Lab 2 – CS 3340

Contents

[1 Overview 1](#_Toc153190027)

[2 Build a Web Page 1](#_Toc153190028)

[3 Add CSS 4](#_Toc153190029)

[4 Merging Cells in Tables 9](#_Toc153190030)

[5 Add & Link Pages 11](#_Toc153190031)

[6 Add a Page with JavaScript 12](#_Toc153190032)

[7 Add a Page with jQuery 15](#_Toc153190033)

[8 Package Assignment for Submission 16](#_Toc153190034)

**Feel free to experiment on the pages you develop below, beyond what the lab specifies.**

To make this document easier to read, it is recommended that you turn off spell checking and grammar checking in Word:

1. Choose: *File, Option, Proofing*
2. At the very bottom, check: “Hide spelling errors…” and “Hide grammar errors…”

# Overview

Lab Objectives:

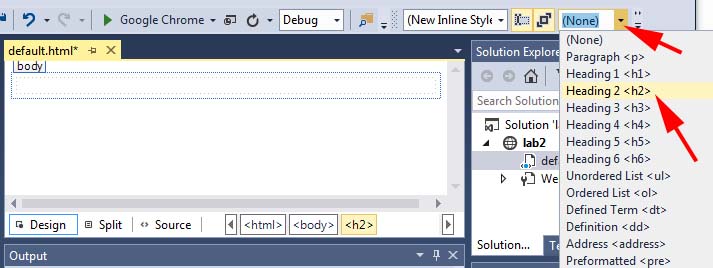
1. Build a webpage that utilizes CSS, JavaScript, and jQuery
2. Link two pages together.

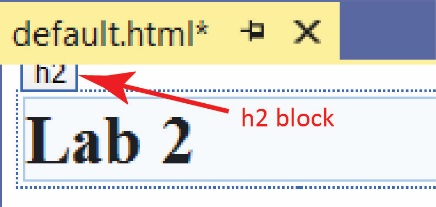
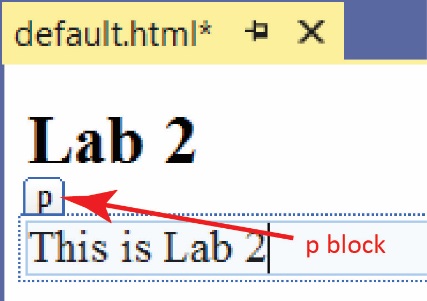
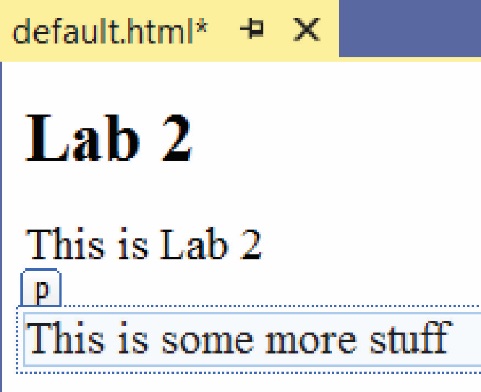
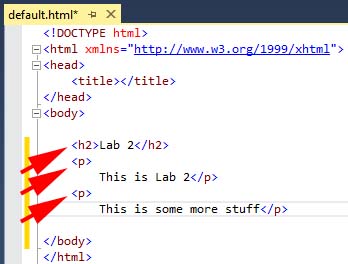
Follow the steps below to complete this lab.

# Build a Web Page

1. **Create Project:** Create an ASP.NET Web Application project named: *lab02\_lastName* (See Lab 01, Section 2 if you need help).
2. **Add a Web Page:** Create an HTML page named *default.html* in your *lab02* project and open in Design view (See Lab 01, Section 3 if you need help).
3. **(Read, no action required)** We being this course by creating HTML pages and the first thing to be said is that VS is **not** a good HTML editor in Design view, it is not designed for that. However, it will work fine for this course. VS Studio does have some features for applying stylistic elements to a page, like fonts and font sizes, alignments, *etc*. However, as we will learn, best practice is to never do styling using HTML. Instead, we will use *Cascading Style Sheets* (CSS).
4. **Create a Heading**:
5. Choose: *Heading 2* from the *Block Format* drop down menu as shown below.

Note: If your display is not large enough, you won’t see this drop down. You can add it by choosing: Add or Remove Buttons (a drop down on the menu bar, in the figure below, it is the control immediately to the left of the drop down that says, “(New Inline Syle)”; the choose: Customize (very bottom), Toolbars, Formatting. I do not see any way to access this from the menus.

