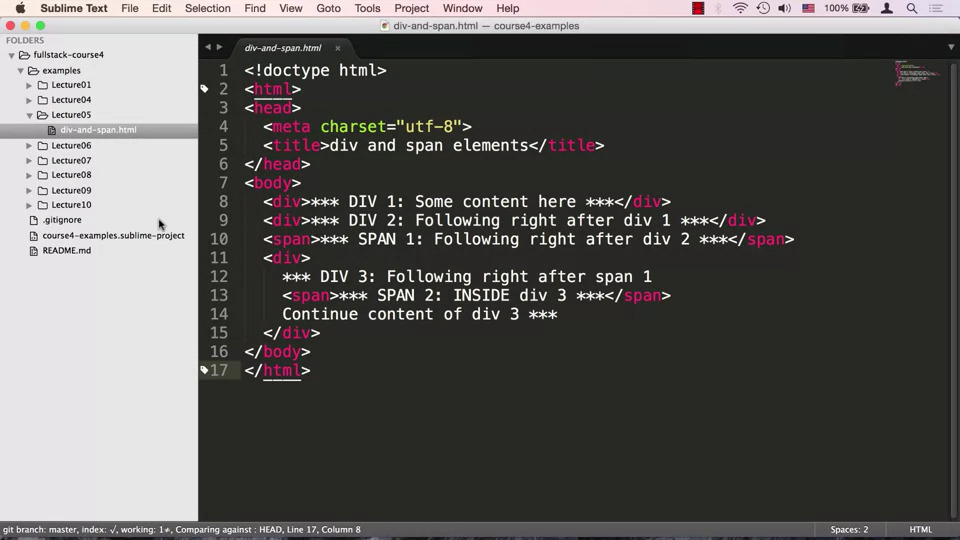
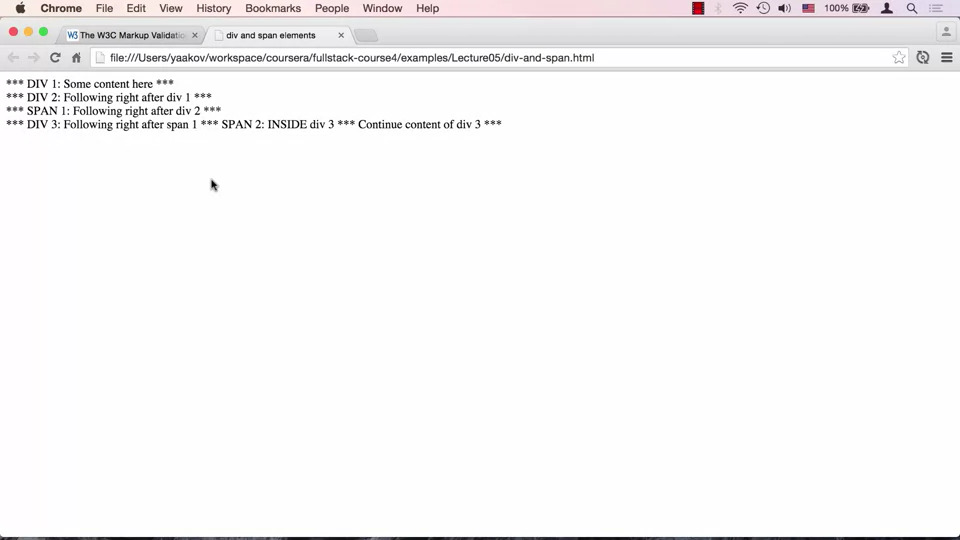
**HTML, CSS and JavaScript for Web Developers-Johns Hopkins**

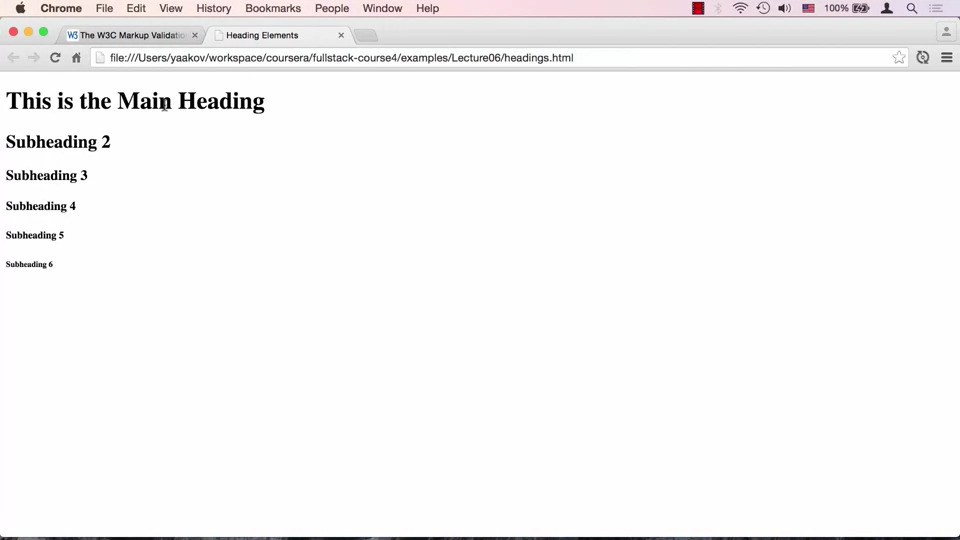
* **Introduction to HTML5**
* **1. What is HTML?**
  + HTML stands for Hyper Text Markup Language
  + Hyper Text is text which contains links to other texts, images etc.
  + Markup means to annotate any content, HTML surrounds the content in tags such as <title>
  + HTML is human readable
  + Language implies that HTML has its own syntax, for example closing tags </div>, HTML also has its own semantic which means these tags mean something to the interpreter
  + Three technologies drive the Web, HTML, CSS, and JavaScript
  + HTML provides the structure, for example it can one heading, a paragraph, HTML however only tells you what the component is, the color and styles are done using CSS, JavaScript adds functionality to the page, for example what happens after a page loads or after a link is clicked
* **2. Relevant History of HTML:**
  + Before 1997, browsers did whatever they wanted, they could all have different tags etc. Around this time, a standard for browsers was set in HTML 4.0. Around 2000 XHTML was introduced, this was a relatively rigid structure. In 2004 another group called WHATWG was introduced. Following 2011 HTML 5 was introduced. Both organizations W3C and WHATWG started working together. W3C has settled on HTML5 where as WHATWG says HTML is evolving
  + Browsers are now capable for updating themselves and are able to get new features
* **Anatomy of an HTML Tag:**
  + At the core of HTML, there is the HTML tag
  + HTML usually have an opening and closing tag, inside there is an element, the closing tag usually matches the opening tag instead of <br> and <hr>
  + For example <p> . . .</p> the paragraph tells us that the content in . . . should be treated as a paragraph
  + Every HTML element can have predefined attributes, an attribute is a name-value pair, for example id attribute is set to myId, this indicates that no other element of any kind is allowed to have its id to be set to myId, <p id=”myId”></p>. No space is allowed between the opening angle bracket and the closing /, space is allowed every where else, attributes can only be specified after the opening tag, the value is surrounded in either double or single quote, if the value itself contains quotes, we need to close them in circle brackets
  + A self-closing tag, does not contain any content and can be used to dynamically insert content, however we need to provide an opening and closing tag for example <p></p> is a self-closing tag
* **4. Basic HTML Document Structure**
  + Every HTML page should start with <!doctype html>, called the doctype declaration, we cannot have space after <, we can have space after doctype
  + If we leave of the declaration, the bowser will think that our page is not following HTML structure
  + The next tag is <html> which contains the entire html document
  + <head> describes what the page will contain, it contains the metadata of the main content
  + We can specify the character set in the meta tag by using charset=’utf-8’, the meta tag does not have a closing tag like </meta>
  + We can specify the title by using <title>, this is required and without it the page is invalid
  + After the <head> tag, we have the <body> tag, this tells us what content will be visible to the user
  + We are nesting HTML tags, for example our <head> tag contains the <title>, one rule to remember is that you have to close the last open tag before you close its parent tag. For example: <p>Coursera is cool! I am learning<span>SO</p>much</span> is wrong because the paragraph tag is closed before the last open tag for span. We should close the span tag first since it was opened last
  + HTML is read from top to bottom
  + Tags can contain upper or lowercase letters
* **5. HTML Content Models**
  + The term Content Model refers to the default behavior the browser applies to the elements belonging to that content model and the nesting rules of those elements
  + Prior to HTML 5, HTML elements were either block level or inline. In HTML 5, Block Level elements are separate from Inline elements
  + Block Level elements will begin on a new line by default, the browser will automatically place them on a new line
  + Block Level elements can contain inline or block level elements
  + Inline elements stay on the same line and can contain other inline elements but not block level elements
  + The <div> tag stands for division and is a block element
  + The <span> tag stands for span and is an inline element

****

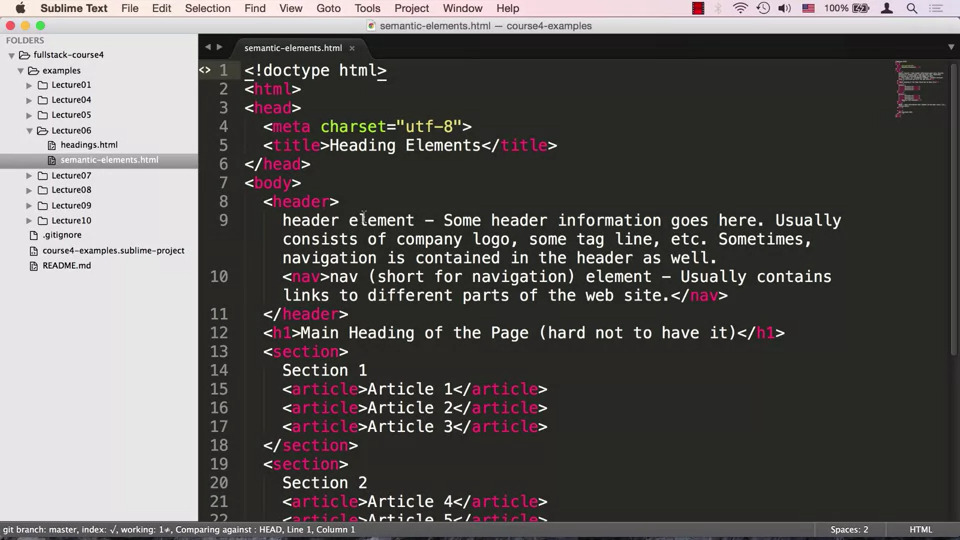
****

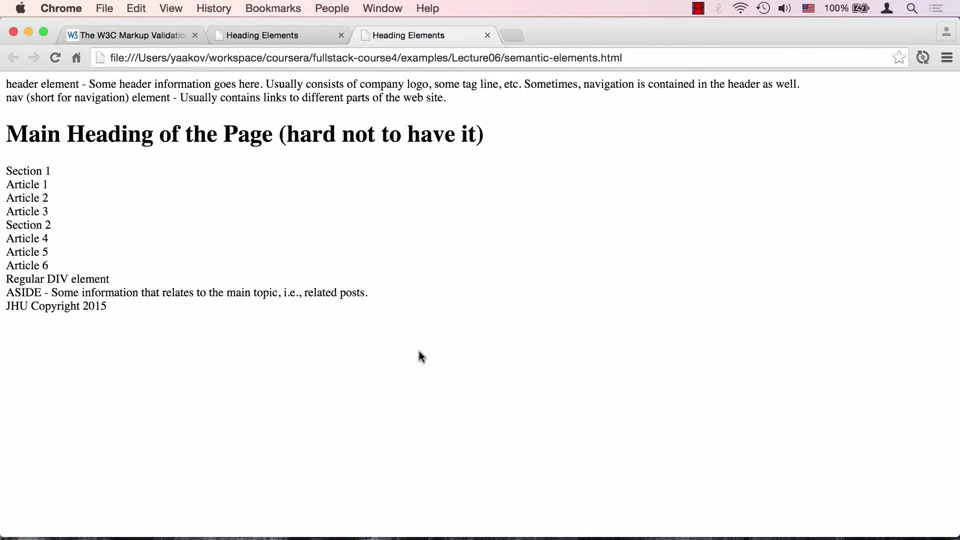
* + The Div1 and Div2 were on their own separate lines, span1 was pushed to a new line since Div2 requires its own line.
  + Div3 requires its own line, so it is on a new line, span2 is sitting inside Div3
  + We can remove all the spaces in our HTML document, and nothing gets changed
  + We cannot close <div> before closing <span> in Div3
* **6. Heading Elements and Semantic Comments**
  + Semantic HTML element is an element that tells us something about the content allowing humans to understand it better

