Basics of HTML and CSS:

Introduction

What I Mean by "Basic" Techniques. We'll range beyond basic HTML and CSS in this class. But here's a list of what I consider some basic techniques.

HTML5:

- Creating a basic HTML5 file.
- Defining a head and body section of your HTML page.
- Structuring your page with div tags.
- Defining basic HTML tags for headings, paragraphs, lists, and links.
- Styling tags with a linked, external style sheet.

CSS3:

- Creating a CSS file
- Defining CSS styles for tags like headings, paragraphs, lists, and links
- Defining ID and class styles for div tags
- Implementing page designs with floated div tags
- Defining styles that apply to multiple tags
- Defining styles that apply to tags only when those tags are embedded in other tags

Setting Up Your Website

I know you're eager to start working with HTML and CSS files. But first we need to do a little prep work—because when you create your files, you'll need a place to put them.

Organizing a Website Folder

The first step is to create a root folder for your website. This is a folder (also a directory) in which all your site content is stored. If you plan to work through this lesson step-by-step, I strongly suggest you create a folder on your desktop named "INFR3120" and store all your website files there.

As you create and work with HTML and CSS files, you'll save them to this folder with a filename that has no uppercase characters, no spaces, and no special characters (like !, ?, or @).

Now you need a reliable code editor, I am using Visual Studio Code, you can use editor of your choice. Do not use a text editor, like Word, or Google Docs, or Windows Notepad! Text editors will corrupt HTML and CSS code with things like smart quotes instead of regular straight quotes.

“ ”
Smart Quotes
" "
Straight Quotes

Creating and Saving an HTML5 File

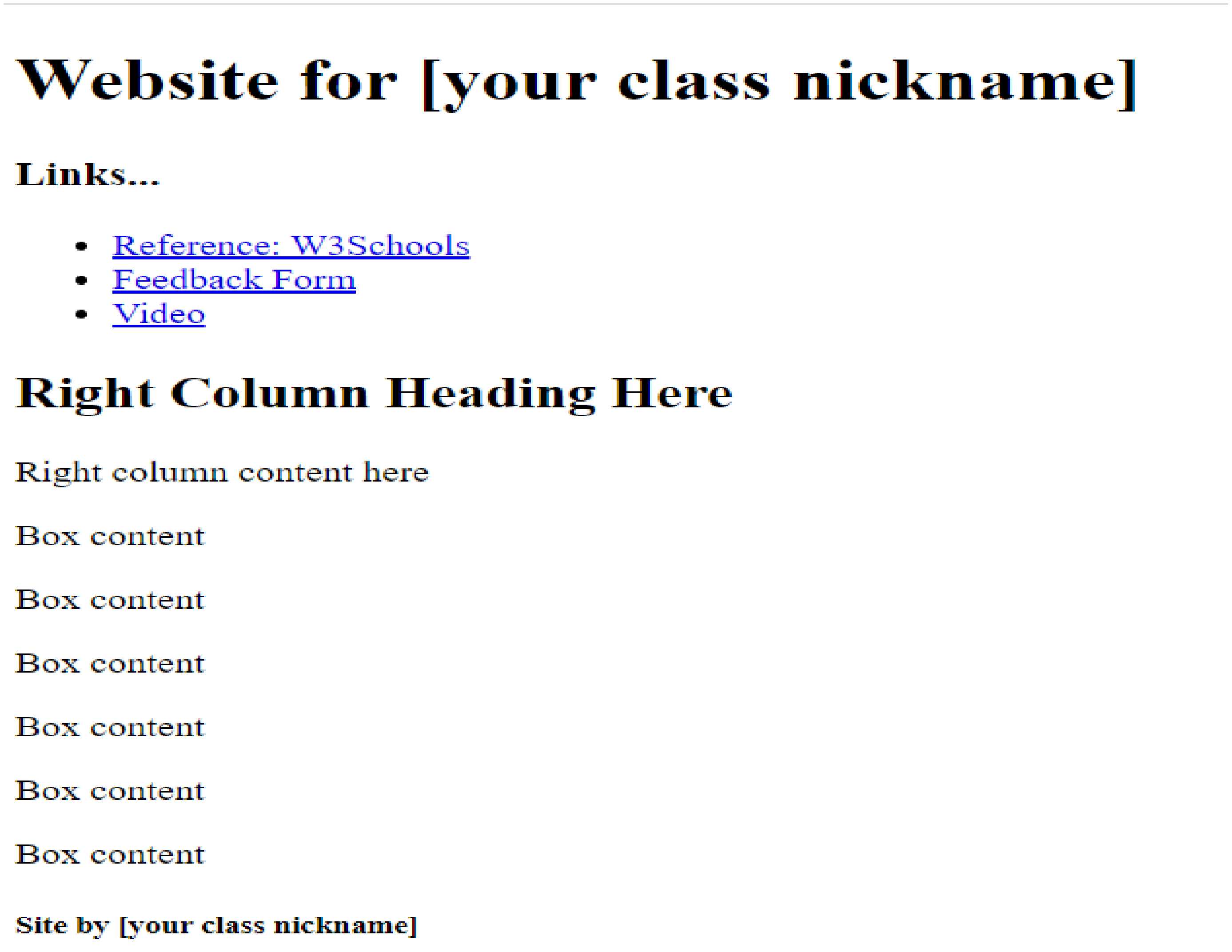
You're ready to explore the template. Copy this HTML into your code editor:

```
<!--The HTML5 doctype declaration is very simple-->
<!DOCTYPE HTML>
<!--All page content is inside the HTML element-->
<html>
<!--Head element content is not visible in a browser window-->
<head>
<!--The UTF-8 character set provides the most support for all symbols and
characters-->
<meta charset="UTF-8">
<title>HTML Template</title>
<!-- The following line links to our style sheet file-->
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<!--Content visible in the browser's window is inside the HTML element-->
<body>
<!--All our content is enclosed in the wrapper ID style-->
<div id="wrapper">
<!--The banner ID style is inside the wrapper ID style-->
<div id="banner">
<h1>Website for [your class nickname]</h1>
</div>
<!--left-column ID style is floated left and used for navigation-->
<div id="left-column">
<h3>Links...</h3>
<ul>
<li> <a href="http://www.w3schools.com/">Reference: W3Schools</a></li>
<li> <a href="feedback.html">Feedback Form</a></li>
<li> <a href="video.html">Video</a></li>
</ul>
</div>
<!--right-column ID style is floated right and used for content-->
```

```
<div id="right-column">
<h2>Right Column Heading Here </h2>
<p>Right column content here </p>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
</div>
<!--The clear class style clears (removes) float-->
<div class="clear"></div>
<div id="footer">
<h5>Site by [your class nickname]</h5>
</div>
</div>
</body>
</html>
```

Before we break down how this page works, save your own version of it. With the code copied into your text editor, choose File > Save. Navigate to your site folder, and save the file as index.html. If your code editor has an option for encoding, choose Unicode (UTF-8).

Code is full of comments, I will explain rest of the code during the session but output of the above code should look like below:


Website for [your class nickname]

Links...

- Reference: W3Schools
- Feedback Form
- Video

Right Column Heading Here

Right column content here

Box content

Box content

Box content

Box content

Box content

Box content

Site by [your class nickname]

Figure 1 Preview of HTML page

Deconstructing an HTML5 Web Page

Let's focus on some important components of a basic Web page. First, note that the HTML includes many comments. The code for a comment is:

<!--this is a comment-->

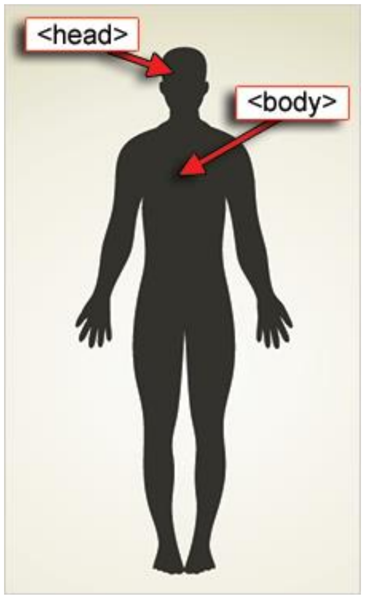
Next, note that we're using the HTML5 document type (doctype) declaration to tell browsers that this is an HTML file. HTML5 has this doctype declaration:

<!DOCTYPE HTML>

Other doctype declarations are for older versions of HTML, and not all browsers support every new element in HTML5. So, should we be using an HTML5 doctype declaration? Yes. Not all browsers support new HTML5 elements, but that doesn't matter when it comes to a doctype declaration. Every browser can interpret the <!DOCTYPE HTML> tag and "learn" that this is an HTML page.

Examining Head and Body Elements

Okay, "head and body elements" sounds as if it belongs in a course for premed students. But no, relax, you're in the right class.



The terms "head" and "body" are based on those parts of the human body. Like all HTML pages, our template has a <head> element and a <body> element. As you'd expect, the <head> element goes at the top.