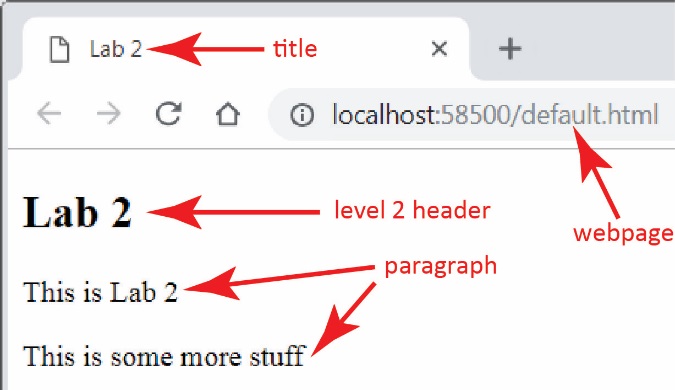


1. An *h2* block is shown on the page. Put your cursor in the *h2* block and type: *Lab* 2. This is a level 2 heading.
2. **Create a Paragraph**:
3. With your cursor at the end of the heading from above, press Enter and a *p* block is shown. Put your cursor in the *p* block and type the text shown. This creates a paragraph.
4. Press Enter and type the text shown. Then save.
5. **Modify Page in Source View**
6. Switch to *Source view* (selecting *Source* at the bottom of the window). Note the following:

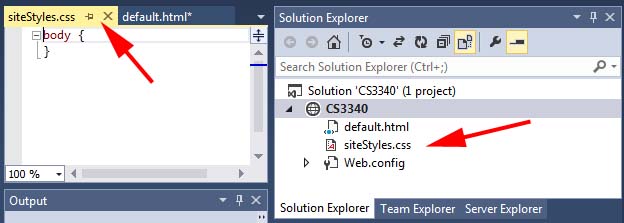
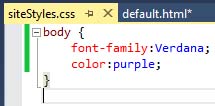
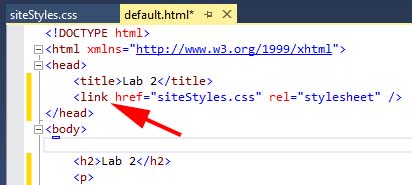
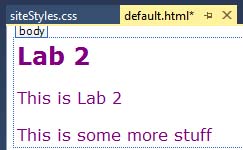
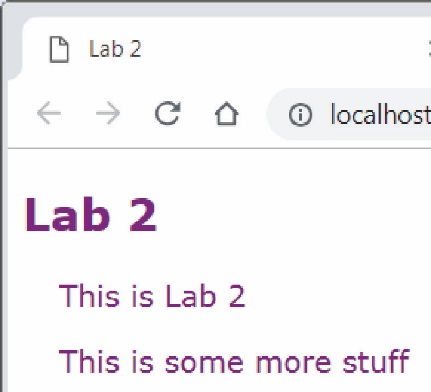
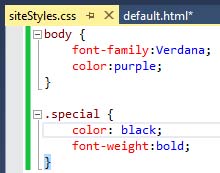
* The contents of a webpage are *marked up* with *HTML tags*.
* Each tag has an opening tag (*e.g.* <h2>) and a closing tag (*e.g.* </h2>).
* Tags can be nested. For example, the *h2* and *p* tags are inside the *body* tag.

1. Type a title between the two *title tags* in: *Lab 2*.

<title>Lab 2</title>

1. View the page on localhost (right-click in page and choose: *View in Browser*).

# Add CSS

1. **Add a CSS Style Sheet to Project**
2. (Read, no action required) HTML has style attributes which can be applied to an HTML tag, and it also has style tags for doing such things as setting font family, font size, colors, alignment, *etc*; however, using these is bad practice. Instead, all styling should be done with *Cascading Style Sheets* (CSS). We will learn more about this in class.
3. Right-click the project in the SE and choose: *Add, Style Sheet*
4. Supply the name, *siteStyles* and choose OK.
5. The style sheet is displayed in the editor with an empty *body* selector. Also, a node is created in the SE.
6. **Add Style Rules to an Element Selector**
7. (Read, no action required) The *body{…}* construct is called an *element selector*. Inside the selector we will write *style rules* which apply to any content inside the *body* tag in the HTML.
8. Type the *style rules* for the *body* as shown on the right. Just start typing and auto-complete will help you.
9. Save the style sheet and close it.
10. **Apply Style Sheet**
11. Redisplay *default.html* in *Source* view.
12. Next, we must apply the style sheet to our web page. Drag the style sheet (*siteStyles.css*) from the SE into the *head* section of the web page. A line will be added as shown below that links the style sheet to this page.
13. Display the page in *Design* view (Select the *Design* tab in lower left). You should see that the *body* style has taken affect.
14. ****View the page on localhost (right-click in page and choose: *View in Browser*).
15. **Create and Apply an Anonymous Class Selector**
16. Display the style sheet. Add the anonymous class selector, *.special* as shown on the right (simply type the text in as shown, be sure and preface with a “.”, or copy/paste).