****

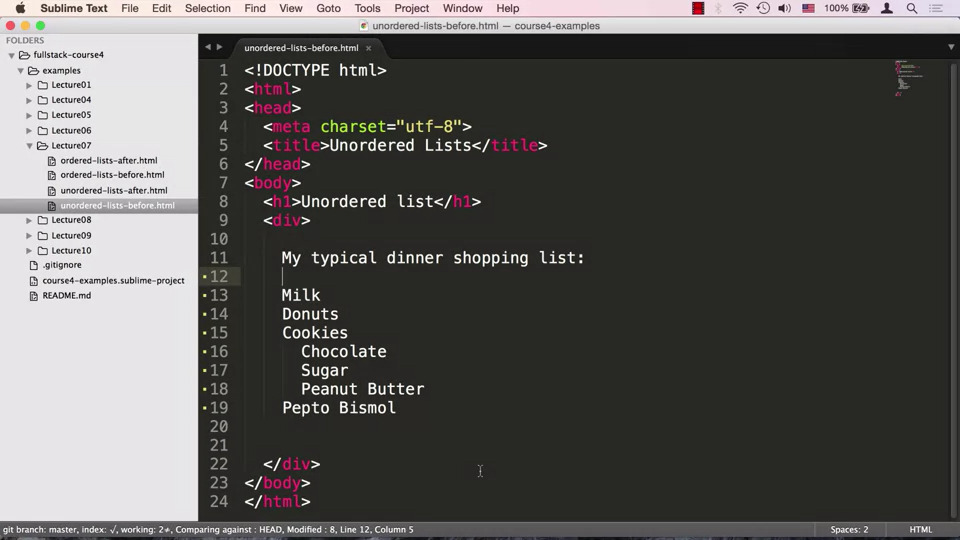
****

* + <h1> heading is the most important where as <h6> is the least important giving it less visual distinction, however these should not be used for styling
  + Using a div would lose the meaning of a heading
  + The heading tag is very important in conveying the main content

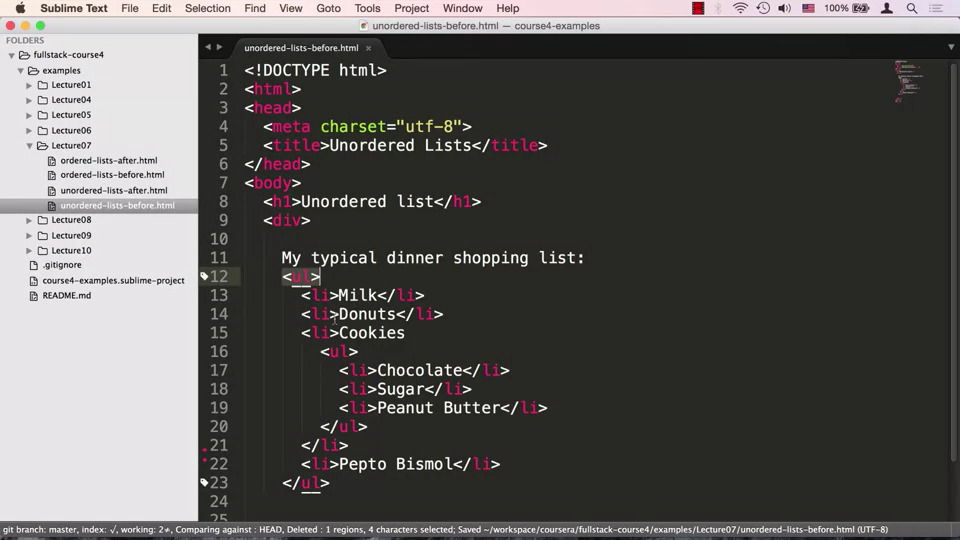




* + Semantic tags such <header> which contains information about the company name, logo, navigation etc. <nav> signifies some content that can be used for navigation
  + <article> is inside <section> tag
  + <aside> is a tag which indicates that this is content which is related to the main content but not as important
  + All these tags are block level elements, we could use <div> but it would be difficult to understand what the page is doing, semantic tags help humans in understanding what the page is doing
* **7. Lists:**
  + Lists allow us to group related content together

****

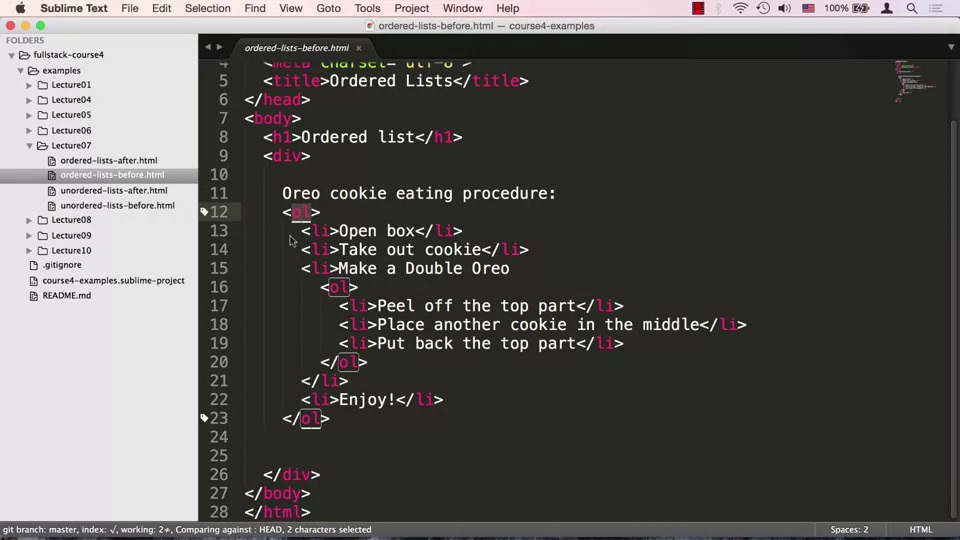
* + We want a bulleted list, however this does not result in a bulleted list



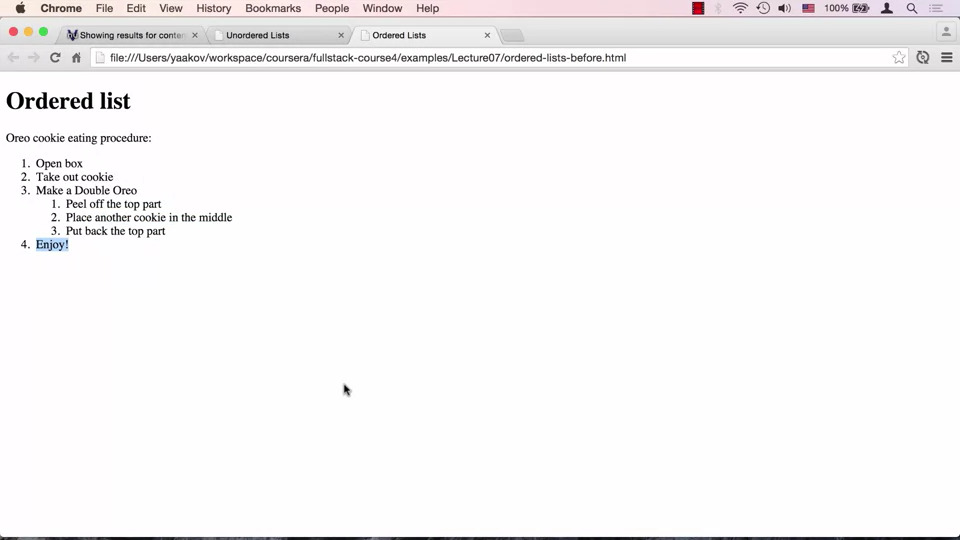
* + We can put this entire list in <ul> tag which stands for unordered list, we can then put each element inside an <li> tag. Cookies is a list itself, so we can create another list using <ul> and <li>



* + If we remove the <li> tag we will get an error since <ul> can only contain <li> tag