Here is the head part of our code:

```html
<!--The HTML5 doctype declaration is very simple-->
<!DOCTYPE HTML>
<!--All page content is inside the HTML element-->
<html>
<!--Head element content is not visible in a browser window-->
<head>
<!--The UTF-8 character set provides the most support for all symbols and
characters-->
<meta charset="UTF-8">
<title>HTML Template</title>
<!-- The following line links to our style sheet file-->
<link rel="stylesheet" type="text/css" href="style1.css">
</head>
```

A meta tag is a label that defines everything in the document. The head element includes the charset meta tag, which in this case defines the character set (the full list of available letters, numbers, and symbols) as UTF-8, which all browsers support. The head element also includes the title element. The <title> tag defines the page title, which appears in a browser's title bar or tab bar. For example, the title of the Internet Movie Database website is "IMDb -Movies, TV and Celebrities."

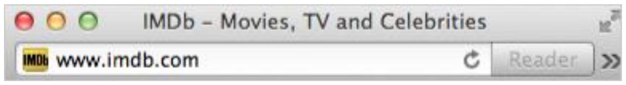


Figure 2 Website title

And the code for that title is:

<title>IMDb - Movies, TV and Celebrities</title>

Before we move past the <head> element, note the link to the file that will be the CSS style sheet for our page. We'll talk more about CSS and about style sheets in this lesson and in later lessons; for now, I just want you to notice the location. In our case both index.html and style.css are in the same folder. Everything that visitors will see in our Web page (other than the title, which appears in a browser title bar) is defined in the <body> element. Let's look into the <body> element of our code:

```html
<!--Content visible in the browser's window is inside the HTML element-->
<body>
<!--All our content is enclosed in the wrapper ID style-->
<div id="wrapper">
<!--The banner ID style is inside the wrapper ID style-->
<div id="banner">
<h1>Website for [your class nickname]</h1>
</div>
<!--left-column ID style is floated left and used for navigation-->
<div id="left-column">
<h3>Links...</h3>
<ul>
<li> <a href="http://www.w3schools.com/">Reference: W3Schools</a></li>
<li> <a href="feedback.html">Feedback Form</a></li>
<li> <a href="video.html">Video</a></li>
</ul>
</div>
<!--right-column ID style is floated right and used for content-->
<div id="right-column">
<h2>Right Column Heading Here </h2>
<p>Right column content here </p>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
</div>
```

```html
<!--The clear class style clears (removes) float-->
<div class="clear"></div>
<div id="footer">
<h5>Site by [your class nickname]</h5>
</div>
</div>
</body>
```

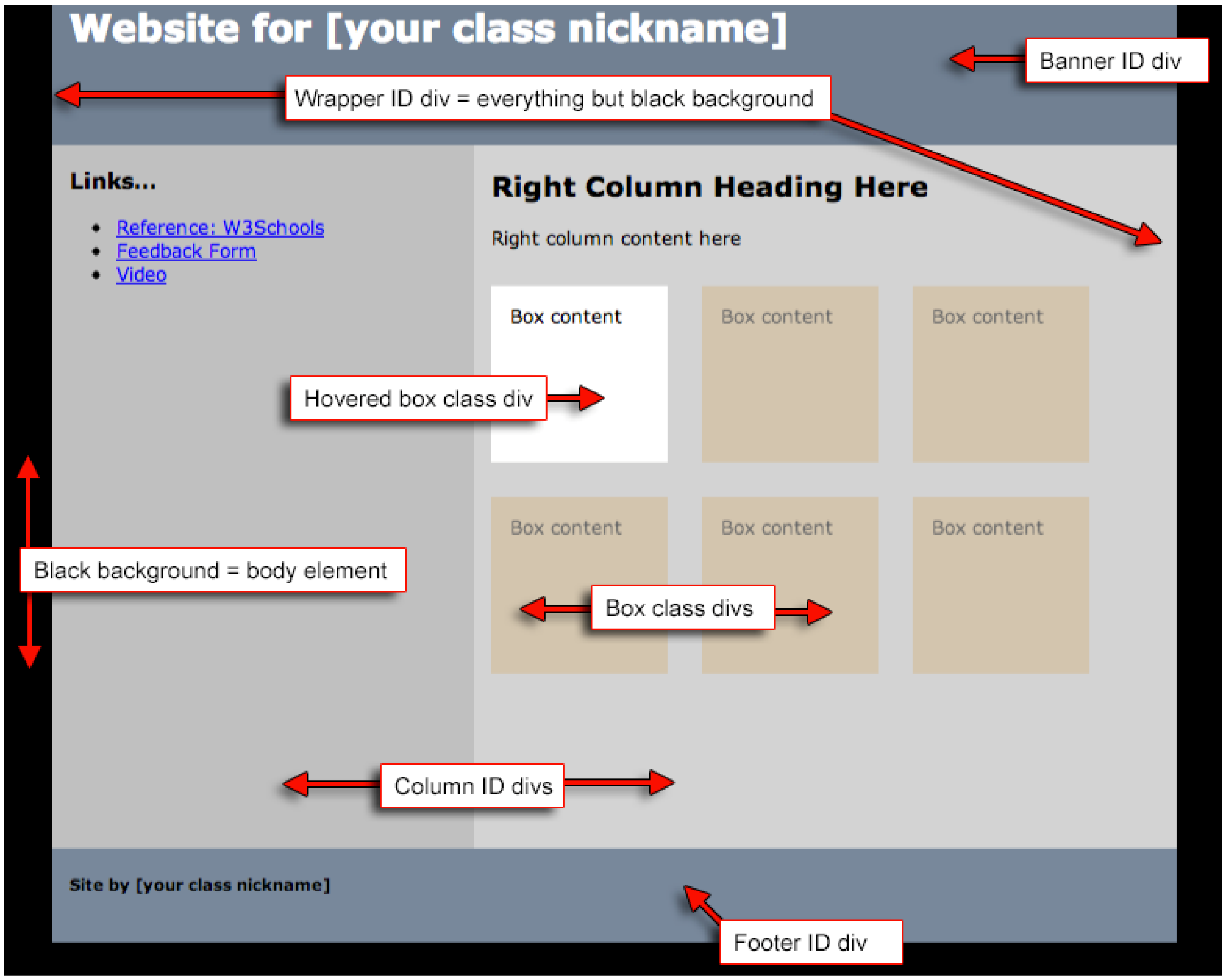
Inside the <body> element, all of the page content is inside a div tag with an ID selector we named “wrapper”. In this case, the div tag that contains all our page content is the "wrapper ID div" for short. By itself, the div tag has no display properties. It takes its "shape"—its width, height, and other attributes—from the ID style associated with it.

Understanding ID Div Tag and Class Div Tag Styles within the wrapper div, you can see these features:

- A banner ID div tag at the top of the page
- Left-column and right-column div tags that are floated (aligned) left and right respectively
- A footer div tag at the bottom of the page

You use ID div tags for elements that occur only once on a page. They're appropriate for our wrapper, our banner, our two columns, and our footer. But they aren't appropriate for the six boxes in the right column.

I discuss more about div during session. For more detail visit (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div)


Website for [your class nickname]
Banner ID div
Wrapper ID div = everything but black background
Links...
Reference: W3Schools
Feedback Form
Video
Right Column Heading Here
Right column content here
Box content
Box content
Box content
Hovered box class div
Box content
Box content
Box content
Black background = body element
Box class divs
Column ID divs
Site by [your class nickname]
Footer ID div

Don’t worry our output will be same like above page but after applying CSS, which we apply later. Those six boxes are identical in shape, so we need a style that we can assign to multiple divs on the page. In our model, that's the box class style. You can see that style applied to six div tags. It looks like this:

```
<div class="box"><p>Box content</p></div>
```

Div tag boxes have replaced tables for page layout. One of the reasons for this replacement is that you can use something called CSS pseudo-classes to animate class styles. You'll see how that's done later, when we deconstruct the CSS for our project. But before we do, I want to draw your attention to the div element with the clear class style.

Here's the line of code I'm referring to.

```
<div class="clear"></div>
```

This div tag "ends" the floating that would otherwise be inherited from (in other words, continue from) the floated boxes that precede it.

Looking at Other Links and Lists

We have other HTML tags in our model, including links (<a>), unordered lists of bullets rather than numbers (<ul>), and list items (<li>). Put them all together, and they look like this:

<ul> <li> <a href="http://www.w3schools.com/">Reference: W3Schools</a></li>

A disclaimer of sorts: Even though I've shown you a "model" HTML template, that doesn't mean you have to design all your pages this way. You could use plenty of other techniques. But the overall approach and specific code I've given you provide a useful starting point for creating basic HTML pages.