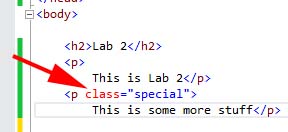
/\* Anonymous Class Selector\*/

.special {

color:black;

font-weight:bold;

}

1. Open the web page, *default.html* in Source view and modify the second *p* tag, as shown on the right, by adding the text: *class=”special”* at the end of the opening paragraph tagand save. Do not preface “special” with a “.”

class="special"

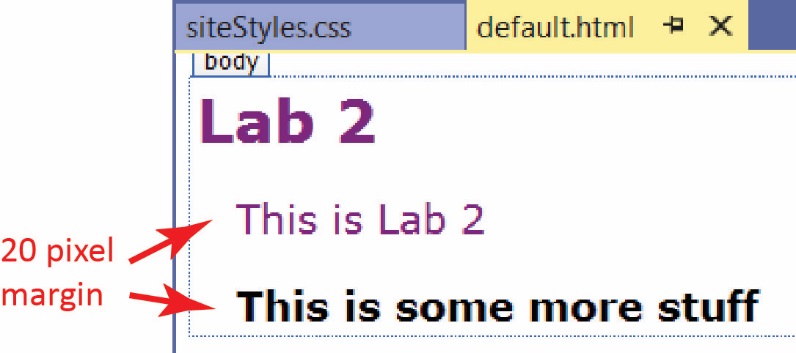
1. (Read, no action required) A *class selector* applies its rules to any tag that has a *class=”special”* attribute inside an HTML opening tag.
2. Display the page in *Design* view. You should see that the *special* style has taken affect.
3. (Read, no action required) This style rule *overrode* the *body* element style defined earlier as it is more specific, *i.e* it is inside the *body* tag.
4. View the page on local host.
5. **Create and Apply another Element Selector**
6. Display the style sheet. Add the element selector, *p* as shown below and save.

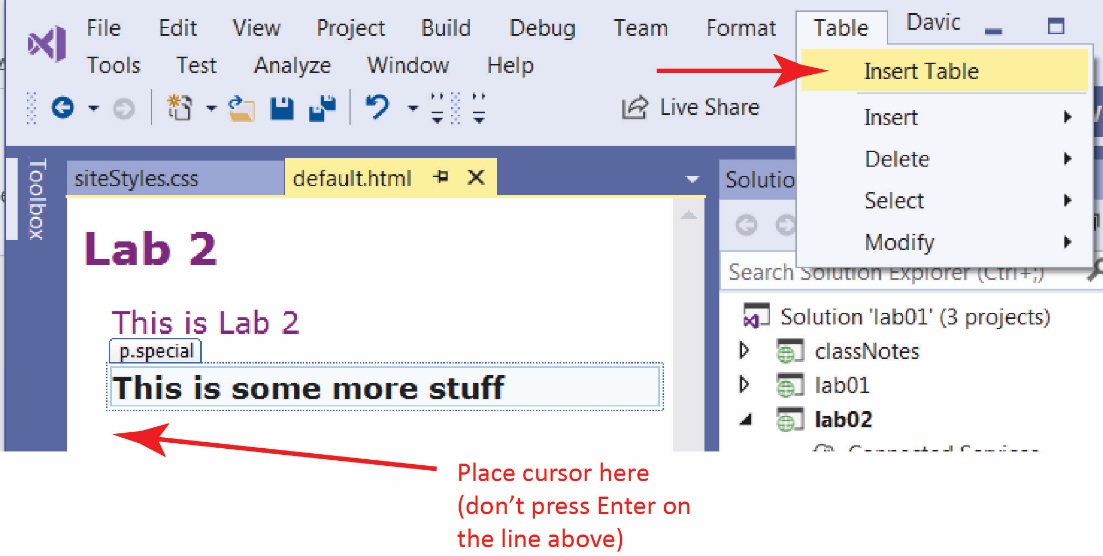
/\* Element Selector\*/

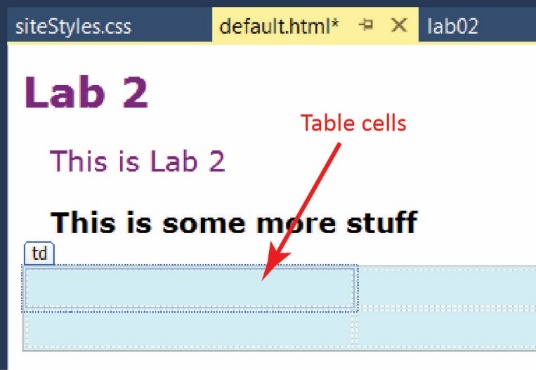
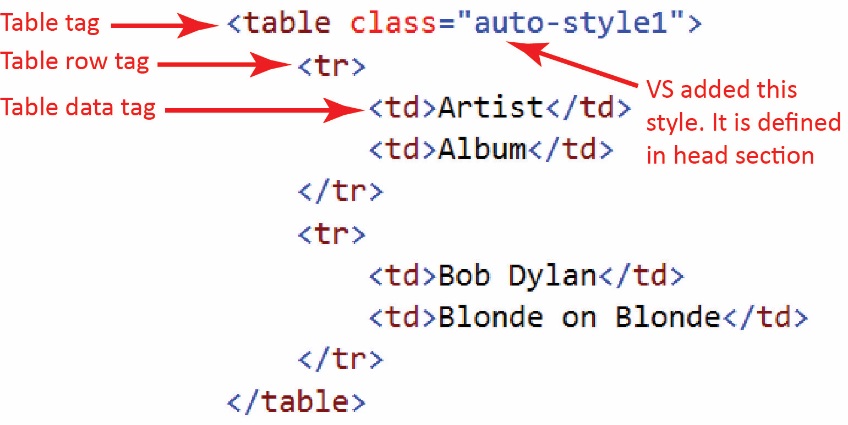
p{

margin-left:20px;

