

# ITEC447

# Web Projects

---

CHAPTER 7 – CLIENT-SIDE SCRIPTS – JAVASCRIPT

# Creating Buttons from Scratch Using CSS

---

- Designers often need to create their own buttons from scratch to achieve a particular look in their sites.
- Using the techniques you learned in previous hours, you can use Cascading Style Sheets (CSS) to make advanced buttons that leave you in control of every element, from font to color to how the button behaves.

# Creating a Classic Box Button

---

- When designers first started to focus on buttons as a navigational tool, they could rely on only basic functionality.
- The most basic button of all is a string of text with a colored box around it.
- The *Classic Box Button* was heavily used on the Web in years past and it utilized the border attribute to mimic a crude 3D effect.
- To understand how to build and use buttons with CSS, the *Classic Box Button* is a good starting point.

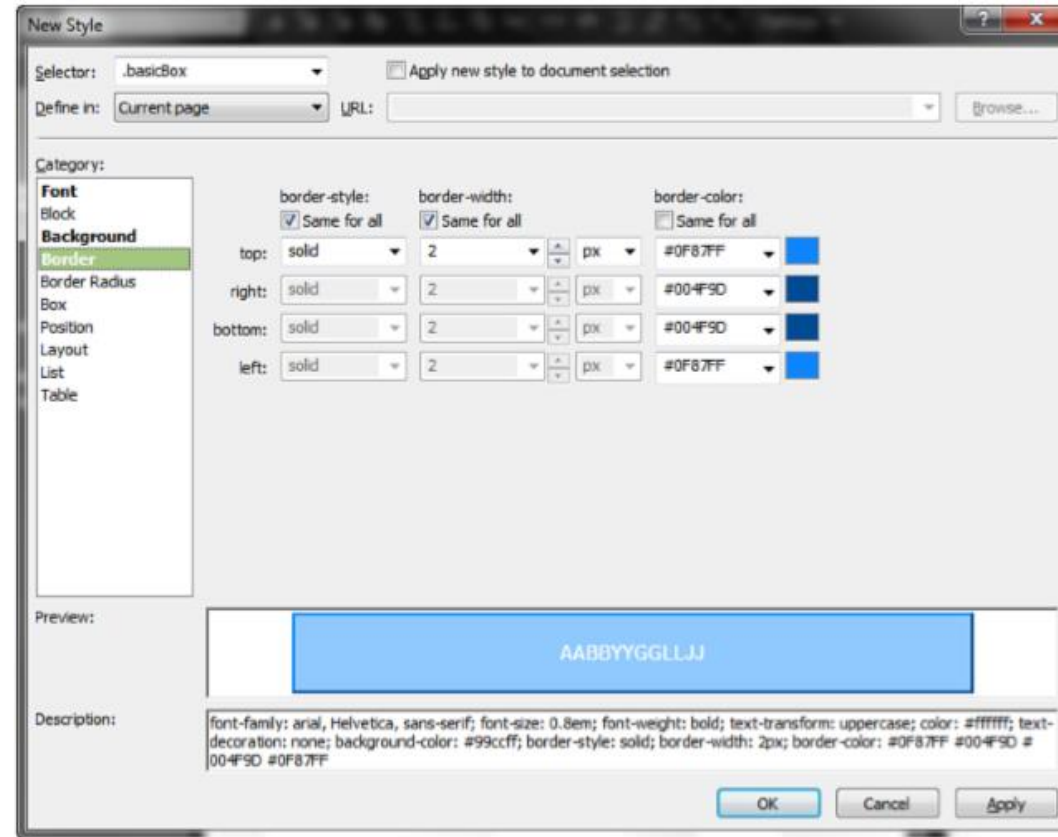


# Creating a Classic Box Button Cont.

---

- Create a new page with the name *buttons.html*.
- Create a new paragraph with the text *Classic Box Button*.
- Highlight the button text and make it a hyperlink pointing back to the current *buttons.html* page.
- Click the *New Style* button and create a new class called *.basicBox*.
- In the *Font* category, set *font-family* to *Arial, Helvetica, sans-serif*; *font-size* to *0.8em*; *font-weight* to *bold*; *text-transform* to *uppercase*; *font-color* to *white (#FFFFFF)*.
- Check the *None* box under *text-decoration* to get rid of the line under the text.
- Under *Background*, set *background-color* to a *light blue (#99CCFF)*.
- Under *Border*, set *border-style* to *solid* for all, *border-width* to *2px* for all, the *border-top* and *border-left* colors to a *medium blue (#0F87FF)*, and the *border-bottom* and *border-right* colors to a *dark blue (#004F9D)*.
- This gives the button the appearance of popping out from the screen.

# Creating a Classic Box Button Cont.



# Creating a Classic Box Button Cont.

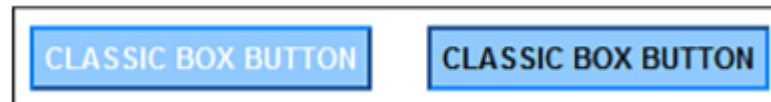
---

- In the *Box* category, set *padding* to *5px* for all four sides.
- Click *OK* to create the new class.
- Select the button text you created, use the tag selector to select the `<a>` tag, and use the *Apply Styles* task pane to apply the new class.
- To make the button react when the user hovers the mouse over it, you also have to make a *:hover* pseudoclass with different styles.
- This is where CSS shows its power: Because you are only changing some of the attributes with the *:hover* state, you don't need to redefine all the ones that don't change.
- By creating a new style with the selector *.basicBox:hover* and changing only those attributes that are different between the two states, you are making less code clutter and using the cascade wisely.

# Creating a Classic Box Button Cont.

---

- With `.basicBox:hover` open in the *Modify Style* dialog, set `font-color` to *black* (`#000000`) and invert the border colors so that the lighter blue is on the right and bottom sides while the darker blue is on the top and left sides.
- This gives the appearance of the button being pressed down when it is hovered over.
- Click *OK* to apply the changes, and then save and test in your browser.



# Creating a Classic Box Button Cont.

---

- As with all other styles, you can create separate styles for each pseudoclass for more advanced visual interaction with the button.
- In this example, the actual styling of the button came from applying a class to the `<a>` tag.
- But this is not the only way to create box buttons.
- You can easily group one or several buttons within a `<div>` or `<span>` class or ID, and create a style instead.
- The result would be the same.



# Creating a Modern Rounded-Corner Button with CSS3

---

- The *Classic Box Button* is called “*Classic*” for a reason—although the functionality is still current, the appearance is not.
- To get the *CSS button* up to modern standards, we need to add some more breathing space, better colors, and the all-important rounded corners.
- By combining the *Classic Box Button* with some CSS3, you can easily make modern buttons that are indistinguishable from those seen on famous websites—because they are made the exact same way!

# Creating a Modern Rounded-Corner Button with CSS3 Cont.

---

- Below the *Classic Box Button*, create a new paragraph with the text *Click me* and make it a hyperlink pointing back to the *buttons.html* page.
- Make a new class called *.modernBox* by using the *New Style* dialog.
- Set the *font-family* to *Arial, Helvetica, sans-serif*; *font-weight* to *bold*; *color* to *#ffffff*.
- Check *None* under *text-decoration*.
- Under *Background*, set *background-color* to *#57D5FF*.
- Set all four *borders* to *solid, 2px, and #CCCCCC*.
- Here come the rounded corners: Under *Border Radius*, set *radius* to *10px*.
- To give the text plenty of whitespace (breathing space), go to *Box* and set *padding top* and *bottom* to *10px* and *left* and *right* to *2em*.
- Finally, to give the button itself some breathing space, go to *Layout* and set *display* to *inline-block*.
- This is done to pull the button out of the regular text flow of the site and make the browser treat it as its own entity.

# Creating a Modern Rounded-Corner Button with CSS3 Cont.

---

- When you save and preview the page in your browser, you'll see a modern-looking rounded-corner button.
- To make a hover state, repeat the process from the earlier example by creating a new style with the selector `.modernBox:hover` that has the `background-color` value set to `#FF9933` and `border-color` set to `#333333`.
- This produces a button with a modern hover state.



# Creating an Advanced Box Button with Images

---

- The modern box button might be easy to create, but in some cases you want a button with more flare.
- That's where the *Advanced Box Button* with Images comes in.
- The key is to replace the background color with a background image.



# Creating an Advanced Box Button with Images Cont.

---

- In this example, you use a different styling technique to apply the same style to multiple buttons:
- Below the *modern box button*, create an *unordered list* with three buttons: *Button 1*, *Button 2*, and *Button 3*.
- Make each list item into a hyperlink back to the current page.
- Make Create a new class called *.advancedBox* by using the *New Style* dialog.
- Under the *List* category, set the *list-style-type* to *none*.
- Click *OK* to create the class and apply it to the `<ul>` tag using the *Quick Tag Selector*.

# Creating an Advanced Box Button with Images Cont.



# Creating an Advanced Box Button with Images Cont.

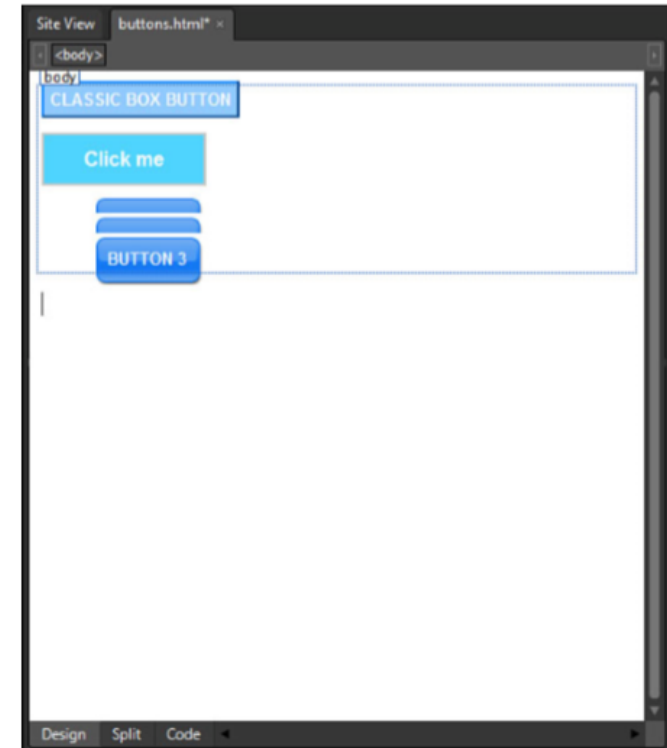
---

- Import the images named *blueButtonUp.gif* and *blueButtonOver.gif* from the lesson files (Hour 15) and save them in a new folder called *buttonGraphics*.
- Make a new style and give it the *selector* name *.advancedBox a*.
- In the *Font* category, set *font-family* to *Arial, Helvetica, sans-serif*; *font-size* to *0.8em*; *font-weight* to *bold*; *text-transform* to *uppercase*; and *font-color* to *white (#FFFFFF)*.
- Check the *None* box under *text-decoration* to get rid of the line under the text.
- In the *Background* category, set *background-image* to the *blueButtonUp.gif* image you just imported, and set *background-repeat* to *no-repeat*.
- Because the image is larger than the text and you want it to surround the text, go to the *Box* category and set *padding* to *13px* on all sides.
- Click *OK* to apply the style.

# Creating an Advanced Box Button with Images Cont.

- In *Design* view, you can see that the new button background is applied, but the buttons are covering each other.
- This is because the list items have not yet been styled, and the browser assumes that the list items are the height of the text without the padding.

The problem with this technique is that it requires two images to work. This might not seem like a big deal, but if the user is on a slow connection, the page is on a slow server, or if there is something else that slows the system down, the user might experience noticeable lag between hovering over the button and the new background image being loaded. One way around this problem is to preload the images using behaviors, but this requires JavaScript to work properly.





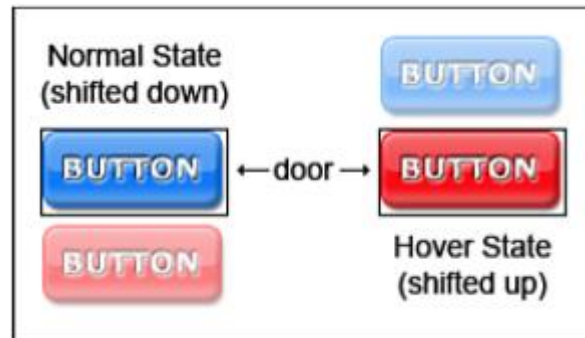
# Creating an Advanced Box Button with Images Cont.

---

- To fix this problem, create a new style with the *selector* name `.advancedBox li` and set the *padding-top* and *padding-bottom* attributes to `13px` to match the link style you just created.
- Click *OK* to apply the new style, and the buttons no longer overlap.
- To create a *hover-over* effect, create a new style with the *selector* name `.advancedBox a:hover`.
- Because the `:hover` pseudoclass inherits all the styling from the main `a` class, all you need to do is change *background-image* to `blueButtonOver.gif`.
- Click *OK*, and then save and preview in your browser.
- Now instead of the background color changing when the mouse hovers over the button, the background image swaps out.
- Because the background image is part of the `a` style, the entire image is clickable, not just the text itself.

# Creating Text-Free Buttons with Sliding Doors

- The problem of preloading content has become more prevalent with the emergence of blogs because many blogs have a lot of scripts running at the same time, and it is important to reduce the load on both the network and the computer as much as possible to make things work smoothly.
- A technique often referred to as *sliding doors* was developed to enable the designer to use one image file as two different backgrounds.
- This is done by creating a file that has two versions of the background, either on top of or to the side of one another.
- The name *sliding doors* refers to the action of literally sliding the background from one side to the other to display only half of the image at a time.



# Creating Text-Free Buttons with Sliding Doors Cont.

---

- You can use a similar technique to hide text content.
- As you have seen, the regular method of creating buttons requires there to be text superimposed on the background.
- If you don't want the text to appear, the quick answer is to simply swap out the text for an image and use it as a button.
- However, if you do this, the link is not visible if the image doesn't load or the visitor uses a text-only browser.
- The way to solve this is to push the text out of the way so that only the background image appears.
- Another reason to use this technique is that designers often want to use custom fonts or font effects in their buttons.
- To do this and retain full accessibility, they need to hide the regular HTML text first.
- In this example, you create a button that uses the sliding doors technique and hides the text at the same time.