Now lets dive into CSS, lets add the below css code into the same folder with the name style.css

```css
@charset "UTF-8";
/* CSS Document */
/* The body tag style applies to all elements on the page */
body {
background-color: black;
font-family: Verdana, Geneva, Arial, sans-serif;
padding:0px;
margin:0px;
}
/* The wrapper ID style is used with a div tag to provide a 960px wide page */
#wrapper {
width: 960px;height: 800px;
margin-left: auto;
margin-right: auto;
background-color:LightSlateGrey ;
}
/* The right-column ID style is floated right */
#right-column {
float: right;
width: 600px;height: 600px;
background-color:LightGray;
}
/* The left-column ID style is floated left */
#left-column {
float: left;
width: 360px;background-color:silver;
height:600px;
}
/* Defining a style for a set of tags separated by commas applies the style to
all tags */
h1,h2,h3,h4,h5,h6,p,li {
margin-left:15px;
}
#footer {
height:80px;
}
```

```css
#banner{
height:120px;
background-color:SlateGrey;
}
/* Defining a style for a set of tags not separated by commas applies the style
only to all tags within the preceding tag*/
#banner h1 {
color: white;
}
/* Advanced Web design relies on class or ID styles, never tables */
.box {
height: 150px;
width: 150px;float: left;
background-color:tan;
margin: 15px;
opacity:.5;
}
/* The following pseudo-class applies to the box class when in a hovered state */
.box:hover {
background-color:white;
opacity:1;
}
/* This clear class style terminates float */
.clear{
clear: both;
}
```

CSS stands for Cascading Style Sheets, a language that lets you control the look and formatting of your Web pages. Let's walk through this code and note key techniques. I opened this file with @charset "UTF-8";. The UTF-8-character set is important in HTML files because it allows browsers to interpret characters and symbols from a wide range of alphabets. I've never found a definitive argument that you need that for a style sheet file, but it doesn't cost anything to add that line of code, and everyone does it. So, I do as well. I've also added extensive comments to this file. Comments in CSS3 look like this: /* CSS Document */

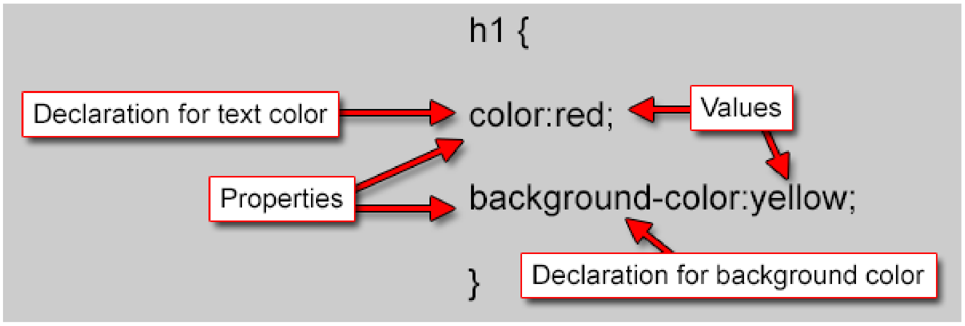
Styles for Tags.

The most important style definition in a CSS3 file is usually for the <body> tag. That's because everything on the page that displays in a browser window goes inside the <body> and </body> tags—in other words, within the <body> element. Like all CSS3 style definitions, the one for the <body> element opens with the name of the element followed by a left curly bracket character, which looks like this: {. And the style definition closes with a right curly bracket:}.

Individual styles within a style sheet are called selectors or rules. For example, if you create a style for the <h1> element, that's a selector. If you create a class style called ".box" (class styles always begin with a"."), that too is a selector. Creating a style for the <body> tag is also a selector. You're starting to see a pattern here, right? Selectors consist of declarations, which are made up of properties and values. Take a look at this example:

```
h1
{
    color:red;
    background-color:yellow;
}
```

This selector is for the <h1> element. It has two declarations: one for text color and one for background-color. Each of those declarations has two parts: a property and a value. In the first declaration in our example, the property is color, and the value is red. In the second declaration, the property is background-color, and the value is yellow.



Use a colon to separate the two parts of every declaration. Another way to put that is: Separate the property (such as background-color) and value (such as black) with a colon.

Defining a Wrapper

Our #wrapper ID style, which defines the div tag box that wraps our entire page content, is 960 pixels(px) wide. A pixel, which is short for "picture element," is a single point within an image. The 960-pixel width has become the standard in all professional Web design work flows. See the reference material on CANVAS for information on the 960 grid. This width works well in all desktop and laptop viewports (monitor sizes) and in many tablet view ports and the number 960 is easily divisible into columns of two, three, six, and so on. That makes it easy to design artwork, create wireframes (sketches) of column layout, and prepare photos and video for the Web.

Applying Float

I mentioned earlier that our right and left columns have float properties (with values of right and left respectively). By default, most HTML elements take up a whole line or row of a page. Defining a left or right float change that and allows content to flow around a box. It's often necessary to define a class style that will clear, or remove, any inherited float properties so that elements on a page revert to filling an entire row. That's necessary in our model after the boxes end and before the footer style is applied to a div tag. So, our CSS style sheet includes this style, which clears any float properties:

```
.clear
```

```
{
    clear: both;
}
```

Using Multiple Styles and Compound Styles

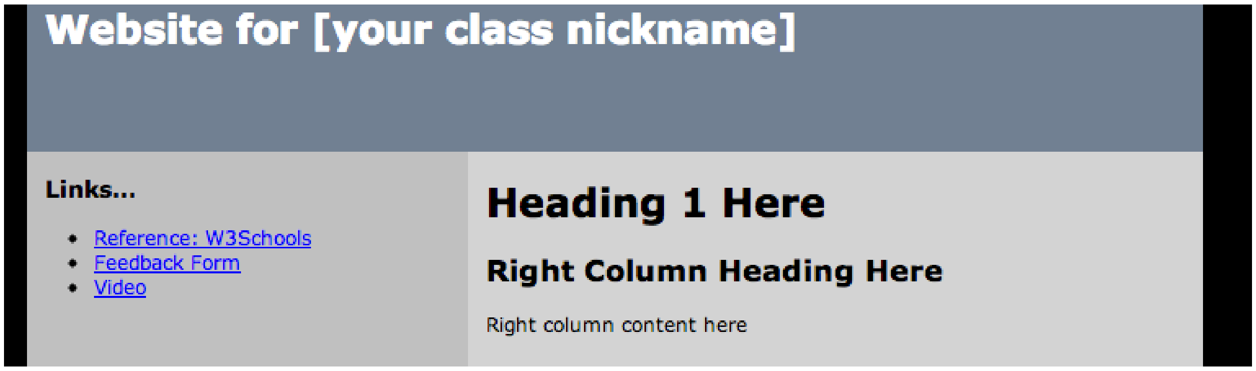
Next, I want to focus on the two style definitions that apply to more than one style. One of those defines a property and value for all six heading levels, the paragraph tag, and the list tag:

```
/* Defining a style for a set of tags separated by commas applies the style to
all tags */
h1,h2,h3,h4,h5,h6,p,li {
margin-left:15px;
}
```

See the commas between the elements in this style rule? That means we're dealing with a multiple style —one that applies to all these elements. This particular multiple style applies a margin that separates the left side of each of these elements from the border of whatever box encloses it. On the other hand, when we list elements in a style definition without commas between them, we create a compound style —one that applies only when the last element is enclosed inside all the elements listed before it. For example, the following style definition displays (text) color as white for the h1 element, but only when that element is inside a #banner ID.

```
/* Defining a style for a set of tags not separated by commas applies the style
only to all tags within the preceding tag*/
#banner h1 {
color: white;
}
```

In other words, the heading text will be white inside the banner but not outside the banner. Our model has an <h1> element in the banner (with the text "Website for [your class nickname]"). If we add an <h1> element to the right column in our model, for example, that element won't adopt the style definition for an <h1> tag that appears inside a #banner style element.


Website for [your class nickname]
Links...
Reference: W3Schools
Feedback Form
Video
Heading 1 Here
Right Column Heading Here
Right column content here

See how "Website for [your class nickname]" is white but "Heading 1 Here" is black? That's because we're using a compound style. The ID styles in our style sheet, like the #banner style, begin with the # symbol in our CSS file. Remember that you can use a particular ID style only once in any single HTML file.

Class styles, which you can use repeatedly in an HTML file, begin with "." You can see this in the .box class style that defines the small boxes in the right column of our page. Pseudo-class styles apply to an element only in a specific state of user activity, like when a user clicks on or hovers his or her mouse over an element. In our model, a pseudo-class changes how our .box class style displays when in a hovered state:

```
/* The following pseudo-class applies to the box class when in a hovered state */
.box:hover {
background-color:white;
opacity:1;
}
```

Defining Opacity

One last thing before we close this compressed survey of a sample CSS file. Note that the pseudo-class style defines the opacity property for the box in its normal state and in a hovered state.