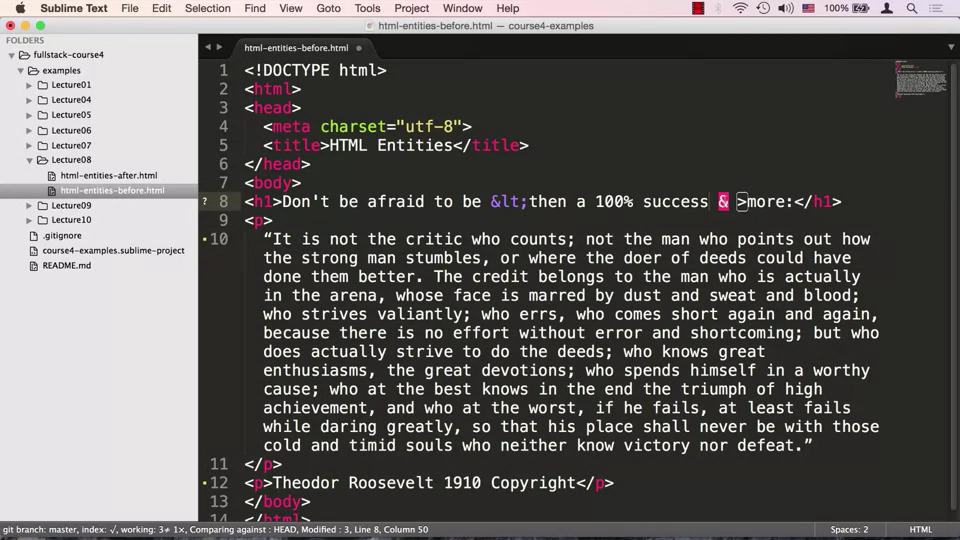


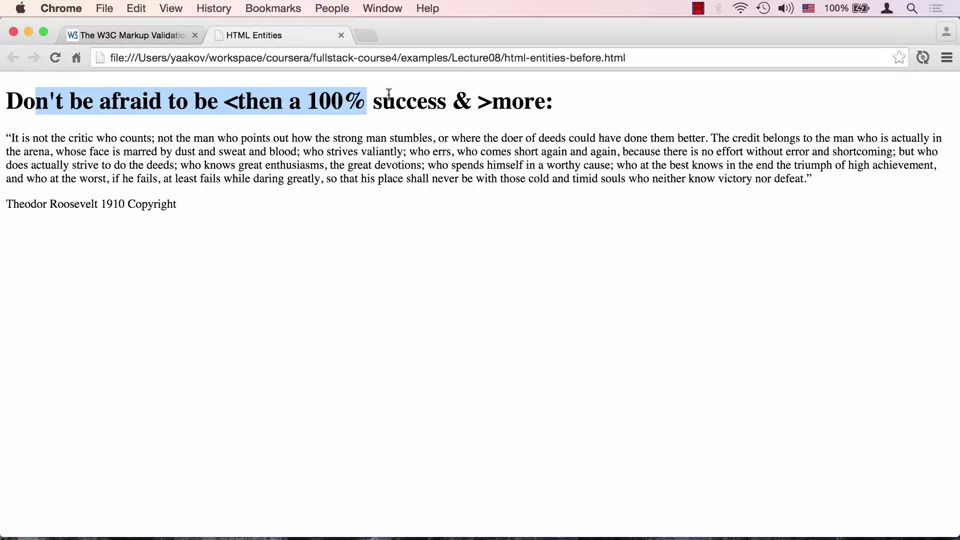
* + We can create an ordered list by using <ol> instead of <ul>



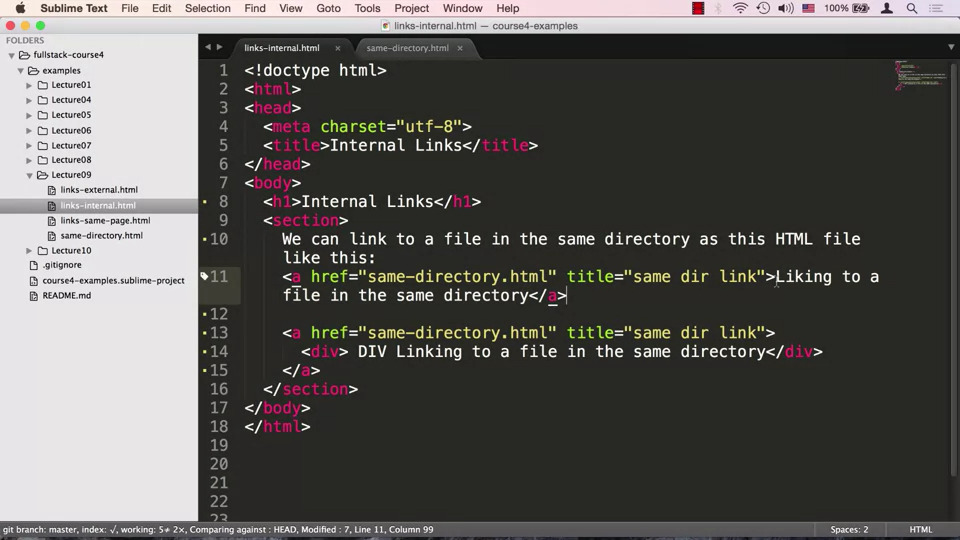
* **8. HTML Character Entity References:**
  + We need a way to differentiate between characters used for syntax and those same characters as content, we need a way to escape them or tell the browser to interpret them as content or syntax

****

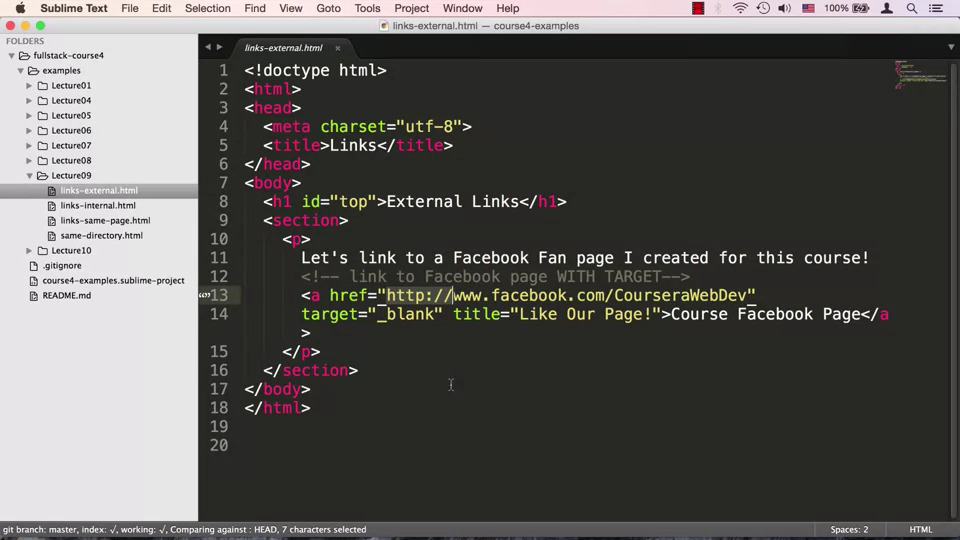
* + 



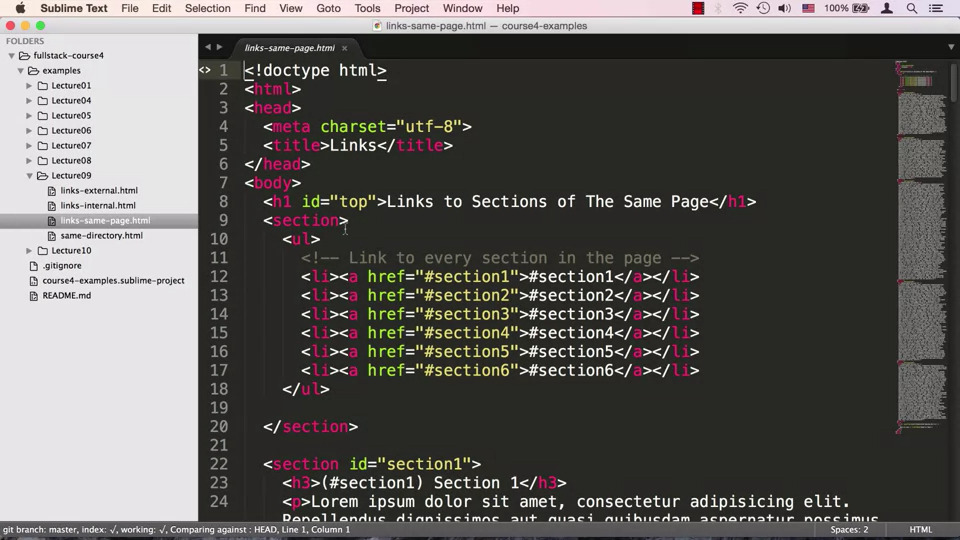
* + We can insert Copyright symbol by using &copy
  + We can specify wrapping which means what will happen when we resize the window, we can do this using &nbsp, we should not use this for spacing, instead we should use <span> along with margin
  + Instead of putting in double quotes, we can use &quotes
* **9 Creating Links:**

****

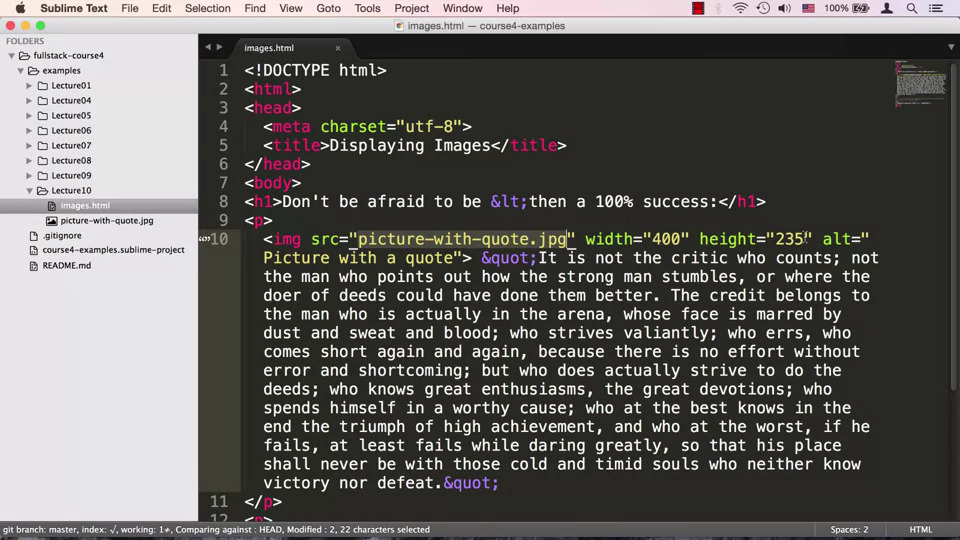
* + We can create a link by using the <a> tag which has the href attribute, the value of href can be a relative or absolute link, in our case we are dealing with internal links, since we are not providing any directory information the browser will think that .html is within the same directory as the code
  + We can specify the title by using the title attribute which helps the visually impaired people in navigating, the content between the <a> tag will be the content seen on the screen
  + We can put in a <div> tag inside the <a> tag, this will be a link on a new line, the <a> tag is a flow and phrasing content/ block and inline element

****

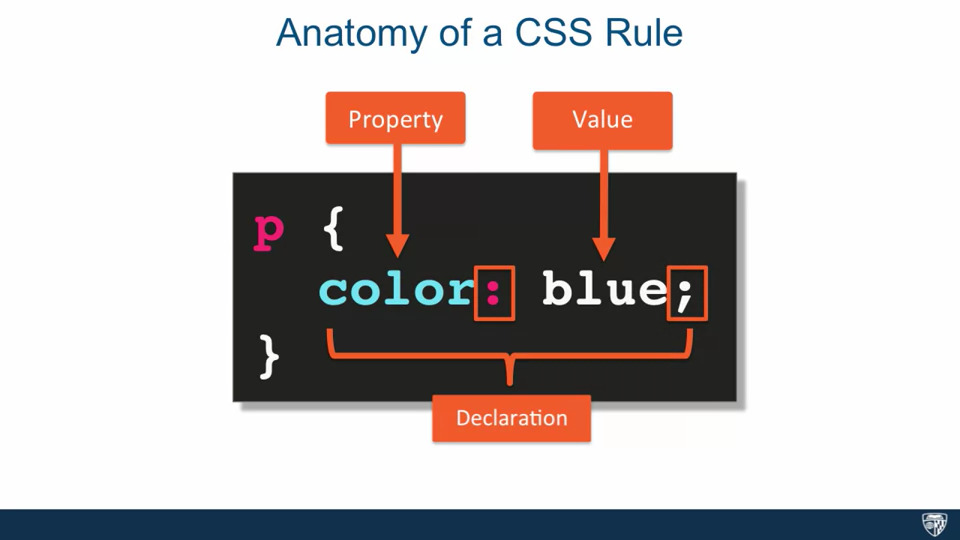
* + For an external link, we can specify target attribute, setting its value to “\_blank”, this forces the browser to open the link in a new tab