# Creating Text-Free Buttons with Sliding Doors Cont.

---

- Import the file named *slidingButtons.gif* into the *buttonGraphics* folder from the lesson files.
- Under the three buttons you just created, add a new subheading and call it *Sliding Doors Button* with *Hidden Text*.
- Below it, add a paragraph with the text *Button* and make it into a hyperlink pointing back to the current page.
- Make a new class called *.slider*.
- Create a span around the new button text using the *Toolbox* panel (make sure the *<span>* tags go on the outside of the *<a>* tags), and apply the *.slider* class to it using the *Quick Tag Selector*.

# Creating Text-Free Buttons with Sliding Doors Cont.

---

```
Site View  buttons.html* x
69 <body>
70
71 <p><a class="basicBox" href="buttons.html">Classic box button</a>
72 <p><a href="buttons.html" class="modernBox">Click me</a></p>
73 <ul class="advancedBox">
74   <li><a href="buttons.html">Button 1</a></li>
75   <li><a href="buttons.html">Button 2</a></li>
76   <li><a href="buttons.html">Button 3</a></li>
77 </ul>
78 <p><span class="slider"><a href="buttons.html">Button</a></span><
79 </body>
80
81 </html>
82
```

# Creating Text-Free Buttons with Sliding Doors Cont.

---

- Make a new style and give it the *selector* name *.slider a*.
- Because you are going to hide the text, you don't need change any of the *Font* attributes.
- In the *Background* category, set *background-image* to *slidingButtons.gif* and *background-repeat* to *no-repeat*.
- To create the sliding effect, you need to change the *position* of the background image and define the visible area within the page.
- The image has two buttons on it: The top one is for the regular state, and the bottom one is for the *:hover* state.
- Set the *(x) background-position* to *left* and the *(y) background-position* to *top*.
- This locks the image in place.

# Creating Text-Free Buttons with Sliding Doors Cont.

---

- To define the visible area of the button, you need to first set the *display* attribute under the *Layout* category to *Block* to create an independent box in which to display the content and then change the size of the box under the *Position* category.
- The *height* of the image is *88px*, and because you will be displaying half of it at one time, the height attribute should therefore be *44px*.
- To contain the active area of the button to the area of the image, set the *width* attribute to *92px*, which is the width of the image.
- Click *OK* to apply the style, and the button appears in *Design* view.

# Creating Text-Free Buttons with Sliding Doors Cont.

---



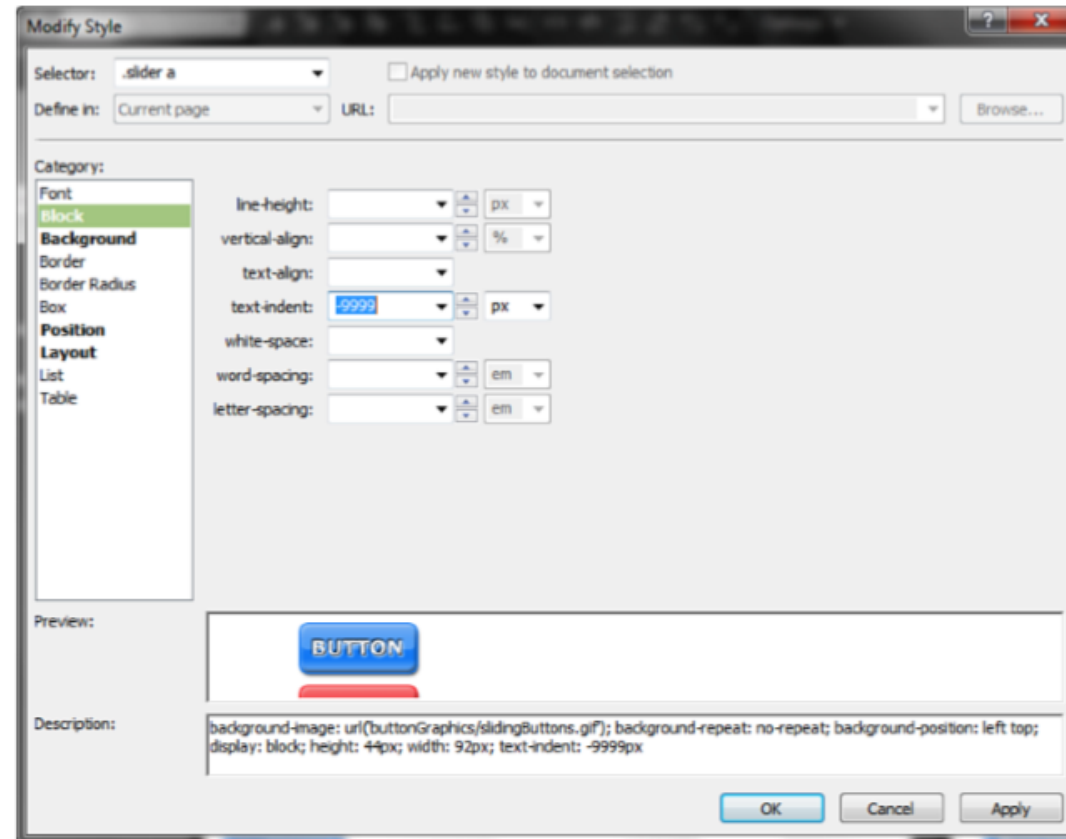


# Creating Text-Free Buttons with Sliding Doors Cont.

---

- As you can see, the button looks the way it should, but the text is still visible.
- To hide the text, open the *Modify Style* dialog for the style, go to the *Block* category, and set the *text-indent* attribute to *-9999px*.
- Click *OK* to save the change.

# Creating Text-Free Buttons with Sliding Doors Cont.



# Creating Text-Free Buttons with Sliding Doors Cont.

---

- Finally, create a new style with the *selector* name *.slider a:hover*.
- Because of the cascade, any pseudoclass inherits all the styling from its parent unless the style is changed.
- Therefore, you need to make changes to only the *background-position* attributes, and the rest of the attributes will stay the same.
- Set (*x*) *background-position* to *0px* and (*y*) *background-position* to *-44px*.
- This shifts the background image up so the bottom of the image aligns with the bottom of the box 44 pixels, and the bottom half is shown in place of the top half.
- Click *OK* and then save and preview in your browser.

# Snippets: An Introduction

---

- One of the many challenges of building websites is the code languages themselves.
- Or more specifically remembering the many complicated syntaxes, formulas, and code configurations used to make certain things happen or create certain effects.
- The code snippets function in Expression Web 4 is a powerful but easily overlooked feature that can and will save you a lot of time and effort once you start using it.
- Out of the box, it comes with a long list of useful snippets for HTML, CSS, JavaScript, jQuery, PHP, and more, and you can create new snippets yourself to serve your specific coding requirements.

# Snippets: An Introduction Cont.

---

- By default, the *Snippets* panel appears in the top-right pane area behind the *Toolbox* tab.
- To see the *Snippets* panel, simply click the *Snippets* tab.



# Snippets: An Introduction Cont.

---

- The *Snippets* panel contains a list of folders, some of which contain snippets and some of which contain further folders.
- The default snippets are stored in folders that correspond with the code language, so CSS snippets live in the CSS folder, HTML snippets in the HTML folder, and so on.
- The snippets are code snippets, so they are used in *Code* view.
- To use a code snippet, you first place your cursor where you want the snippet to appear in *Code* view and then use one of two methods to insert the snippet:
- Either double-click the snippet directly in the *Snippets* panel or press *Ctrl+Enter* to jump to the Snippet search function and then start typing in the name of the snippet you want to use.
- The search function will narrow down your options and display only snippets with names that match what you have written.

# Snippets: An Introduction Cont.

---



# Snippets: An Introduction Cont.

---

- To insert a snippet from a text search, use the arrow keys to highlight the correct snippet and press *Enter*.
- Alternatively, you can simply double-click the highlighted snippet.
- You can also reset your search by clicking the *X* at the right side of the search area.
- When you select a snippet—either by clicking it once with your mouse or by navigating to it with search and the arrow keys—you will see the name, description, and code output for the selected widget in the preview area at the bottom of the *Snippets* panel.
- This gives you the ability to check whether the snippet is the correct one before you insert it.



# Use a Snippet to Create a Custom Tooltip

---

- To see how to use the snippets and understand how to make them work, let's start by creating a custom tooltip function using a *CSS snippet*:
- Create and open a new HTML page called *snippetsDemo.html* in *Split* view.
- At the top of the page, create a paragraph and write *MyKipple.com* is a great place to find information about kipple.
- In *Code* view, scroll to the top and place your cursor right before the end `</head>` tag.
- Press *Enter* to create a few new empty lines.
- In the *Snippets* panel, find the *Style Block* snippet and double-click it to insert it in the `<head>` tag.
- This inserts the standard code block that contains CSS code in HTML pages.
- With your cursor on the empty line between the beginning and end `<style>` tags, press *Ctrl+Enter* to make a snippet search.
- Type *Tooltip*, and one snippet is revealed called *Tooltip with CSS*.
- Press *Enter* or double-click *Tooltip with CSS* to insert the code snippet.

# Use a Snippet to Create a Custom Tooltip Cont.

---



# Use a Snippet to Create a Custom Tooltip

## Cont.

---

- It would be logical to assume that once we've added the snippet, everything should work automatically.
- However, that's not the case.
- Apart from *the jQuery UI Widgets* snippets, which we'll cover later on in this hour, most of the snippets are just starting points for more work.
- Case in point, the *tooltip* snippet does nothing unless we add the *.tooltip* class to some content and create a span to contain the tooltip itself.
- If you select the *Tooltip with CSS* snippet again in the *Snippets* panel and look at the description in the bottom of the panel, you see the following instructions:
- *"If you want to display a tooltip, just add the class 'tooltip' to an element and write your Tooltip Text in a span tag within the element."*

# Use a Snippet to Create a Custom Tooltip Cont.

---

- In *Code* view, find the beginning `<p>` tag and attach the `.tooltip` class.
- Still in *code* view, place your cursor after the word *MyKipple.com*, insert a new `<span>` tag, and write *The home of my kipple*.
- Then end with the `</span>` tag.
- If you click inside *Design* view to refresh the page, you'll see that the new tooltip does not appear.
- But if you save the page and preview it in your browser, the tooltip will appear when you hover over any of the text in the paragraph.
- This is a very basic demo of the *Tooltip with CSS* snippet, and the tooltip itself is not particularly attractive, but it provides you with the building blocks to create advanced tooltips of your own simply by adding a class and a span tag to an element (think image).

# CSS Snippets

---

- The CSS snippets are divided into three main folders: *Action*, *Effect*, and *Text*.
- All the snippets in these folders provide standard CSS markup that can be inserted either in the head of the HTML page itself (contained within a standard `<style>` block provided by the *Style Block* snippet) or in a CSS file or external style sheet.
- To apply a CSS snippet to an element on the page, you attach the snippet class to that element.
- Keep in mind that a lot of these functions are CSS3, meaning they will not work in all browsers.

# CSS – Action

---

- *Replace text with image*—The *.replace* class replaces text with a defined image when text is hovered over.
- *Tooltip with CSS*—The *.tooltip* class displays a tooltip wrapped in a `<span>` tag when the containing element is hovered over.

# CSS – Effect

---

- *Box Shadow*—The *.boxShadow* class adds a box shadow to the containing box. The size and color can be changed by changing the values.
- *Circle*—The *.circle* class draws a circle with CSS. The size and color can be changed by changing the values.
- *Horizontal-Vertical align*—The *.align* class aligns the element to the horizontal and vertical center of the browser window.
- *Image Button*—The *.imageButton* class attached to the input button element replaces the default button with a custom image of your choice.
- *Opacity*—The *.opacity* class sets the opacity (transparency) of the element. *0.0* or *0* is fully transparent; *1* or *100* is fully opaque.
- *Rotate Box*—The *.rotate* class rotates the element box by degrees.
- *Rounded Corners*—The *.round\_border* class adds rounded corners to the element box.

# CSS – Text

---

- *Change Color of Selected Text*—Globally changes the color of text across the entire page when highlighted with a mouse.
- *Disable Select Text*—The `.unselect` class prevents the visitor from selecting text in the element for copying and such.
- *First letter in color and uppercase*—Globally sets the first letter of each element red and uppercase. Can be specified by changing style element from `:first-letter` to `p:first-lettedsdf`, for example.
- *Text Ellipsis*—The `.ellipsis` style adds an ellipsis (...) to the end of text when not all text is displayed due to overflow (the box is too small to display all text).
- *Text Shadow*—The `.text-shadow` class adds text shadow. Limited browser support at this time.
- *Text Outline*—The `.outline` class uses the text-shadow attribute and white text to create the appearance of outlined text.



# Doctypes Snippets

---

- The *Doctypes* snippets provide the default doctype calls for the standard doctypes used on the Web today.
- They are as follows:
- HTML 4.01 Strict
- HTML 4.01 Transitional
- HTML 5 (default in Expression Web 4)
- XHTML 1.0 Strict
- XHTML 1.0 Transitional
- XHTML 1.1
- The doctype is inserted as the first line of code on a page.
- A page should never have more than one doctype.

# HTML Snippets

---

- The HTML snippets are a mixed bag of `<head>` and `<body>` snippets:
- *Definition List*—Add in `<body>`. Inserts a standard definition list in the body of the page. The `<dt>` tag wraps the definition keyword while the `<dd>` tag wraps the definition itself.

# HTML – Hyperlink

---

- To be added in the `<body>` of the page:
- *Add to favorites*—Creates an “Add to favorites” (that is, bookmark this page) link to your page.
- *Mail to link*—Creates a click-to-email link. Replace “@@email@@” with the target email address.
- *Open in new window*—Creates a link that opens the target in a new page using JavaScript.

# HTML – Meta

---

- To be added in the *<head>* of the page:
- *HTML 5 UTF-8 Meta*—Sets the character set of the page to UTF-8.
- *Meta keywords*—Adds meta keywords to the page. The cursor is automatically placed where keywords are to be added.
- *Meta refresh*—Tells the browser to auto-refresh the page and go to a different URL after a set time. Can also be used to refresh the page. Commonly used on newspaper sites to ensure new content is always displayed.

# HTML – IE Meta Tags

---

- IE meta tags are used to force various versions of Internet Explorer to interpret the content of the page in accordance with different standards.
- These are highly specialized functions most often used when deploying old code or when targeting old browsers.
- They are added at the very top of the `<head>` tag before any files such as style sheets are called.

