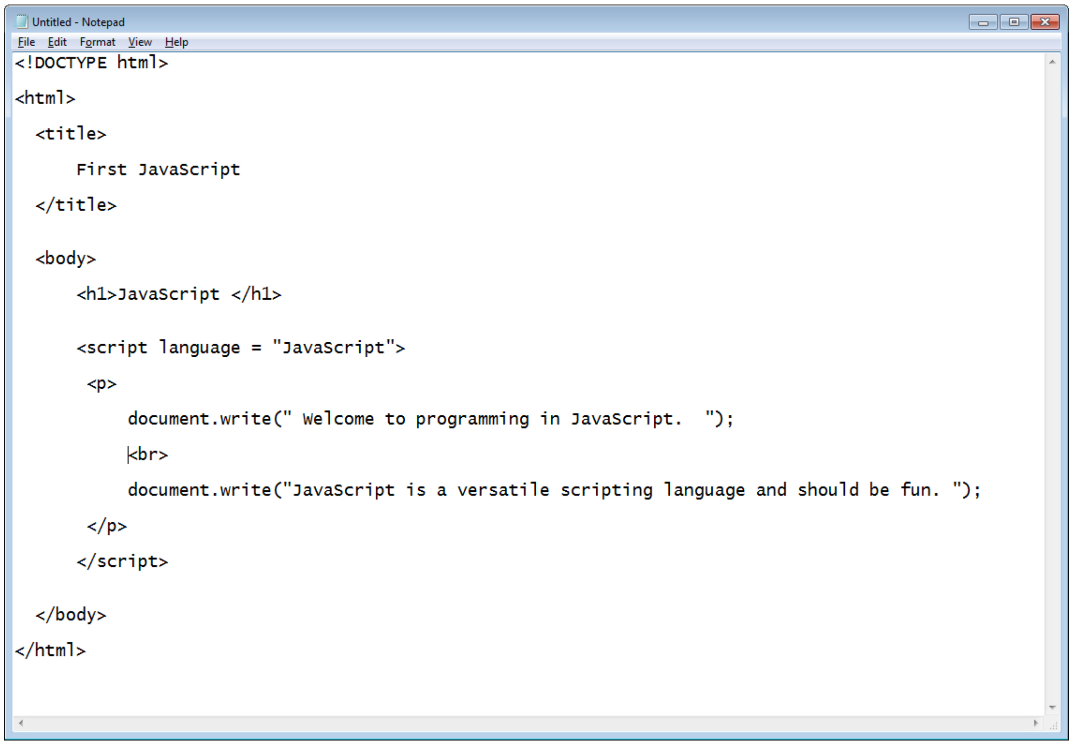
Save your file.
The test has been a success.
You have created your first web application. Congratulations.

3. *JAVASCRIPT* SOURCE CODE

Thus far it has been demonstrated how HTML is used to create and display the content of a web page. The web pages on their own; however, are unable to interact with the user. To provide interactive functionality, forms and form controls are incorporated into the web page. The form controls, however, must have coded instructions written for their applicable events such as clicking on them, changing the selected item of a combo box, clicking on a radio button, etc. The coded instructions (**source code** or **code**) must be written in a scripting language that the web browser (e.g. *Internet Explorer*) will interpret and execute. Source code written in a scripting language is called a **script**. There are many scripting languages in use. In this course the *JavaScript* language shall be presented.

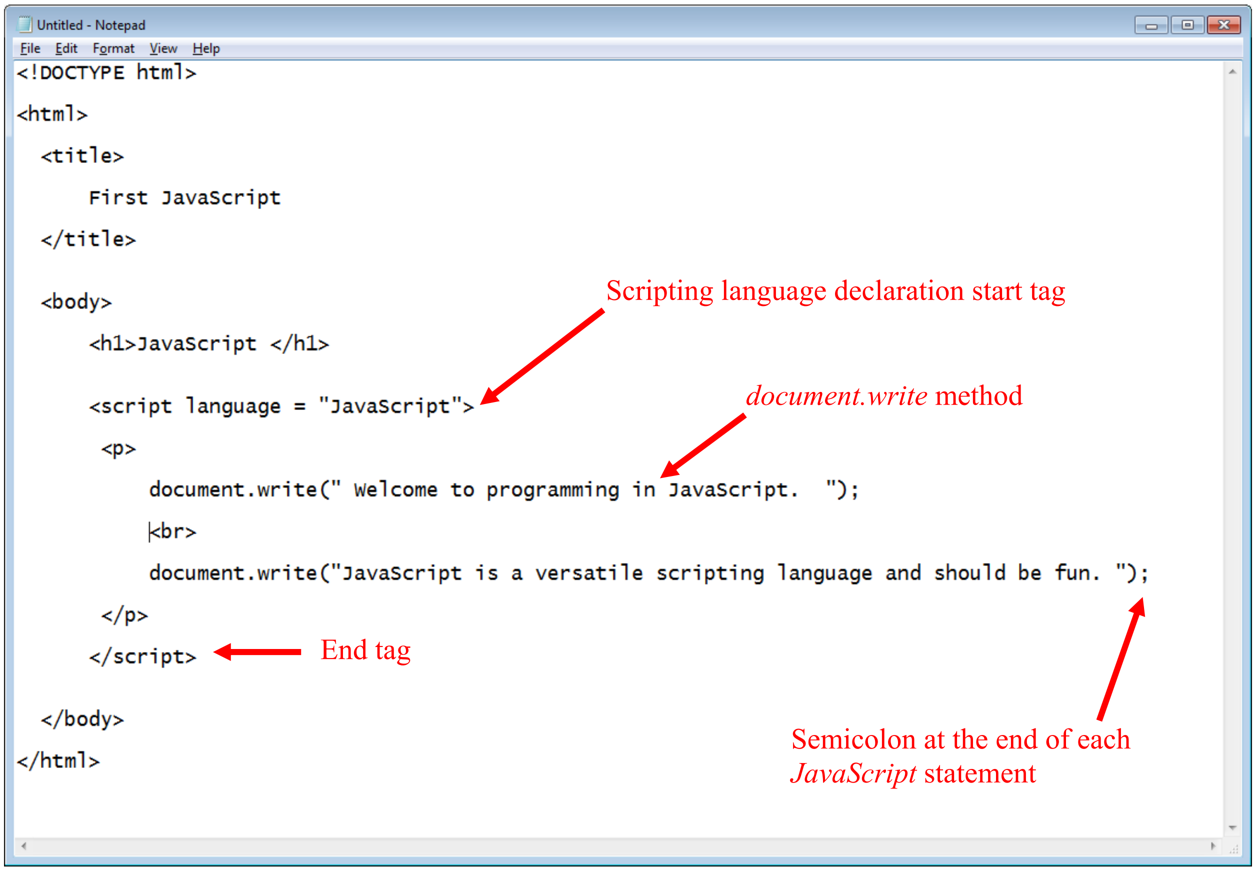
3.1 *JavaScript* and HTML Tags

Open a new session of *Notepad* (or your preferred text editor), and type the following to create a new web page.



```
<!DOCTYPE html>
<html>
  <title>
      First JavaScript
  </title>

  <body>
      <h1>JavaScript </h1>

      <script language = "JavaScript">
       <p>
           document.write(" Welcome to programming in JavaScript.  ");
           <br>
           document.write("JavaScript is a versatile scripting language and should be fun. ");
       </p>
      </script>

  </body>
</html>
```

Study the HTML tags and the *JavaScript* code.


Untitled - Notepad
File Edit Format View Help
Scripting language declaration start tag
document.write method
End tag
Semicolon at the end of each *JavaScript* statement

```
<!DOCTYPE html>
<html>
  <title>
      First JavaScript
  </title>

  <body>
      <h1>JavaScript </h1>

      <script language = "JavaScript">
       <p>
           document.write(" Welcome to programming in JavaScript.  ");
           <br>
           document.write("JavaScript is a versatile scripting language and should be fun. ");
       </p>
      </script>

  </body>
</html>
```

The <*script*> tag declares the beginning of the coding in the scripting language. The *language* attribute is used to declare the specific scripting language being used. *JavaScript* is the default scripting language for *HTML5*, the current standard for HTML. Therefore, if it is the original intent to use *JavaScript*, the language attribute may be omitted. In other words, if the language attribute is not specified *JavaScript* is automatically invoked. Alternately, the *type* attribute may be used for similar purpose as follows:

<script type="text/javascript">

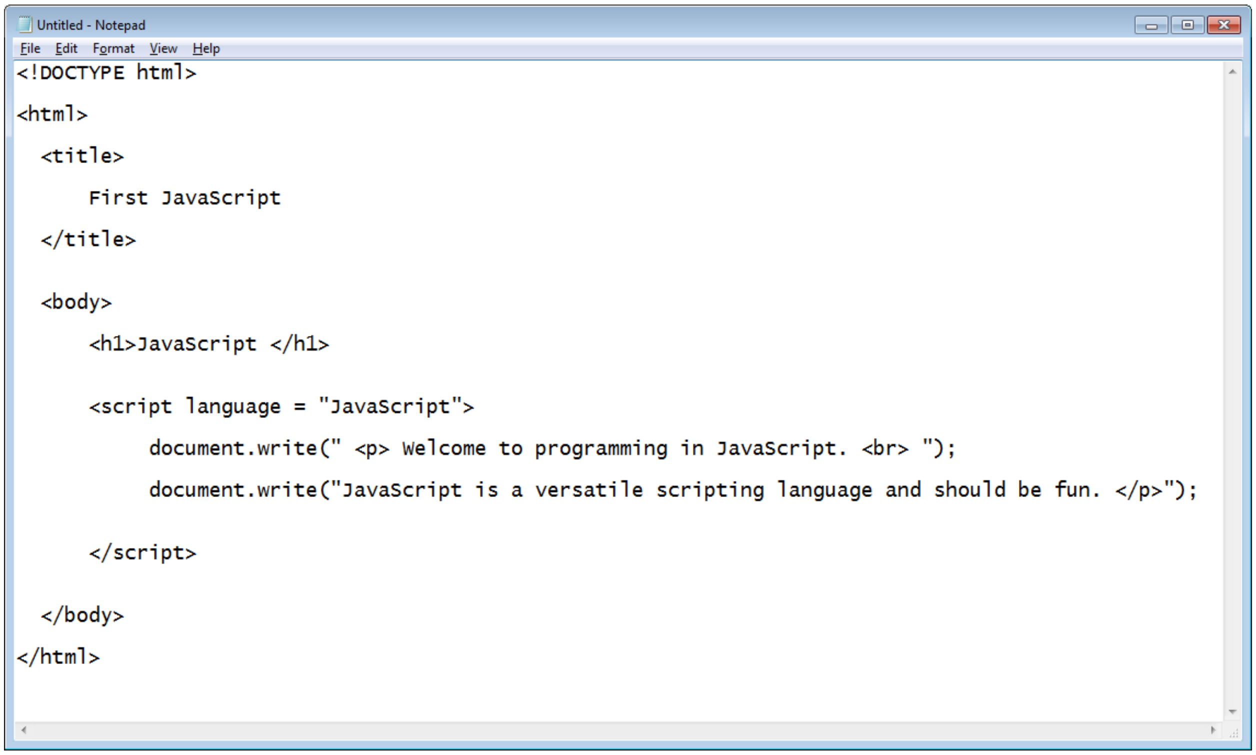
The </*script*> tag declares the end of the script.