****

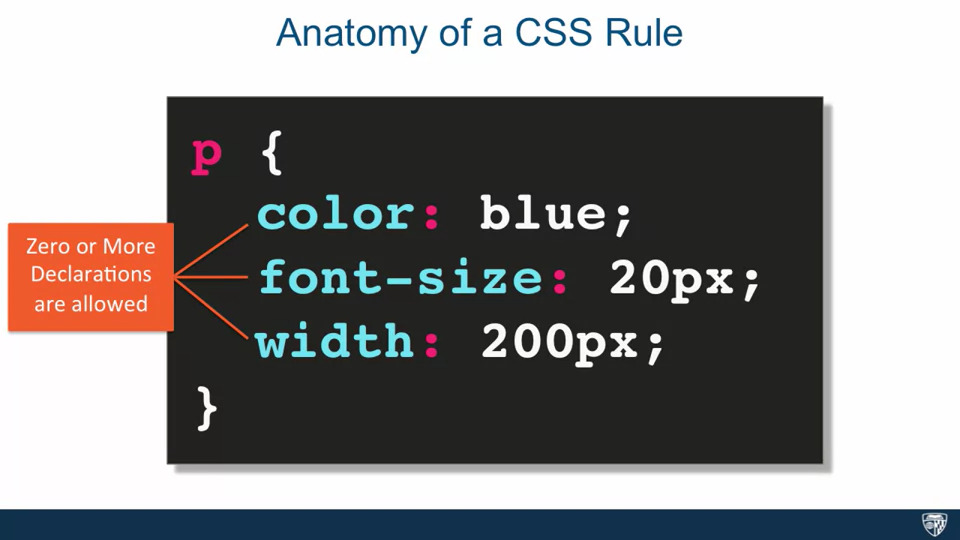
* + We can create links that point to a certain section in our HTML page, called fragment identifiers, we set the value of href as #section\_sectionnumber. When creating section we <section id=”section\_sectionnumber”>, another way is to create an <a> tag
* **10. Displaying Images:**

****

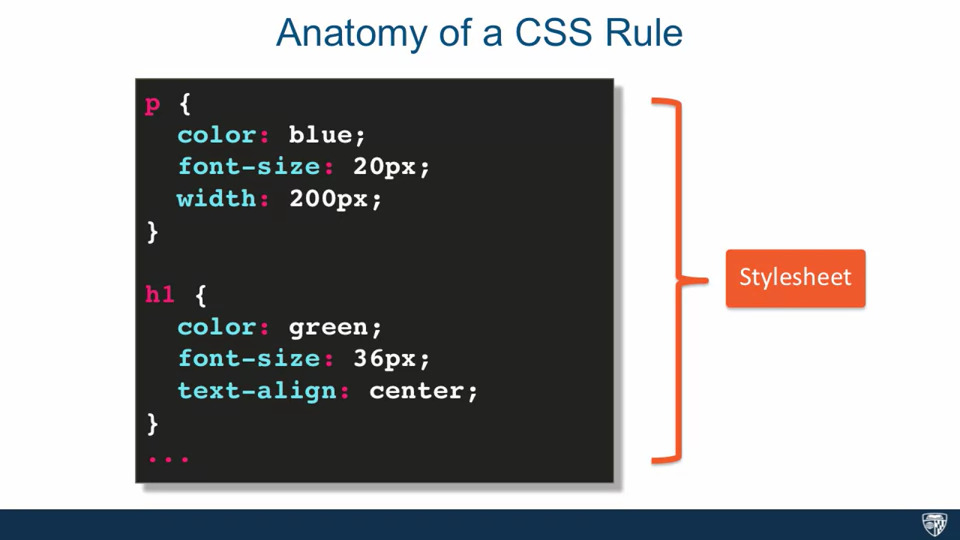
* + Comments start with <!-- and close with -->
  + We can insert images by using <img> tag , the src attribute can be an absolute or relative link, we can also specify width and height, the alt attribute can include description.
  + The img tag is an inline tag
  + Specifying the width and height can overcome browser loading time for the image, since the browser knows that space has been allocated
* **Introduction to CSS3**
* **11. Power of CSS:**
  + While HTML defines the structure of the content, structure along is not enough, we need to style our content, CSS or Cascading Style Sheet allows us to style our content
* **12. Anatomy of a CSS Rule:**
  + CSS works by associating rules with HTML elements

****

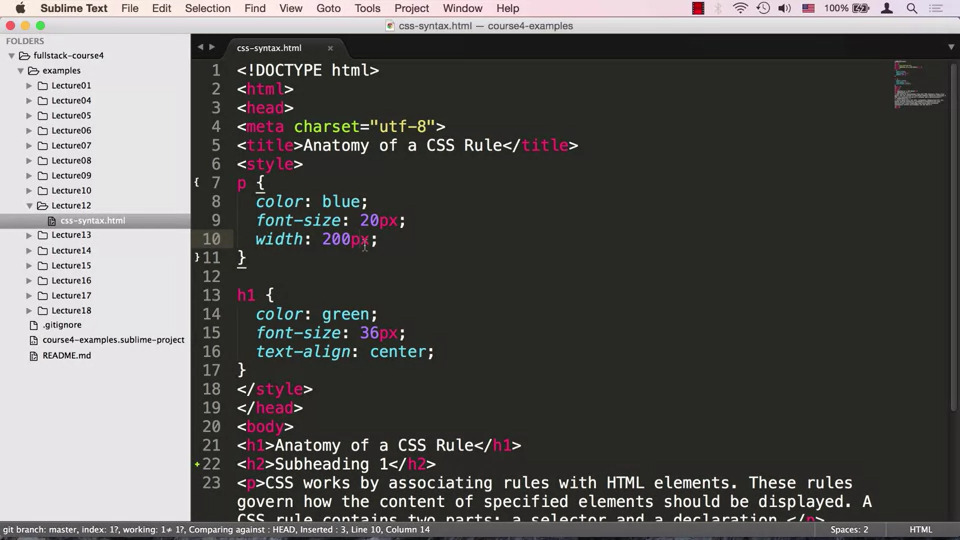
* + A CSS rule consists of a selector, the p tag or paragraph tag is an example, it is saying that whatever rule I am about to tell should apply to all paragraph tags in the HTML page. The tag is followed by curly braces, inside them is declaration which consists of a property and value, the property and value are separated by a colon and is terminated by a semi-colon

****

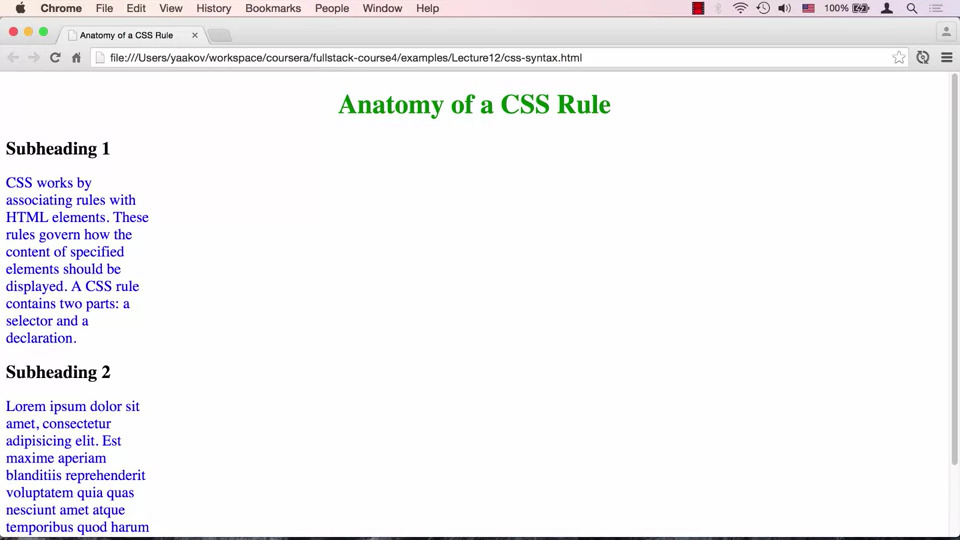
* + A CSS rule can have several declarations

****

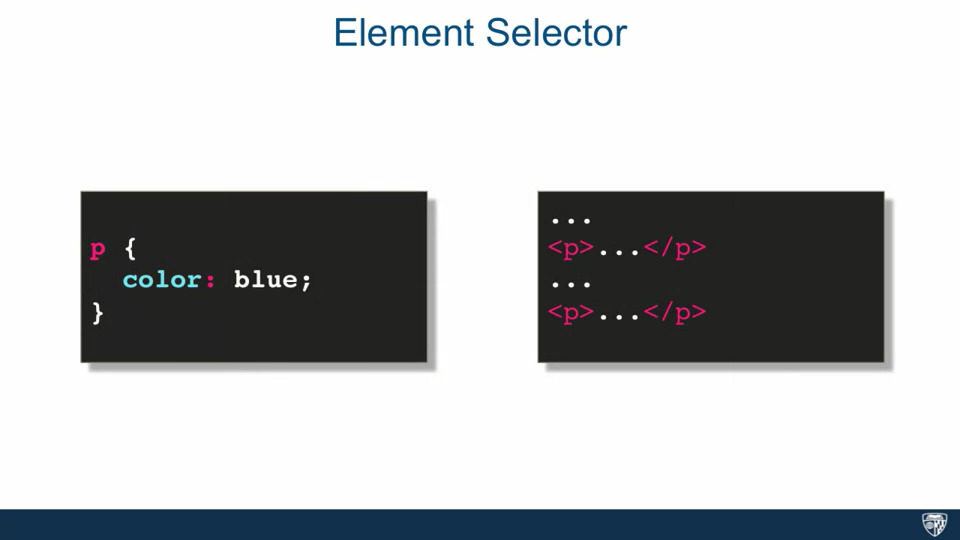
* + A collection of CSS rules is known as a stylesheet

****

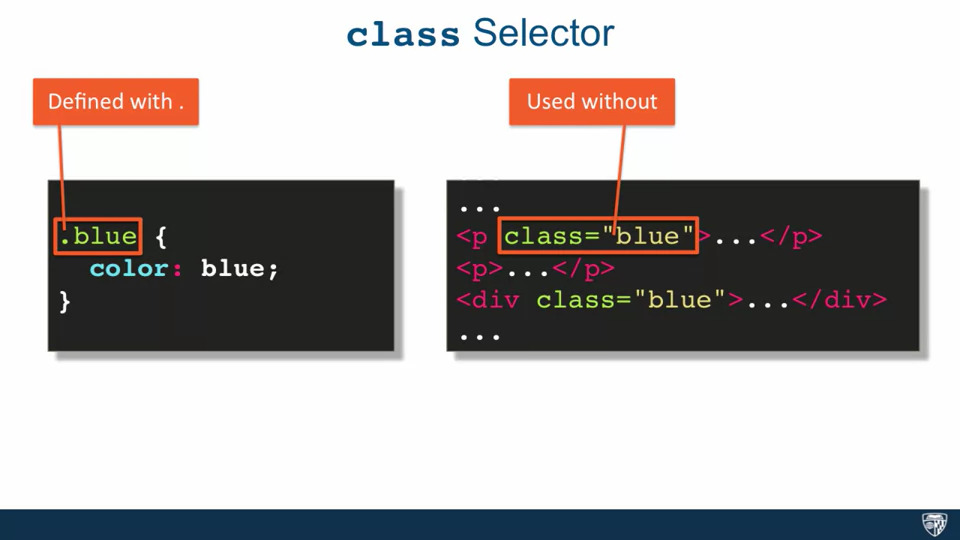
* + In this HTML document we have a heading h1, a couple of h2 and a paragraph represent by <p>
  + The CSS is put directly into the HTML document in this case, our <p> tag will be blue, will have a font-size is 20 pixels and width is 200 pixels. Our h1 tag will be green and will be in the center