# JavaScript Snippets

---

- The *JavaScript* snippets contain most normal functions, elements, and control structures used when writing *JavaScript*, in addition to a few handy conversion functions that are fully built out.
- Combined with the *JavaScript IntelliSense* support in Expression Web 4, you have a fully stocked and powerful JavaScript coding application right at your fingertips.

# jQuery Snippets

---

- *jQuery* is a popular *JavaScript* library that makes it easy to perform complicated actions and add cool effects and functionality to your site.
- *jQuery* is open source, and a large community of jQuery enthusiasts on the Web publishes tutorials, plugins, and information about the library and how to use it.

# jQuery – General

---

- To run *jQuery* scripts, you must first call the *jQuery* library itself from within the page.
- The location of the library can be your own site or a Content Delivery Network.
- Microsoft hosts several versions of jQuery you can reference, and so does Google.
- The *General jQuery* snippets call the *jQuery* or *jQuery UI* libraries from different locations:
- *jQuery script reference – MS CDN*—Calls the jQuery library from Microsoft’s Content Delivery Network.
- *jQuery UI script reference – Local*—Calls the jQuery UI library from a location on your domain or one of your choosing.
- *jQuery UI script reference – MS CDN*—Calls the jQuery UI library from Microsoft’s Content Delivery Network.
- *Document Ready*—Inserts a generic document-ready block. The document-ready block will wrap (contain) all the jQuery code and ensure that it is running when the rest of the content on the page is loaded.



# jQuery – Forms

---

- The *Forms* snippets under jQuery allow you to add automated functionality to your forms.
- Once you've added a jQuery library, you can insert any or all of these *Forms* snippets in the *Document Ready* snippet for enhanced functionality.
- Each snippet has a description explaining the functionality.
- Most of them are designed to target form elements with specific IDs that are inserted in the snippet.

# jQuery – Styling

---

- *jQuery* is often used to change styling attributes on elements either by adding, changing, or removing classes or by performing other actions on the elements that can't be applied with static CSS or HTML.
- *Add-Remove class on hover*—Adds a class named `.hover` to the element with the defined ID when it is hovered over. The `.hover` class is removed when the item is no longer being hovered over.
- *Highlight input label*—For use in forms. Highlights the form label of the defined element when the form field is in focus.
- *Zebra stripes*—Adds a class named `.oddClass` to every odd element in lists and tables. Combined with CSS, this can produce a nice zebra stripe look to the list or table for easier reading.

# PHP Snippets

---

- As with the *JavaScript snippets*, the *PHP snippets* offer up a long list of standard elements, functions, and control structures for writing PHP.

# Stray Snippets

---

- In addition to all the snippets just listed, there are two stray snippets that are not added to a folder.
- You have already used one of them—the *Style Block*—but the other one is so unusual it requires special mention:
- *Web slice*—The web slice is a technology introduced with Internet Explorer 8 that allows the website owner to create a mini version of the page that a visitor can pin to her Favorites bar in the browser and check later.
- The snippet will introduce into the page the necessary code to make such a web slice work.

# Using the jQuery UI Widgets Snippets

---

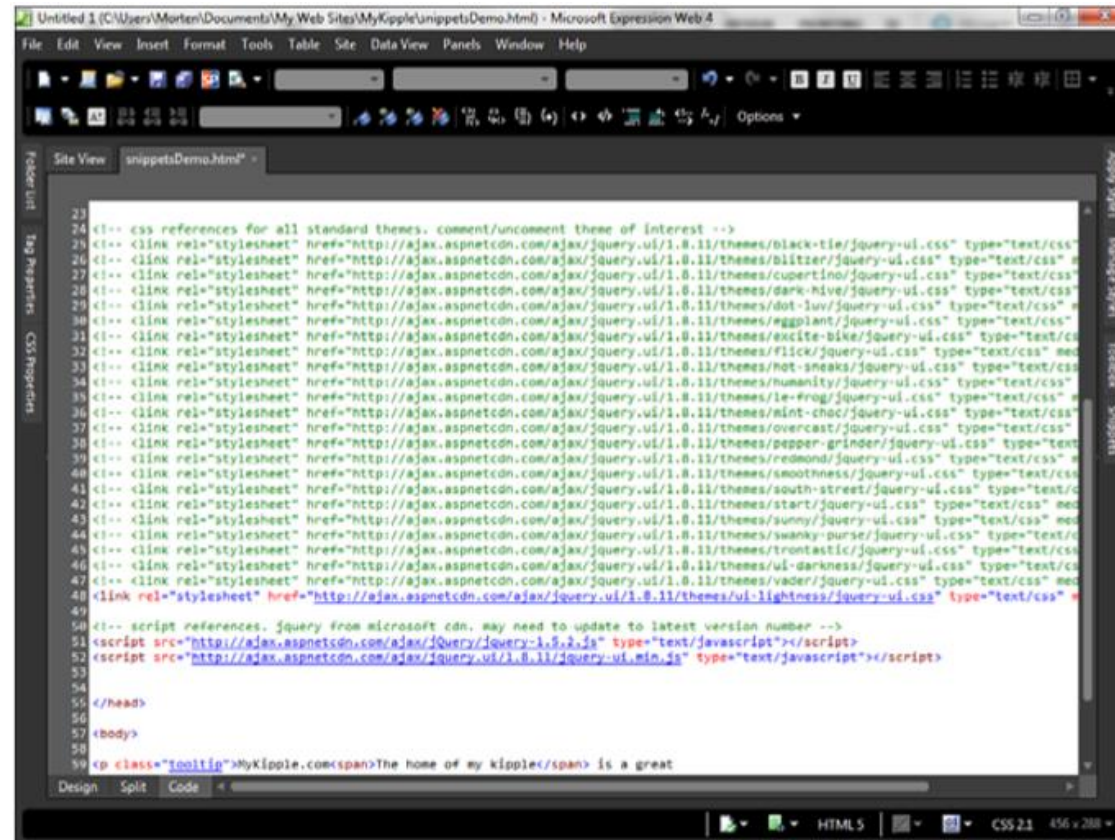
- *jQuery UI* is a variant of the standard jQuery library that allows you to do some pretty nifty things with minimal effort.
- The *jQuery UI* library has a series of widgets built in that you can hook into and create some very advanced effects on your site with only a small amount of code.
- And Expression Web 4 ships with three of these widgets all built out and ready to use.
- They are an *accordion*, a *tabbed box*, and an *interactive calendar*.
- All these three widgets can be dropped in anywhere on your site and used out of the box or with only minor configurations.
- But before you can use these widgets, you have to add the jQuery UI head to your page!

# Using the jQuery UI Widgets Snippets Cont.

---

- In *Code* view, scroll to the top of the page and create a new line right before the end `</head>` tag.
- In the *Snippets* panel under *jQuery, UI Widgets*, find and click the *jQuery UI head widget*.
- This inserts 30 lines of code, most of which is commented out.

# Using the jQuery UI Widgets Snippets Cont.



The screenshot shows the Microsoft Expression Web 4 interface with a code editor displaying HTML code for jQuery UI themes. The code includes comments and links for various themes, followed by script references for jQuery and jQuery UI. The code is as follows:

```
23 <!-- css references for all standard themes. comment/uncomment those of interest -->
24 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/black-tie/jquery-ui.css" type="text/css"
25 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/blitzer/jquery-ui.css" type="text/css"
26 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/cupertino/jquery-ui.css" type="text/css"
27 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/dark-hive/jquery-ui.css" type="text/css"
28 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/dot-luv/jquery-ui.css" type="text/css"
29 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/eggplant/jquery-ui.css" type="text/css"
30 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/excite-bike/jquery-ui.css" type="text/css"
31 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/flick/jquery-ui.css" type="text/css"
32 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/hot-sneaks/jquery-ui.css" type="text/css"
33 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/humanity/jquery-ui.css" type="text/css"
34 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/le-frog/jquery-ui.css" type="text/css"
35 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/mint-choc/jquery-ui.css" type="text/css"
36 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/overcast/jquery-ui.css" type="text/css"
37 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/pepper-grinder/jquery-ui.css" type="text/css"
38 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/redmond/jquery-ui.css" type="text/css"
39 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/smoothness/jquery-ui.css" type="text/css"
40 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/south-street/jquery-ui.css" type="text/css"
41 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/start/jquery-ui.css" type="text/css"
42 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/sunny/jquery-ui.css" type="text/css"
43 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/swanky-purse/jquery-ui.css" type="text/css"
44 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/trontastic/jquery-ui.css" type="text/css"
45 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/ui-darkness/jquery-ui.css" type="text/css"
46 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/vader/jquery-ui.css" type="text/css"
47 <!-- <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/ui-lightness/jquery-ui.css" type="text/css"
48 <link rel="stylesheet" href="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/themes/ui-lightness/jquery-ui.css" type="text/css"
49
50 <!-- script references. jquery from microsoft cdn. may need to update to latest version -->
51 <script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.8.2.js" type="text/javascript"></script>
52 <script src="http://ajax.aspnetcdn.com/ajax/jquery/ui/1.8.11/jquery-ui.min.js" type="text/javascript"></script>
53
54
55 </head>
56
57 <body>
58
59 <p class="tooltip">MyKippie.com<span>The home of my kippie</span> is a great
```

# Using the jQuery UI Widgets Snippets

## Cont.

---

- The snippets consist of three elements:
- Calls to all the standard theme style sheets for jQuery UI (all the commented out lines + the one active style sheet reference), a call to the jQuery library as hosted by Microsoft CDN, and a call to the jQuery UI library also hosted by Microsoft CDN.
- When you add a jQuery UI Widget, the *jQuery* and *jQuery UI* libraries will kick in and the widget will use the theme style sheet for the presentation elements.



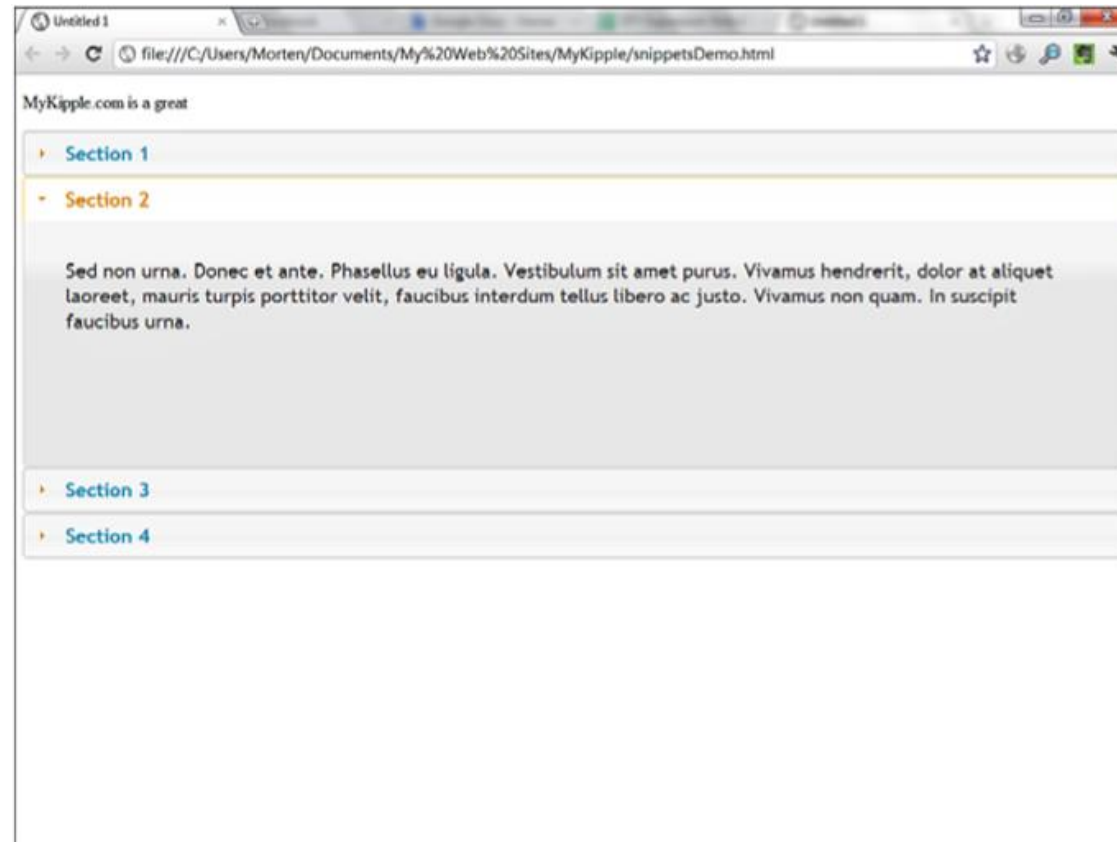
# Create a Dynamic Accordion Effect with the jQuery UI Widgets Snippet

---

- Now that you have the *jQuery UI head* installed in your page, you can use the *jQuery UI Widget snippets* to add some cool functions to it:
- In *Code* view, scroll down and place your cursor directly underneath the paragraph you created earlier.
- Press *Ctrl+Enter* to go to *Snippet search* and type *Accordion*.
- This highlights the *Accordion jQuery UI Widget* snippet.
- Press *Enter* to insert the code snippet into your page.
- Save the page and preview it in your browser.

# Create a Dynamic Accordion Effect with the jQuery UI Widgets Snippet Cont.

---



# Create a Dynamic Accordion Effect with the jQuery UI Widgets Snippet Cont.

---

- The accordion works by revealing the text and other content within each panel and collapsing the other open panel when you click the corresponding tab.
- Going back to Expression Web and looking at the source code, you'll see that the accordion is in fact nothing more than a series of nested divs and headings marked up with standard HTML.
- That means you can change the content of the accordion.
- If you want to change the appearance of the accordion itself, you can do so either by commenting out the active stylesheet link in the *jQuery UI head* and uncommenting one of the many other default style sheets or by writing your own stylesheet based on the default ones.

# Calendar and Tabs Widgets

---

- The *Calendar* and *Tabs jQuery UI Widgets* snippets work much the same way as the *Accordion* snippet.
- You add them to the page the same way and they are controlled by the same style sheet and the same *jQuery* and *jQuery UI* reference.
- Therefore, even if you have an *accordion*, a *calendar*, and a *tab* on the same page, you only need one *jQuery UI head*.
- Also, if you change the style sheet, all three widgets will change appearance at the same time.