All *JavaScript* code must written within the *<script>*... *</script>* tags. A *<script>*... *</script>* must embedded within the *<html>*... *</html>* tags. *JavaScript* code consists of statements which tell the browser what to do. Each *JavaScript* statement must end with a semicolon (;). HTML tags may be intermixed with the *JavaScript* code and embedded in the script area. Note that *JavaScript* is a case-sensitive language.

To facilitate printout, reading, and review of codes, it is a good practice to space out the codes and offset specific blocks of code to distinguish them from other blocks and hence make them readily identifiable. This is called **indenting**. In this example the *<script>* ... *</script>* tags as well as their content have been indented to make them distinct and easily identifiable.

The *document.write* ***method*** sends the input string to the browser to be displayed. HTML tags may be embedded in the string passed to the *document.write* method.

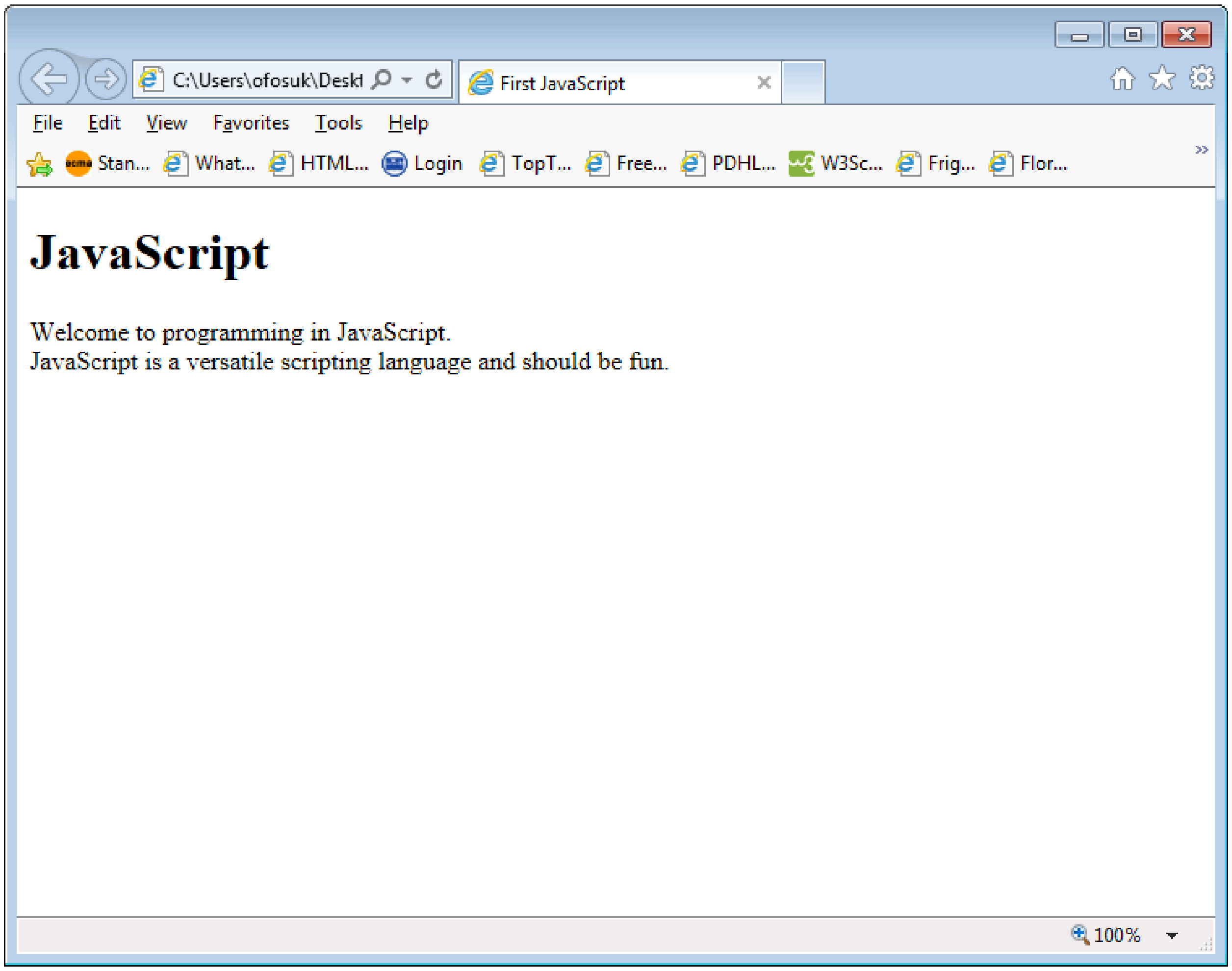
Therefore the paragraph tags and the line break tag may be embedded in the *document.write* statements as follows.



```
<!DOCTYPE html>

<html>

  <title>

      First JavaScript

  </title>


  <body>

      <h1>JavaScript </h1>


      <script language = "JavaScript">

           document.write(" <p> Welcome to programming in JavaScript. <br> ");

           document.write("JavaScript is a versatile scripting language and should be fun. </p>");


      </script>


  </body>

</html>
```

Save your text file as ***MyFirstJavaScript.html***, in your *JavaScript_files* folder on your Desktop.
Close your file.
From its directory location right click on the file ***MyFirstJavaScript.html***.
Select *Open with*, and select your favorite web browser.

The web page opens to display the content per the HTML tags and the JavaScript code.


C:\Users\ofosuk\Deskt
First JavaScript
File Edit View Favorites Tools Help
Stan... What... HTML... Login TopT... Free... PDHL... W3Sc... Frig... Flor...
100%

JavaScript

Welcome to programming in JavaScript.
JavaScript is a versatile scripting language and should be fun.

3.2 Commenting

Comments are added to codes to explain in plain language what the codes are doing. Comments are also useful in enabling third parties to review your codes. In some cases projects may be suspended due to other work and priorities. Comments are vital in the case whereby a new programmer comes in to take over a project, or whereby a project is being developed by multiple teams with each team building some specific aspect of the program. Having good, detailed, and descriptive comments in the codes will enable the programmer to get up to speed on the project in a timely manner. There are no rules as to how much commenting is required. A good rule of thumb is to add as many comments as necessary for a non-expert third party to be able to review your comments and gain a reasonable understanding of what your program is doing.

In *JavaScript* comments are implemented using the "//" at the beginning of a line. Anything typed to the right of the "//" on that line becomes a comment and will be ignored by the web browser and considered to be not part of the code. For example:

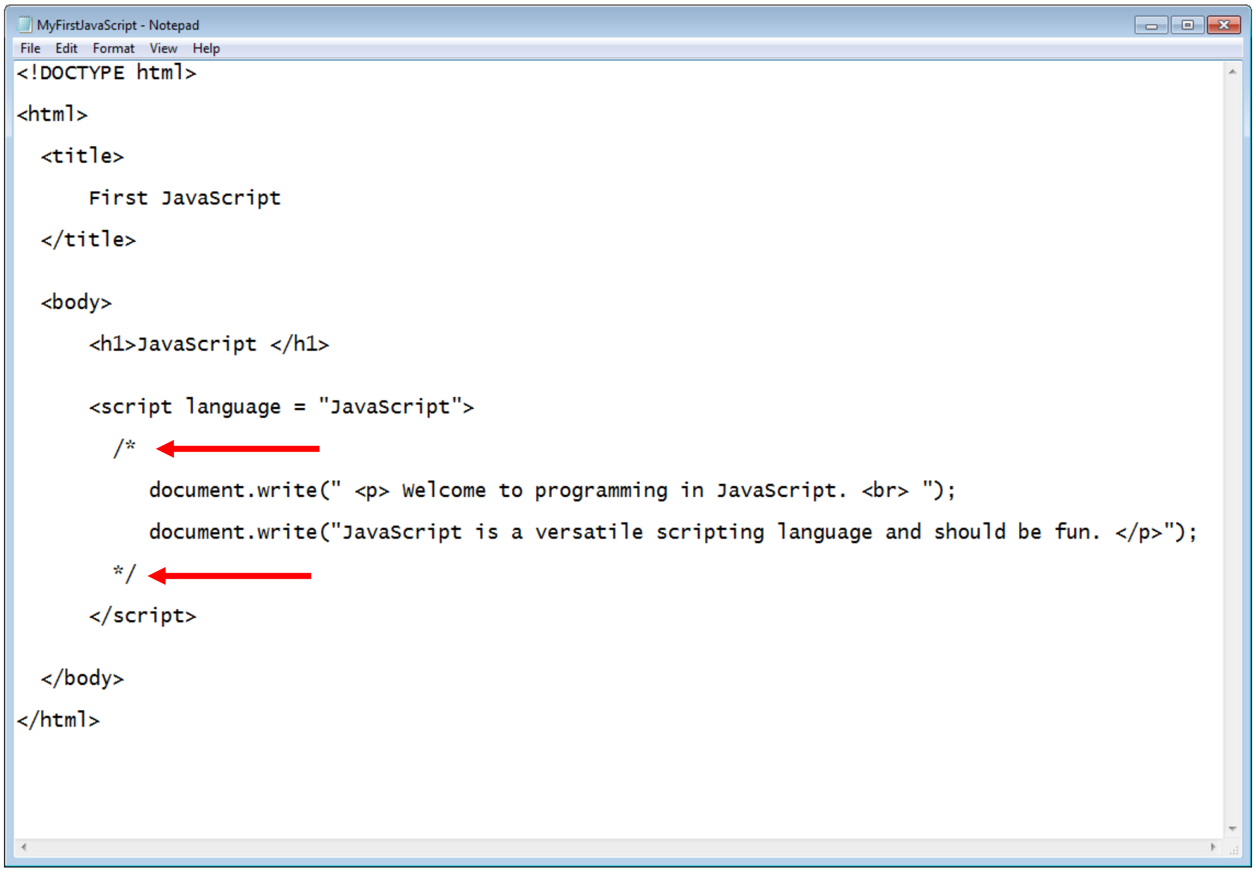
```
<script>
            // introduce yourself to your audience
            document.write("Hello and welcome to our company website. <br>");
            document.write("We are a full service Civil Engineering company");
                        :
</script>
```

The first line within the script is a comment and will be ignored by the browser. The browser will then display the text within the *document.write* statements.

Another commenting technique is to use "/*" to start a comment and "*/" to close the comment. This method may be used to implement a comment over several lines.

Open your file ***MyFirstJavaScript*** in your text editor.