****

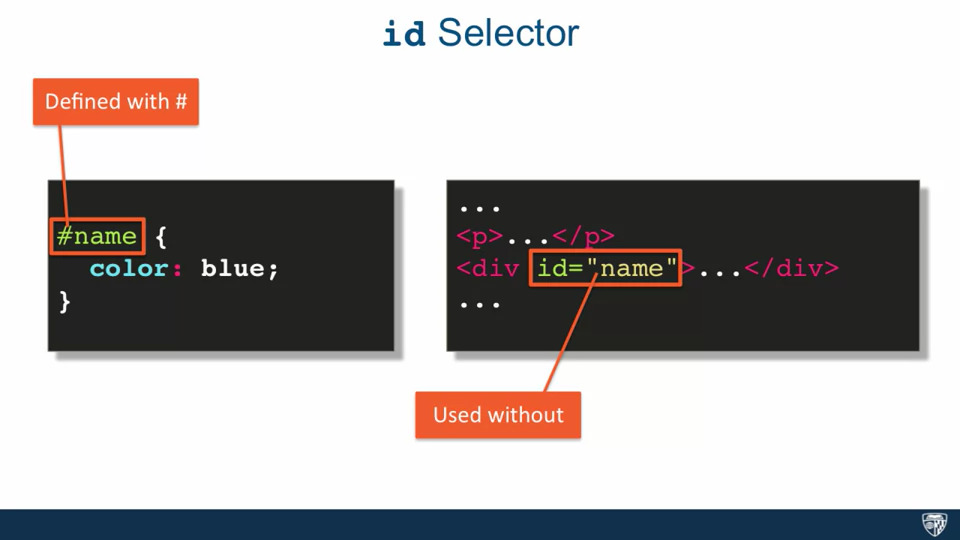
* **13. Element, class, & id Selectors:**
  + CSS selectors are used to determine which HTML elements or set of elements to apply the CSS to.
  + There are 3 different selectors: element, class and id

****

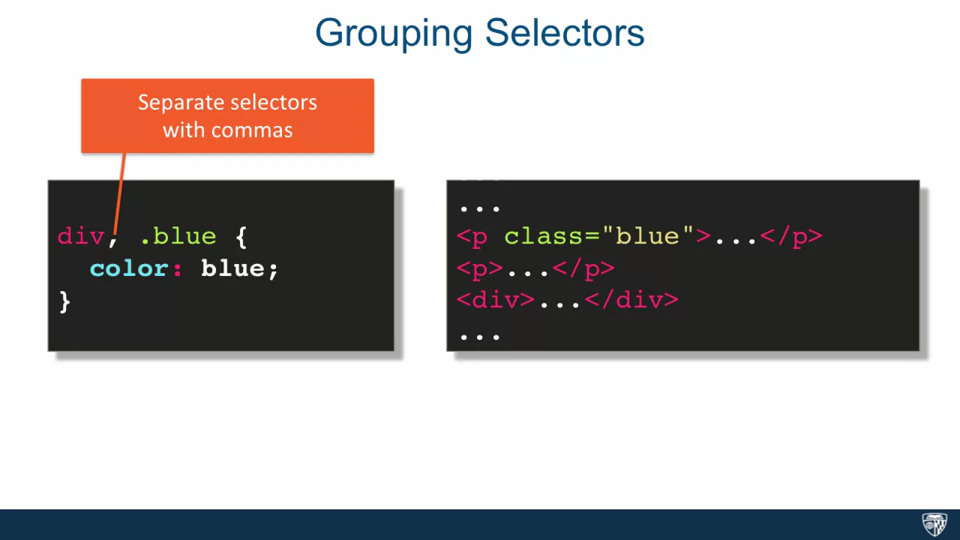
* + The element selector is the user just specifying the element name, in this case p for paragraph, says that every paragraph in our HTML document should be blue

****

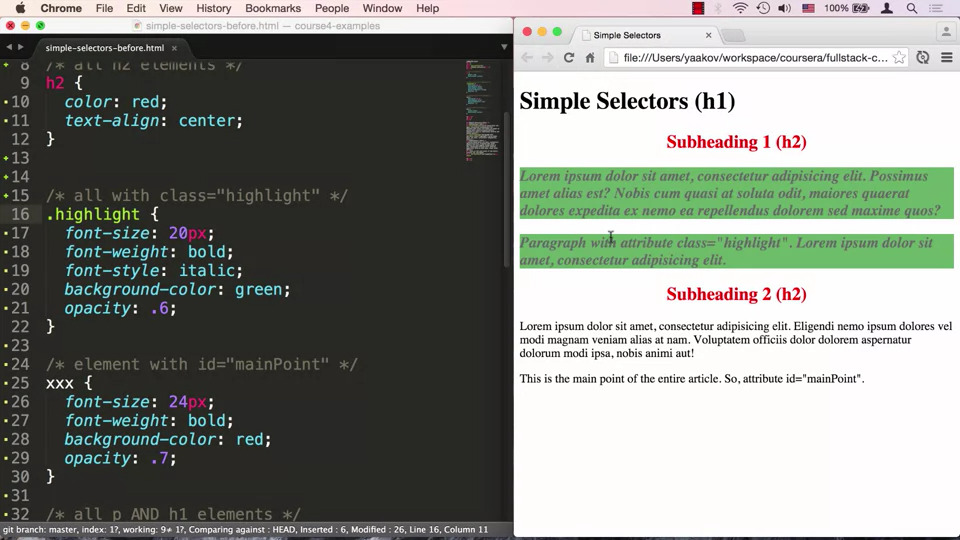
* + We have a class selector, we specify it with a dot and class name, in this case we have a blue class which will color whatever targets with blue. The class selector requires a change in the HTML document, users must add class attribute to their HTML tags. We define the class with a dot but when using the class we don’t put in the dot

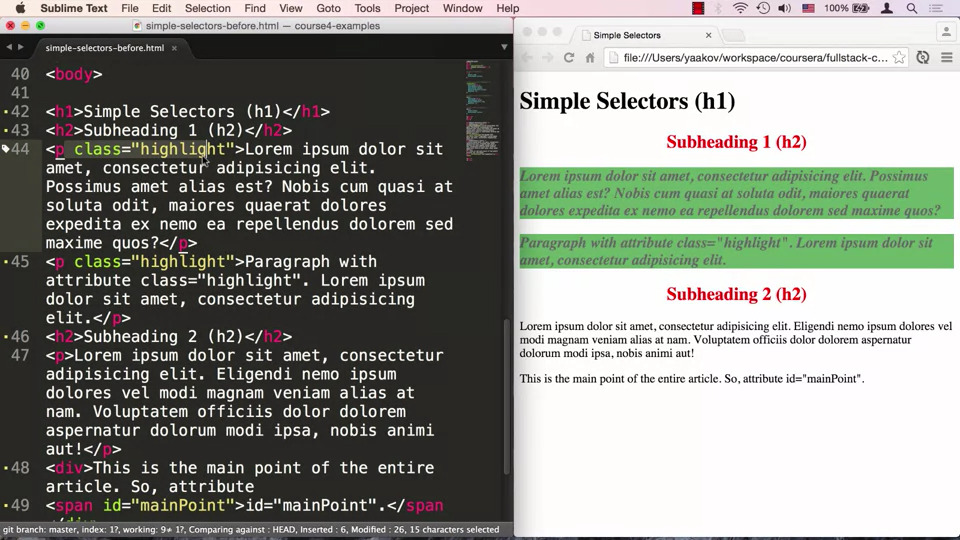
****

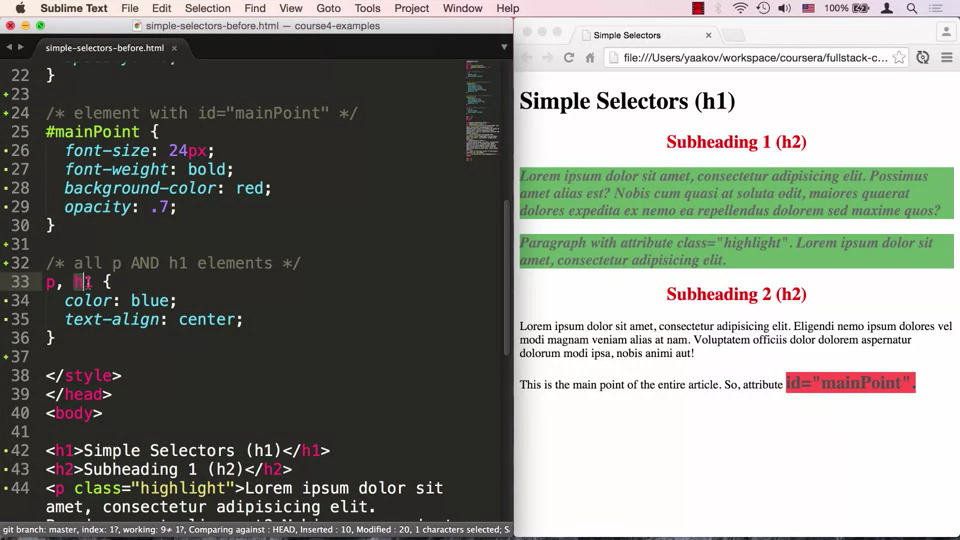
* + The way you specify an id selector is by specifying the value of an id of an element within your HTML document, preceded by a pound sign. If we defined an id selector as name, it will be applied to an HTML tag which has an attribute id set to name