}

1. (Read, no action required) As stated earlier, an *element selector* affects all instances of the element (tag) in the web page. Thus, in our page both paragraphs will have a left margin of 20 pixels. Note that the class selector (*.special*) will still apply to the second paragraph.
2. ****Display the page in *Design* view. You should see that the the two paragraphs are indented. Note that the second paragraph also has the *.special* style applied.
3. View the page on local host.
4. **Add a Table**
5. Make sure your page is in *Design* view. Position your cursor as shown in the figure below. Do not press Enter as it will create a paragraph. We do not want the table we are about to create to be inside a paragraph.
6. Choose: *Table, Insert Table*.



1. Do the following on the *Insert Table* dialog:
2. Set it so that it has 2 rows and 2 columns.
3. Change the *Borders, Size* property to 1.
4. Change the *Background, Color* property to some light color (choose: *More colors*) (don’t choose red as we will use that for something else)
5. Accept all the other defaults
6. Choose: *OK*. The result is shown on the right. You should see the faint outline of the 4 cells.
7. Type the text shown below into the table (use your favorite Artist and Album). Note that the *body* style (purple text, verdana font-family) is applied.
8. View the page in *Source* mode. Note the following:
9. The table is defined with *table* tags: <table>…</table>
10. The table is composed of rows: <tr>…</tr>
11. Each row contains table data: <td>…</td>
12. VS added the “auto-style1” shown on the table tag.
13. The style is defined in the head section as a class style as shown below. It could be moved to the (external) style sheet, *siteStyles.css*; however, we will leave it where it is. In class I will talk about when to put styles in an external style sheet and when to put them in the head section of the page itself.
14. View the page on local host.
15. **Create and Apply another Element Selector**
16. Display the style sheet. Add the element style shown below:

/\* Element Selector\*/

**td {**

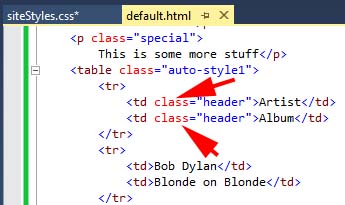
**color:red;**

**}**

1. View your page in design mode, and then view on localhost.
2. **Create and Apply a Class Selector**
3. Display the style sheet. Add the class selector shown below. This is similar to an anonymous class selector, but only applies to a <td> tag.

/\* Class Selector\*/

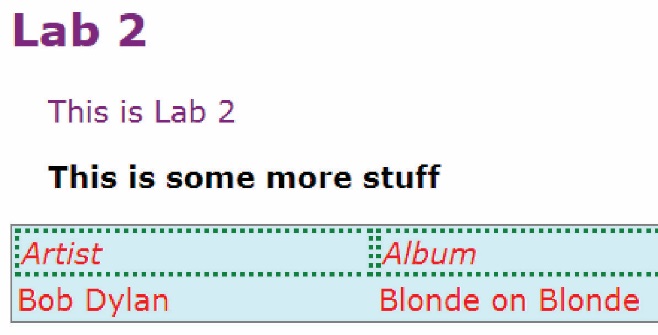
td.header{

 font-style:italic;

border:dotted;

border-color:green;

}

1. Open the web page, *default.html* in Source view and modify the first two *td* tags as shown on the right. *E.g.* add the text: *class=”header”.*
2. Display the page in *Design* view. You should see that the styles have taken affect. View the page on localhost.
3. **(Read, no action required)**

* HTML elements can have more than one style. In the example below, the paragraph will be styled according to *center* and *large.*

<p class="center large">This paragraph refers to two classes.</p>

* You can also group selectors:

h1, h2, p {  
  text-align: center;  
  color: red;  
}

# Merging Cells in Tables

1. Create a new page, *tables.html*.
2. Add a table with 4 rows and 4 columns, and set the border width to 1. This only sets the outside border.
3. View the page on local host, and you can see that there are no borders around the cells.
4. Remove the *style* tag from the *head* section (*i.e.* remove the text shown below). We are going to write a better set of styles to display the cell borders.