# Calendar and Tabs Widgets Cont.

---

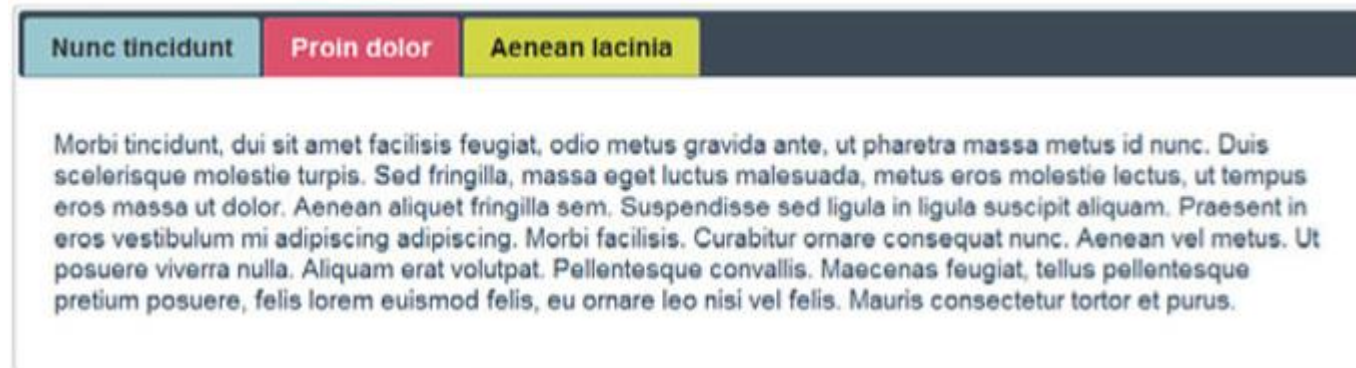
- The *Calendar* widget adds an interactive calendar to a date form field or date field on the page.
- When the user clicks the field, the calendar opens up so that the user can select a date without punching in any numbers.



# Calendar and Tabs Widgets Cont.

---

- The *Tabs* widget can be seen as a horizontal version of the *accordion* widget.
- When added in the page, it creates a tabbed box where the contents of each tab are displayed when the tab is clicked.
- Like with the *accordion*, the HTML code is straightforward and easy to edit and configure.



# Creating and Editing Snippets

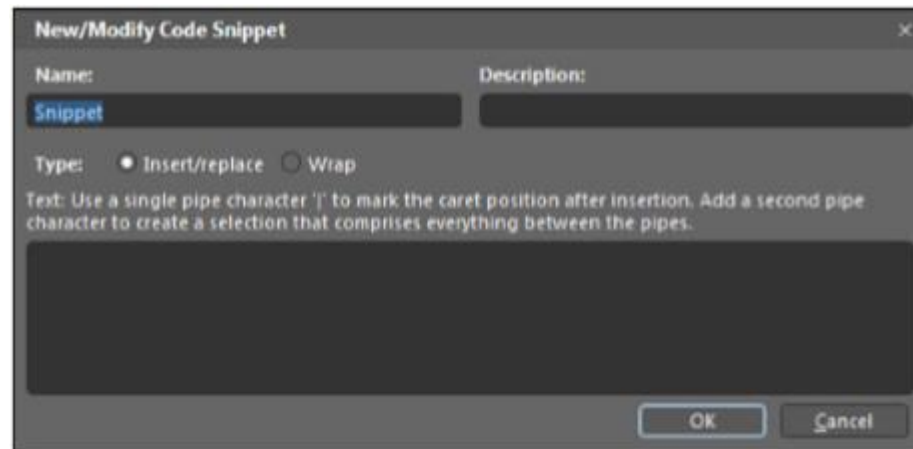
---

- In addition to using the default snippets that ship with Expression Web 4, you have the ability to create your own custom snippets to add to your preconfigured coding arsenal.
- That way, as you encounter code snippets you use a lot, you can add them to the panel for easy access later.
- Adding folders and snippets is done from the *Snippets* panel.

# Add a New Snippet

---

- Go to the *Snippets* panel and make sure every folder is collapsed and that no folder is selected.
- Click the *Options* button and select *New Folder* from the drop-down menu.
- This creates a new folder called *Snippet*.
- Right-click the *new folder*, select *Rename*, and rename it *Custom Snippets*.
- With the new folder selected, click *Options* again and select *New Snippet*.
- This opens the *New/Modify Code Snippet* panel.





# Add a New Snippet Cont.

---

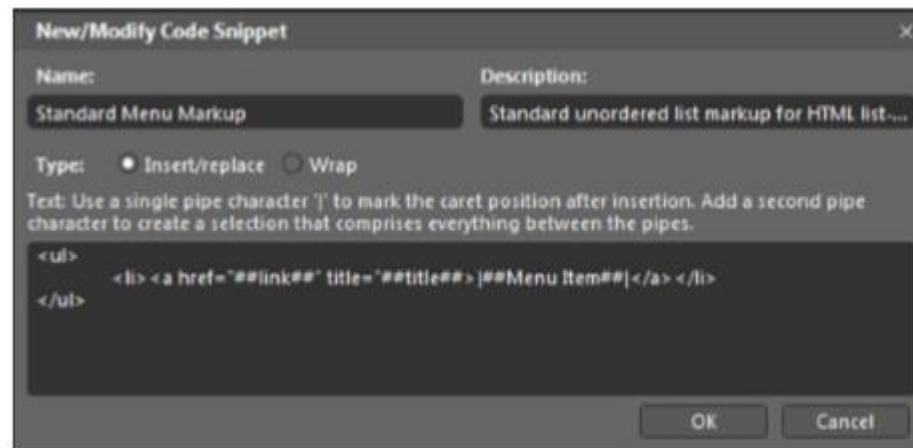
- Give the new snippet the name *Standard Menu Markup* and the description *Standard unordered list markup for HTML list-based menu*.
- Under *Type*, leave *Insert/Replace* checked.
- In the text area, write out the code for a standard unordered list with one list item with a link inside, as follows (the ## segments indicate areas that must be edited once the list is inserted):

```
<ul>  
<li><a href="##link##" title="##title##">##Menu Item##</a></li>  
</ul>
```

# Add a New Snippet Cont.

---

- To tell Expression Web where to put the cursor, you can insert a pipe character (|) anywhere in the snippet.
- If you want Expression Web to highlight a portion of the snippet for immediate replacement, you can wrap it in pipe characters.
- In this case, we want Expression Web to immediately highlight the *##Menu Item##* section so that when you start typing, you are typing in the menu item name.
- Therefore, place a pipe character on either side or both sides, like this: |##Menu Item##|
- Click *OK* to create the new snippet.



# Add a New Snippet Cont.

---

- Now that the snippet is made, you can insert it anywhere in your code by double-clicking it from the *Snippets* panel.
- When you insert the snippet, the *##Menu Item##* text is immediately highlighted.
- If you start typing, it is replaced with the new text.
- Editing an existing snippet is done using the same dialog.
- To edit an existing snippet, either right-click the snippet and select *Edit Snippet* from the menu or select the snippet and click the *Edit* button at the top of the preview area in the *Snippets* panel.

# Building a Functional Menu

---

- Menus are a great tool for simplifying navigation in a website.
- They enable you to group several different pages or links together visually and give the user an intuitive and interactive experience when surfing your site.
- There are many types of menus—from the standard vertical list menu to horizontal menus to drop-down menus of all shapes and sizes.

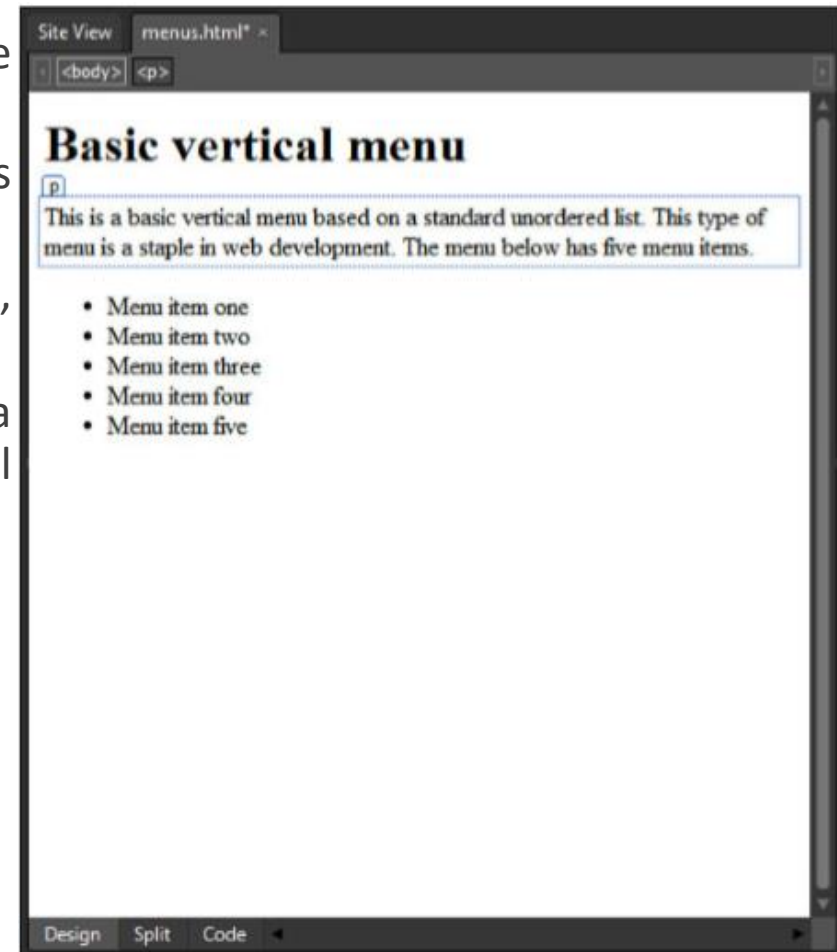
# Making the Basic Vertical Menu Exciting

---

- The most basic standards-based menu available is the vertical unordered list.
- Not only is it easy to build (all you need to do is create an unordered list in which each list item is a link), but it is also easy to maintain, and it works no matter what browser the visitor uses.
- Now, create a new folder called *menus* with a page called *menus.html* to work from.

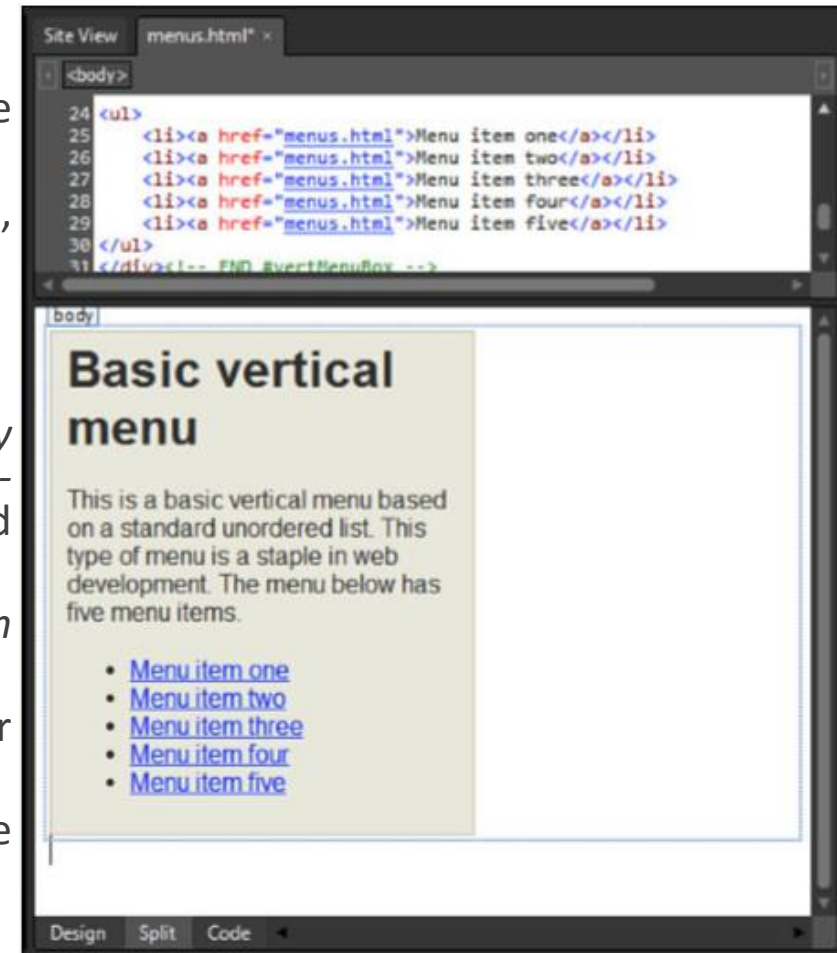
# Making the Basic Vertical Menu Exciting Cont.

- If you haven't already done so, create and open the *menus.html* page in *Design* view.
- Create a standard unordered list with five menu items named *Menu Item One*, *Menu Item Two*, and so on.
- Directly above the unordered list, insert a new paragraph, type *Basic Vertical Menu*, and change the style to *H1*.
- Press *Enter* to create a new paragraph and then insert a couple of lines of text explaining that this is a basic vertical menu based on an unordered list.



# Making the Basic Vertical Menu Exciting Cont.

- The whole idea of a menu is to create navigation.
- To do so, make each of the menu items a hyperlink back to the *menus.html* page.
- Switch to *Split* view and create a div to wrap the heading, paragraph, and list.
- Give the div the ID *vertMenuBox*.
- Now it's time for the styling.
- First, create a new ID style for *#vertMenuBox* and set *font-family* to *Arial, Helvetica, sans-serif*; *color* to *#333333*; *background-color* to *#EBEADF*; and all four *borders* to *solid, 1px*, and *#C4C2AB*.
- To give the content some breathing space, set *top* and *bottom padding* to *5px*, and *right* and *left padding* to *10px*.
- Finally, to constrain the size of the menu, set *width* (found under *Position*) to *250px*.
- Click *OK*, and you see that the menu is now contained in a nice box with a beige color.



