W3 WWD Notes

Remember many of the assignments build off of each other in this course. It is critical that you do not fall behind. Learn to recognize when you are stuck and get help from peers or the instructor. There are also many resources available online about HTML and CSS. If there is a specific topic that is not making sense a quick search will bring up many examples.

**BYU-Idaho Online Learning**

Video Transcript

Class and ID Attributes

*[Text saying “Class and ID Attributes” is shown on a blank background.]*

**Female:** Class and ID attributes. To target an element in CSS as a selector, we've been referring to them by their element name.

*[The text on the screen changes. CSS targeting the element names is shown. The paragraph element is having its color changed to #000. The background color of all h1 elements are being changed to yellow.]*

But there are two attributes for elements that can identify them with a different name. Because we might have a whole lot of p tags on our page. But maybe we only want to apply some CSS to only one or some of them. That's when classes and ID attributes come in handy. They can be used in HTML elements and also targeted as a selector in CSS. The developer can choose the name or value of IDs and classes, but they shouldn't start with the number. These ID and class attributes are placed inside the beginning tag of the element.

*[The text changes to just a paragraph element. The beginning tag says* ***“<p class=”anyname”>”.]***

Then later when CSS is used, the ID and class values can be used to target just that element.

*[The text changes to CSS. The selector for the CSS is* ***“.anyname“****.]*

IDs and classes don't affect the presentation of an element unless there's a CSS rule, selecting that class or ID that indicates it should be displayed differently. If I had four paragraphs and I only wanted every other paragraph to have a light gray background, how do I target just some of the paragraphs in CSS and not others?

*[A website is shown. The title is “Ways to Stay Warm Outdoors”. Four paragraphs are shown giving tips on how to stay warm outdoors.]*

If I target the p element, they'd all be shaded.

*[CSS is shown changing all p tags background color to #ccc. The background color of all four paragraphs in the website are now gray.]*

I could give the paragraphs that I want shading to a class name of shaded.

*[An image of the website's HTML appears and the other two images are removed. The text* ***“class=’shaded’”*** *is highlighted and is in the opening tag of both the first and third paragraph tags.]*

And in CSS, I target not the p tag, but the class name of shaded. To show that it's a class, we put a dot or a period in front of it. (.)

*[An image of the CSS is shown. The text* ***“.shaded { background-color: #ccc; }”*** *is shown.]*

This tells CSS to look for a class name. Now I can put a background color and it will apply to only the paragraphs with that class Name.

*[The HTML and CSS images are removed and the website image is shown again. This time the first and third paragraphs are highlighted gray.]*

IDs work similarly, but an ID attribute is used to uniquely identify the element from others on the page.

*[The image of the website is removed. Text appears saying* ***“<h1 id=”main-topic”> My Title Here </h1>” is shown.]***

This means I can't apply one ID name to multiple elements like I did the class shaded.

*New text underneath appears saying* ***“<h2 id=”main-topic”> My Sub Title Here </h2>”****. The text* ***“id=”main-topic””*** *is crossed out.]*

If I give my H1 an ID of “myheader”, I can't use the ID with that value my header anywhere else in this HTML document.

*[The text is removed. An image of the HTML of the website is shown. The text* ***“id=”myheader””*** *is in the opening tag of the <h1> element and is highlighted.]*

If I do, my code will not validate. To apply CSS to that ID, I could use the ID name with a hashtag in front of it. (#)

*[The image of the HTML is removed. An image containing CSS is shown. The selector of an element is* ***“#myheader”****.]*

This tells CSS that you're looking for an ID called my header. So **dots are for classes and hashtags are for IDs**. And CSS can be applied to only that one class with the same results.

*[The image containing CSS is removed. The image of the websites HTML is shown again. This time the word “class” has replaced “id” in the opening tag of the <h1> element.]*

So there's always some confusion as to **why we need IDs**. If you go on to learn JavaScript later, for example, ID attributes can be used to allow the script to work with that one particular element and no other. So there are reasons that you might use IDs that will make more sense to you later. **Elements can also have more than one class.** For example, if I wanted to emphasize this text to be more bold and red, I can apply a class of emphasized to it.