<style type="text/css">

.auto-style1 {

width: 100%;

border-style: solid;

border-width: 1px;

}

</style>

1. Change the *class* from:

<table class="auto-style1">

To:

<table class="main">

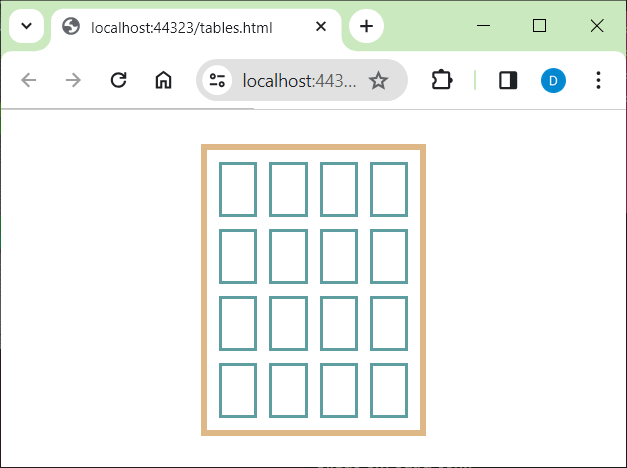
1. Add these selectors to *siteStles:*

table.main {

margin-left: auto;

margin-right: auto;

border: thick solid burlywood;

 border-spacing: 10px;

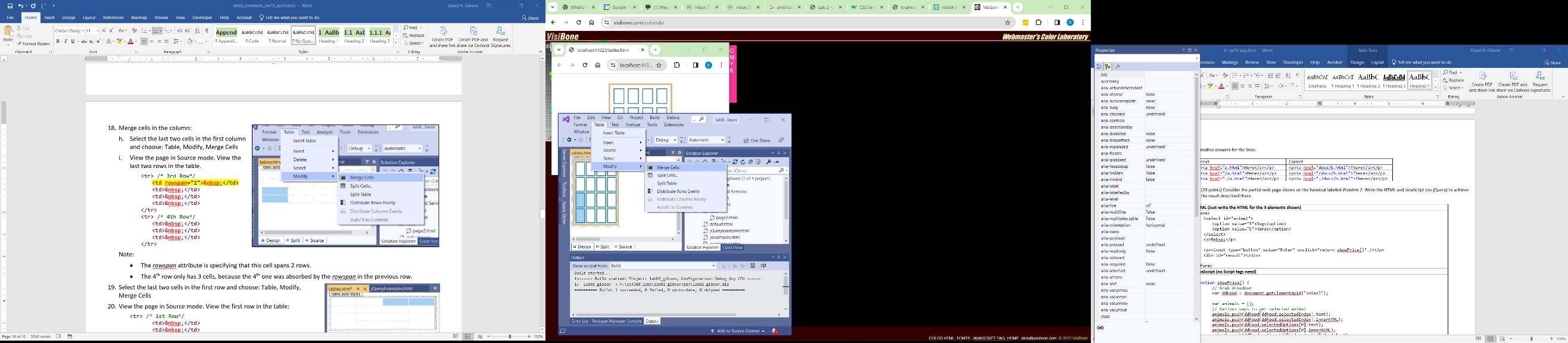
}

table.main td {

border: medium solid cadetblue;

padding: 10px 10px 10px 10px;

}

1. Link the style sheet to this page: drag *siteStyles.css* from the SE to the *head* section of *tables.html.*
2. View the page on local host as shown on the right.
3. Merge cells in the column:
4. Select the last two cells in the first column and choose: Table, Modify, Merge Cells
5. View the page in Source mode. View the last two rows in the table.

<tr> /\* 3rd Row\*/

<td rowspan="2">&nbsp;</td>

<td>&nbsp;</td>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr> /\* 4th Row\*/

<td>&nbsp;</td>

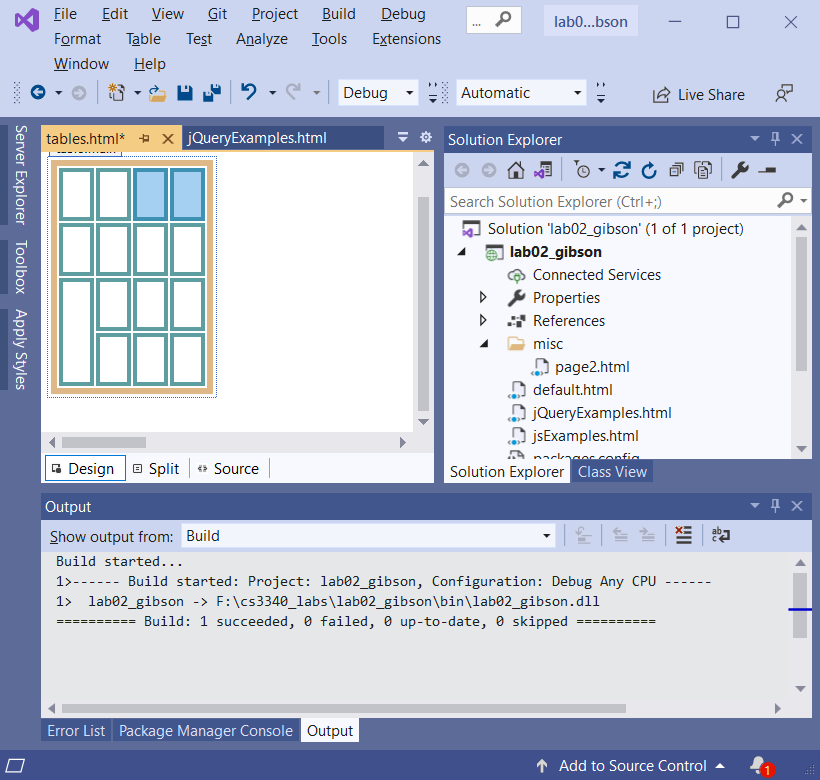
<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