# Making the Basic Vertical Menu Exciting Cont.

---

- To remove the list bullets and align the menu items with the left side of the box, create a new style with the selector `#vertMenuBox ul` and set the *padding* and *margin* values to `0px` and the *list-style-type* under *List* to *none*.
- Click *OK* to apply the new style.
- Next up are the links.
- Create a new style with the selector `#vertMenuBox ul li a` and set *font-size* to `0.8em`, *text-transform* to *uppercase*, *color* to `#666666`, and check *None* under *text-decoration*.
- To style each of the links so they are farther apart, set the *top* and *bottom padding* to `5px` and create a further visual separation by setting *border-bottom* to *dotted*, `1px`, and `#333333`.
- Finally, set *display* to *block* to make the links into block-level elements.
- Click *OK* to apply the new style.
- Create a new style with the selector `#vertMenuBox ul li a:hover` and set *color* to `#333333`.
- Click *OK*, save the page, and test it in your browser.



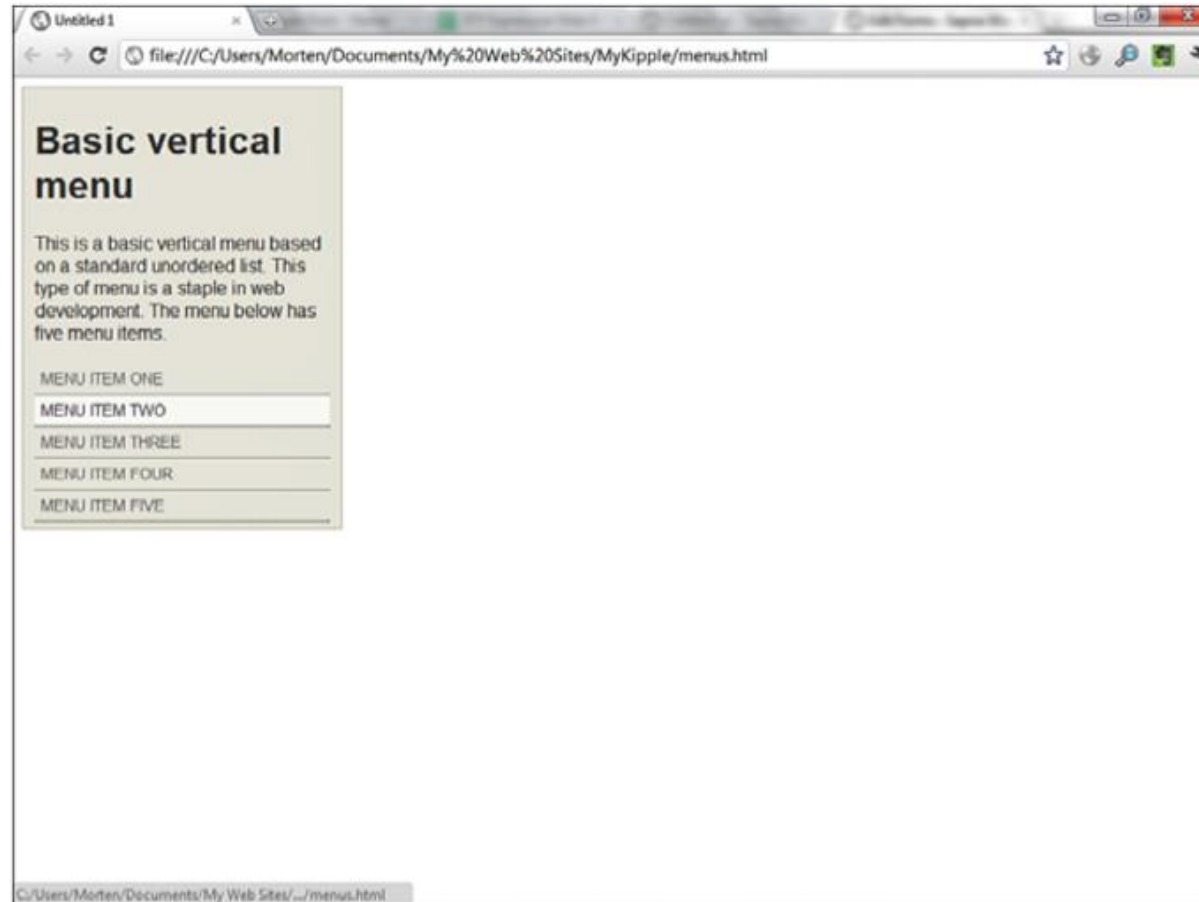
# Making the Basic Vertical Menu Exciting Cont.

---

- Now, you have a basic vertical menu with some styling that makes it look less boring. But it's still pretty basic. To make it stand out, add a couple of extra style tricks to the menu.
- Open the *Modify Style* dialog for the `#vertMenuBox ul li a` style and set *padding-left* to `5px`.
- Click *OK* to apply the change.
- Open the style with the selector `#vertMenuBox ul li a:hover` and set *background-color* to `#FAFAF5`.
- Click *OK* to apply the new style, save, and test the page in your browser.
- By setting the list item links to block-level elements and giving them a lighter background color on the hover state, you are both making the menu buttons easier to click because they are larger and making them more visually attractive.

# Making the Basic Vertical Menu Exciting Cont.

---



# The Horizontal Menu—Laying a List on Its Side

---

- As you saw in the preceding exercise, making a vertical menu based on an unordered list look interesting with CSS isn't all that hard.
- However, you don't always want your menus to be vertical.
- In many cases, a horizontal menu looks better and is more functional.
- But how do you create a horizontal menu from an unordered list when the unordered list by its very nature is a list, and therefore vertical?
- The answer is simple.
- Lay it on its side!

# The Horizontal Menu—Laying a List on Its Side Cont.

---

- Below the *Basic Vertical List box* you just created (and outside the box), create a new unordered list with the same five list items as before, and make each a link back to the current page.
- In *Split* view, wrap the new list in a div with the ID `#horizontalMenuBox`.
- Create a new style with the selector `#horizontalMenuBox` and set *font-family* to *Arial, Helvetica, sans-serif*; *background-color* to `#EBEADF`; *border* to *solid, 1px, #C4C2AB*; and *margin-top* to *1em* to create some space between the *Basic Vertical Menu* and the new one you are about to create.
- Create a new style with the selector `#horizontalMenuBox ul` and set *padding* and *margin* to *0px* for all sides.
- To get the menu items to appear on one line instead of as a list, create a new style with the selector `#horizontalMenuBox ul li` and set *display* to *inline* under the *Layout* category.
- Click *Apply*, and you see that the list items are now on one line.

# The Horizontal Menu—Laying a List on Its Side Cont.

The screenshot displays a web editor interface for a file named 'menus.html'. The top section shows the HTML source code, with lines 67 through 80. The code defines a horizontal menu box with five items, each linked to 'menus.html'. The bottom section shows the rendered design view, which includes a text box explaining that this is a standard unordered list and a visual representation of the horizontal menu with five items: 'Menu item one' through 'Menu item five'.

```
67 <li><a href="menus.html">Menu item five</a></li>
68 </ul>
69 </div><!-- END #vertMenuBox -->
70 <div id="horizontalMenuBox">
71 <ul>
72 <li><a href="menus.html">Menu item one</a></li>
73 <li><a href="menus.html">Menu item two</a></li>
74 <li><a href="menus.html">Menu item three</a></li>
75 <li><a href="menus.html">Menu item four</a></li>
76 <li><a href="menus.html">Menu item five</a></li>
77 </ul>
78 </div>
79 </body>
80
```

on a standard unordered list. This type of menu is a staple in web development. The menu below has five menu items.

MENU ITEM ONE  
MENU ITEM TWO  
MENU ITEM THREE  
MENU ITEM FOUR  
MENU ITEM FIVE

Menu item one Menu item two Menu item three Menu item four Menu item five

# The Horizontal Menu—Laying a List on Its Side Cont.

---

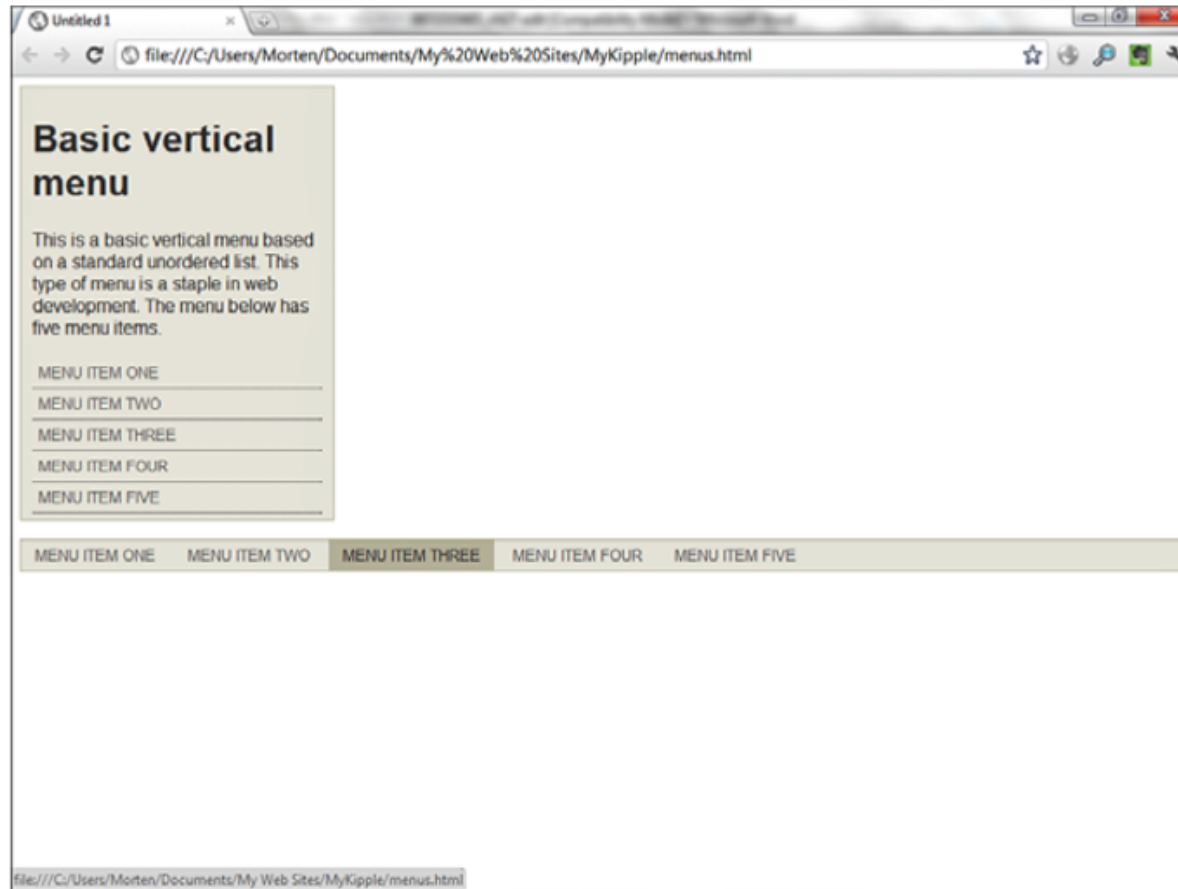
- To give the menu some more weight, you want to create some vertical space above and below the links.
- However, if you go to *Box* and set *padding-top* and *padding-bottom* to *10px* and click *Apply*, you see that nothing changes.
- To change the line height and thereby the vertical space above and below the list items, go to the *Block* category and set *line-height* to *25px*.
- Click *OK* to apply the style.
- Right now, the menu items are stacked almost directly against one another.
- To give them some breathing room, create a new style with the selector *#horizontalMenuBox ul li a* and set *padding-right* and *padding-left* to *1em*. (Step 6)
- To match the vertical space created earlier, set *padding-top* and *padding-bottom* to *5px*.
- At the same time, set the *font-size* to *0.8em*, *text-transform* to *uppercase*, *color* to *#666666*, and *text-decoration* to *none*.
- Click *Apply* to see the changes take effect.

# The Horizontal Menu—Laying a List on Its Side Cont.

---

- Finally, to make the menu items react when the visitor hovers her mouse over them, create a new style with the selector *#horizontalMenuBox ul li a:hover* and set *color* to *#333333* and *background-color* to *#C4C2AB*.
- Click *OK* to apply the new style, save, and test the page in your browser.
- As you can see, the items in the horizontal menu react in much the same way as in the vertical one, with one major difference: The entire hover area is a link.
- This is because you used the padding in step 6 to expand the link area to cover an area larger than the text itself.
- You can use the same technique to make the entire area of the vertical menu into a link.

# The Horizontal Menu—Laying a List on Its Side Cont.





# Pure CSS Drop-Down Menus: A Clean Alternative

---

- Now that you've created a vertical and a horizontal menu, it is time to merge the two into a more advanced drop-down menu in which the main menu is horizontal and the submenus are vertical.
- As you have learned, if you want your page to be as accessible as possible and standards-compliant and cross-browser compatible, you must base all of your menus on ordered or unordered lists.
- That way, the menu is still meaningful if the visitor is on an old computer or uses a text-only or text-to-speech browser.
- The ideal solution for making drop-down menus should be to create an unordered list with sub-lists styled using CSS.

# Step 1: Make a Menu List

---

- To start things off, we need to mark up the actual links we are going to convert to menu buttons.
- This is done using an unordered list.
- In the *menus.html* page, insert a new div underneath the code for the horizontal menu and give it the ID *#dropDownMenu*.
- Inside the new div, create another copy of the unordered list menu from above by copying and pasting everything within the `<ul>` tags.