*[The image of the HTML is changed. The <h1> tag has an id instead of a class. The second paragraph has the <span> tag surrounding the first sentence. The opening span tag has a class called “emphasize” in it. CSS is shown on the right selecting the “emphasize” class. The CSS changes the text to be bold and the color is changed to “darkred”. An image of the website is shown. The first sentence of the second paragraph is bold and darkred.]*

If I also want another paragraph emphasized, I could give it two class names. So now it has the shaded class name and the emphasise class name.

*[The image of the website is removed. The image of the HTML is shown. The first paragraph’s opening tag now* ***says “<p class=’shaded emphasize’>”.*** *The text* ***“class=’shaded emphasize’”*** *is highlighted.]*

In CSS, I have two different rules for each of those class names.

*[The image of the website is shown. The first paragraph is bold and red with a gray background. The second paragraph’s first sentence is bold and red. The third paragraph’s background is gray.]*

You could even have an ID and a class in one attribute. I can add the emphasized class to my H1 element that already has an ID.

*[The image of the website is changed. The title is now bold and red.]*

As your website grows larger with more and more HTML pages, it's nice to have just one CSS file so you don't have to reuse the same CSS for parts of the page that remain the same. But simple element name selectors can affect multiple pages in unexpected ways sometimes. So using meaningful classes and IDs can help organize your CSS for multiple pages as well.

*[End of video]*

**BYU-Idaho Online Learning**

Video Transcript

HTML File Basics

*[The screen is open to an IDE. The file “index.html” is open and is displaying HTML code.]*

**Female:** Let's look at some basics that you should follow when creating an HTML page from beginning to finish. When you begin an HTML file, you should always follow a few simple naming rules. It's strongly suggested that you save your file names in all lowercase. This will cut down on mistakes that might be made later. The first character of the file names should be a letter. Don't start the name with a number. There shouldn't be any spaces in the file name. If you want to separate words, you can use a dash or an underscore, but no other characters. These rules apply to folders, image files, or other resource files that you can name when you create your website as well. Remember, the file with the name index.html will always be recognized as the homepage of your website. So when the user types in **the domain name of your website** like Amazon.com, **the index page will always come up first as the homepage**. Other sub-pages or child pages on your website. Don’t have to have any special name. You can name them whatever you want, but try to remember to keep it short and simple and follow the naming conventions.

*[The instructor opens a file called “top\_product.html”. The file is blank.]*

I'm going to use VS Code, and their shortcut here, and type an exclamation mark and tab or enter.

*[The instructor types the exclamation mark and a list of suggestions opens. The first suggestion is highlighted. When the instructor hits the tab. The file removes the exclamation mark and autofills the file with the* ***tags <!DOCTYPE html>, <html lang=”en”>, a <header> with <meta> tags and a <title> tag, and a <body> tag.]***

This brings in the beginning of our HTML page. Notice the Meta tags that come in. Let's talk about those. HTML lets you add additional information about your document using Meta tags. The Meta tags can include name-value pairs describing properties of your HTML document. **These tags are not visible on your webpage, but belong in the head section of your HTML.** This is where we have information about our page, not the content that shows up on our page. I could add my own stating who the author of the document is.

*[The instructor adds the text “****<meta name=”author” content =”Sue Smith”>” in between the <head></head> tags.****]*

It’s not required and it does not affect the look of the document in any way, but it is just additional information about our page if you want to include it. The next meta tag tells us what character set we're using in our document.

*[The instructor highlights the code “<meta charset=”UTF-8”>”]*

**Utf-8** refers to the Unicode character set. Unicode can handle any international language characters. So it's a good one to use on your webpage. The next meta tag tells Microsoft to use the latest rendering engine edge.

*[The instructor highlights the code* ***“<meta http-equiv=”X-UA-Compatible” content=”IE=edge”>”****]*

This one is only needed in a few instances. The last metatag is important and I would always include it on any HTML document.