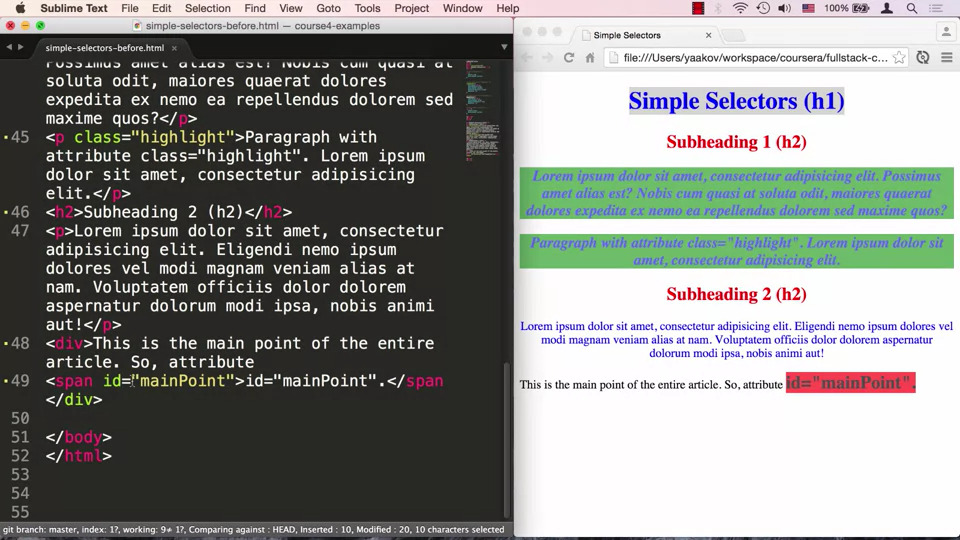
****

* + To write more efficient rules we can group together selectors, here we have two selectors div for the div element and blue for the class blue. Both the div and anything with class=”blue” will appear as blue

****

****

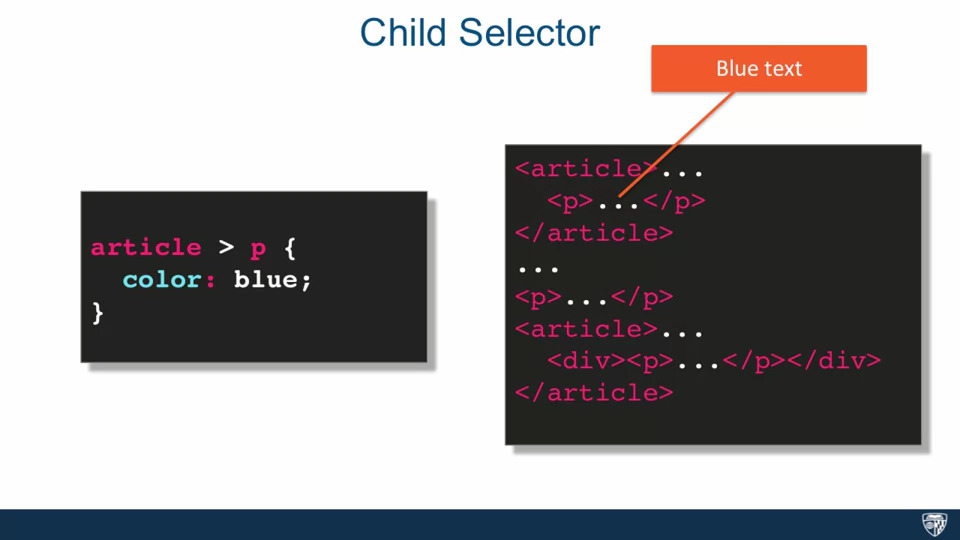
****

****

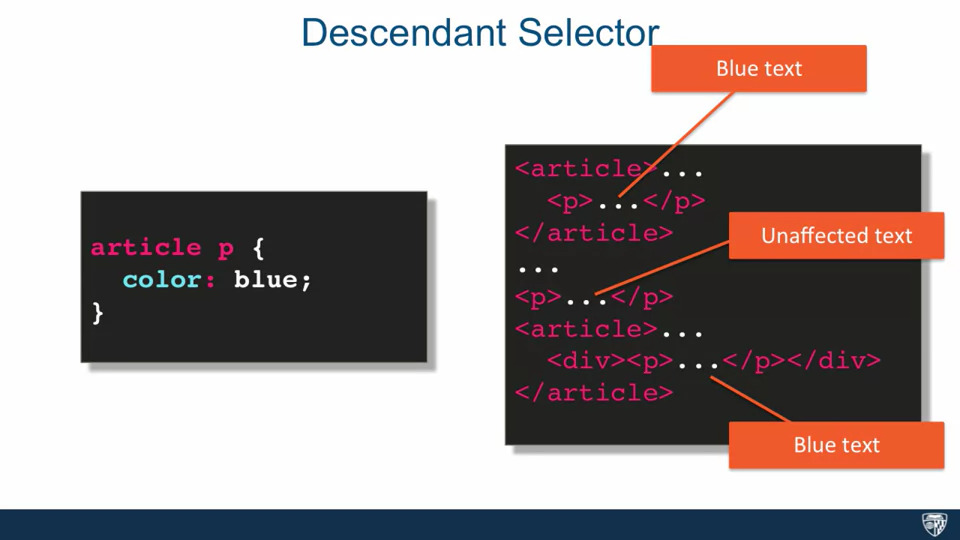
* + We want all h2 elements to be red and in the center thus we define an element selector called h2. We want to effect every single element which has the class set to highlight, so we will define a class called highlight. Next we specify an id selector called mainPoint. Lastly we want to take all elements that are p and h1 group them and apply styling to them, we do this by doing p, h1 and define a rule.
  + The id selector is the least reusable one since by HTML rules, a particular Id attribute value can only appear once in the document, the id mainPoint cannot be used for any other element/tag
* **14. Combining Selectors:**
  + Combining selectors is a very powerful technique and allows us to target dumb elements



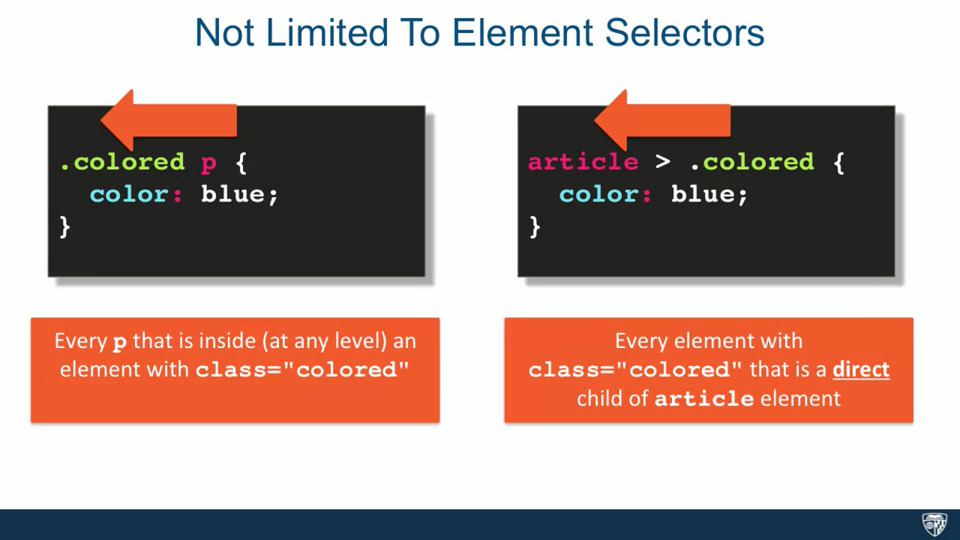
* + In this case we have an Element with Class selector, we want to target every p that has class=”big”. There should be no spaces between the first and second selector. Only p with class big will be affected

****

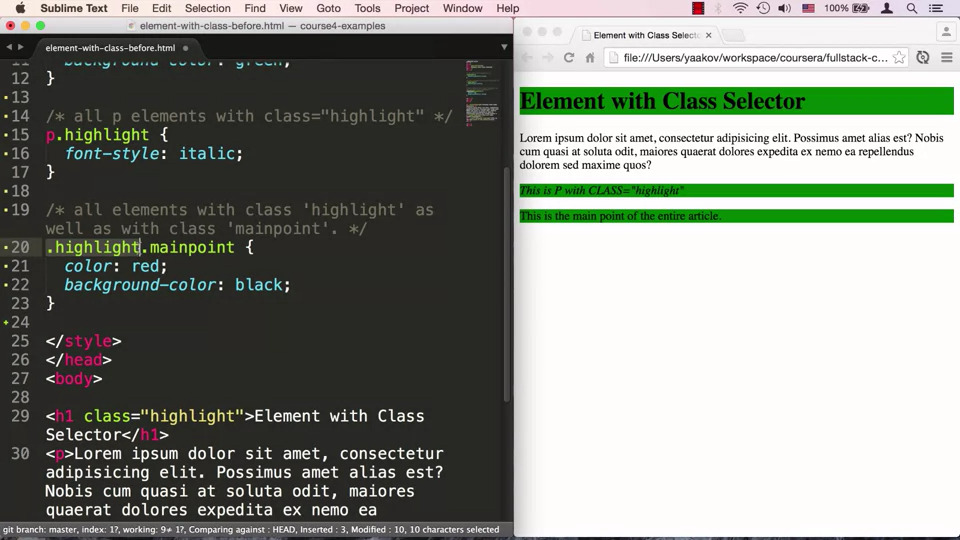
* + In this case we have a child selector, it’s a selector followed by > and another selector, it is read from right to left, in this case we want to target every p element which is a direct child of an article.

****

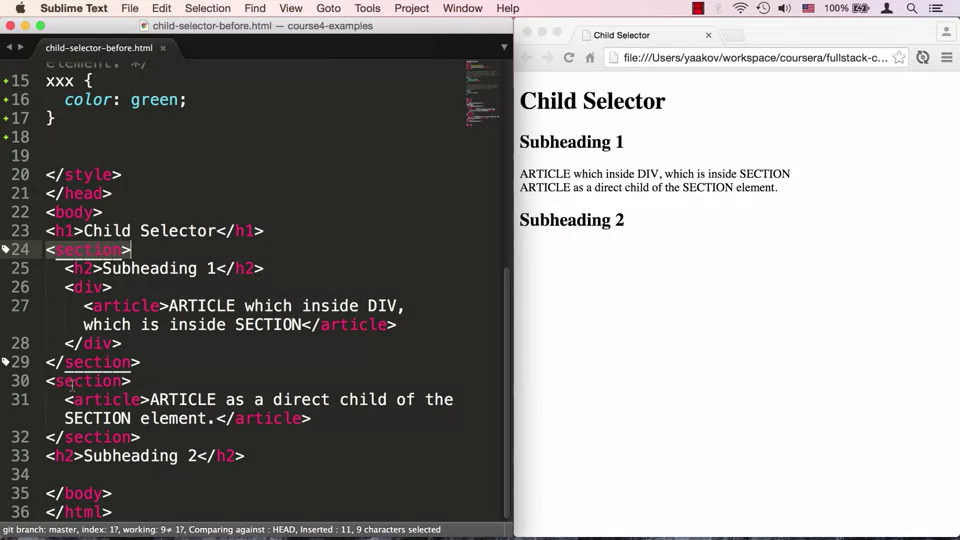
* + We can have a descendant selector, it’s a selector space and another selector. It is read from right to left. In this case we want to target every p element that is inside at any level of an article element. In this case even though <p> is inside <div> it is still inside article.