# Step 1: Make a Menu List Cont.

---

```
Site View  menus.html* x
67
68
69 <body>
70 <div id="vertMenuBox">
71 <h1>Basic vertical menu</h1>
72 <p>This is a basic vertical menu based on a standard unordered list. This
73 of menu is a staple in web development. The menu below has five menu ite
74 <ul>
75 <li><a href="menus.html">Menu item one</a></li>
76 <li><a href="menus.html">Menu item two</a></li>
77 <li><a href="menus.html">Menu item three</a></li>
78 <li><a href="menus.html">Menu item four</a></li>
79 <li><a href="menus.html">Menu item five</a></li>
80 </ul>
81 </div><!-- END #vertMenuBox -->
82 <div id="horizontalMenuBox">
83 <ul>
84 <li><a href="menus.html">Menu item one</a></li>
85 <li><a href="menus.html">Menu item two</a></li>
86 <li><a href="menus.html">Menu item three</a></li>
87 <li><a href="menus.html">Menu item four</a></li>
88 <li><a href="menus.html">Menu item five</a></li>
89 </ul>
90 </div>
91
92 <div id="dropDownMenu">
93 <ul>
94 <li><a href="menus.html">Menu item one</a></li>
95 <li><a href="menus.html">Menu item two</a></li>
96 <li><a href="menus.html">Menu item three</a></li>
97 <li><a href="menus.html">Menu item four</a></li>
98 <li><a href="menus.html">Menu item five</a></li>
99 </ul>
100 </div>
101 </body>
102
103 </html>
104
```

# Step 1: Make a Menu List Cont.

---

- Create sub-lists for three of the new menu items.
- In *Split* view, select the *Menu Item Two* list item.
- In *Code* view, place the cursor right before the closing `</li>` tag and then press *Enter* to create a new line.
- Type `<ul>` to create a new unordered list (*IntelliSense* closes the tag for you), press *Enter* again to create a new line, and create a sub-list item by typing `<li>`.
- Create three sub-list items named *Sub List Item One*, *Sub List Item Two*, and so on.

# Step 1: Make a Menu List Cont.

The screenshot shows a web editor with two panes. The top pane displays the HTML code for a menu, and the bottom pane shows the rendered output.

```
92 <div id="dropDownMenu">
93   <ul>
94     <li><a href="menus.html">Menu item one</a></li>
95     <li><a href="menus.html">Menu item two</a>
96       <ul>
97         <li>Sub List Item One</li>
98         <li>Sub List Item Two</li>
99         <li>Sub List Item Three</li>
100      </ul>
101     </li>
102     <li><a href="menus.html">Menu item three</a></li>
103     <li><a href="menus.html">Menu item four</a></li>
104     <li><a href="menus.html">Menu item five</a></li>
105   </ul>
106 </div>
```

The rendered output shows a menu structure with five main items. The first two items, 'Menu item one' and 'Menu item two', are highlighted with a light green background. 'Menu item two' has a sub-menu with three items: 'Sub List Item One', 'Sub List Item Two', and 'Sub List Item Three'. The other three main items are 'Menu item three', 'Menu item four', and 'Menu item five'. The rendered output is as follows:

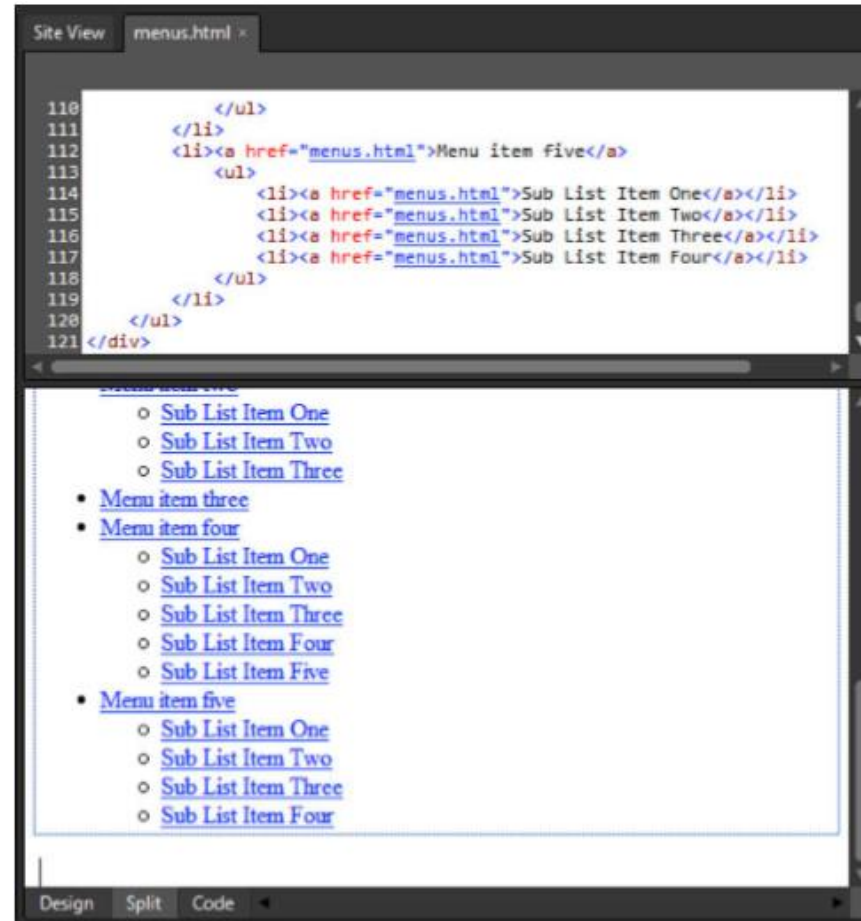
- MENU ITEM ONE
- MENU ITEM TWO
  - Sub List Item One
  - Sub List Item Two
  - Sub List Item Three
- MENU ITEM THREE
- MENU ITEM FOUR
- MENU ITEM FIVE

# Step 1: Make a Menu List Cont.

---

- Now that you have a sub-list, go back to *Design* view and make all the new items into links that point back to the current page.
- Repeat the same steps for two of the other main menu items.

# Step 1: Make a Menu List Cont.



```
110     </ul>
111   </li>
112   <li><a href="menus.html">Menu item five</a>
113     <ul>
114       <li><a href="menus.html">Sub List Item One</a></li>
115       <li><a href="menus.html">Sub List Item Two</a></li>
116       <li><a href="menus.html">Sub List Item Three</a></li>
117       <li><a href="menus.html">Sub List Item Four</a></li>
118     </ul>
119   </li>
120 </ul>
121 </div>
```

- [Menu item three](#)
  - [Sub List Item One](#)
  - [Sub List Item Two](#)
  - [Sub List Item Three](#)
- [Menu item four](#)
  - [Sub List Item One](#)
  - [Sub List Item Two](#)
  - [Sub List Item Three](#)
  - [Sub List Item Four](#)
  - [Sub List Item Five](#)
- [Menu item five](#)
  - [Sub List Item One](#)
  - [Sub List Item Two](#)
  - [Sub List Item Three](#)
  - [Sub List Item Four](#)

# Step 2: Style the Main Menu

---

- Now you have a working set of hyperlinks, but it looks nothing like a menu.
- The next step is to turn the menu into a horizontal one like you did in the menu previously in this hour.
- Create a new style with the ID *#dropDownMenu*.
- Set *font-family* to *Arial, Helvetica, sans-serif*; *background-color* to *#EBEADF*; *border* to *solid, 1px, #C4C2AB*; and *margin-top* to *1em* to create some space.
- Create a new style with the selector *#dropDownMenu ul* and set *padding* and *margin* to *0px* for all sides.
- Note that, in *Design* view, all the list items line up to the left side of the box regardless of what level they are at.
- To ensure that the menu retains its height, go to *Position* and set *height* to *25px*.

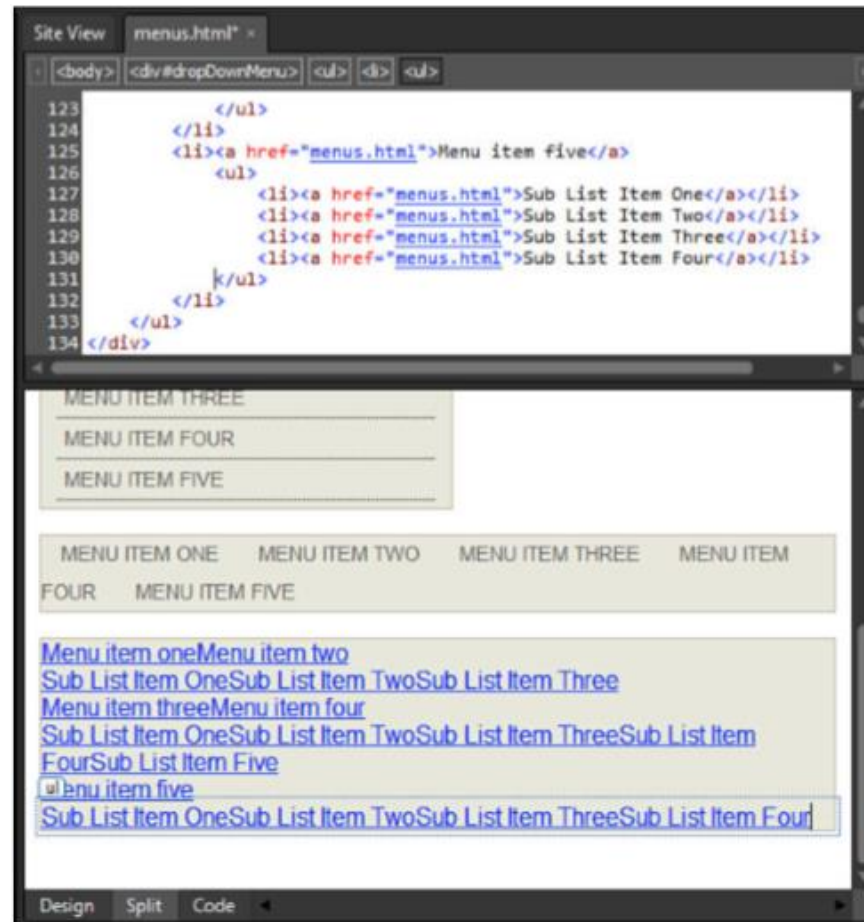


# Step 2: Style the Main Menu Cont.

---

- Create a new style with the selector *#dropDownMenu ul li* and set *display* to *inline* under the *Layout* category.
- In the preceding menu, this was enough to align the menu items left to right rather than top to bottom, but if you click *Apply*, you see that the list items still stack vertically.
- This is because the sub-lists are too wide to fit on one line and are considered one item.

# Step 2: Style the Main Menu Cont.



The screenshot shows a web development IDE with two panes. The top pane displays HTML code for a menu structure, and the bottom pane shows the rendered output.

```
123     </ul>
124   </li>
125   <li><a href="menu.html">Menu item five</a>
126     <ul>
127       <li><a href="menu.html">Sub List Item One</a></li>
128       <li><a href="menu.html">Sub List Item Two</a></li>
129       <li><a href="menu.html">Sub List Item Three</a></li>
130       <li><a href="menu.html">Sub List Item Four</a></li>
131     </ul>
132   </li>
133 </ul>
134 </div>
```

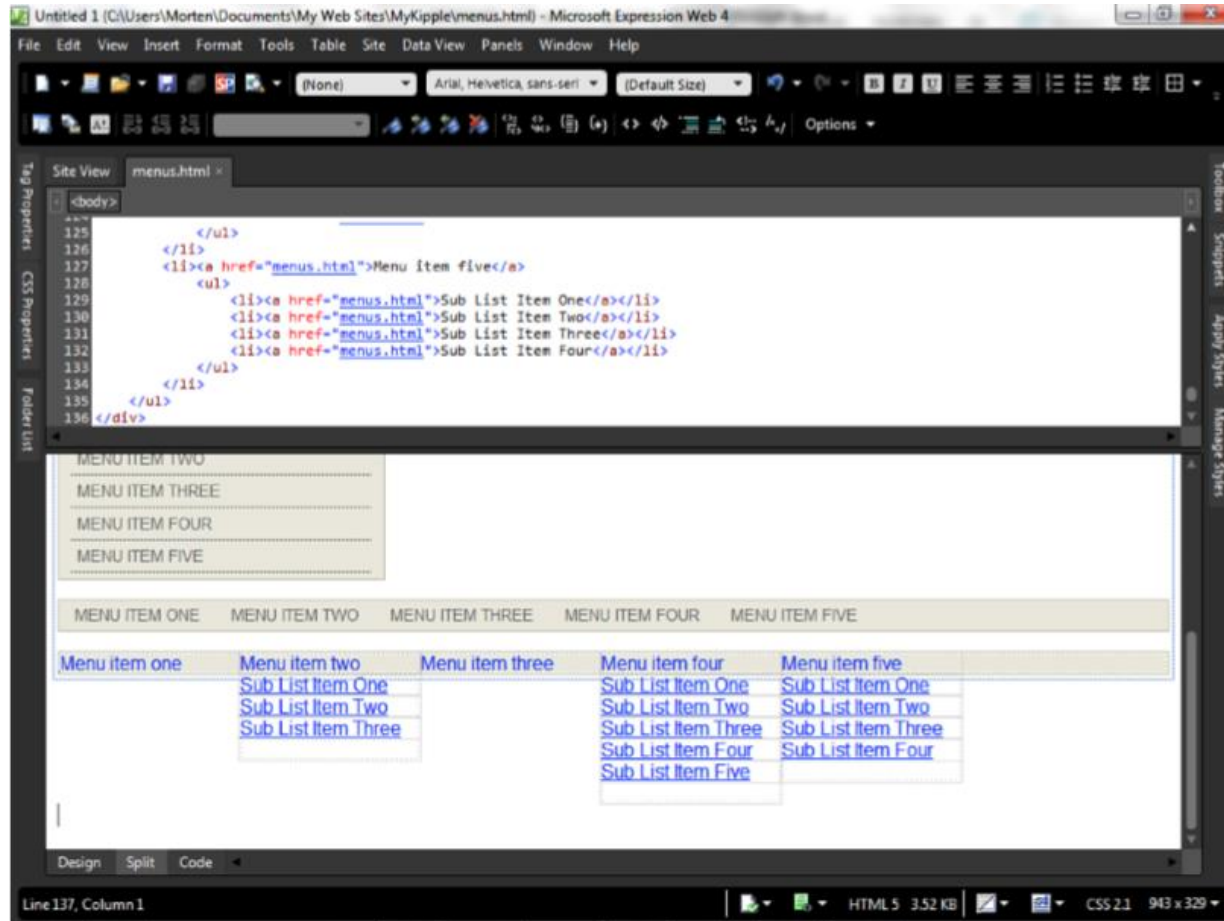
The rendered output shows three menu styles: a vertical list, a horizontal list, and a list with sub-lists. The sub-lists are rendered as separate lines of text, each preceded by a small square icon.

# Step 2: Style the Main Menu Cont.

---

- To get the new menu to stack properly, you need to do two things:
- Define a specific width for each list item and make them float to the left.
- Still in the *#dropDownMenu ul li* style, set *width* under *Position* to *150px*.
- You can set it wider if you want, but any narrower and the text in the submenu items won't fit.
- Then, go to the *Layout* category and set *float* to *left*.
- Click *Apply* again, and you see that now the main menu items line up from left to right, and the submenu items appear as vertical lists under their respective parent item.