*[The instructor highlights the code* ***“<meta name=”viewport” content=”width-device-width, initial-scale=1.0”>”****]*

It allows developer, the developer to take control over the viewport or visible area of a web page, no matter what device is being used. Whether the user is seeing your web page on a phone, tablet, laptop, or desktop, the width equals device width sets the width of the page to follow the screen width of the device. And the initial scale equals 1.0, sets the initial zoom level when the page is first loaded in the browser. So this initial zoom at 1.0 or 100%. When used in conjunction with media queries, which we'll touch on later, it can help make pages be responsive, or in other words, look good on any device. As you create your file, make sure you're using semantic elements for your content.

*[The instructor opens a file called “*staywarm.html*”. Code in HTML is written.]*

Most tags describe what is inside them. And our semantic, for example, the nav tag has a navigation contained between this opening and closing tag.

*[The instructor highlights the nav opening and closing tag. Inside is a list of links to other pages on the website.]*

But some elements are not semantic like divs or span tags. These tags tell nothing about their content. They simply create a wrapping element around their content, usually for CSS presentation purposes, not because it gives their element meaning. So, for example, I might not have a good tag that has a semantic fit with the banner information.

*[The instructor highlights a div with the ID “****banner-msg****”.]*

But I know that I'll be doing CSS and all that content together later. So, I place a div with the class of banner message around it. Also, the span element is an example of a non-semantic tag.

*[The instructor highlights a span tag with the class “emphasize”.]*

I might want the color of just that text “have an outdoor heat source” to be a different color. With CSS, it doesn't really have a semantic meaning to that content. Why use a nav tag when a div tag would work and your pages navigation or menu will look just the same on the page. The reason it's important is because screen readers and search engines use this extra information. And it's important to give our pages as much meaningful information about our content as we can. This makes it accessible to those navigating our page with that site. And also for search engines like Google to make more sense out of our page and make it appropriately. It's also just good practice to do it. Once we're done with our file, we should always **validate the code**. Validating your HTML CSS is important. Validating Is just like it sounds, making sure our code is valid or coded properly. **W3c is the World Wide Web Consortium or international community that maintains and develops web standards.** They have a validator that you should use to validate your HTML. It’s at [validator.](https://www.google.com/url?q=http://validator.w3.org&sa=D&source=editors&ust=1737462865972576&usg=AOvVaw3ksLi2Ada1_F8mLQ5zeIM5)[w](https://www.google.com/url?q=http://validator.w3.org&sa=D&source=editors&ust=1737462865972900&usg=AOvVaw3Zns6uze7NinU4LhHRf2is)[3.org](https://www.google.com/url?q=http://validator.w3.org&sa=D&source=editors&ust=1737462865973065&usg=AOvVaw1mRTbRhFxOeM2KTEqSwb_T).

*[The instructor goes to validator.w3.org. The website has a banner saying “markup Validation Service”. There are three tabs that say, in order, “Validate by URL”, “Validate by File Upload”, and “Validate by Direct Input”.]*

Here you can check your webpage HTML code by either placing the web address of your page or uploading your HTML file, or by pasting your HTML code in directly.

*[The instructor selects the “Validate by Direct Input” and pastes in the code in an input field labeled “Enter the Markup to validate:”. The instructor hits a button with text saying “Check”.]*

Then when you check it, it will then let you know if you have any errors.

Then you should correct any errors so that your page will validate with no errors. Here's what it would look like once it validates.

*[Text is added in a green box that says “Document checking completed. No errors or warnings to show.”]*