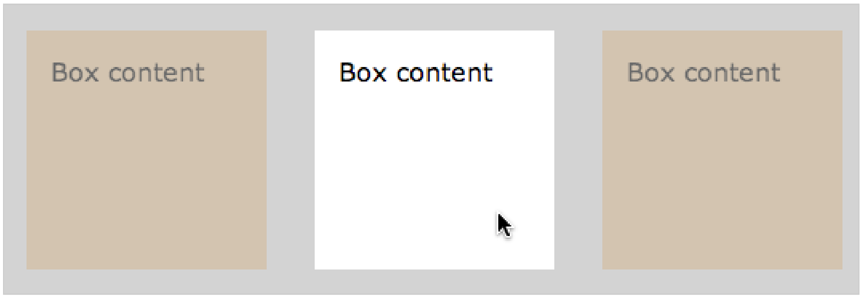


Figure 3 Changing the background color and opacity of a hovered box

If you have 100% opacity, or opacity with a value of 1, you get an opaque box. At the other end of the scale,0% opacity (a value of 0) defines a transparent box. What about an opacity value of 50% or .5? That defines a box that's halfway transparent (or halfway opaque—it's the same thing). In a normal state (when not hovered over), the box displays with 50% opacity (or a value of .5). That property and value "dim" the boxes by making them semi-transparent. But when a user hovers his or her cursor over the box, the opacity changes to 100%. That makes the box "pop" when someone hovers over it.

Defining and Applying a Color Scheme

Website color schemes are composed of five colors. Why do Web designers, who rarely agree on anything, agree on this? It's because you need five colors to distinguish different elements of your pages: banners, sidebars, text color, and so on. Generally, anything less than five colors makes your site look dull.

Why not use eight or nine colors, then? Constraining yourself to five colors gives your site a coherent theme. If there are more than five colors in your site design, visitors are overwhelmed with a cacophony of color. The site feels noisy and cluttered. Also keep in mind, In Web design, black, white, and shades of gray or dark blue generally don't count. You can use them on top of and in addition to the colors in your color scheme.

We won't go into the detail of color schemes, it's a complete course but just look into the basics. Color schemes can be either complementary or harmonious. Complementary color schemes involve mixing "opposite" colors from the standard color wheel. For example, you might build a color scheme around blue and orange, or you might prefer yellow and purple.

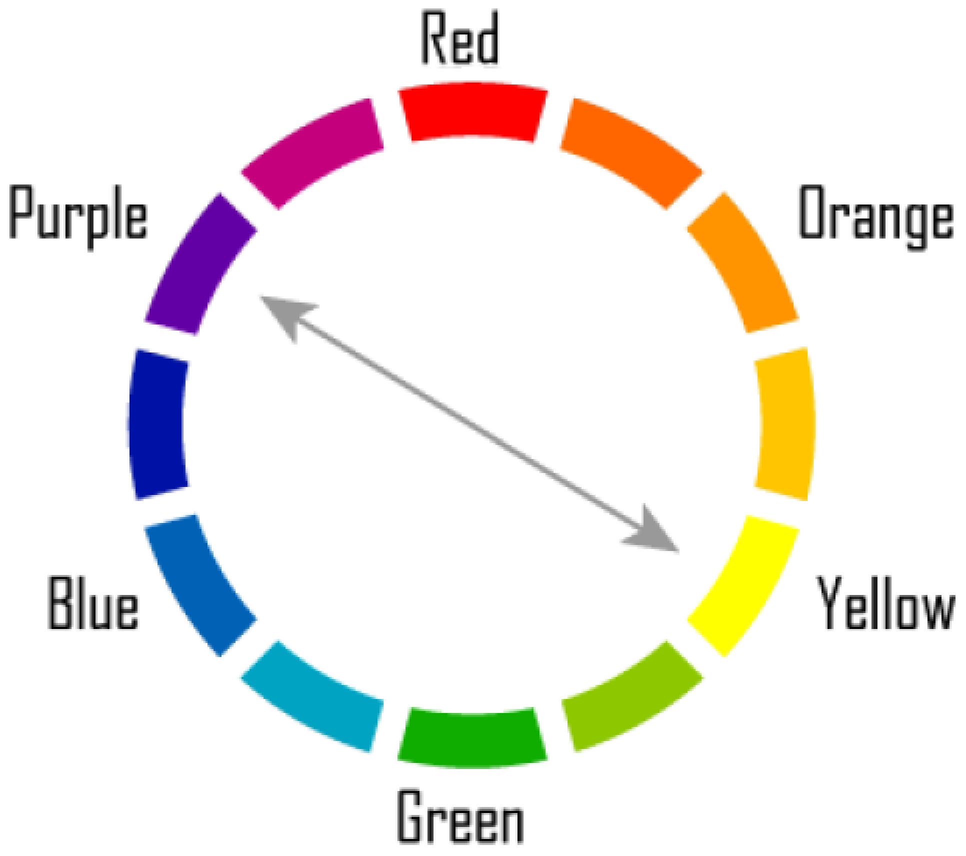
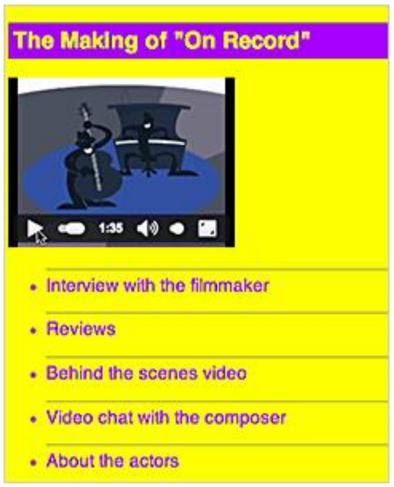


Figure 4 Complementary colors on the color scheme

Here's a Web page that effectively uses complementary colors:



The Making of "On Record"

- Interview with the filmmaker
- Reviews
- Behind the scenes video
- Video chat with the composer
- About the actors

Another approach is to build harmonious color schemes around gradations of a color. For example, yellow, green, and blue are next to each other on the color wheel. You might build an effective color scheme based on gradations of those colors (like lime green and aqua).

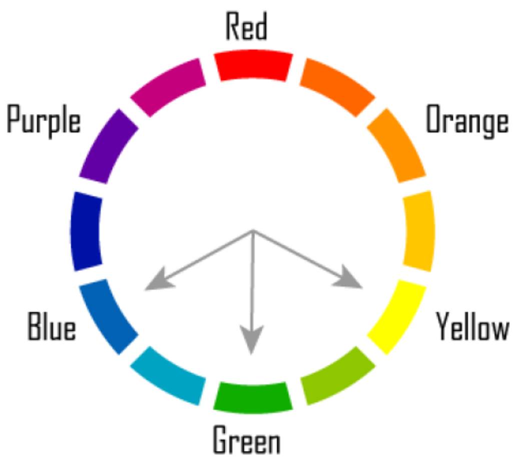
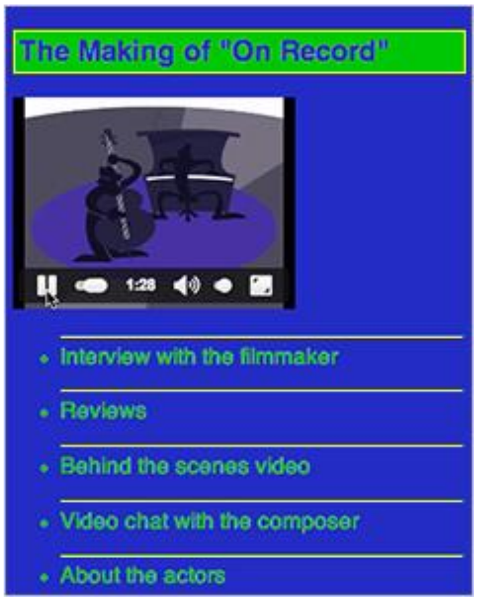


Figure 5 Harmonious color scheme

And here's an example of how a harmonious color scheme can look:


The Making of "On Record"
1:28
Interview with the filmmaker
Reviews
Behind the scenes video
Video chat with the composer
About the actors

You can find many tools for generating color schemes online. The most powerful is Adobe's Color site.

To create a color scheme in Color, follow these steps:

1. In a browser, go to color (https://color.adobe.com/create). If you haven't used this site before, you'll need to create an account. If that's the case, click **Sign Up**, and follow the instructions.
2. You can search Color's massive set of color schemes for one you like, or you can create your own. Let's start with the search option. Click **Explore**, and then go to the search box. In that box, type the name of a color (like orange). You can also search by concept (like "happy" or "scary").