Note:

* The *rowspan* attribute is specifying that this cell spans 2 rows.
* The 4th row only has 3 cells, because the 4th one was absorbed by the *rowspan* in the previous row.

1. Select the last two cells in the first row and choose: Table, Modify, Merge Cells
2. View the page in Source mode. View the first row in the table:

<tr> /\* 1st Row\*/

<td>&nbsp;</td>

<td>&nbsp;</td>

<td colspan="2">&nbsp;</td>

</tr>

Note:

* The *colspan* attribute is specifying that this cell spans 2 columns.
* There are only 3 cells in the first rows because the last cell spans two columns.

# Add & Link Pages

A skill you need to have is to be able to link to another page in the same folder and in sub folders. Pay attention to the *href* attribute in the examples below.

1. Add new page, *page1* and link to *default:*
2. Add a new HTML page named: *pages1.html.* Open in *Design* mode.
3. Type this text at the top of your new page: “Page 1”
4. On the next line, type this text: “Lab 2 Home”
5. Select the text and then choose: *Format, Convert to Hyperlink*
6. Choose: *Browse,* select *default.html*, *OK, OK*.
7. View the Source and verify that *anchor* tag was added:

<p><a href="default.html">Lab 2 Home</a></p>

1. Set the title to: “Lab 2 – Page 1”
2. View the page on local host and verify that the link works correctly.
3. Link *default* to *page1*
4. Open *default.html* in Design view
5. At the bottom of the page, type this text: “Page 1”
6. Select the text and then choose: *Format, Convert to Hyperlink*
7. Choose: *Browse,* select *page1.html*, *OK, OK*.
8. View the Source and verify that *anchor* tag was added:

<p><a href="page1.html">JS Examples</a></p>

1. View the page on local host and verify that the links works correctly. You should now be able to navigate back and forth between the two pages.
2. Add new page, *page2* in a subfolder, *misc*, and link to *default:*
3. Right-click the project node in the SE and choose: *Add, New Folder* and then type the name: *misc*.
4. Right-click the *misc* node in the SE and choose: *Add, HTML Page* and type the name: *pages2.html.*
5. Open *pages2* in *Design* mode.
6. Type this text at the top of your new page: “Page 2”
7. On the next line, type this text: “Page 1”.
8. Select the text and then choose: *Format, Convert to Hyperlink*
9. Choose: *Browse,* select *page1.html*, *OK, OK*.
10. View the Source and verify that *anchor* tag was added:

<p><a href="../page1.html">Page 1</a></p>

1. Set the title to: “Lab 2 – Page 2”
2. View the page on local host and verify that the link works correctly. You can move to *page1* but you can’t move back down to *page2*, yet.
3. Link *page1* to *page2*
4. Open *page1.html* in Design view
5. Beside the link, “Lab 2 Home”, type this text: “Page 2”
6. Select the text and then choose: *Format, Convert to Hyperlink*
7. Choose: *Browse,* select *page2.html*, *OK, OK*.
8. View the Source and verify that *anchor* tag was added:

<p><a href="misc/page2.html">Page 2</a></p>

1. View the page on local host and verify that the links works correctly. You should now be able to navigate back and forth between the two pages.

# Add a Page with JavaScript

1. Add new page, *jsExamples* and link to *default:*
2. Add a new HTML page named: *jsExamples.html.* Open in *Design* mode.
3. Type this text at the top of your new page: “Lab 2 Home”, and create a link back to *default.html.*
4. Set the title to: “Lab 2 – JS Examples”
5. Open *default.html* in Design mode and create a link, “JS Examples”, to *jsExamples.*
6. View the page on local host and verify that the links work correctly.
7. **Add and Call a JavaScript Function** – We will add a button to the page. When it is pressed, it will call a JavaScript function that prompts the user for a number and then displays the square root.
8. Open *jsExamples.html* in Source view.
9. Add the *script* tag below, which contains a JS function named *squareRoot*, in the *head* section, after the *title* tag.