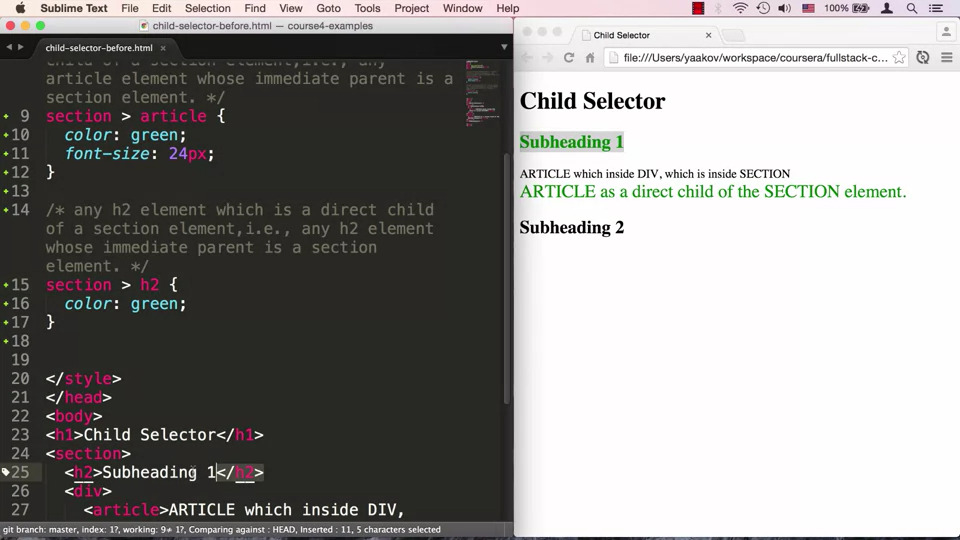
****

* + If have a .colored p then every p element that is inside at any level an element with class=“colored” will be targeted. If we have article > .colored, then every element with class= ”colored” that is a direct child of article element

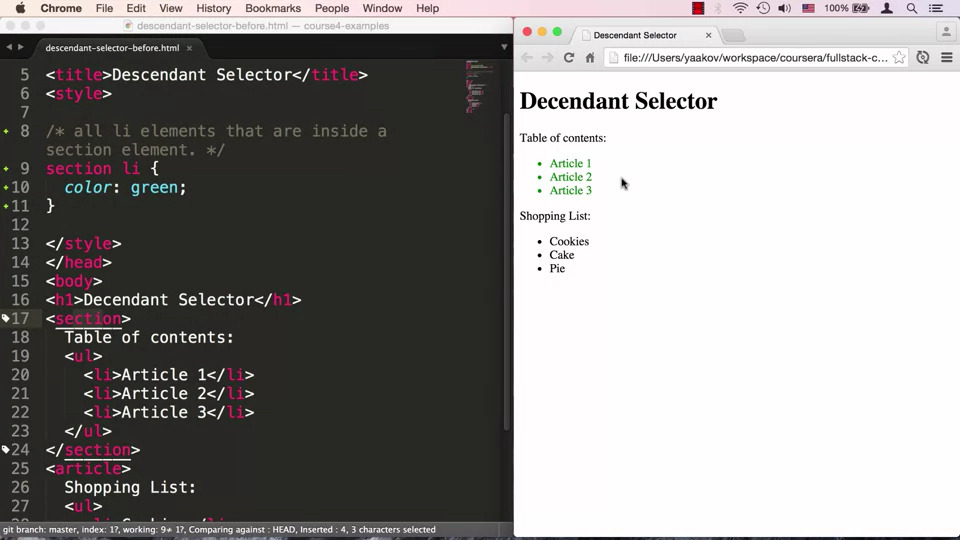
****

* + In this example we have an h1 tag whose class is highlight, we have div tag which has two classes applied to it, mainPoint and highlight. To target everything with class highlight, we can do .highlight, next we want to target all p elements with class highlight, we can do this using p.highlight. Next we want all elements with class highlight and mainPoint, we do this by doing .highlight.mainpoint

****

****

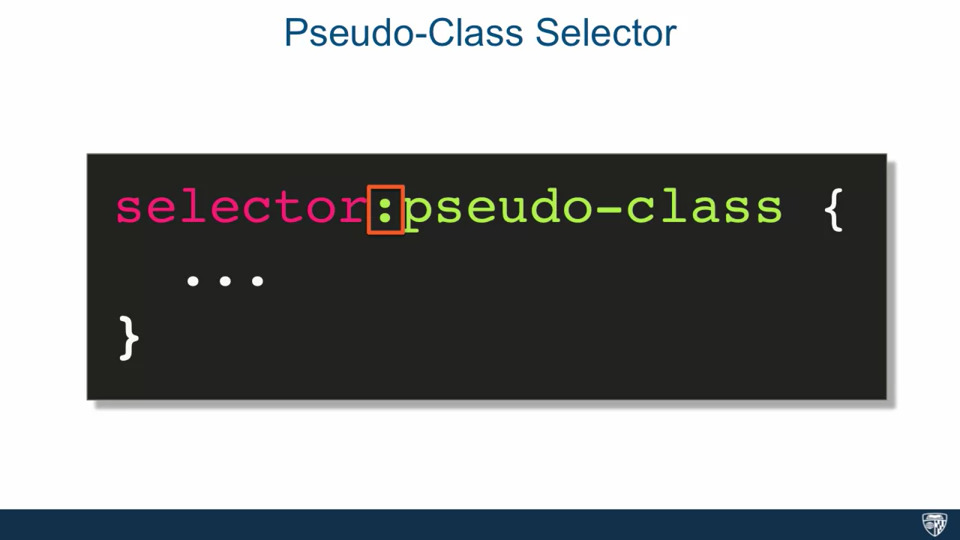
* + In this example we have a couple of section tags, the first section tag has an h2 tag, a div tag which contains inside it an article tag, another section tag which contains an article tag, and a separate h2 tag. First we want to target any article element who’s immediate parent is section, we can do this using section > article, next we want to target any h2 element which is a direct child of a section element, we can do this using section > h2.

****

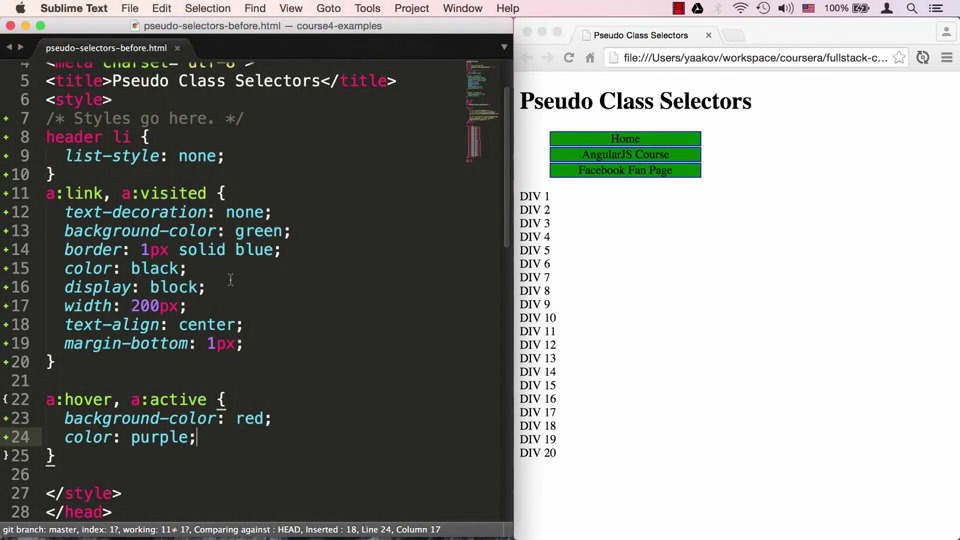
* + In this example we have unordered lists sitting inside the section and article elements. We want to target all li elements which are inside the section element, we can do this using section li, even though ul separates li, since we used descendant selector it should get applied

****

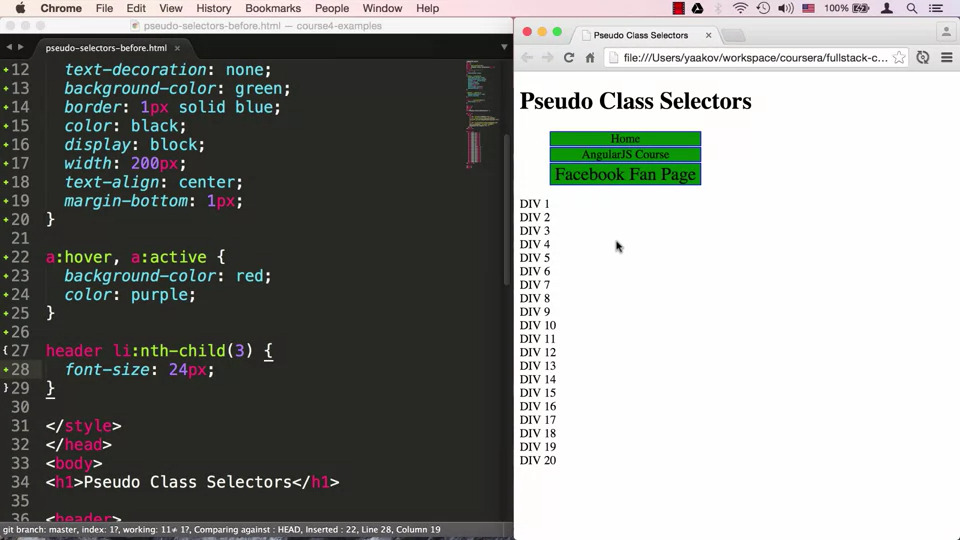
* **15. Pseudo Class Selector:**
  + Pseudo Class Selectors address targeting structures that cannot be targeted by simple selectors, or they targeting the ability to style based on user interaction of the page, for example we want the styling to change if the user hovers over the content

****

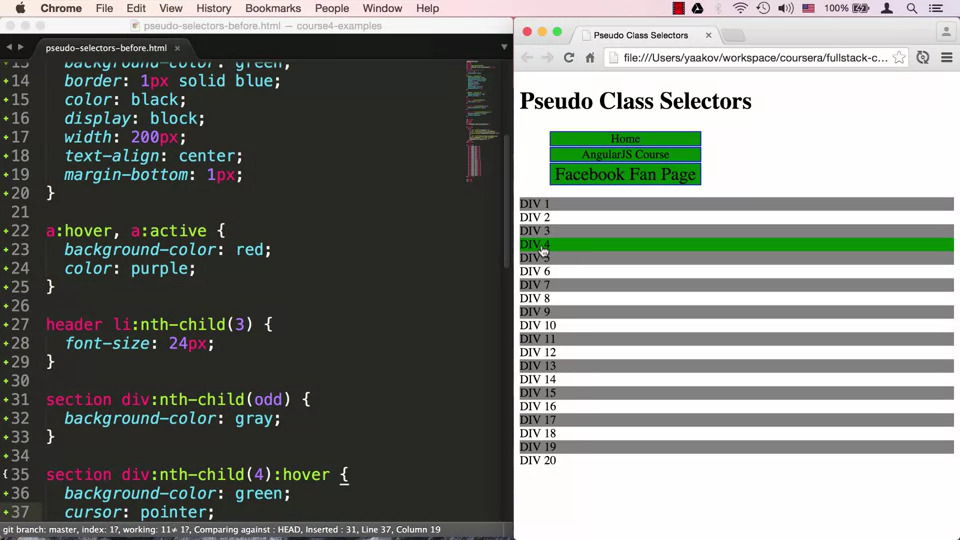
* + We can specify a pseudo class selector by specifying some selector:pseudo-class.
  + There are many pseudo class selectors such as :link, :visited, :hover, :active, :nth child