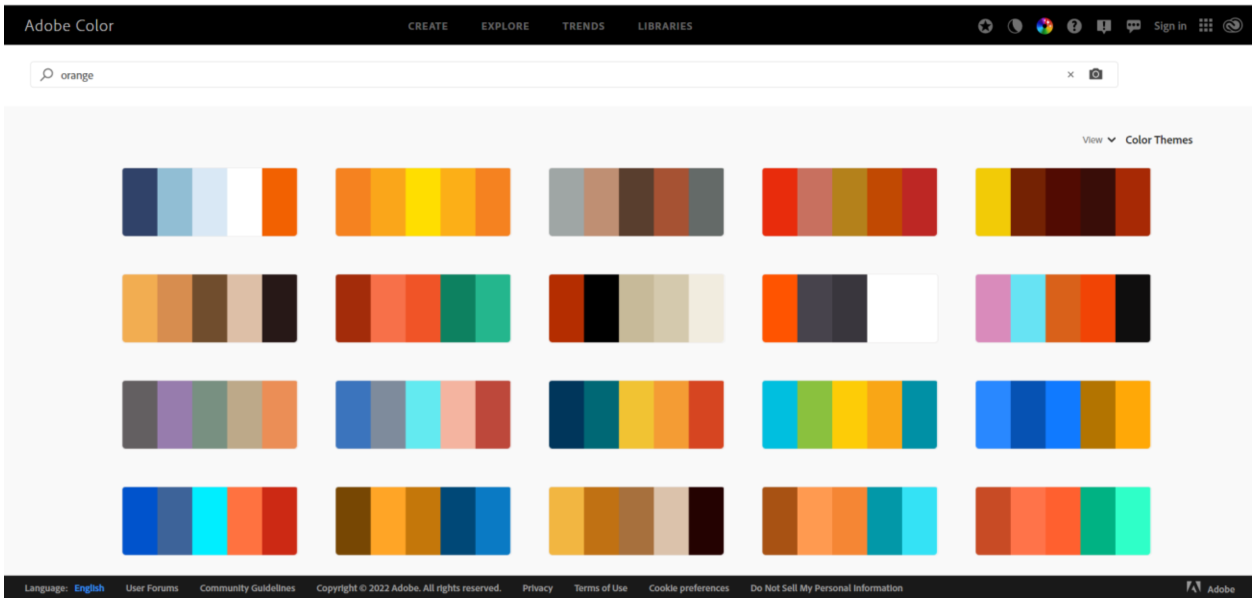
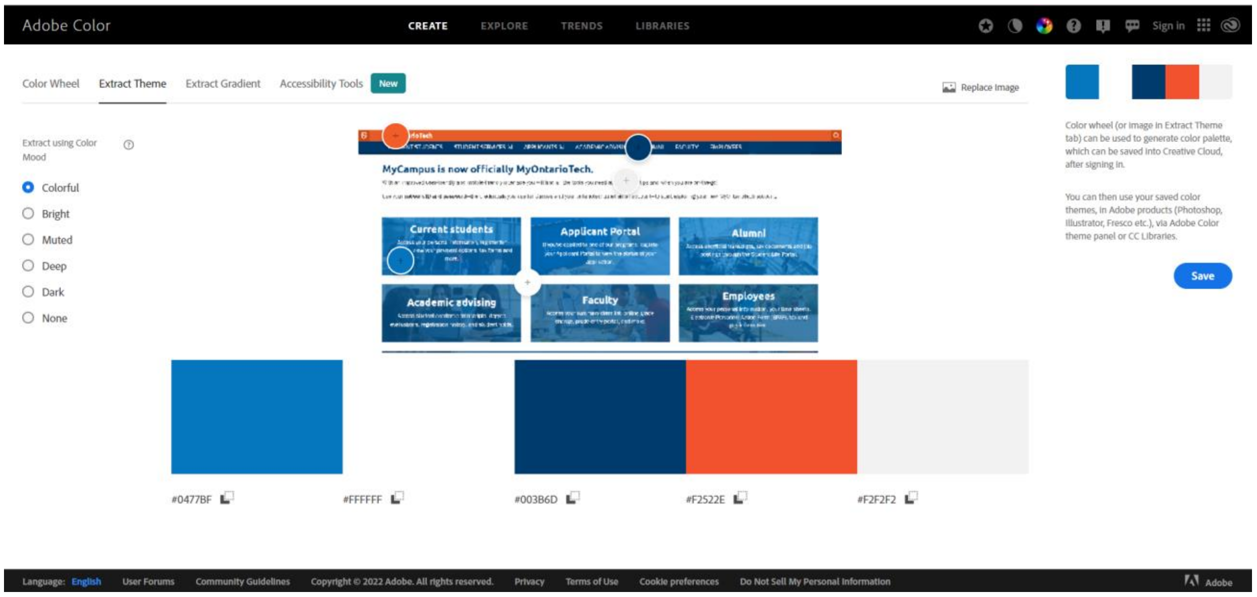


Figure 6 Adobe color showing orange color schemes

3. To define your own color scheme, click the Create link (it's under the search box).
4. In the new window that opens, Color lets you create from a color wheel or from an image. In Create and then Extract Theme option is valuable for designers; you can upload an image, and

Color will extract color scheme options from it. With the image uploaded, you can move any of the five small circles over different parts of the image to extract a color for your color scheme.



Extract using Color Mood

- [✓ Colorful]
- Bright
- Muted
- Deep
- Dark
- None

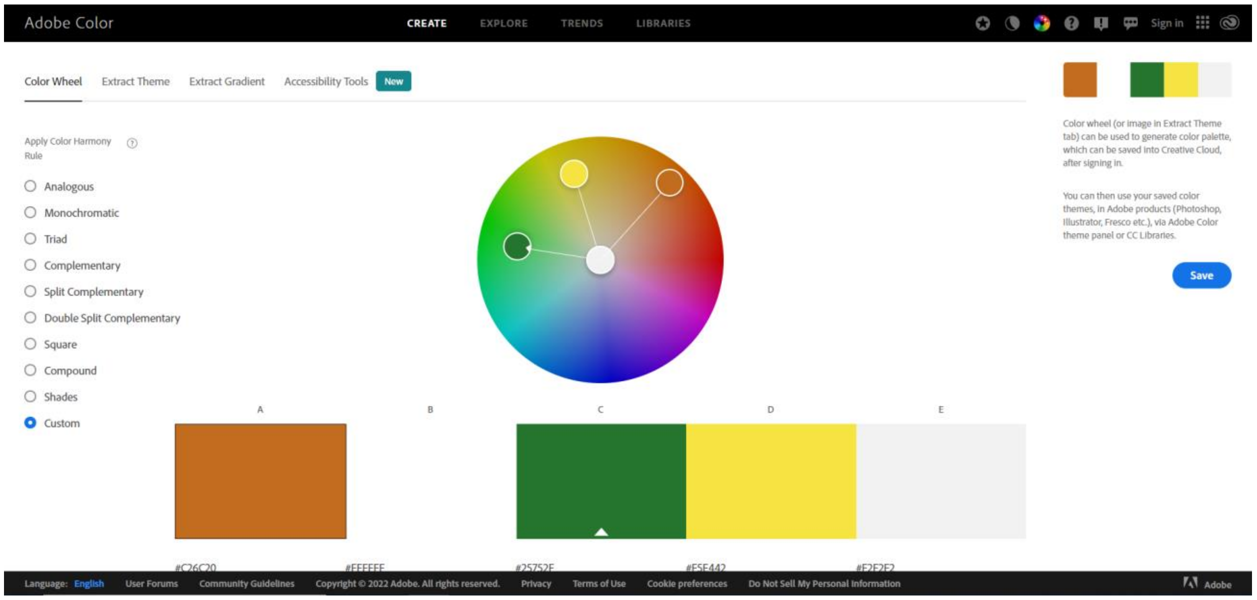
#0477BF #FFFFFF #003B6D #F2522E #F2F2F2

Color wheel (or image in Extract Theme tab) can be used to generate color palette, which can be saved into Creative Cloud, after signing in.

You can then use your saved color themes, in Adobe products (Photoshop, Illustrator, Fresco etc.), via Adobe Color theme panel or CC Libraries.

Figure 7 Color Scheme based on image

5. To define a color scheme from a color, click the different areas of the color wheel.



Apply Color Harmony Rule

- Analogous
- Monochromatic
- Triad
- Complementary
- Split Complementary
- Double Split Complementary
- Square
- Compound
- Shades
- [✓ Custom]

#C26C20 #FFFFFF #25752E #F5E442 #F2F2F2

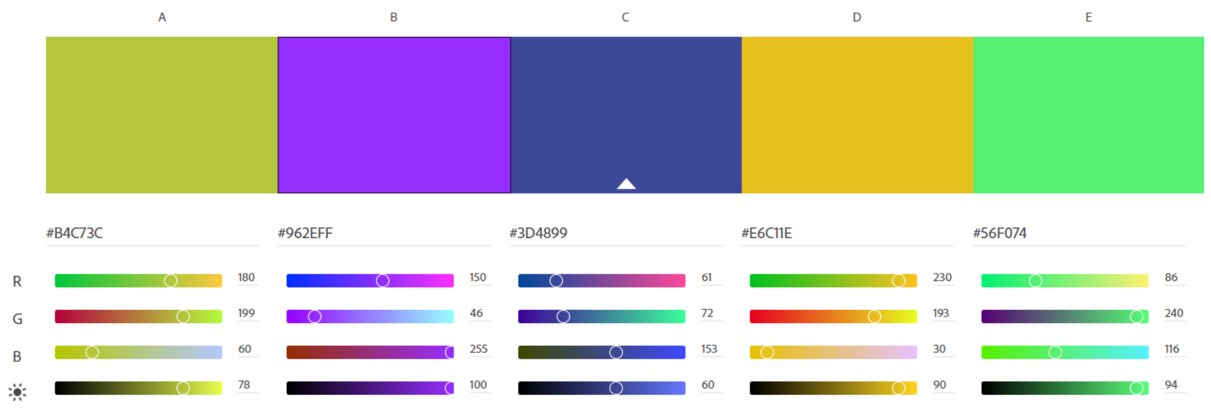
Color wheel (or image in Extract Theme tab) can be used to generate color palette, which can be saved into Creative Cloud, after signing in.