# Step 2: Style the Main Menu Cont.

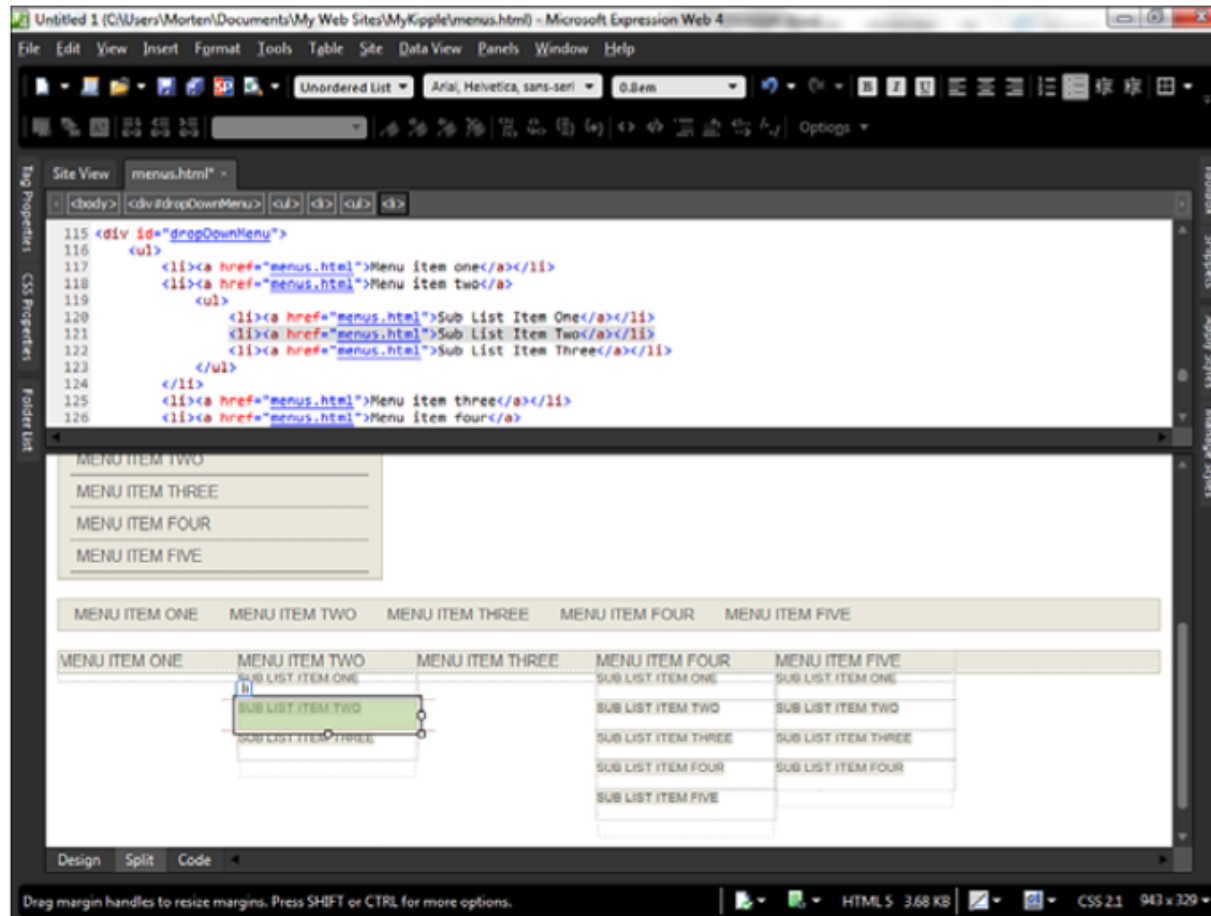


## Step 2: Style the Main Menu Cont.

---

- Because we are operating with two layers of list items, the basic font styling should be done in the *li* style: Still in the *#dropDownMenu ul li* style, set *font-size* to *0.8em* and *text-transform* to *uppercase*.
- To give the buttons some breathing space, set *height* under *Position* to *25px*.
- Click *OK* to apply the changes.
- You now need to style the links like you did earlier.
- Create a new style with the selector *#dropDownMenu ul li a*.
- Under *Font*, set *color* to *#666666* and set *text-decoration* to *none*.
- Go to *Background* and set *background-color* to *#EBEADF* manually, or you can use the *Eyedropper* tool and pick the background color from one of the other menus.
- Click *Apply*, and you see that the links now have the right font and background color, but that the background is visible only directly behind the links.

# Step 2: Style the Main Menu Cont.

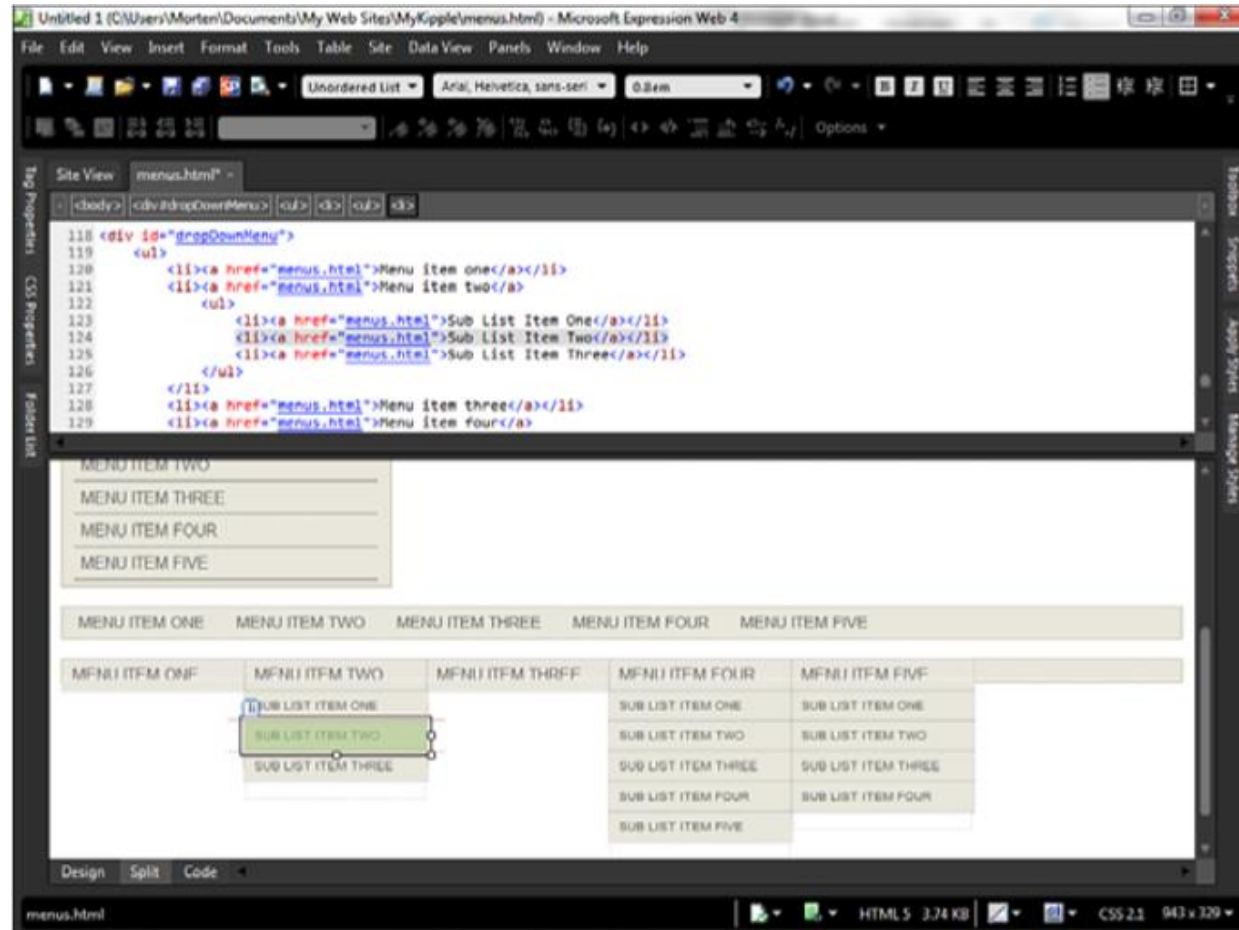


## Step 2: Style the Main Menu Cont.

---

- For the menu to look and function properly, the link area needs to extend beyond the text to cover the allotted area in the boxes the `li` style has created.
- In the earlier horizontal menu, you used the `padding` attribute to do this, but this time, you use the `line-height` and `display` attributes.
- Still in the `#dropDownMenu ul li a` style, go to *Block* and set the *line-height* attribute to `25px` to match the height you set in the `#dropDownMenu ul li` style earlier.
- Next, go to *Layout* and set *display* to `block`.
- Finally, set *padding-left* to `8px` to create some space between the left edge and the text.
- Click *OK*, and the backgrounds now fill out the correct areas.

# Step 2: Style the Main Menu Cont.





## Step 2: Style the Main Menu Cont.

---

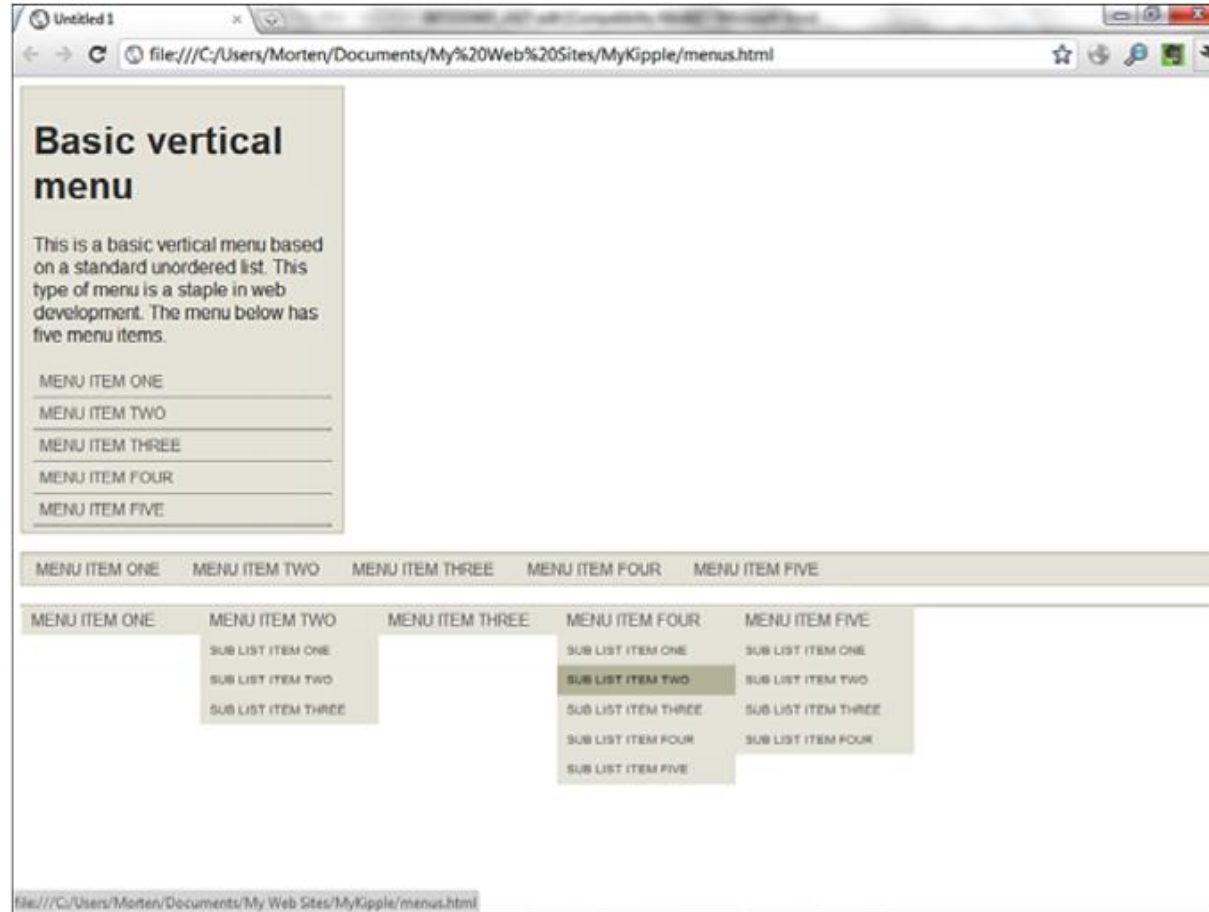
- Now that the buttons extend to fill the required area, create a new pseudoclass with the selector *#dropDownMenu ul li a:hover* and set *color* to *#333333* and *background-color* to *#C4C2AB*.
- Click *OK*, save the page, and test it in your browser.

# Step 3: Make the Drop-Down Menus Drop Down

---

- As you can see, the main menu and submenus line up correctly, but the submenus are all visible all the time.
- However, the whole point of drop-down menus is that they drop down only when the visitor hovers over them.
- To achieve this, use the *:hover* pseudoclass with the *visibility* attribute to hide the submenus.