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

**function squareRoot() {**

**var val = window.prompt("Gimme a number", 483);**

**var x = parseFloat(val);**

**var result = Math.sqrt(x);**

**window.alert("The sq.root of " + x + " is " + result.toFixed(3));**

**}**

**</script>**

Note:

* Variables can hold any type of value (int, double, Boolean, string, object) and do not have to be declared. I prefer to declare them with the *var* keyword.
* *window.prompt* displays a message in a modal dialog. The user can type in a value and dismiss the dialog with an *OK* or *Cancel.* The user’s response is returned.
* *window.alert* displays a message in a modal dialog. The user can dismiss the dialog with an *OK* or *Cancel.*

1. Add button – Add this text to the *body* section, below the link:

<p><button id="btnSqRoot" onclick="squareRoot()">Square Root</button></p>

Note:

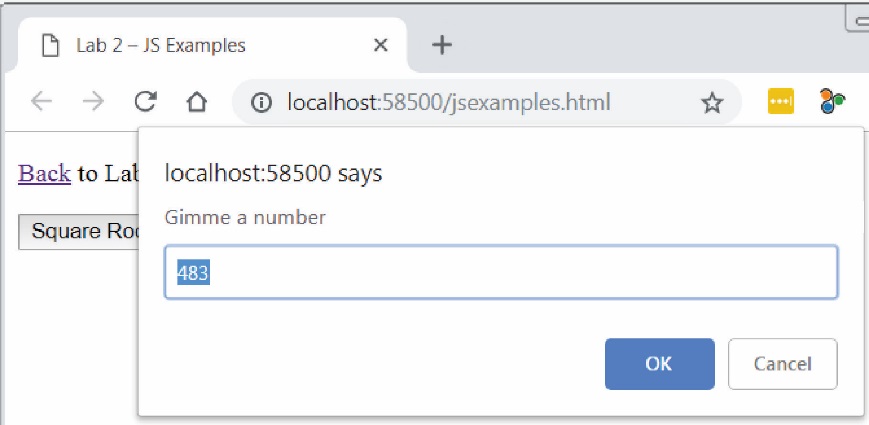
* There is an *id* attribute that is intended to uniquely (though not enforced) identify this button should code need to use the button.
* There is an *onclick* attribute which specifies the behavior that is to occur when the button is pressed. In this case, the *squareRoot* JS function is called.
* Earlier versions of HTML used: **<input id="Button1" type="button"...** to render a button, which is a single tag. HTML 5 also defines the button tag as shown above. An identical version of the button above is shown below using the *input* tag. You may see this style in the notes from time to time.

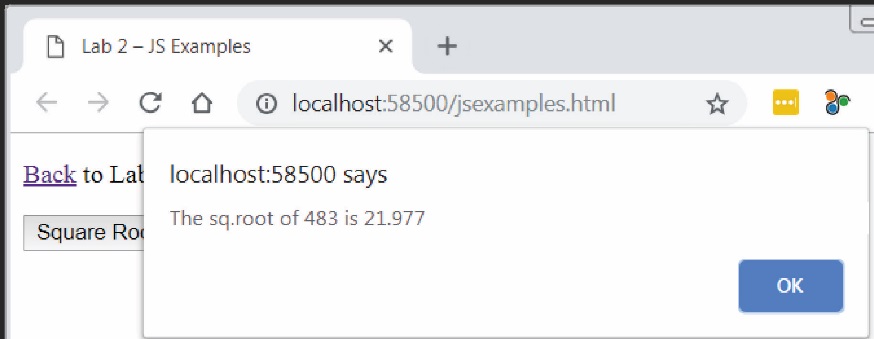
**<p>**

**<input id="Button1" type="button" value="Press Me" onclick="squareRoot()" />**

**</p>**

1. View the page on localhost. The page should display and the button should perform as expected.





1. **Add some more JavaScript**
2. Add this function inside the *script* tag in the *head* section:

function greeting() {

var name = document.getElementById("txtName");

var result = document.getElementById("result");

result.innerHTML = "Hi, " + name.value;

}

Note:

* This line:

var name = document.getElementById("txtName");

finds the HTML element with an *id* attribute of “txtName”, which is of course is the text field. Thus, the text field object is referenced by the *name* variable. Similarly, the second line finds the HTML element with an *id* attribute of “result”, which is a paragraph. Thus, the paragraph object is referenced by the *result* variable.

* The third line sets the *innerHTML* property of the *result* paragraph to contain the text shown above. The value typed in by the user is contained in *name.value*

1. Add this text to the end of the *body* section

<p>

What is your name?