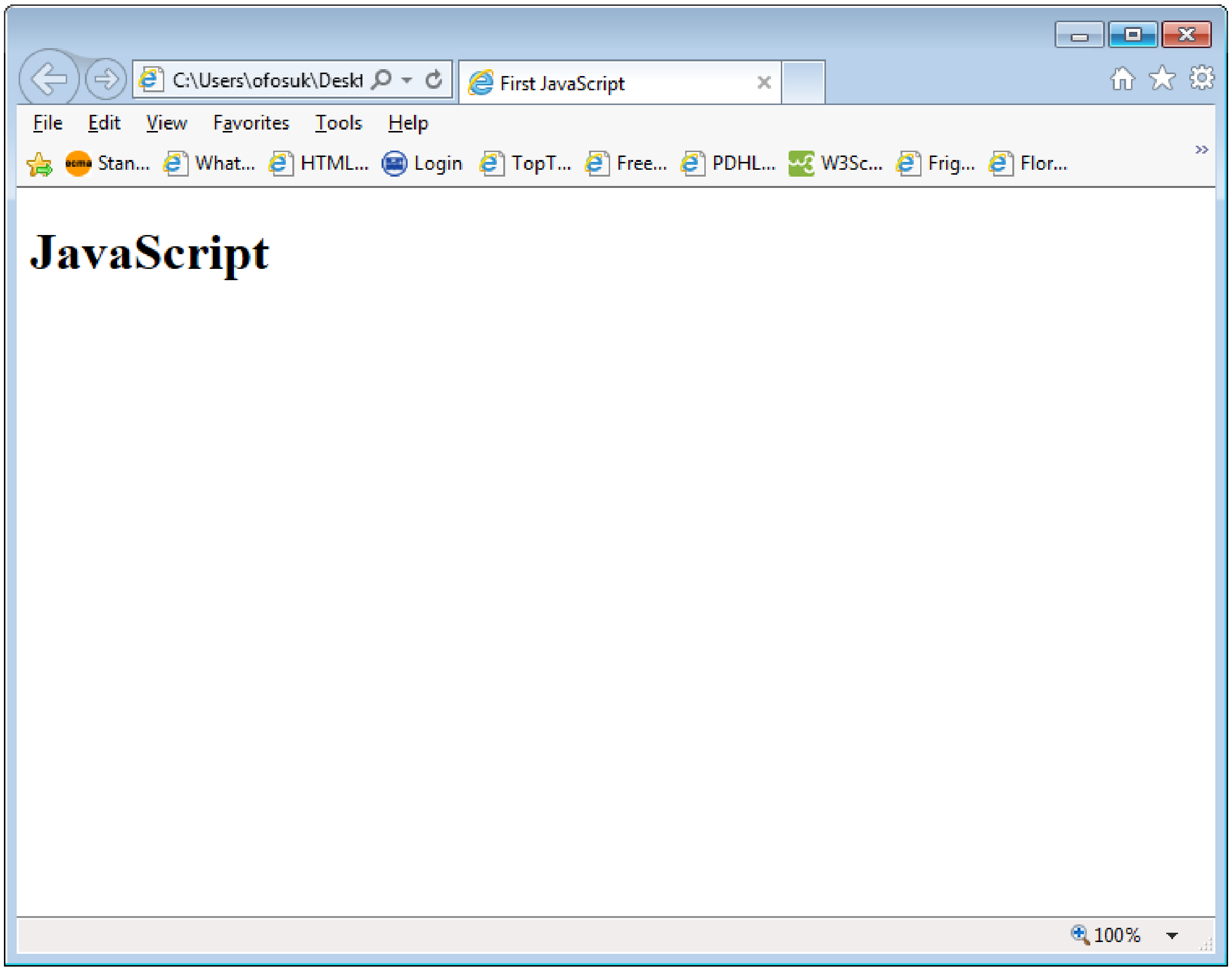
Add comment marks as follows:



```
<!DOCTYPE html>
<html>
  <title>
      First JavaScript
  </title>

  <body>
      <h1>JavaScript </h1>

      <script language = "JavaScript">
        /*
           document.write(" <p> Welcome to programming in JavaScript. <br> ");
           document.write("JavaScript is a versatile scripting language and should be fun. </p>");
        */
      </script>

  </body>
</html>
```

Save and close your file.
From its folder location, double click on your file to open it in your web browser.

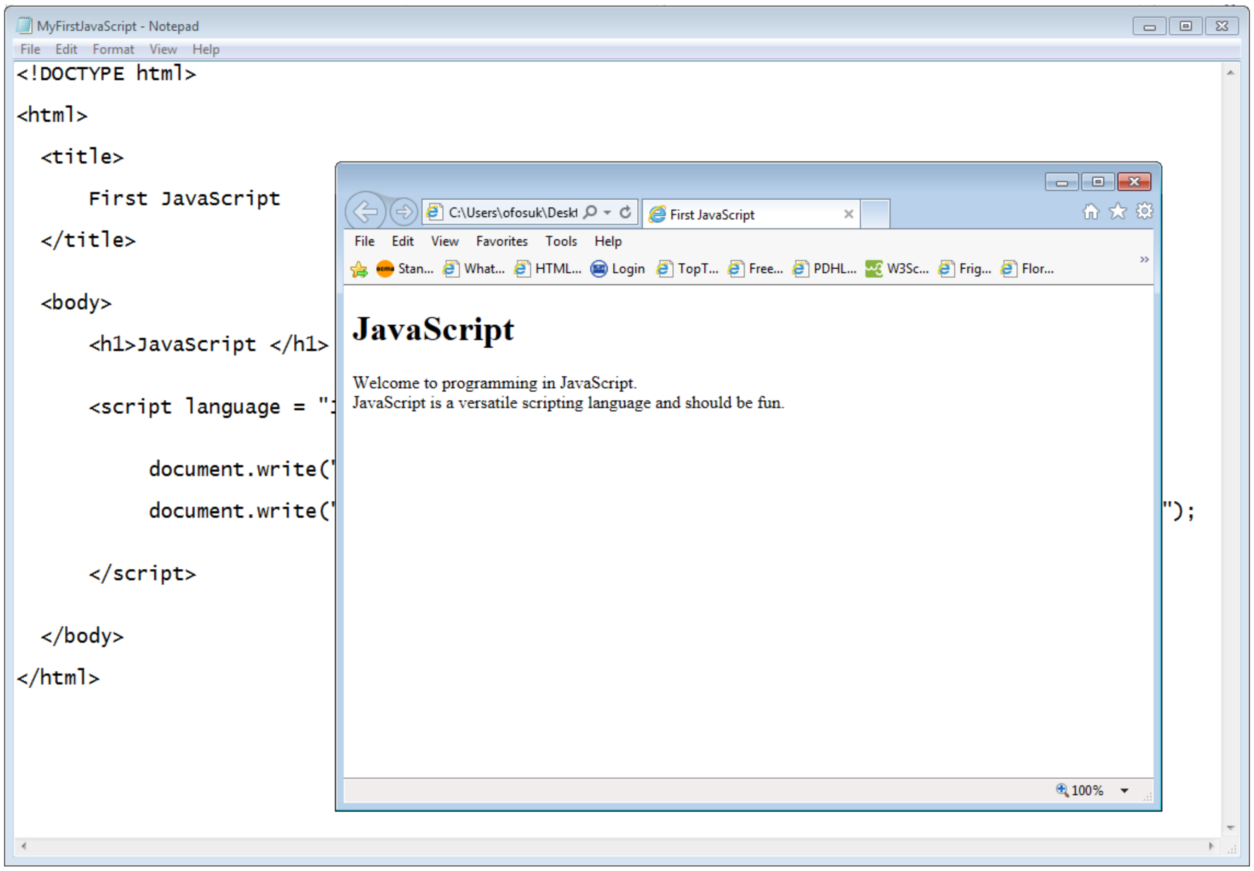
The *document.write* statements have been commented out. They are now part of a multiline comment and are therefore ignored by the browser.


C:\Users\ofosuk\Deskt
First JavaScript
File Edit View Favorites Tools Help
Stan... What... HTML... Login TopT... Free... PDHL... W3Sc... Frig... Flor...
100%

JavaScript

Reopen your file in your text editor.
Delete the comment marks. Save your file.
Refresh your web browser.

The web page content reappears.


MyFirstJavaScript - Notepad
File Edit Format View Help
C:\Users\ofosuk\Deskt
First JavaScript
File Edit View Favorites Tools Help
Stan... What... HTML... Login TopT... Free... PDHL... W3Sc... Frig... Flor...
100%

```
<!DOCTYPE html>
<html>
  <title>
      First JavaScript
  </title>

  <body>
      <h1>JavaScript </h1>

      <script language = "

           document.write('
           document.write('                                                      ");

      </script>

  </body>
</html>
```

JavaScript

```
Welcome to programming in JavaScript.
JavaScript is a versatile scripting language and should be fun.
```

Save your file.
Close your file.

Both commenting techniques may be used within a single HTML document. The programmer may mix and match them as desired to convey the comments as deemed necessary and appropriate.

4. VARIABLES

4.1 Definition

A **variable** is a portion of computer memory that is reserved to store specific data. A variable is created by declaring it, in other words "announcing" to the browser that you are creating a variable. As in algebra, variables represent values and expressions, and facilitate calculations and other operations.

The *var* keyword is used to declare a variable.

```
var Hours;
var HourlyRate;
var Salary;
var LastName;
```

After the variable has been created (declared) it is "empty". In a process called **initialization**, a value is assigned to the variable using the assignment operator, the equal (=) sign. The assignment operator works such that the right hand side is evaluated first, and the result stored at the variable location named on the left hand side of the assignment operator. Variables may be assigned values, expressions or the output of operations involving other values and variables, etc.

```
Hours = 24;
HourlyRate = 18.55;
Salary = Hours * HourlyRate;
LastName = "Johnson";
```

Note that text values must be placed in quotation marks to assign them to the variable. Before the initialization step the value of the variable is **undefined**.

The declaration and initialization may be combined on one line

```
 var Hours = 24;
var HourlyRate = 18.55;
var Salary = Hours * HourlyRate;
var LastName = “Johnson”;
```

More than one variable may be declared on one line with one *var* statement, by separating them with commas.

```
var Hours = 24,  HourlyRate = 18.55 , Salary = Hours * HourlyRate , LastName = “Johnson”;
```

The above declaration can span over multiple lines as long as each line ends with a comma.

```
var Hours = 24,  HourlyRate = 18.55 ,
Salary = Hours * HourlyRate ,
LastName = “Johnson”;
```

4.2 Variable Names

The names chosen for your variables must follow the following rules

1. Must start with a letter, a *$* sign, or and underscore (_)
2. After the first character, any combination of letters, numbers and underscores may be used
3. A variable name cannot contain a space
4. Variable names are case-sensitive

Variable names may be similar to algebraic characters, e.g. *x*, *y*, etc., or they may be descriptive, e.g. *MonthlySalary*, *ProductWeight*, etc. For descriptive variable names that consist of a combination of words, a common convention is to start each word with an upper case letter, e.g. *DateOfBirth*. Another convention is to start the first word with lower case and the subsequent words with upper case. This is called camel-case letters, e.g. *dateOfBirth*. Another convention is to separate the constituent words with underscores, e.g. *date_of_birth*. In general, you are free to design and adopt your own naming convention(s). Naming conventions promote consistency and provide a common platform when an application (or the components of an application) is being developed by multiple programmers or by multiple independent teams.

4.3 Data Types

Data type refers to the type of data stored in a variable, e.g. a number, a text string, a Boolean value, an array, etc. At initialization a value is assigned to the variable. The type of data assigned to the variable, e.g. a number, becomes the data type of the variable. If another value of a different type is assigned to the variable later in the code, e.g. a text string, the variable becomes that data type. In other words, in *JavaScript*, unlike in many other languages, the data type of a variable can change within the application based on the context. This is a programming concept known as **loose typing**, or **dynamic typing**.

Data types in *JavaScript* are classified into primary (or primitive) data types, composite data types, and special data types.

4.3.1 Number Data Type

This type is a primary data type in *JavaScript*. It consists of positive and negative integers and decimal values. *JavaScript* is an **untyped language**, therefore unlike other languages, *JavaScript* does not define different types of numbers such *Integer*, *Long*, *Double*, *Currency*, etc. Numbers may be represented as exponents using scientific notation. Numbers in hexadecimal and octal number notation can also be used.

Examples:

```
var myConversionFactor = 0.6699 ; //decimal value
var ProjectCost = 2265240.68 ;
var ProjectDuration = 635; //integer
var myLargeNumber = 1.69e3 ;  // this is 1.69*10^3
```

If a calculation results in a number beyond the range of number values provided by *JavaScript*, *Infinity* will be returned. For example, dividing a number zero will yield *Infinity*.