Why is this important? First, it can help you debug your code. If your page is not rendering like you expected it to, validating can help show you where your errors are. Also it helps you learn good practice, having clean code, and help with site performance. It's a sign of professionalism. And those hiring you later might look at your portfolio or webpages and validate them to see if you have good code. Also, search engines like Google will rank your website much better if your code validates. So get into a good habit of always validating your HTML code. You can also validate CSS at [j](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865973930&usg=AOvVaw0-Pb5oy4yyTJCpBdj2759-)[igsaw](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974084&usg=AOvVaw1isxdjs8e5VEyJ9kY2lOfS)[.w](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974199&usg=AOvVaw3MuNyTPA83zFheMKNva1KW)[3.org](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974320&usg=AOvVaw0DUrmDEwBCjpczAjtaenth)[/](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974469&usg=AOvVaw0HW25B-Hy4qy9fPgk-ysC7)[css](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974635&usg=AOvVaw0o1wkD7VSDWl1kDhMFwe7S)[-v](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974793&usg=AOvVaw2y6UDvMdIU7UgOmB-ZQ6v1)[alidator](https://www.google.com/url?q=http://jigsaw.w3.org/css-validator&sa=D&source=editors&ust=1737462865974911&usg=AOvVaw0vBCeSau_kuAlQbpM-KOyl).

*[End of video]*

**BYU-Idaho Online Learning**

Video Transcript

Wireframes

It's important to plan out your website and each webpage before you start coding in HTML or CSS. We started this planning last week with the site plan. Part of the site plan was the wireframe of the homepage. **Wireframing** plays an important role in your design process. Wireframes will depict what the webpage will look like or the interface of our page. They can be something like this that is hand written. Or they can also be created using different tools, like we saw with this one, that they don't really portray the different colors per se or the actual images of the page. It's just a visual representation of the page. Too many images or colors might distract from focusing on the layout. At this point, we're just interested in the basic structure of the page and which elements will be using to create the page will be creating a wireframe. Later in the course this week we're looking at completed wireframes. If I were building a website for a client, I would I wouldn't want to spend a lot of hours coding out all of my HTML and CSS to then show the client for the first time, I would create a wireframe like this to present to them to get feedback. At this point, if they want to changes, that would be simpler to change than if I had coded at all. So not only as wireframing a good design practice, it saves a lot of time in the long run with your development process. With a wireframe showing the design of our page. We're ready to start converting that wireframe into HTML. But let's take a look at this wireframe and decide how our page elements might be grouped. We can see that the top of the page is the logo and navigation. So I'm going to group this area. So this is the navigation and the logo area. It's followed by a scrolling banner with text over the banner and images, a product gallery, and then a footer with contact information and copyright information, sitemap and social media icons. **These two parts of the page will remain the same for every page on our website.** As the users of our site go from page to page and the site, the top portion of the page, **the navigation logo, and the bottom portion, the footer, will stay the same.** These are going to logically **go in their own groups. Everything out on the page between the navigation and footer will most likely change from page to page.** So we're going to also group that section together as well. Okay, within that part, if we look at this wireframe, we can see a few more groupings.

Notice, notice the banner scrolling area here has a different purpose from the product gallery. That is here. They not only have different functions, but they will also be designed or laid out differently as well. So I'm going to group those together as well. One group for the banner image here, and one grouping for the product gallery here. So for example, each banner image also might have some text that goes with each image. So we see some text here on top of the image. For design purposes, we might also want to group each of the images and the text for each of the banner displays together. Also notice that each of their products also have a headline and a description or a title of the product with them as well. So we can also group those together within the product gallery. If we know later that we'll be doing some CFS the same on all of them. So we can see that the groupings can nest inside of each other. As we study the wireframes, this will correspond with how our HTML might be nested inside each other. As we create these HTML elements, Let's look at the wireframe for the homepage of the whitewater rafting site. Again, we can see some law to keep logical groupings here, header section here. And of course a footer. And these two areas won't change from page to page. In between that, we have the main section that will change from page to page. And at the top here we have a hero or banner section. And you'll see it kinda goes up behind the menu, but all grabber right here for now, other pages won't go up behind the menu. And we also have this area here that has three images with some icons and text, and an area here with an image and a message area here. Again, within these groups, we have three of very similar groupings here with an image, an icon, and some texts that we might group together. And maybe inside this one we might group this headline paragraph and buttoned together as well. So you can see how the groups are nested in ITA, inside of other groups. And when we get to the HTML structure that we start typing out, we're going to see how the elements and class attributes are used to group them all properly.

*[End of video]*