****

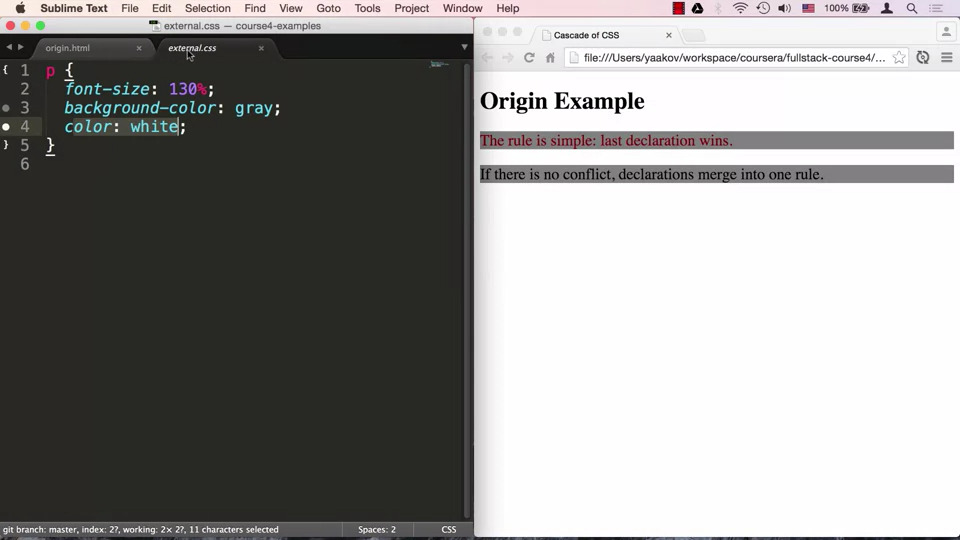
* + In this example, we have a header, in the header we have unordered list, followed by a section which contains a bunch of divs. We want to style the unordered list as a menu buttons. We want to target the list items, we can do this by using descendant selector, header li{}, we can turn of the bullet points by saying list-style: none. We then want to style the links inside the li elements to look like buttons. Links are not easy to style since they have states, these states can be expressed using our pseudo classes, here we are targeting two states, one is a:link and the other is a:visited, we group both these. Link is as the name suggests a link, visited means that after a link is clicked a different style can be applied to it. We can start by removing the underline by using text-decoration: none, set background color to green using background-color: green etc. The a element is inline and block level element, we want our buttons to be block level elements, we can do this using display: block. We can specify width, text-align and margin of the buttons as well. We want to interact with the buttons when we hover over them such as pointer should change when we move over. We can do this by defining two more states of the link, a:hover which is when the user hovers over the element, in this case the “a” element, and a:active is when the user clicks an element but hasn’t released it, since we don’t want to differentiate in this case, we will group them together. We can then specify what happens for a:active and a:hover by specifying background color to turn red when we move over the button and the color of the text to turn purple.

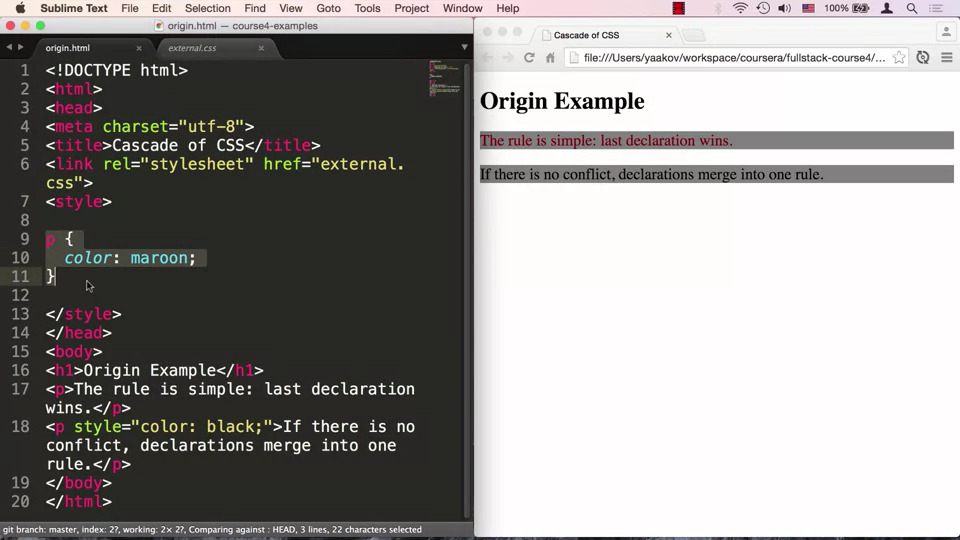
****

* + The nth-child pseudo selector allows us to target a particular element within a list, in our case we want to make the Facebook page button bigger. In our HTML, the Facebook link is located in the header section and there are three li elements, the Facebook link is the third li element. We can go back to our style tag and specify that we want to target the li element that is inside the header element and we want to target the 3 child, we do this using header li:nth-child(3), and we specify that we want font size to be bigger.
  + We now want to target our div elements inside section and we want every odd member of div to have a background color of grey. We can do this using section div:nth-child(odd) and then set background color to gray.

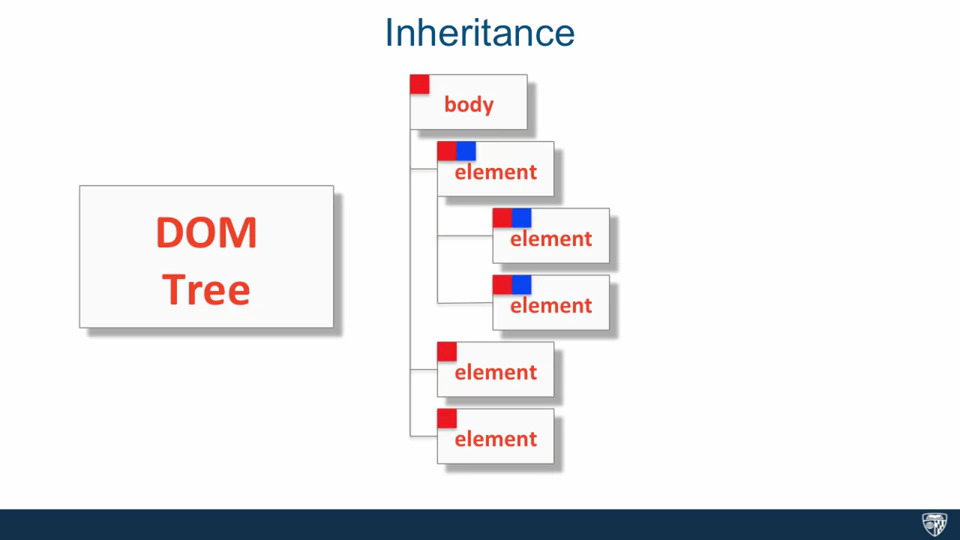
****

* + If we want the fourth element in this list to have a different hover behavior we can combine pseudo selectors, first we target section div:nth-child(4), and then add :hover, so section div:nth-child(4):hover. We can then specify the background color and pointer change.
* **16. Style Placement:**
  + Our choice of placing styles affects how reusable the styles and which style declaration overrides the other.
  + So far we have been specifying styles inside the style tag. We can specify a CSS style, for example we can apply it directly on an element by providing the style attribute, we don’t need a target since we are inside the element such as <p style=”text-align: center;”> I am centered!</p>. This is called inline styling and is most restricted and least preferred way of styling
  + When we have multiple HTML pages, we need external style sheets, we can specify it in the HTML document using link, for example : <link rel=”stylesheet” href=”style.css”>, the rel tells the browser that it a style sheet, and href is set “style.css” , and should be in the same directory as the HTML document.
  + We always want an external CSS style sheet
* **17. Conflict Resolution Part 1:**
  + The Cascade algorithm is at the core of using CSS style sheets, the cascade combine the importance, origin, specificity and source order of the applicable style declarations to determine exactly which declaration should be applied to any given element and if there's a conflict, how to resolve that conflict. In other words, how to tell which CSS rule wins
  + To have a working knowledge of the Cascading algorithm, we need to understand origin, merge, inheritance, specify
  + Origin Precedence kicks in when two declarations are in conflict in other words they specify the same property for the same target, the rule is that the last declaration wins, HTML is read from top to bottom, the lower they are in the page, the more precedence they have. We can think of the entire content of the external CSS where cut and pasted in the head tag

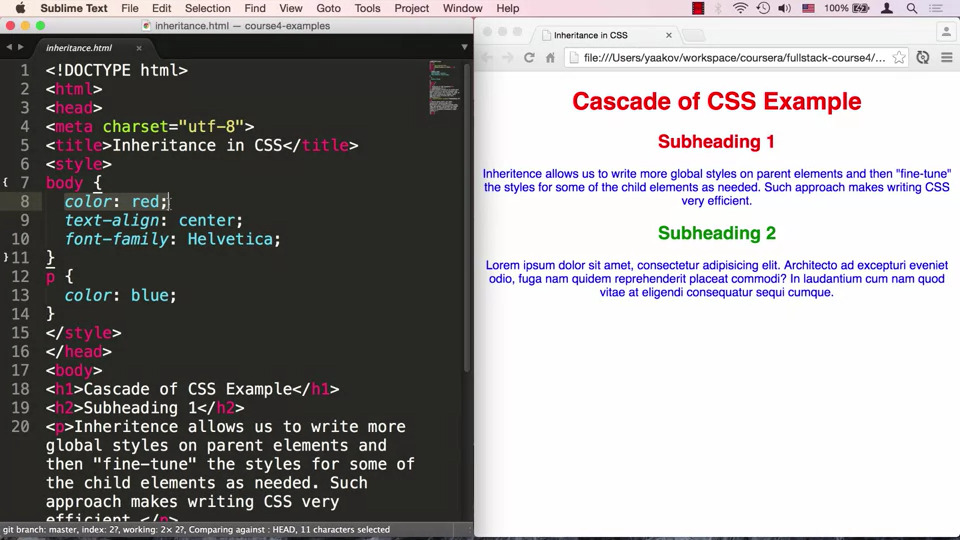
****

****

* + When different CSS declarations do not conflict, that is they target the same element but the CSS properties with which they target are different, we use Declaration Merge, for example a declaration for font size and declaration for color since they are two different properties, when they target the same element, the element will get both of them. In this case we have an external CSS declaration for the p tag which sets the color of the text to white, inside the HTML document we have a style tag inside the head which overrides the p tag and sets color to maroon. We then declare a paragraph, inline where we specify the color the text as black. We should think that the external CSS was cut and pasted inside the style tag, since the maroon declaration overrides it, our paragraph is maroon, the same thing for black, this is origin precedence. Our text still has a gray background, which is declaration merge.

****

* + Inheritance states that if we specify a CSS property, then all the grand children will inherit that property

****

* + It is common practice to specify some CSS declarations for the Body tag, in it we can specify some details which can apply to the whole HTML document. In this case we specified a font for the entire document, we then specified a specific color blue for the paragraph, even though each paragraph will inherit the color red but it is overridden with blue. Even though we did not specify design for h1, h2 tags, since they are inside the body tag they inherit its style.