If a numerical operation results in a value that is not numerical, *JavaScript* will return the special reserved word *NaN*, meaning “Not a Number”

4.3.2 String Data Type

This is a *JavaScript* primary data type. A string is a sequence of characters delimited by double quotation marks. Strings are used to store and manipulate text.

```
var myFirstName = “Mike” ;
var myLastName = “Williams” ;
var myFullName = myFirstName + “” + myLastName ;
```

JavaScript also provides special characters that can be added to strings to create characters that cannot be typed directly. For example:

```
document.write (“My name is Jack Brown. \n I am a licensed P.E.”) ;
```

The “\n” special character creates a line break and sends the cursor to the next line to continue. In the above example, the second sentence will be displayed on the next line. A list of special characters in *JavaScript* can be found at other sources (Microsoft, 2014b).

4.3.3 Boolean Data Type

This is a *JavaScript* primary data type. A Boolean type has a “true” or “false” value. Boolean variables are commonly used to control the flow of the code of an application.

```
var confirmationCheckVariable = true ;
```

Note that in the assignment of the Boolean value double quotation marks are not used.

4.3.4 Null Data Type

This is a *JavaScript* special data type. It has only one value, *null*. The contents of an existing variable can be erased by assigning *null* to it.

```
var confirmationCheckVariable = null ;  //the variable is now "empty"
```

4.3.5 Undefined Data Type

This is a *JavaScript* special data type. A declared variable that has never had a value assigned to it has an *undefined* value.

```
 var unitPrice  = 6.25 ;
var Quantity ;
var subtotal = unitPrice * Quantity ; /*an error will occur here as the Quantity
                                          value is undefined*/
```

4.3.6 Arrays

An array is a composite data type in *JavaScript*. An array is a variable that holds multiple values. The individual values are called the **elements** of the array. Each element is identified by its unique address in the array called its **index**. In *JavaScript*, the first element is of index zero [0]. The elements of an array may be of different data types. In large and complex applications, arrays drastically reduce the number of variables needed.

An array is created as follows,

```
var NameOfArray = new Array( ) ;

          NameOfArray[0] = firstelement ;
          NameOfArray[1] = secondelement ;
          NameOfArray[2] = thirdelement ;
            :
            :
          NameOfArray[n] = nthelement ;
```

Alternately an array may be declared as follows,

```
var NameOfArray = new Array(firstelement, secondelement,   , nthelement ) ;
```

or

```
var NameOfArray = [firstelement, secondelement,   , nthelement ] ;
```

Example,

```
var ProjectTeam = new Array( ) ;
          ProjectTeam[0] = “Project Manager” ;
          ProjectTeam[1] = “Design Engineer” ;
          ProjectTeam[2] = “Engineering Assistant” ;
          ProjectTeam[3] = “Drafter” ;
```

Which yields the array

$$\begin{bmatrix} Project\ Manager \\ Design\ Engineer \\ Engineering\ Assistant \\ Drafter \end{bmatrix}$$

4.3.7 Objects

An object is a composite data type in *JavaScript*. An object is data that has properties and methods stored with it. The properties are values that describe the object whereas the methods are actions that the object can perform. *JavaScript* objects have built-in methods.

Consider a computer. The properties of the computer would include type, brand, color, monitor, operating system, processor, etc. The methods would include display, copy, save, delete, print, reboot, etc.

An object is created as follows

```
var NameOfObject = new Object( ) ;

            NameOfObject.firstproperty = firstpropertyvalue ;
            NameOfObject.secondproperty = secondpropertyvalue ;
            NameOfObject.thirdproperty = thirdpropertyvalue ;
               :
               :
            NameOfObject.nthproperty = nthpropertyvalue ;
```

To declare our computer object,

```
var ourComputer = new Object( ) ;
            ourComputer.type = "desktop" ;
            ourComputer.brand = "Dell" ;
            ourComputer.color = "grey" ;
            ourComputer.monitor = "20-inch LCD" ;
```

ourComputer.operatingsystem = "Windows7" ;
ourComputer.processor = "Intel 7Series" ;

To **call** an object method, the syntax is as follows,

NameOfObject.MethodName ( ) ;

For example,

ourComputer.Reboot ( ) ;

All of the data types we have learned thus far can in one way or the other be modelled as objects.

4.4 Operators

JavaScript has a full range of operators for arithmetic computations and as well as assignment operators.

Arithmetic operators are used to perform computations between variables and values. The arithmetic operators are summarized in Table 4.1.

Assignment operators are used to assign values to variables. The *JavaScript* assignment operators are summarized in Table 4.2.

Table 1: Arithmetic operators

Operator	Description	Example	Result

+: Addition	Adds values/ variables together	*x = 6;*
		y = 5;
		z = x + y ;
		document.write (z) ;
			11

- : Subtraction	Subtracts a value/ variable from another	*x = 12;*
		y = 5;
		z = x - y ;
		document.write (z) ;
			7

-: Negation	Negates a value/ variable	*z = 6 ;* *z = - z ;* *document.write (z) ;*	-6

*: Multiplication	Multiplies values/ variables	*x = 6;*
		*document.write (7*x);*
			42

/ : Division	Divides a value/ variable by another	*x = 40;*
		y = 5;
		document.write (x/y) ;
			8

%: Modulo	Remainder of division of a value/ variable by another	*x = 27;*
		y = 5;
		z = x % y ;
		document.write (z) ;
			2

++: Increment	Increases a variable by the increment amount	*x = 2 ;* *x = x++ ;* *document.write (x) ;* *x = ++4 ;* *document.write (x)*	3 6
++: Decrement	Decreases a variable by the decrement amount	*x = 9 ;* *x = x- - ;* *document.write (x) ;* *x = - -2 ;* *document.write (x);*	8 7

Table 2: Assignment operators

Operator	Example	Equivalent to:	Result
=	*z = 6;* *document.write (z) ;*		6
+ =	*y = 2 ;* *y += 5;* *document.write (y) ;*	*y = 2 ;* *y = y + 5;* *document.write (y) ;*	7
-=	*y = 5 ;* *y -= 2;* *document.write (y) ;*	*y = 5 ;* *y = y - 2 ;* *document.write (y) ;*	3
*=	*y = 5 ;* *y *= 2;* *document.write (y) ;*	*y = 5 ;* *y = y * 2 ;* *document.write (y) ;*	10
/ =	*y = 20 ;* *y /= 5;* *document.write (y) ;*	*y = 20 ;* *y = y / 5 ;* *document.write (y) ;*	4
% =	*x = 18 ;* *x %= 5;* *document.write (x) ;*	*x = 18 ;* *x = x % 5 ;* *document.write (x) ;*	3

Note that the addition operator can be used for number and string **operands** (inputs) and combinations thereof. For example,

```
var string1 = “Project”, string2 = “Engineer”, ProjectNumber = 12 ;

document.write(string1 + “ “ + string2) ;
document.write(string1 + ProjectNumber) ;
```

which yield “Project Engineer” and “Project12” respectively.

4.5 Order of Precedence

The order of precedence determines the order in which arithmetic operations will be performed in a mathematical expression. The **order of precedence** for arithmetic operations is as follows:

1st.	Exponentiation
2nd.	Multiplication and division
3rd.	Addition and subtraction

Operations of the same level are performed from left to right.

Example: The expression $2+3*6^2$

```
document.write(  2+3*6^2  ) ;
```

Under the order of precedence the exponentiation will be conducted first.
Next, the result of the exponentiation will be multiplied by the adjacent value.
Third, the addition will be performed. The result therefore will be 110.

If the above result was not the original intent, but rather to conduct the addition first, then **parenthesis** may be inserted appropriately to control the calculation. The expression within parenthesis is always evaluated first and its value is then used in the remainder of the calculation. For example,

```
document.write(  (2+3)*6^2  ) ;
```

This yields the result 180.

It must be noted that *JavaScript* does not support implied algebraic operations. For example, the expression 3(X-Y) will not be syntactically valid. It will have to be typed as 3*(X-Y) where X and Y are variables that have been declared earlier in the code.

5. FUNCTIONS

5.1 Definition

A **function** is a block of code containing a sequence of instructions that will be executed when called. A function may be **called** from any point in the *JavaScript* code associated with a web page. Functions are commonly called when an event associated with a form control occurs, for example clicking on a button can fire the *onclick* event which calls some function to perform some tasks or run some calculation(s). Functions therefore are a means by which form controls provide the functionality of interactive web pages.

5.2 Function Declaration

The syntax for declaring a function is:

```
function NameOfFunction ( )
          {
            :
            Block of code to be executed
            :
          }
```

The curly brackets (*{ }*)define the start and end of the block of code that will be run when the function is called.

In some cases it is necessary to supply the function with input values which will be needed to execute some command in the function code. These inputs are called **arguments** and are said to be **passed to** the function. A function declaration with arguments is as follows:

```
function NameOfFunction (argument1, argument2,   , argumentn )
            {
               :
               Block of code that needs the arguments for successful execution
               :
            }
```

The arguments must be listed in the order in which they will be encountered in the code block.

Example:

```
/*this function calculates wages based on the Hours and Rate of an employee passed into the
function*/
 function calcWages (37,  24.75 )
            {
              document.write( Hours * Rate) ;
            }
```

The first variable encountered in the function code block takes the value of the first argument, and so on and so forth. An argument may be a variable that has been created and manipulated elsewhere in the application.

5.3 Function Names

Function names must obey the following rules.

1. Must start with a letter or an underscore (_)
2. After the first character, any combination of letters, numbers and underscores may be used
3. No special characters other than the underscore are permitted
4. A function name cannot contain a space
5. Function names are case-sensitive

Naming conventions for functions follow the guidelines for naming variables.

5.4 Returning a Value

In some cases the intent is for the function to run some calculation and return a value. The value will be returned at the location where the function was called. A *return* statement must be added to the function code block for it to return a value. The syntax is of the form

```
 function calcWages (37,  24.75 )
              {
              return  Hours * Rate ;
              }
```

The value of the calculation will be sent back to the location where the function was called. Also, and very importantly, upon executing the *return* statement, the function will terminate. The *return* keyword terminates the execution of the function. The return statement can therefore be used to exit out of a function. For example,

```
 function calcWages (37,  24.75 )
              {
                document.write( Hours * Rate) ;
                 return ;
              }
```

5.5 Scope of Variables

The **scope** of a variable refers to its “lifetime” or where the variable is “alive”. A variable declared within a function only “lives” while that function is being executed. Once the function completes execution and exits, the variable ceases to exist. Due to the fact that this variable only exists within the function and only “lives” while the function is active, this kind of variable is referred to as a **local variable**.

A variable declared outside of a function is available to, and can be used by or be manipulated by, all functions and all scripts associated with the web page. Such a variable is referred to as a **global variable**. A global variable "dies" only when the web page is closed.

It is important to note that if a variable that has not been declared is assigned a value, that variable becomes a global variable regardless of whether the assignment took place within a function or not.

5.6 Built-in Functions

To facilitate programming, *JavaScript* has some functions that are built into modern web browsers and can be used as needed in the development of web-based applications. A limited selection of *JavaScript* built-in functions is presented in Table 4.3.