You can then use your saved color themes, in Adobe products (Photoshop, Illustrator, Fresco etc.), via Adobe Color theme panel or CC Libraries.

Figure 8 Defining colors using color scheme

6. Drag circles on the color wheel to adjust the color scheme. Play with this feature a bit, and you'll discover some intuitive features. For instance, dragging one color adjusts the rest of the colors.
7. Dragging colors toward the center of the color wheel creates muted (less bright) colors.
8. Finally, if you're comfortable defining colors with HSV, RGB, CMYK, LAB, or hexadecimal color codes, you can enter those values directly below any color swatch to apply a specific color to a swatch digitally.

9. Now it's the time save five hexadecimal values for our color scheme. Copy and paste the five hexadecimal values into a comment in our css file.



	#B4C73C	#962EFF	#3D4899	#E6C11E	#56F074
R	180	150	61	230	86
G	199	46	72	193	240
B	60	255	153	30	116
☀	78	100	60	90	94

Figure 9 Extracting Hexadecimal values from adobe color

10. Then paste those values into a comment in the CSS style sheet file in your code editor. Your comment code should look something like this (except, of course, that your values will be different):
/* Color scheme hexadecimal values: B4C73C 962EFF 3D4899 E6C11E 56F074 */

Applying a Color Scheme

Now that we've pasted our color scheme hexadecimal values into a comment in our style sheet, we can replace the rather generic shades of gray in the page we created with our own colors. Here's an example of a custom color applied to the body element called background color:

```css
body {
background-color: #B4C73C;
font-family: Verdana, Geneva, Arial, sans-serif;
padding:0px;
margin:0px;
}
```

Experiment with a variety of color values from your color scheme in different elements.

Now before hosting our website, we need to perform some testing's: We need to make sure our websites are ready for the world. That means, among other things:

- Validating code for errors
- Testing all the links
- Checking spelling
- Evaluating for accessibility —how easy it is for people with disabilities to access our site
- Testing for browser issues

Are all these little tasks important? Yes. Carefully testing your site before uploading helps you to make it inviting, impressive, and professional. And, to get negative for a moment, any of these things can severely undermine the impact of even a well-designed site that has appealing content:

- A misspelled word on your home page
- A bad link to your "About Us!" page
- An HTML error message that pops up in a browser
- Links that are inaccessible to folks who can't use a mouse, either because of their system environment (like a mobile device) or because of physical limitations

Of course, as you create page content, you check for mistakes. You test pages in a browser on your local computer before uploading them. But as websites grow to dozens, hundreds, or thousands of pages, you need to rely on more systematic testing techniques—and I'll show you how to do that.

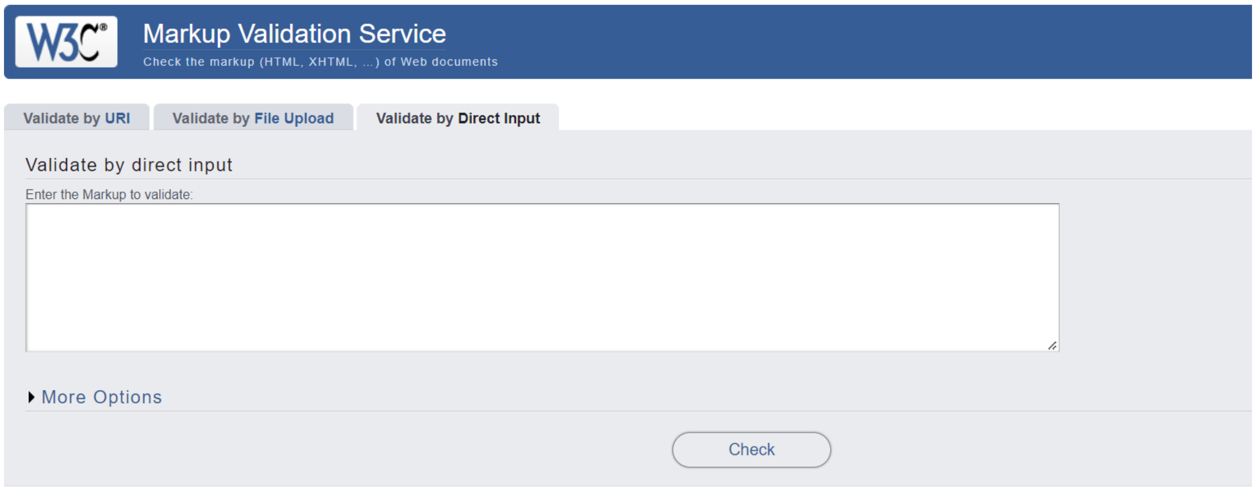
Testing and Troubleshooting Your Site

Our site's almost ready to upload to the Web—but first we should test and troubleshoot it. The most basic and important way to test a Web page is to save it in your code editor and open it with a browser (Run the code). Beyond that, valuable online tool scan validates (test) your code, your links, and the accessibility of your site.

Validating Your Code and Links

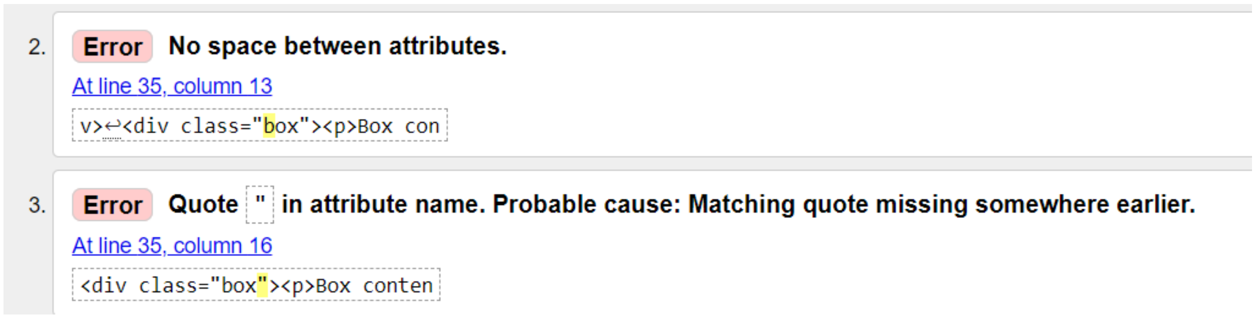
What if you open your Web page in a browser, and it doesn't look like it's supposed to? The problem could be an error in your HTML or CSS code. To troubleshoot, use an online validation system to identify errors. To test HTML code, follow these steps:

1. Open W3C Markup Validation Service (http://validator.w3.org/) in your browser. You'll see the markup validation service that the World Wide Web Consortium (W3C) provides. The W3C sets international standards for the Web.


W3C
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Validate by URI
Validate by File Upload
Validate by Direct Input
Validate by direct input
Enter the Markup to validate:
More Options
Check

2. Click the Validate by Direct Input tab at the top of the page. Doing this allows you to test code before you upload it. Since we include a doctype declaration in our file, you don't need to expand the More Options triangle that appears near the middle of the page.
3. Copy and paste your HTML into the Enter the Markup to validate box, and click the button that says check.

4. If there are no errors in your code, you'll see a message confirming that. Sometimes this message includes a "Notes and Potential Issues" list that identifies potential compatibility issues that aren't critical.
5. If there are errors in the code, it will show you the error along with hints on how to fix those errors.

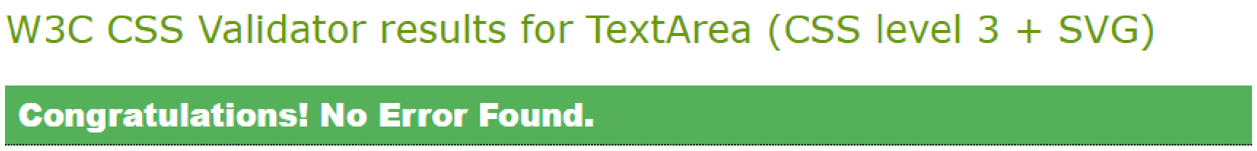


2. Error No space between attributes.
At line 35, column 13
```
v>↩<div class="box"><p>Box con
```
3. Error Quote " in attribute name. Probable cause: Matching quote missing somewhere earlier.
At line 35, column 16
```
<div class="box"><p>Box conten
```

Testing CSS Code

You can test CSS code the same way we just tested HTML code, but you'll use a different Web page at W3C. To test CSS, go to

1. W3C CSS Validation Service (http://jigsaw.w3.org/css-validator/).Just as we did with HTML, click the By Direct Input tab, and paste your CSS code into the box.
2. Then click the Check button. The results will be either a Congratulations message or a list of errors with advice on how to fix them.