<input id="txtName" type="text" />

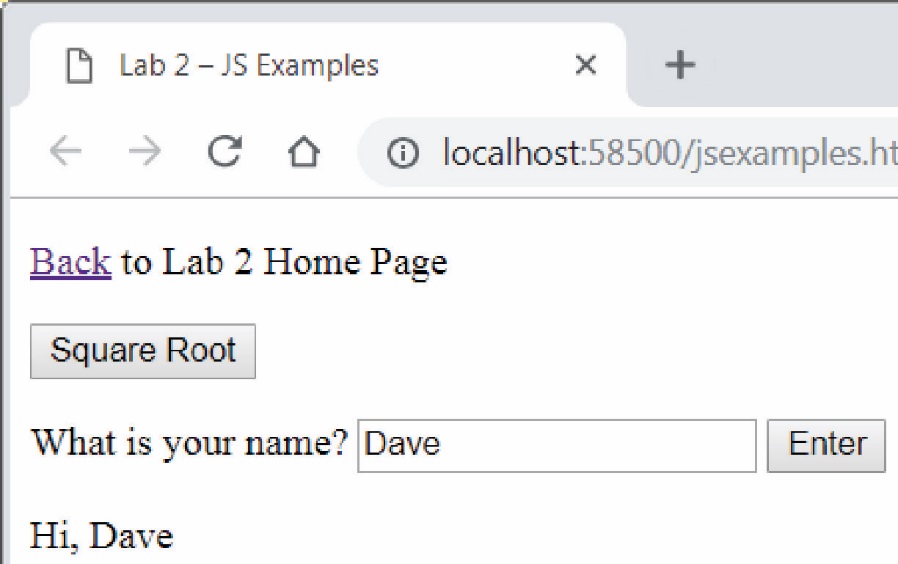
<input id="Button2" type="button" value="Enter" onclick="greeting()" />

</p>

<p id="result" class="special"></p>

Note:

* The first *input* tag has a *type* attribute that is set to “text.” This will display a text field where the use can type something. The *id* attribute, “txtName” uniquely identifies this text field so that the code can access the value that was typed in.
* The second *input* tag is a button, and when it is pressed, the *greeting* javascript function is called.

1. View the page on localhost. Type in a name and press Enter. The page should perform as expected.

# Add a Page with jQuery

1. Add new page, *jQueryExamples*:
2. Link *default* and *jQueryExamples* to each other (in both directions).
3. Set the title to: “Lab 2 – jQuery Examples”
4. At the bottom of the page, type this text: “jQuery Examples”
5. View the page on local host and verify that the link works correctly.
6. **Add a Button and TextField** – Open your *jQueryExamples* page in *Source* view and copy this HTML to the bottom (just above the closing body tag)

<p id="ageStuff">How old are you?

<input id="txtAge" type="text" />

<button id="btnAge">Enter</button>

</p>

<p id="result" class="special"></p>

1. **Download and Include the jQuery Library**
2. Add this line to the *head* section.

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

1. **Add a jQuery Function** –
2. Open your page in *Source* view
3. Copy the code below into the *head* section, below the previous *script* tag:

<script type="text/javascript">

$(document).ready(function () {

$("#btnAge").click(function () {

var but = $(this);

var butText = but.text();

if (butText == "Enter") {

$("#result").text("You are " + $("#txtAge").prop('value') + " years old");

but.text("Reset");

}

else {

$("#ageStuff").fadeOut(1000);

$("#ageStuff").promise().done(function () {

$("#result").text("");

$("#txtAge").prop('value', '');

but.text("Enter");

$("#ageStuff").fadeIn(1000);

$("#txtAge").focus();

});

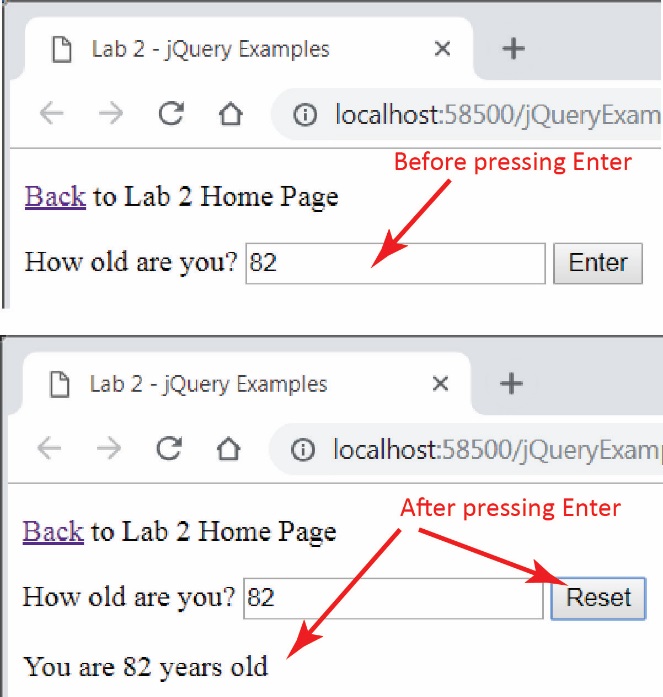
}

});

});

</script>

Note: jQuery will be covered in another tutorial and in class.

1. **Test Page** – Right-click in page and choose: *View in Browser*. Test the new button
2. Enter a value and press Enter.
3. The resulting display. Now press Reset and see what happens: everything fades out, then fades back in with the text field empty and the text on the button changed back to “Enter”

# Package Assignment for Submission

1. Close VS and zip your *lab02\_lastName* project folder and submit on Blazeview in the *lab 02* dropbox.

If you need further directions, see Lab 1, Stage 4.

**You are done!**