Table 3: *JavaScript* built-in functions

Function	**Description**	**Example**	**Result**
Math.random( )	Returns a random number between 0 and 1	Math.random( ) ;	e.g. 0.8596
Math.max( )	-Returns the maximum of a number of arguments	*y = 2 ;* *x = 5;* *z =max(8,x,y) ;*	8
Math.round( )	Rounds the argument to the nearest integer	*y = 5.65 ;* *x = round(y) ;*	6
Math.log( )	Returns the logarithm of an argument	*y = 5 ;* *y = Math.log(y) ;*	0.6989
Math.sqrt( )	Returns the square root of an argument	*x = 100 ;* *y = Math.sqrt(x) ;*	10
Math.sin ( )	Returns the sine of an argument (in radians)	*x = Math.sin(Math.PI/4) ;*	0.8509
Math.PI	Returns the value π	*Math.PI ;*	3.142

Table 3 (Continued): *JavaScript* built-in functions

Function	**Description**	**Example**	**Result**
Math.E	*Returns Euler's constant, 2.718*	*P = Math.E ;*	2.718
document.getElementById( )	*Access an HTML element by its ID attribute*	User enters "5" in text box with *id="Txt" getElementById("Txt").value;*	5
document.getElementByName()	*Access an HTML element by its Name attribute*	User enters "17" in text box with *name="Salary" getElementById("Salary").value;*	17
parseInt( )	*Parses a string and returns an integer*	*parseInt("004") ;*	4
parseFloat( )	*Parses a string and returns floating point number (decimal)*	*parseFloat("11.33 feet") ;*	11.33
String ( )	*Converts a number to a string*	*x= "00" +String(6) ;*	006
eval( )	*Evaluates an expression in a string*	*var x = 5 , y = 8;* *var z = eval("x*y") ;*	40
toLowerCase( )	*Converts a string to all lower case*	*q = toLowerCase("BjO") ;*	bjo
concat( )	*Concatenates strings*	*x = "hi" ;* *y = " " ;* *z = "there" ;* *p = x.concat(y, z) ;*	hi there

Table 3 (Continued): ***JavaScript*** **built-in functions**

Function	**Description**	**Example**	**Result**
getDate( )	*Returns the day of the month*	*//today is 4/11/14* *x = getDate( ) ;*	11
getDay( )	*Returns the current day*	*//today is 4/11/14* *y = getDay( ) ;*	Friday

reverse( )	*Reverses the order of the elements in an array*	*var x1 = new Array( ) ;*
		x1[0] = 5 ;
		x1[1] = 9 ;
		reverse(x1) ;
			x1[0] = 9
			x1[1] = 5

sort( )	*Sorts the elements of an array in numerical order*	var y = [5, 9, 3] ; x = sort(y) ;	[3, 5, 9]
split( )	-*Splits a string into an array of substrings based on a separator*	*x= "Hi there, Sir" ;* *p = split(x, ",") ;*	Hi there Sir
splice( )	*Adds / removes elements of an array based on position in array and number of elements to be removed/added*	var *y = ["5"," "3, "2"] ;* *x = splice(y, 2, 0, "4", "9");*	[5,3,4,9,2]

Other built-in functions can be found at various sources (ECMA International, 2011), (Microsoft, 2014a).

5.7 Practicum #2

In this exercise the web page developed in Practicum #1 will be updated using functions to give functionality to the buttons and in the process creating a fully functional interactive web page.

Make a copy of your Practicum #1 file and name it ***Practicum #2.html***. You may save it to a new directory or continue working in your original directory. If you choose to work in a new directory you must also copy the cascading styles sheet ***(.css)*** file along with your new Practicum file.

Review the formula for the calculation of the yellow interval duration.
Create script tags. In the script tag content declare a function (using a function name of your choice that meets all function name rules) that will perform the calculations.

Within the function, declare variables for each input in the formula. Use intuitive/ descriptive variable names that meet all variable naming rules.
Add comments as necessary to inform a third party what the function is doing.

```
<script type="text/javascript">
 function fnYellowCalc ( )
          {
            //declare variables needed for the calculation
            var reacTime, speed, decel, gradient ;
          }

</script>
```

Assign values to the variables. The values will be coming from the relevant controls on the form. Use the *id* attribute of a control to assign the control's value to the variable.

```
<script type="text/javascript">
 function fnYellowCalc ( )
          {
            //declare variables needed for the calculation
            var reacTime, speed, decel, gradient ;

            /*now assign the controls' values to the variables, identifying each
            control by its unique id */
            reacTime = document.getElementById("txtReacTime").value;
            speed = document.getElementById("optSpeed").value;
            decel = document.getElementById("txtDecel").value;
            gradient = document.getElementById("txtGrade").value;
          }
</script>
```

Next, convert all variables to numerical types to ensure the mathematical calculations are conducted accordingly.

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

 function fnYellowCalc ( )

            {
               //declare variables needed for the calculation
               var reacTime, speed, decel, gradient ;

               /*now assign the controls' values to the variables, identifying each
               control by its unique id */
               reacTime = document.getElementById("txtReacTime").value;
               speed = document.getElementById("optSpeed").value;
               decel = document.getElementById("txtDecel").value;
               gradient = document.getElementById("txtGrade").value;

            //now ensure variable values are converted to numbers for calcs
               reacTime = parseFloat(reacTime);
               speed = parseFloat(speed);
               decel = parseFloat(decel);
               gradient = parseFloat(gradient);

            }

</script>
```

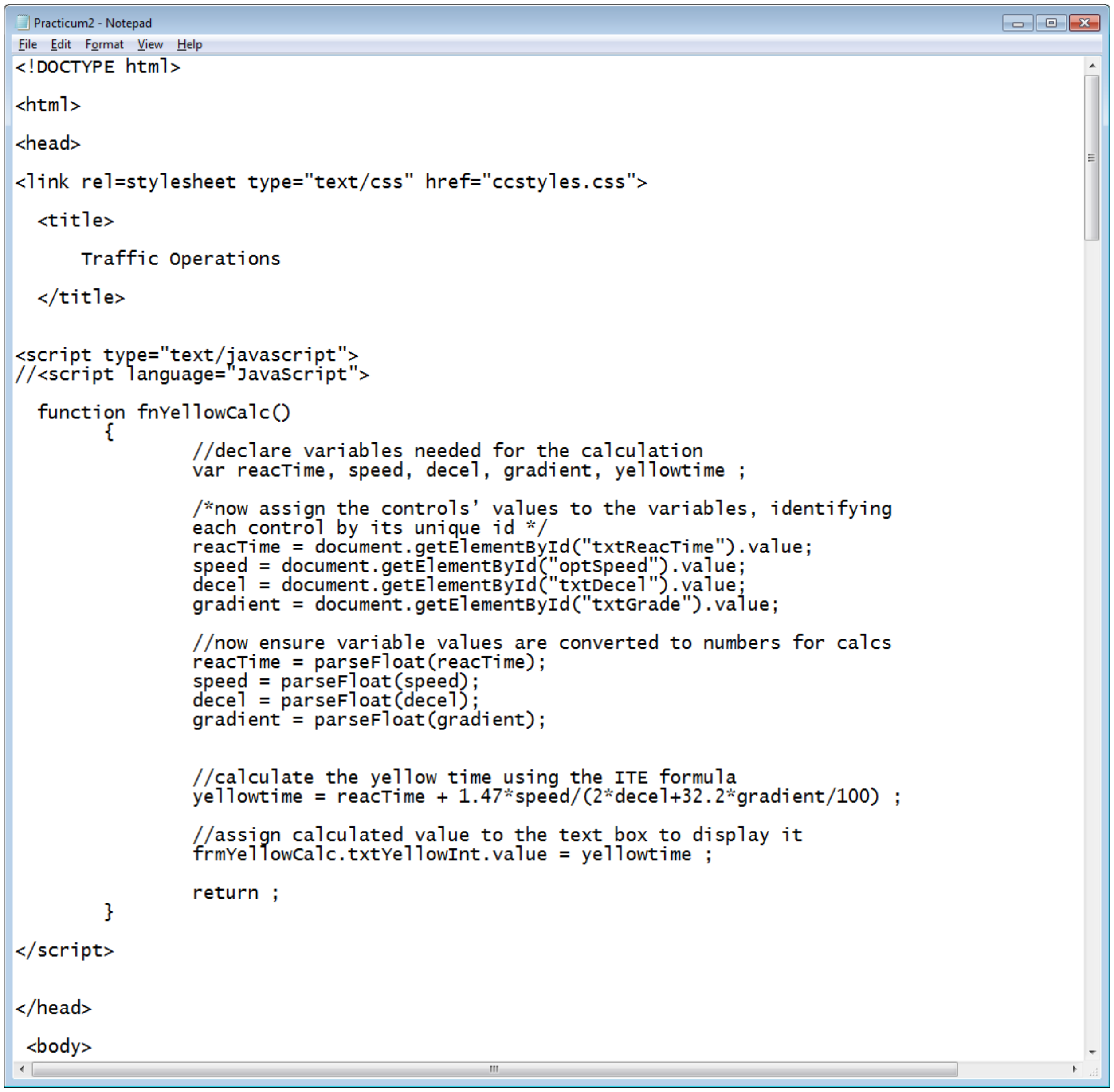
With all variables assigned and their values converted to floating point numbers, we can now invoke the yellow time formula to calculate a value. Once calculated the value/ variable needs to be assigned to the form control that will display it.
Add the *return* keyword to terminate the function.

```
<script type="text/javascript">
 function fnYellowCalc ( )
            {
              //declare variables needed for the calculation
              var reacTime, speed, decel, gradient, yellowtime ;

              /*now assign the controls' values to the variables, identifying each
              control by its unique id */
              reacTime = document.getElementById("txtReacTime").value;
              speed = document.getElementById("optSpeed").value;
              decel = document.getElementById("txtDecel").value;
              gradient = document.getElementById("txtGrade").value;

            //now ensure variable values are converted to numbers for calcs
              reacTime = parseFloat(reacTime);
              speed = parseFloat(speed);
              decel = parseFloat(decel);
              gradient = parseFloat(gradient);

              //calculate the yellow time using the ITE formula
              yellowtime = reacTime + 1.47*speed / (2*decel +32.2*gradient / 100) ;

              //assign calculated value to the text box to display it
              frmYellowCalc.txtYellowInt.value = yellowtime ;

              return ;
            }
</script>
```

This function will be called anytime a user clicks on the *Calculate* button for the yellow time duration. Therefore, an event needs to be added to that *Calculate* button's properties that will invoke the function anytime a user clicks on the button. The code (typed on one line) is:

```
<button type = "button"  name = "calcYellowTime"
                                        onclick = "fnYellowCalc ( )">Calculate</button>
```

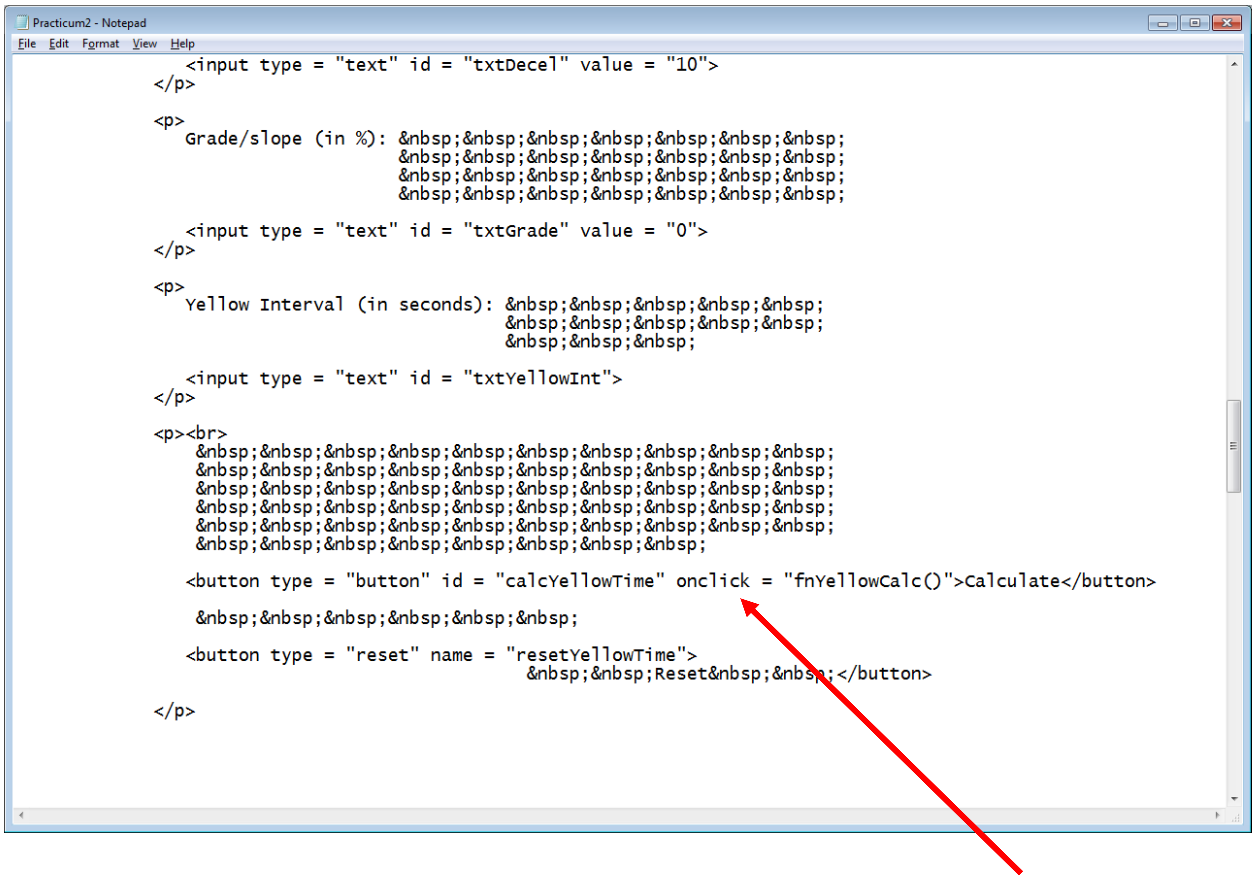
Implement the function codes in your document.
Embed the script in the *<head>* *</head>* tags.



```
<!DOCTYPE html>

<html>

<head>

<link rel=stylesheet type="text/css" href="ccstyles.css">

  <title>

      Traffic Operations

  </title>


<script type="text/javascript">
//<script language="JavaScript">

  function fnYellowCalc()
        {
                //declare variables needed for the calculation
                var reacTime, speed, decel, gradient, yellowtime ;

                /*now assign the controls' values to the variables, identifying
                each control by its unique id */
                reacTime = document.getElementById("txtReacTime").value;
                speed = document.getElementById("optSpeed").value;
                decel = document.getElementById("txtDecel").value;
                gradient = document.getElementById("txtGrade").value;

                //now ensure variable values are converted to numbers for calcs
                reacTime = parseFloat(reacTime);
                speed = parseFloat(speed);
                decel = parseFloat(decel);
                gradient = parseFloat(gradient);


                //calculate the yellow time using the ITE formula
                yellowtime = reacTime + 1.47*speed/(2*decel+32.2*gradient/100) ;

                //assign calculated value to the text box to display it
                frmYellowCalc.txtYellowInt.value = yellowtime ;

                return ;
        }

</script>


</head>

 <body>
```

The scripts may be placed anywhere in the HTML document. It is common practice, however, to place them in the *<head>...</head>* as a means of separating the scripts from the content of the main body of the web page.

Insert the *onclick* event value for the *Calculate* button.

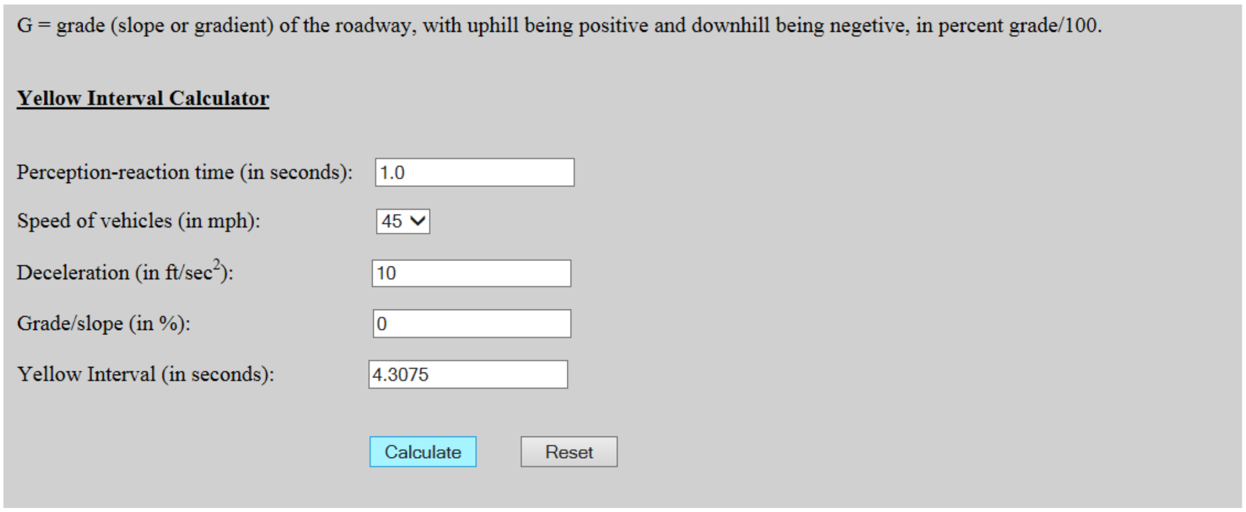


Practicum2 - Notepad

```
        <input type = "text" id = "txtDecel" value = "10">
      </p>

      <p>
        Grade/slope (in %): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

        <input type = "text" id = "txtGrade" value = "0">
      </p>

      <p>
        Yellow Interval (in seconds): &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                                      &nbsp;&nbsp;&nbsp;

        <input type = "text" id = "txtYellowInt">
      </p>

      <p><br>
         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

        <button type = "button" id = "calcYellowTime" onclick = "fnYellowCalc()">Calculate</button>

         &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

        <button type = "reset" name = "resetYellowTime">
                                &nbsp;&nbsp;Reset&nbsp;&nbsp;</button>

      </p>
```

Save your file.
Test your file in the web browser.

Using the default values, click on the *Calculate* button.
A yellow time is calculated and displayed.



G = grade (slope or gradient) of the roadway, with uphill being positive and downhill being negetive, in percent grade/100.

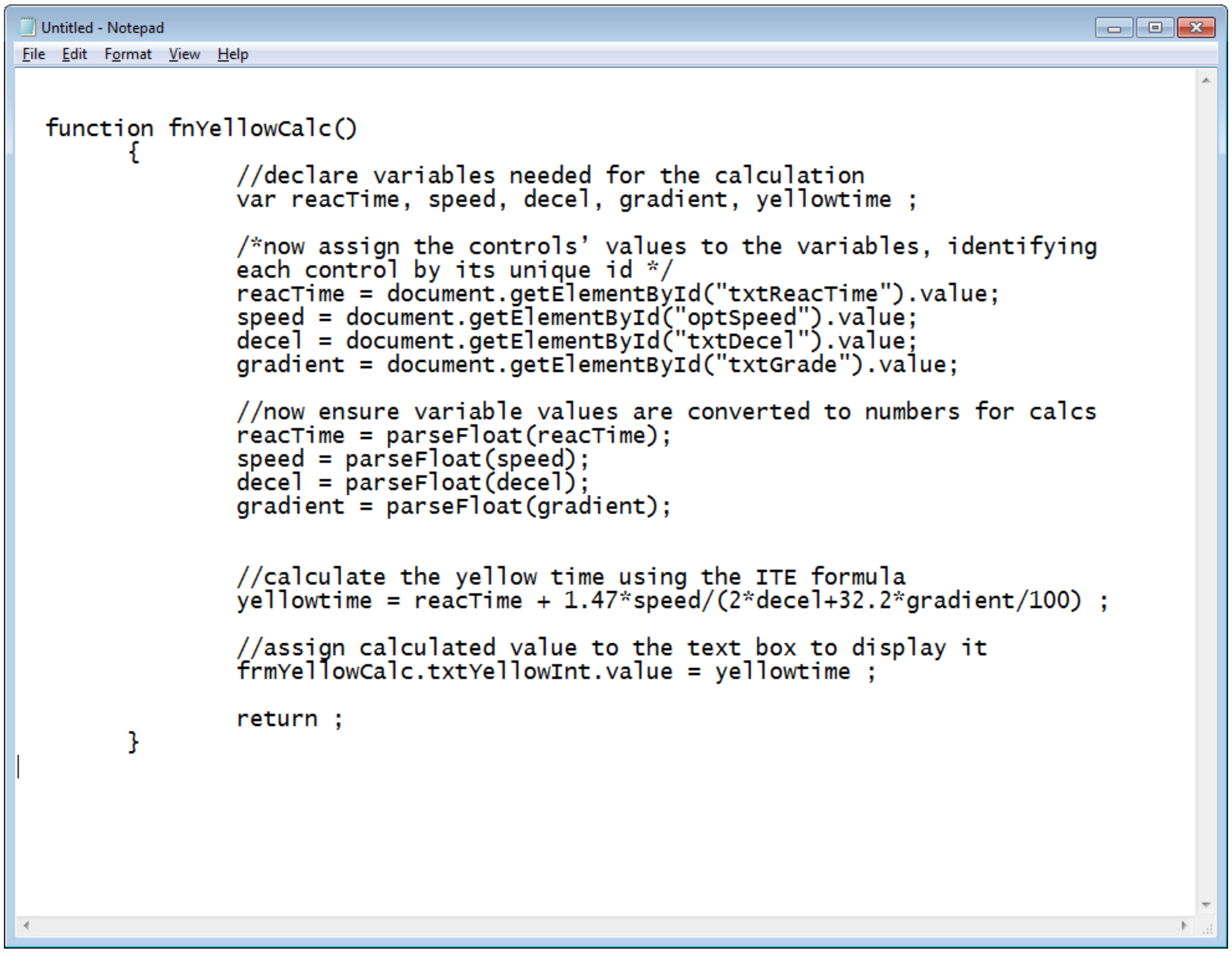
Yellow Interval Calculator

Perception-reaction time (in seconds):	1.0
Speed of vehicles (in mph):	45
Deceleration (in ft/sec^2):	10
Grade/slope (in %):	0
Yellow Interval (in seconds):	4.3075

Conduct a manual calculation to confirm your result.
Save your file.
The test is a success.

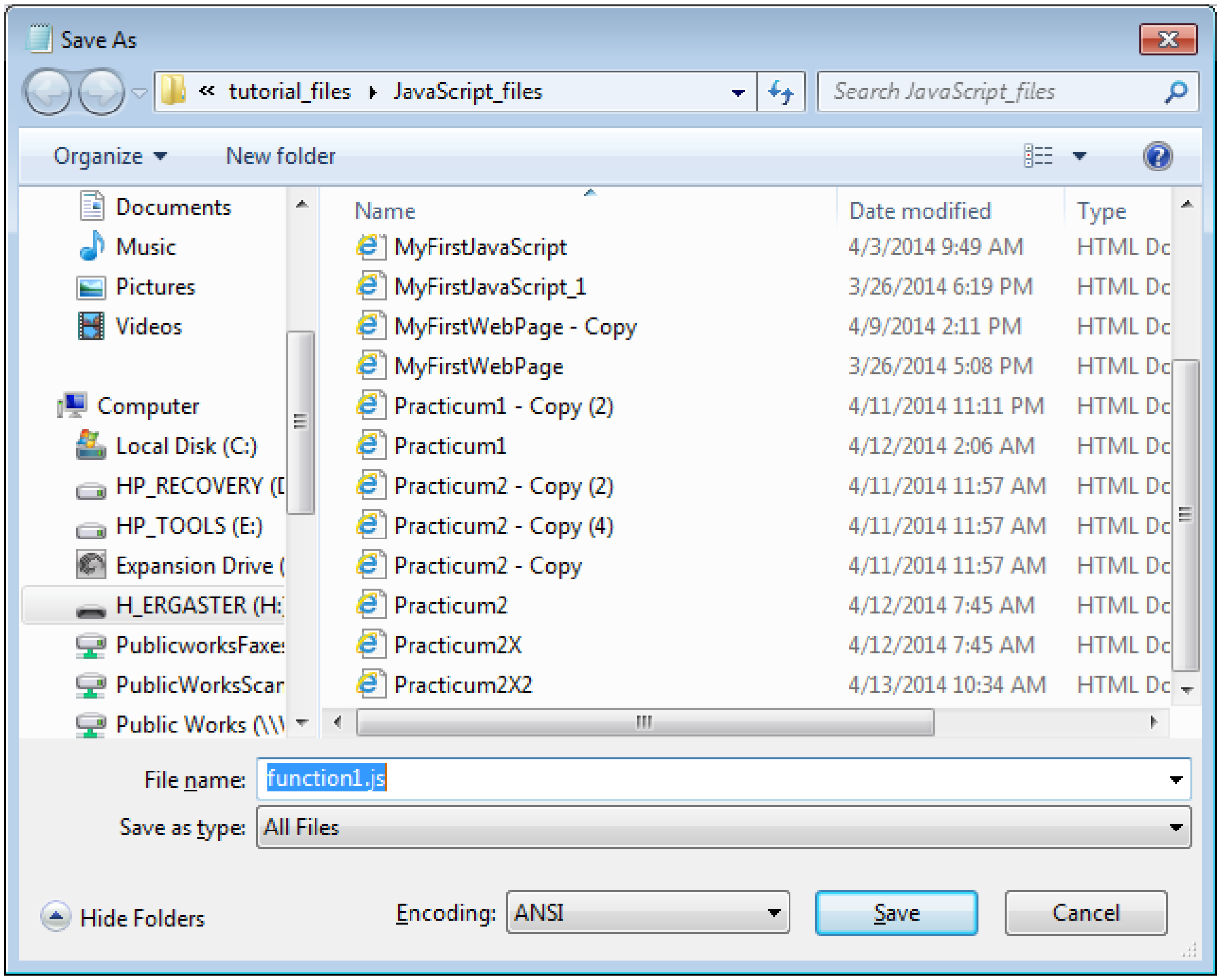
Alternately, the script may be written in a separate file, a *JavaScript* (***.js***) file.
Cut and paste the function for calculation of the yellow time to a new session of *Notepad* (or the text editor you are using).

Note that the *<script>…</script>* tags themselves are not to be added in the code in the *JavaScript* file.