# Step 3: Make the Drop-Down Menus Drop Down Cont.



# Step 3: Make the Drop-Down Menus Drop Down Cont.

---

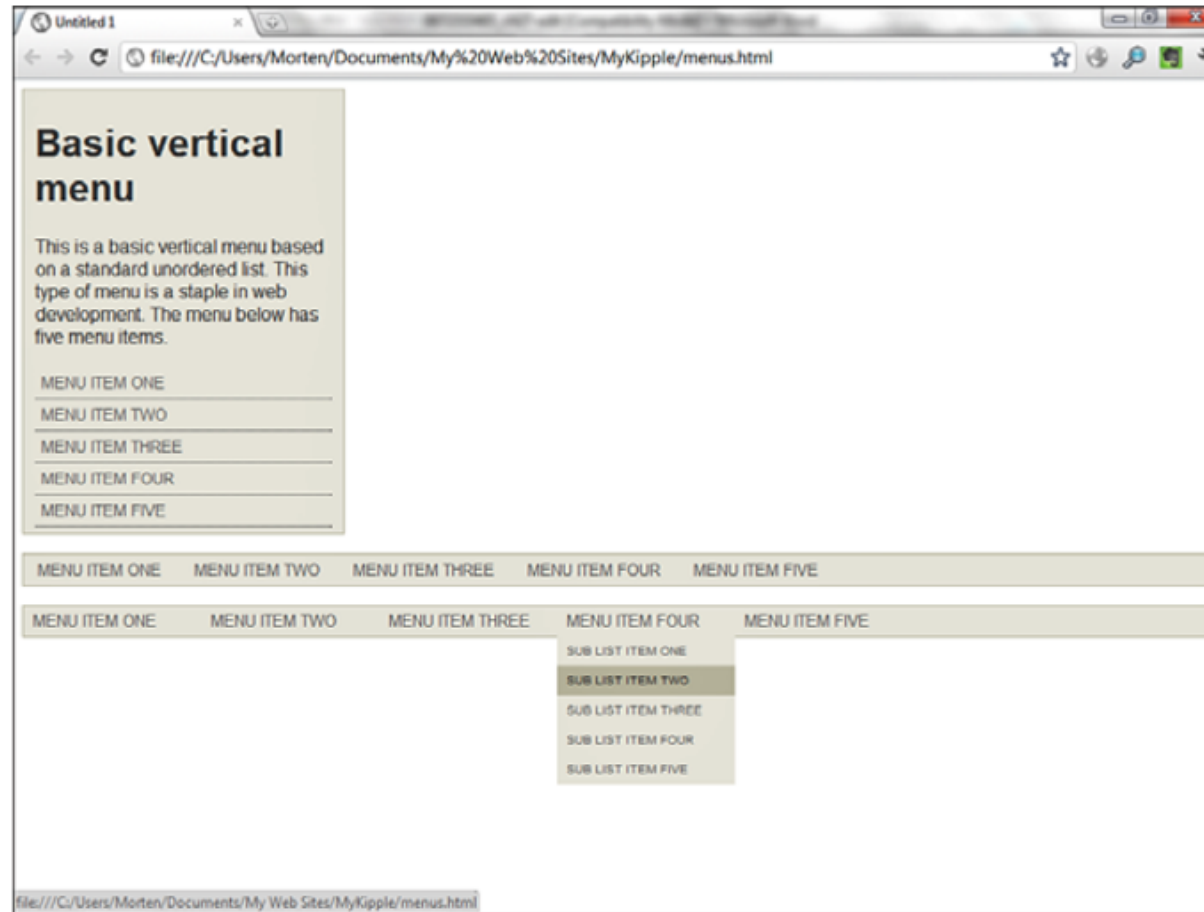
- Hide the submenus so that they are invisible unless the user triggers them.
- To do so, create a new style with the selector `#dropDownMenu ul li ul`.
- This style affects only the unordered lists contained within a list item (that is, the submenus).
- Under the *Layout* category, set *visibility* to *hidden*.
- Click *OK* to apply the modified style.
- Now, the submenus are no longer visible in *Design* view.
- Create a new style with the selector `#dropDownMenu ul li:hover ul`.
- This style is a pseudoclass that triggers when the visitor hovers over a main menu list item and affects any unordered list contained within that list item.
- Under the *Layout* category, set *visibility* to *visible*.
- Click *OK* to apply the new style.

# Step 3: Make the Drop-Down Menus Drop Down Cont.

---

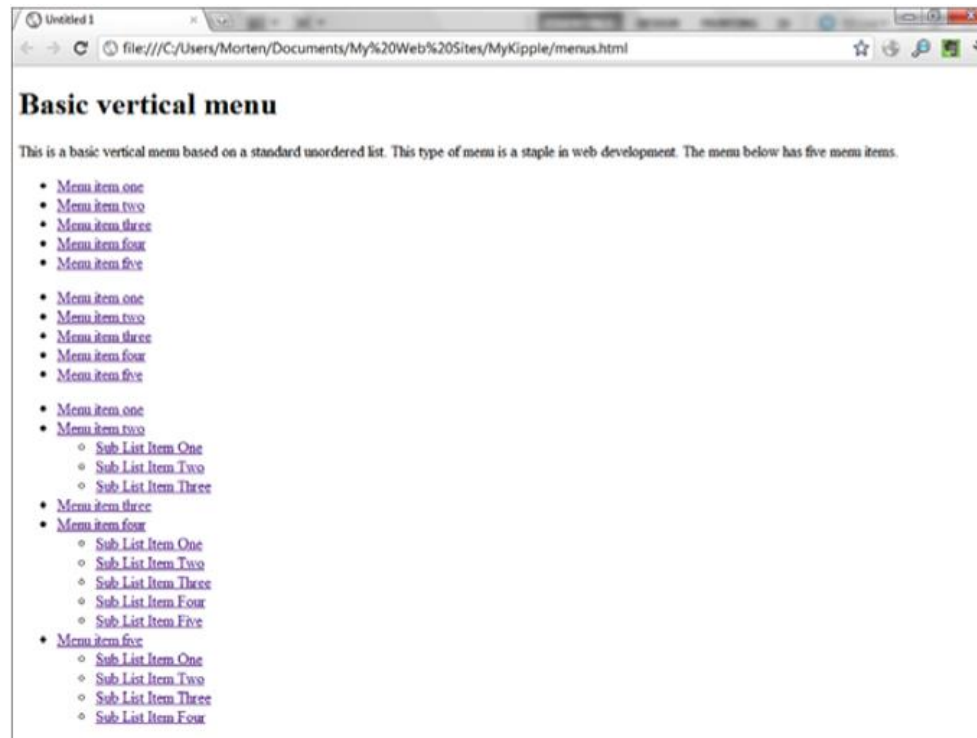
- To ensure that the drop-down menu items appear on top of and not behind content or below them on the page, open the selector *#dropDownMenu*, go to *Position*, and set *position* to *relative* and *z-index* to *10*.
- The *z-index* value places the entire menu in a higher level than the rest of the content on the page.
- Save and preview the page in your browser.
- You see that the drop-down menus now work the way they should.
- Furthermore, the menu is 100% CSS-based, which means it works without any code additives, such as JavaScript.
- However, most important, it is fully legible if the visitor uses a text-only or text-to-speech browser.

# Step 3: Make the Drop-Down Menus Drop Down Cont.



# Step 3: Make the Drop-Down Menus Drop Down Cont.

- As you can see in the following figure, with styles turned off, the CSS-based menus revert to their original form, which is standard unordered lists with sub-lists.
- Not only is the menu easier to read, but also the layout, and ordering is intuitive to the visitor even without styles.



# Styling the Submenus to Make Them Stand Out

---

- Right now, there is no visual difference between the main menu items and the submenus.
- But, depending on the design of the site, it can sometimes be a good idea to give the visitor visual clues that separate different types of content from each other.
- A simple way of doing this is to give the submenu items a different set of styles than the main menu items.

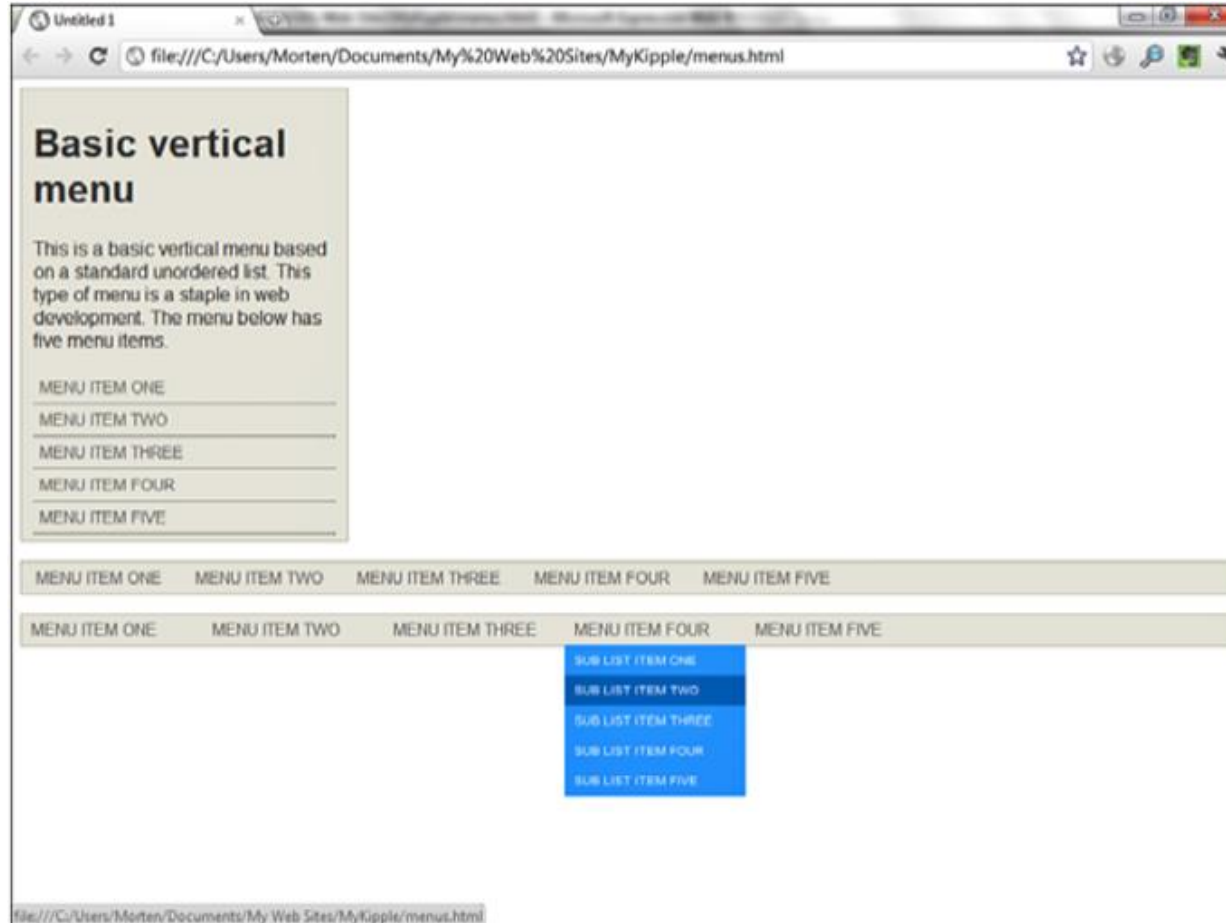


# Styling the Submenus to Make Them Stand Out Cont.

---

- Create a new style with the selector *#dropDownMenu ul li ul li a*.
- This style affects only the links inside list items that are housed inside another list item.
- Under *Font*, set color to *#FFFFFF* and under *Background*, set *background-color* to *#3399FF*.
- This gives the submenus a blue background color.
- Click *OK* to apply the new style.
- Because of the cascade, unless you specify something different, the hover state of the submenu is styled by the *#dropDownMenu ul li a:hover* pseudoclass.
- To change the hover state, you need to create a new pseudoclass with the selector name *#dropDownMenu ul li ul li a:hover*.
- Set the color to *#FFFFFF* and set the *background-color* to *#0065CA*.
- Click *OK*, and save and test the page in your browser.
- With the new styles applied, the submenu now has a distinct look that is different from the main menu.

# Styling the Submenus to Make Them Stand Out Cont.



# Create an Image-Based Menu for the MyKipple Site

---

- Now that you know how to make a functional menu, let's apply that knowledge to the *MyKipple* site and, in the process, make it even more advanced by applying image-based backgrounds.
- With the *default.html* file open in *Split* view, create a div with the ID *#mainMenu* and place it inside the header.
- In *Code* view, create a new unordered list with three buttons named *Home*, *Gallery*, and *Contact*.
- Make each of them a link pointing back to *default.html*.
- Create a new style in *kippleStyles.css* and give it the selector *#mainMenu ul*.
- Set *width* to *100%* so that the box spans the entire width of the header box.
- Create a new style in *kippleStyles.css* and give it the selector *#mainMenu ul li*.
- Under *Block*, set *line-height* to *30px*.
- Under *Layout*, set *display* to *inline*.
- This aligns the buttons on one line.

# Create an Image-Based Menu for the MyKipple Site Cont.

---

- Create a new style in `kippleStyles.css` and give it the selector `#mainMenu ul li a`. Set `color` to `#000000`, `text-align` (found under Block) to center, `width` to 110px, `height` to 35px, and under Layout, set `display` to block and `float` to left.
- Create a pseudoclass for `a:hover` and set `color` to `#FFFFFF` and `text-decoration` to `none`.
- At this point, you should have a basic three-item menu along the bottom left side of the `#header` div.
- The buttons work, but there are no backgrounds.
- Now, you assign separate graphic backgrounds to each of the three buttons using custom classes.

# Create an Image-Based Menu for the MyKipple Site Cont.



# Create an Image-Based Menu for the MyKipple Site Cont.

---

- Import the three files, *button26.jpg*, *button32.jpg*, and *button2c.jpg*, from the lesson files for hour 19 and place them in the *Graphics* folder.
- In *kippleStyles.css*, create a new style with the selector *.blue*.
- Set *background-image* to *button26.jpg*, *background-repeat* to *no-repeat*, and *height* to *35px*.
- In the *Manage Styles* panel, right-click the new *.blue* style and select *New Style Copy* from the pop-up menu.
- This creates an exact copy of the style.
- Rename it *.green* and change *background-image* to *button2c.jpg*.
- Click *OK* to save the new style, and use the exact same technique to create a third style with the selector *.purple*.
- In *Code* view, find the anchor tag for the first of the three buttons and place your cursor directly after the letter *a*.
- Press the spacebar and type *class="blue"*.
- This applies the *.blue* class to the first button.

# Create an Image-Based Menu for the MyKipple Site Cont.

---

- In *Design* view, click the second button and use the *Quick Tag Selector* to select the `<a>` tag.
- In the *Manage Styles* panel, right-click the `.green` style and select *Apply Style* from the pop-up menu.
- In *Design* view, click the third button and make sure the `<a>` tag is highlighted in the *Quick Tag Selector*.
- In the *Apply Styles* panel, click the `.purple` style to apply it.
- As you can see, you now have three buttons with three different backgrounds.
- If you paid attention to the earlier lessons in this hour, you noticed that the styling of this menu is the same used to create the drop-down menu, which means that if you want to, you can expand the menu to include drop-down features later!

# Create an Image-Based Menu for the MyKipple Site Cont.