W3C CSS Validator results for TextArea (CSS level 3 + SVG)

Congratulations! No Error Found.

Checking Links

W3C also provides a link-testing page, but it works only after you upload your site. So, bookmark this page now, and use it later:

1. W3C Link Checker (http://validator.w3.org/checklink). Link validation takes time because the validator searches your entire site and tries every link. When testing is complete, you can click the Go to the results link to see a list of errors. The link validator provides detailed explanations of what the problems are and how you can fix them.

Checking Your Spelling

Misspelled words doom a website. They send a message that your message isn't worth taking seriously and since I can't spell worth a darn, that's a big problem for me. Even if you're an excellent speller, it's a wise move to check your spelling before posting a site.

Many code editors include spell-checking for regular text. Text Wrangler (for Mac) includes spell-checking, and plug-ins are available that add spell-checking to Notepad++ for Windows. Squiggly red underlining appears under suspected spelling errors, and you can view corrections by right-clicking (or CTRL + clicking on a Mac) on the underlined words.

Will your website work for everyone who accesses it? Find out by using WAVE.

Testing for Accessibility

WAVE stands for Web Accessibility eValuation Tool. That site tests pages for accessibility. For example, will your website work well for hearing-impaired or sight-impaired people? Will color-blind people be able to comprehend the content? Will people with motor disabilities be able to navigate the pages? To test your page in WAVE, go to WAVE Web Accessibility Evaluation Tool (http://wave.webaim.org/). The site works for uploaded pages only, so bookmark this link as well for future reference.

When you enter a URL in the WAVE site, you see an annotated version of the URL. WAVE identifies accessibility problems, and it highlights features that make a page more accessible. One of the most important accessibility features is proper use of ALT properties with images. Alt property values are short text strings that describe the image. Spaces, uppercase and lowercase characters, and special characters (like "&" or "!") are fine. Alt properties describe content to users who have limited vision.

Next Step set up remote hosting server. Upload your web page. Free hosting links are shared on CANVAS and rest I will show you that in class and TA will help you in Lab!!

Improving Your Site Design with Gradient Backgrounds and Custom Fonts

Introduction:

Radical changes are taking place in the aesthetics and technology of Web design. New techniques have opened up due to increasing support for different design effects created with CSS3. Here we discuss about two of those techniques—gradient backgrounds and custom fonts—and I'll explain how they can make your sites more attractive and engaging.

Using Gradient Backgrounds

When I say gradient backgrounds, I'm talking about backgrounds to pages, layout boxes, and other elements. In the figure below, I've applied a gradient to the body element background, the banner element background, and the right column element of our page layout.

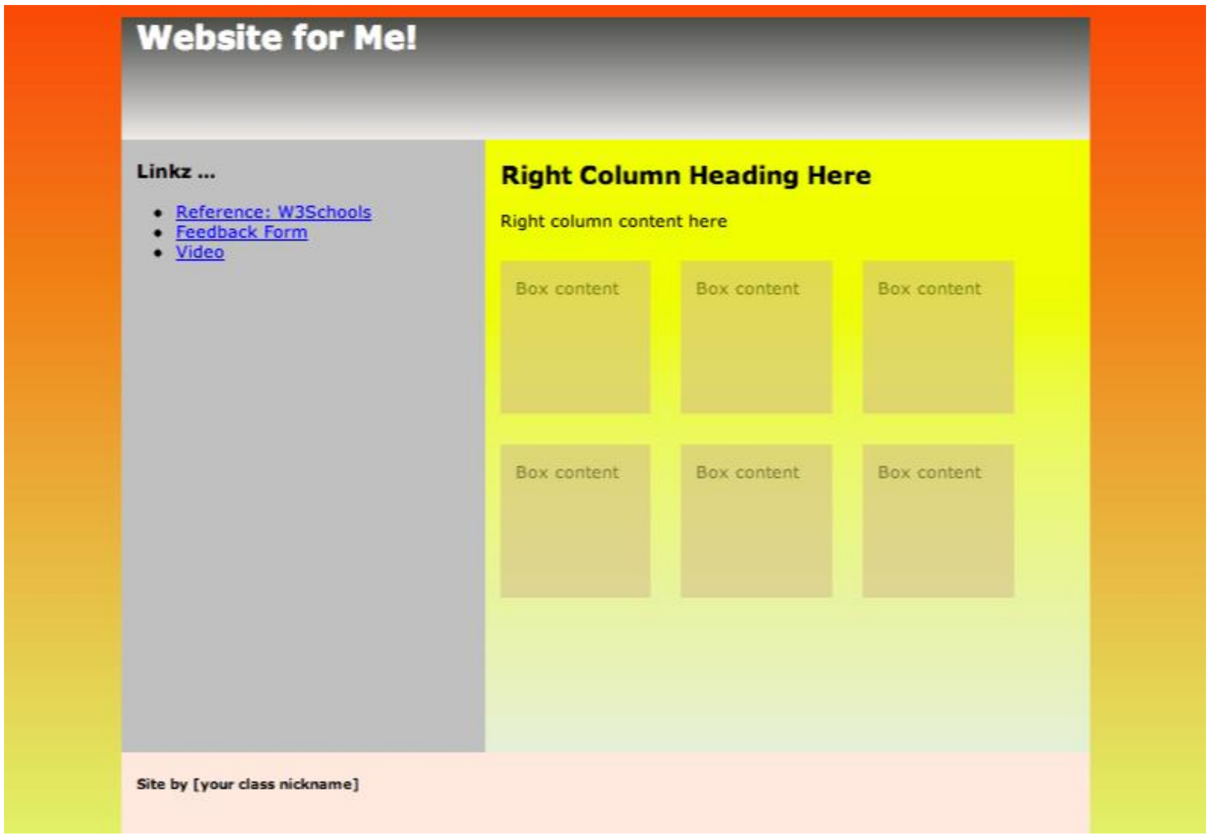


Figure 10 Gradient backgrounds applied to the body, banner, and right column

Traditionally, designers created backgrounds like this in Adobe Illustrator, Adobe Photoshop, or another drawing application and then saved them as PNG or JPEG images. The designer then applied CSS coding to "tile," or repeat, the image from top to bottom or from right to left.

CSS3 technique more powerful because you can generate complex gradients that do more than blend from top to bottom, or right to left. Look closely at the gradient I applied to the banner in the image below. It's darker in the upper left, and it blends to lighter in the lower-right corner of the banner.

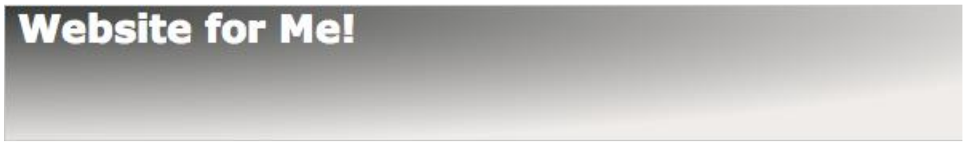


Figure 11 A more complex gradient background

CSS3 gradients are more efficient than old-style tiling images because there are no images involved! There is no image to download.

Discovering Custom Fonts

Fast-downloading effects is the use of custom Web-based fonts (often just called Web fonts). Most of the time, Web designers still rely on a user's computer environment to supply font support. So, if you apply the Forte font to a style definition, only visitors who have Forte installed on their computers would see that font. Most users would see a substitute font instead.

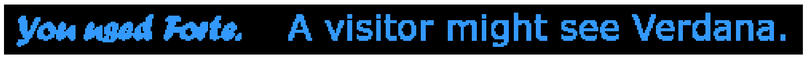


Figure 12 Example of substitute font

If you really needed to present text in Forte, you could create a graphic file and save it in JPEG or PNG (or GIF) format. This approach has many downsides, though. It isn't possible to copy and paste this graphical text. The user can't search for it. And using a graphic image to present text adds download time to a page.

The advanced Web design solution is to use custom Web-based fonts that allow us to use a wide range of font faces in our site design. These fonts work even for visitors who don't have the font installed on their computers. Users have to download Web fonts. But the sources that supply these fonts have blazing-fast servers, so users aren't burdened with noticeable wait time, even with a 3G connection on a tablet or smartphone.