```
function fnYellowCalc()
      {
              //declare variables needed for the calculation
              var reacTime, speed, decel, gradient, yellowtime ;

              /*now assign the controls' values to the variables, identifying
              each control by its unique id */
              reacTime = document.getElementById("txtReacTime").value;
              speed = document.getElementById("optSpeed").value;
              decel = document.getElementById("txtDecel").value;
              gradient = document.getElementById("txtGrade").value;

              //now ensure variable values are converted to numbers for calcs
              reacTime = parseFloat(reacTime);
              speed = parseFloat(speed);
              decel = parseFloat(decel);
              gradient = parseFloat(gradient);


              //calculate the yellow time using the ITE formula
              yellowtime = reacTime + 1.47*speed/(2*decel+32.2*gradient/100) ;

              //assign calculated value to the text box to display it
              frmYellowCalc.txtYellowInt.value = yellowtime ;

              return ;
      }
```

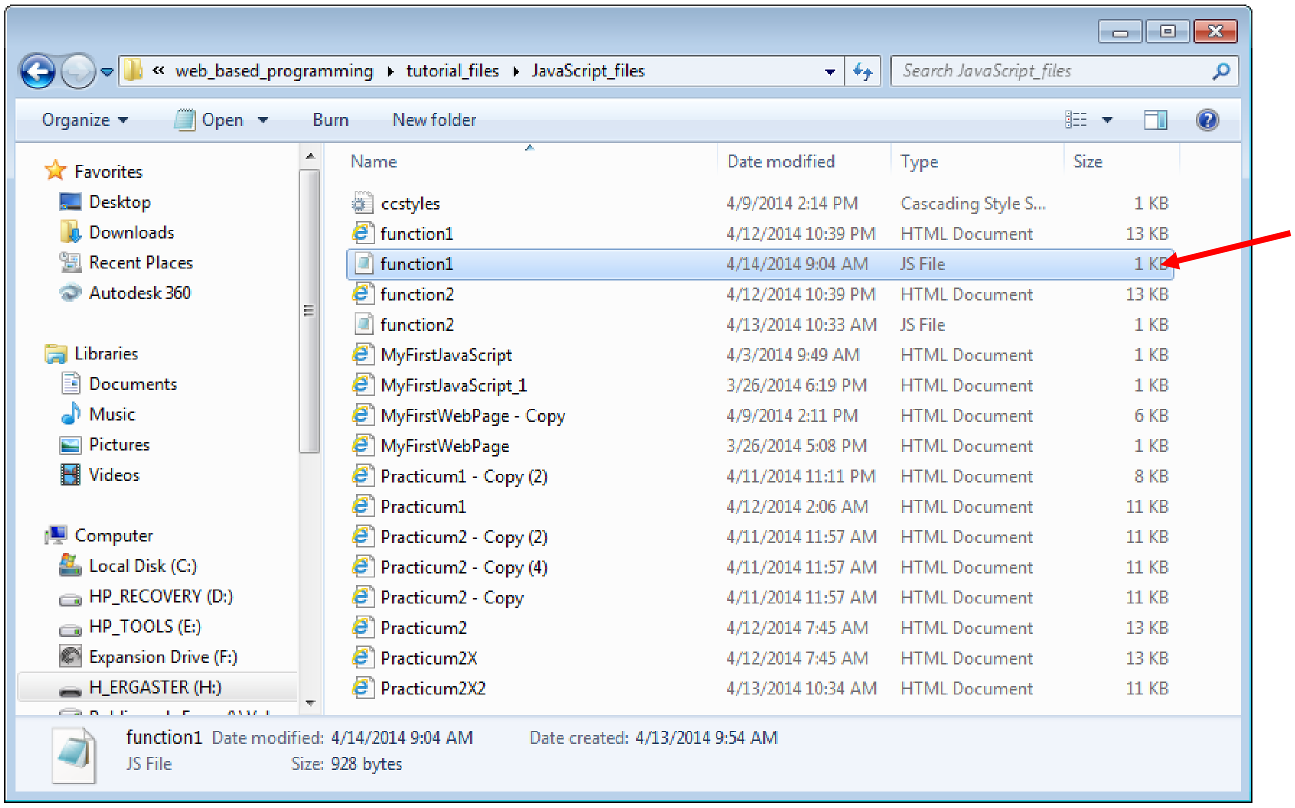
Click on *File* - *Save As* to save your file.
Navigate to your working directory.
Give your file a name of your choice (in this case the name *function1* will be used).
Append the suffix ***.js*** to the name of the file, and select *All Files* in the *Save as type* box.
Click on *Save* to save your ***.js*** file.



Name	Date modified	Type
MyFirstJavaScript	4/3/2014 9:49 AM	HTML Dc
MyFirstJavaScript_1	3/26/2014 6:19 PM	HTML Dc
MyFirstWebPage - Copy	4/9/2014 2:11 PM	HTML Dc
MyFirstWebPage	3/26/2014 5:08 PM	HTML Dc
Practicum1 - Copy (2)	4/11/2014 11:11 PM	HTML Dc
Practicum1	4/12/2014 2:06 AM	HTML Dc
Practicum2 - Copy (2)	4/11/2014 11:57 AM	HTML Dc
Practicum2 - Copy (4)	4/11/2014 11:57 AM	HTML Dc
Practicum2 - Copy	4/11/2014 11:57 AM	HTML Dc
Practicum2	4/12/2014 7:45 AM	HTML Dc
Practicum2X	4/12/2014 7:45 AM	HTML Dc
Practicum2X2	4/13/2014 10:34 AM	HTML Dc

File name: function1.js
Save as type: All Files
Encoding: ANSI

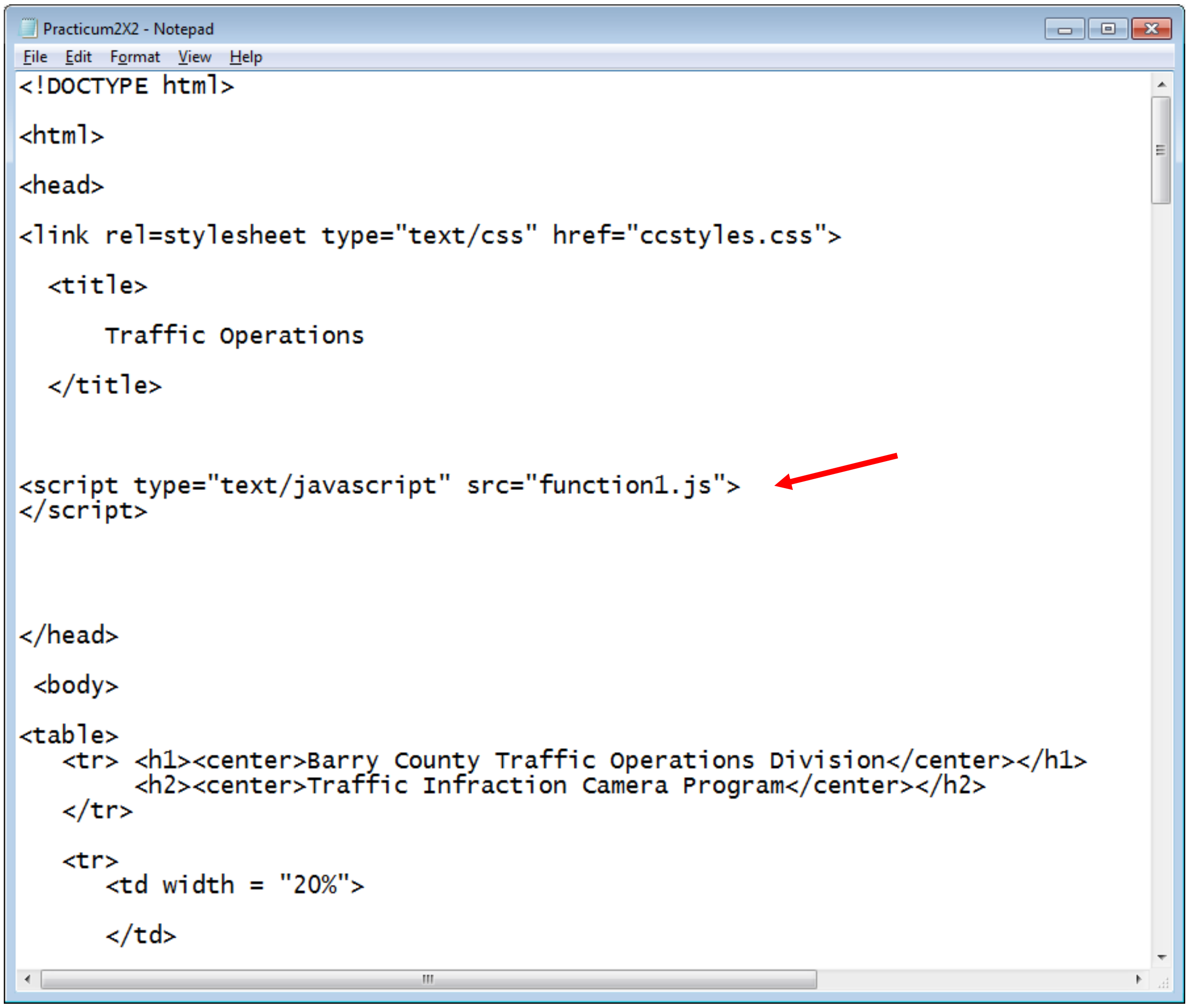
Review your working directory and identify your ***.js*** file.
Note the icon for a ***.js*** file.



Name	Date modified	Type	Size
ccstyles	4/9/2014 2:14 PM	Cascading Style S...	1 KB
function1	4/12/2014 10:39 PM	HTML Document	13 KB
function1	4/14/2014 9:04 AM	JS File	1 KB
function2	4/12/2014 10:39 PM	HTML Document	13 KB
function2	4/13/2014 10:33 AM	JS File	1 KB
MyFirstJavaScript	4/3/2014 9:49 AM	HTML Document	1 KB
MyFirstJavaScript_1	3/26/2014 6:19 PM	HTML Document	1 KB
MyFirstWebPage - Copy	4/9/2014 2:11 PM	HTML Document	6 KB
MyFirstWebPage	3/26/2014 5:08 PM	HTML Document	1 KB
Practicum1 - Copy (2)	4/11/2014 11:11 PM	HTML Document	8 KB
Practicum1	4/12/2014 2:06 AM	HTML Document	11 KB
Practicum2 - Copy (2)	4/11/2014 11:57 AM	HTML Document	11 KB
Practicum2 - Copy (4)	4/11/2014 11:57 AM	HTML Document	11 KB
Practicum2 - Copy	4/11/2014 11:57 AM	HTML Document	11 KB
Practicum2	4/12/2014 7:45 AM	HTML Document	13 KB
Practicum2X	4/12/2014 7:45 AM	HTML Document	13 KB
Practicum2X2	4/13/2014 10:34 AM	HTML Document	11 KB

function1 Date modified: 4/14/2014 9:04 AM Date created: 4/13/2014 9:54 AM
JS File Size: 928 bytes

Open your HTML file and replace your function code for the calculation with a link to the ***.js*** file as follows:



```
<!DOCTYPE html>

<html>

<head>

<link rel=stylesheet type="text/css" href="ccstyles.css">

  <title>

      Traffic Operations

  </title>


<script type="text/javascript" src="function1.js">
</script>



</head>

 <body>

<table>
    <tr> <h1><center>Barry County Traffic Operations Division</center></h1>
         <h2><center>Traffic Infraction Camera Program</center></h2>
    </tr>

    <tr>
       <td width = "20%">

       </td>
```

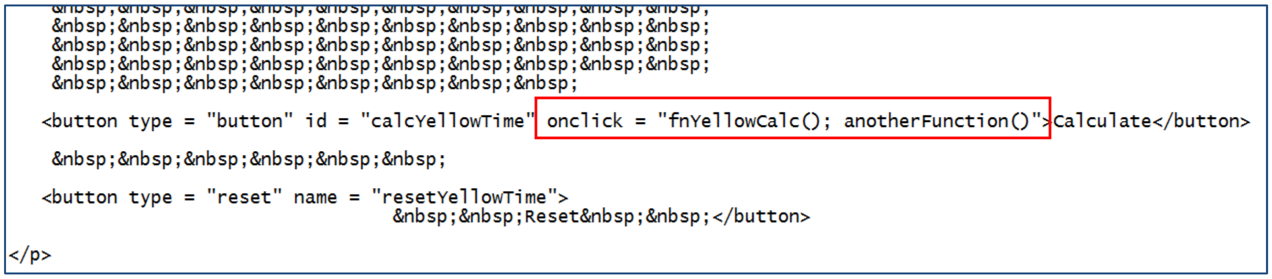
Save your file.
Test your file to confirm it performs as intended.

<u>Some points to consider:</u>

1. The ***.js*** file may be called by multiple web pages that need its functionality. In that case rather than copying and pasting or reproducing the script/function in each of the relevant web pages, the ***.js*** file may be called as needed.

2. In the above example, the ***.js*** file and the HTML file were in the same directory. As such the ***.js*** file was called in the link in the script tag by its name only. If the ***.js*** file and /or the HTML file are in separate directories, then the reference to the ***.js*** file must include the full path to the file.

3. The *onclick* events used so far involves one function being called. If necessary, more than one function may be added to an event. In such cases the function calls will be separated by a semicolon. The functions may be in a script(s) in the HTML file, or may be in the same or separate ***.js*** files, or some may be in the HTML file and others in ***.js*** file(s). For example:



```
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

   <button type = "button" id = "calcYellowTime" onclick = "fnYellowCalc(); anotherFunction()">Calculate</button>

    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

   <button type = "reset" name = "resetYellowTime">
                              &nbsp;&nbsp;Reset&nbsp;&nbsp;</button>

</p>
```

<u>Assignment for Independent Study:</u>

Review the narrative for the calculation of the All-Red Clearance interval. Add a script to calculate the All-Red Clearance interval when the relevant button is clicked on.

This completes Practicum #2. You have created your first fully functional interactive web page.

6. DEPLOYING WEB-BASED APPLICATIONS

6.1 Web Hosting

All the web pages (HTML files) developed in this course are currently residing on your computer (in the Desktop folder *JavaScript_files*), unless you moved or renamed the folder and/or files. The web pages must be uploaded to the web to go "live" and be viewable by the global internet audience. In order to do so, your files must be uploaded to a hosting web server. Many large companies and institutions such as universities and colleges have such facilities and allow members' web pages to be hosted on their servers for non-commercial use. Some internet service providers and cable companies may offer web hosting services to their customers, typically for a fee or periodic paid subscription. Otherwise individuals and small businesses may use a commercial web hosting service. There is an abundance of web hosting companies on the web. The costs for such services are varied and depend on the purpose of the website, e.g. personal use versus online retail business, etc. The customer signs up for an account and a domain name(s),
e.g. *ww.kwabenaofosu.com*, etc. The customer then uploads their files to designated folders in their account for them to go "live". The web hosting products generally come with additional incentives and benefits such as multiple domain names, free web page editing software, and free email accounts, as well as features to help online businesses such as database support, and e-commerce tools. Some web hosts may provide the service for free to their customers, but the customers may have to agree to carry advertising on their web pages. It is the sole prerogative of the customer to research and decide which hosting service is right for them.

After uploading your files to a web host it is important that the web pages and scripts be tested. For example, images should be checked for proper display. Forms and controls such as buttons, check boxes, and combo boxes should be clicked on to verify that the relevant scripts associated with them are executed and produce valid results. Links should be clicked on to verify that they open, and open to the correct location(s). Links and scripts that were set up to reference files in other folders will typically need to be reconfigured to reflect the path(s) to the file(s) and folder(s) as it now exists on the web hosting server.

6.2 Debugging HTML Files

HTML codes and *JavaScript* scripts should be reviewed and thoroughly tested frequently as they are being developed. Web browsers do not throw error messages (when an error or omission is encountered in the code) in a similar fashion and extent as a typical programming language, e.g.

Visual Basic. If your HTML code contains errors, the web browser will show what it can and ignore the rest of the code, or it will interpret the code as-is and display that outcome. It is the web developer's responsibility to frequently test the code and address problems as they arise. Verify or otherwise that the browser displays the web page content correctly and executes the scripts as intended. Test your codes and scripts frequently, block by block, line by line, tag by tag, rather than writing the entire code before testing it. In the latter scenario, it will be much more difficult to identify and isolate any problems.

6.3 Web Browser Compatibility

It is pertinent to note that different web browsers (and versions thereof), have some slight differences in the way they interpret and execute some HTML and *JavaScript* commands. This must be kept in mind in the development and testing of your applications. In this course series, all codes were developed and tested with the *Internet Explorer* (version 10) web browser by *Microsoft*. It is the programmer's responsibility to test that the codes will work as intended on other web browsers.

6.4 Getting Help

There is currently an abundance of help information on *JavaScript*, web page design, and HTML, particularly on the web. These include official (peer-reviewed) and unofficial sources, websites, academic work, professional presentations, tutorial videos (YouTube, etc.), user groups, online forums, downloadable code snippets, etc., etc. Typing a *JavaScript* topic in a search engine will typically yield hundreds if not thousands of results. It is strongly recommended that all codes developed be tested thoroughly before deployment.

7. CONCLUSION

This course has presented a broad overview of the fundamentals of the Hypertext Markup Language (HTML) forms and the *JavaScript* scripting language. Several examples and illustrations from many fields of specialty were presented to illustrate the creation and editing of interactive web pages using HTML and *JavaScript* scripts.

In this course participants developed a prototype interactive website using HTML codes and *JavaScript* scripts. All codes were typed in using the *Notepad* text editor only. The prototype website is interactive and can be used to input data and output results, as well as execute an algorithm.

This course has enabled participants to identify situations where web-based programming is relevant and will be of advantage to the practicing professional. Practitioners are strongly encouraged to look out for situations in their domains of expertise where web-based programming solutions are applicable and will be of benefit to their work and their organization.

Web development as well as web-based programming require a careful and meticulous approach and can only be mastered and retained by practice and repetition.

Good Luck and Happy Programming.

REFERENCES

ECMA International. (2011, June). *Standard ECMA-262 ECMAScript Language Specification Edition 5.1*. Retrieved March 26, 2014, from ECMA International: http://www.ecma-international.org/publications/standards/Ecma-262.htm

FunctionX Inc. (2011a). *Hypertext Markup Language*. Retrieved December 21, 2013, from FunctionX Tutorials: http://www.functionx.com/html/index.htm

FunctionX Inc. (2011b). *JavaScript Tutorial*. Retrieved March 21, 2014, from FunctionX Tutorials: http://www.functionx.com/javascript/index.htm

Microsoft. (2014a). *JavaScript Functions*. Retrieved April 1, 2014, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/6fw3zxcx(v=vs.94).aspx

Microsoft. (2014b). *Special Charachters (JavaScript)*. Retrieved April 5, 2014, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/2yfce773(v=vs.94).aspx

Mozilla. (2014). *Array-JavaScript*. Retrieved May 1, 2014, from Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

w3schools.com. (2014). *JavaScript and HTML DOM Reference*. Retrieved 5 5, 2014, from w3schools.com: http://www.w3schools.com/jsref/default.asp

World Wide Web Consortium. (2014). *HTML5*. Retrieved Aprl 9, 2014, from World Wide Web Consotium (W3C): http://www.w3.org/TR/2014/CR-html5-20140204/