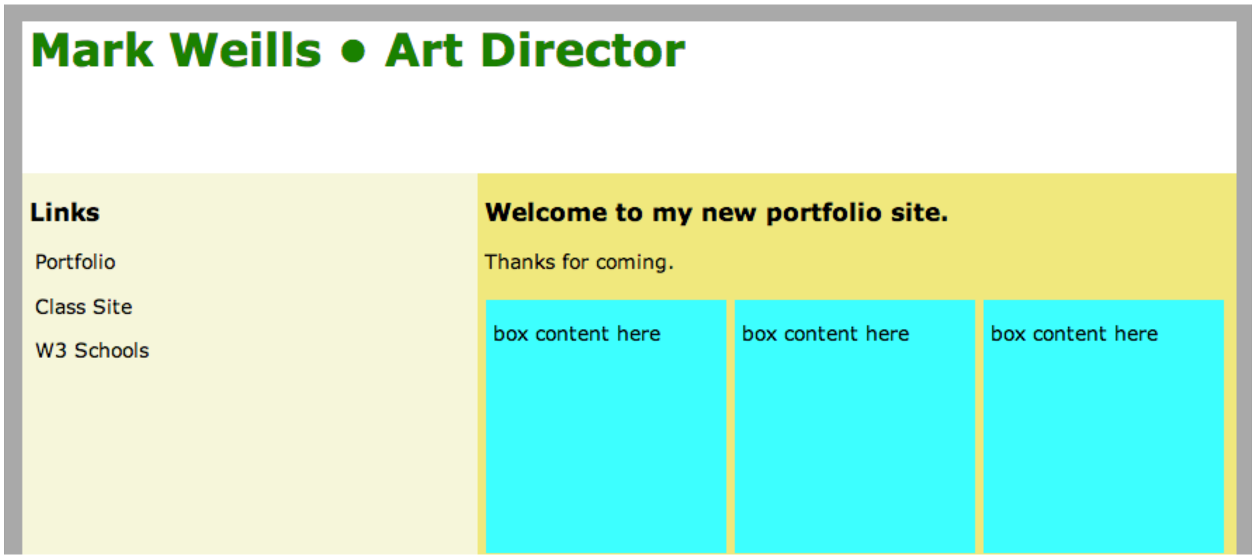


Figure 13 Example of web-font

Applying CSS3 gradients and Web fonts isn't hard, and it gives you creative freedom with no extra download times for your users.

Creating Gradients for Different Browsers

With CSS3 gradients, what appears to the visitor as an image is something you created entirely with CSS3. But what about browsers that don't support CSS3? We'll apply the principle of "graceful degradation" for these older browsers. Visitors to our site whose browsers don't support CSS3 will simply see a solid-color background from our color scheme. That's not such a bad option. Their pages will still download quickly. The pages won't be innovative, but they won't be ugly either.

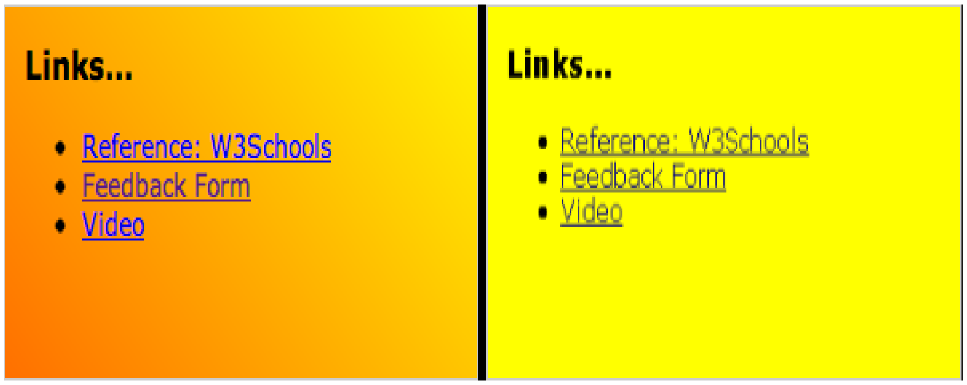


Figure 14 A CSS3 gradient background on the left defaults to a solid color

Here's the basic syntax for a gradient property:

background: linear-gradient (<to direction>, <color stop>, <color stop>);

Let's analyze this piece by piece.

- The property is background.
- The value is linear-gradient followed by a definition within parentheses.
- The to direction part of the value defines the direction of the color blend.
- The <color stop> sections define a color and control where that color becomes fully visible. (Why call this point a "stop"? That terminology comes from programs like Adobe Illustrator. But it's more helpful to think of that value as the point where a color can start or end.)

Let's break down a simple example. In the CSS code below, I've set the direction as "to bottom," meaning the gradient begins at the top of the element and ends at the bottom. I chose red as the first color, and it starts at 0%—the top of the element. The ending color for the gradient blend is yellow, and the background becomes fully yellow at 100% of the height of the element. In other words, the gradient ends at the bottom of the element.

background: linear-gradient (to bottom, red 0%, yellow 100%);

That's the generic syntax for defining a CSS3 gradient.

This generic syntax isn't complicated. Unfortunately, not all browsers don't yet have a single way to define CSS3 gradients . . . and at this writing, not all of them support the generic syntax. So, we need different definitions for each browser.

To support all modern browsers, we need to create four versions of our CSS3 gradient syntax. They look like this:

```
body {
background: yellow;
background: -moz-linear-gradient(top, red 0%, yellow 100%);
background: -webkit-linear-gradient(top, red 0%, yellow 100%);
background: -o-linear-gradient(top, red 0%, yellow 100%);
background: linear-gradient(to bottom, red 0%,yellow 100%);
font-family: Verdana, Geneva, Arial, sans-serif;
padding:0px;
margin:0px;
background-color: black;
font-family: Verdana, Geneva, Arial, sans-serif;
padding:0px;
margin:0px;
}
```

The -webkit prefix applies to Chrome and Safari. (Webkit is the underlying program that powers both those browsers.) The -moz prefix applies to Firefox -moz is short for Mozilla, the group that produces Firefox. The -o prefix applies to Opera.

It's good practice to include that generic option as well, since someday all browsers may agree to use it.

What about creating a horizontal gradient? To transform a gradient from top-to-bottom to left-to-right, replace "top"' with "left" in the prefixed declarations, and replace "to bottom" with "to right" in the generic.

Before we wrap up our exploration of linear gradients, let's explore how to create an angled gradient. To angle a gradient background, replace "left" or "top" with a defined angle value, and follow that (without a space) by "deg" because you're indicating the number of degrees. Angle values can be positive (like 45deg) or negative (like -45deg).

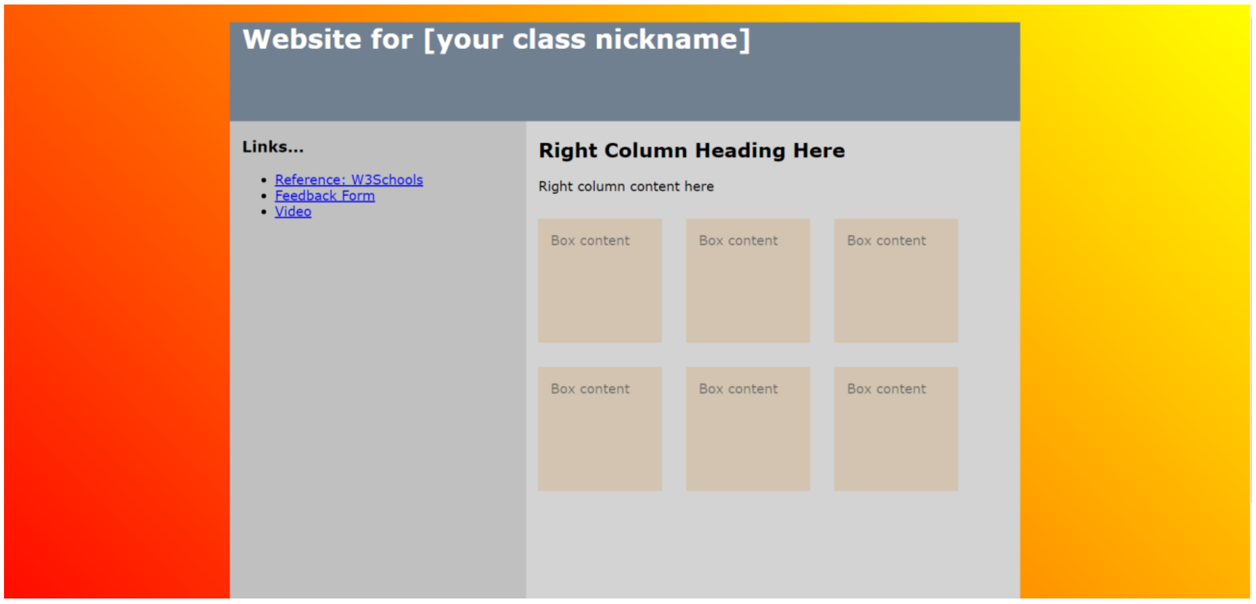


Figure 15 Gradient with 45 degree

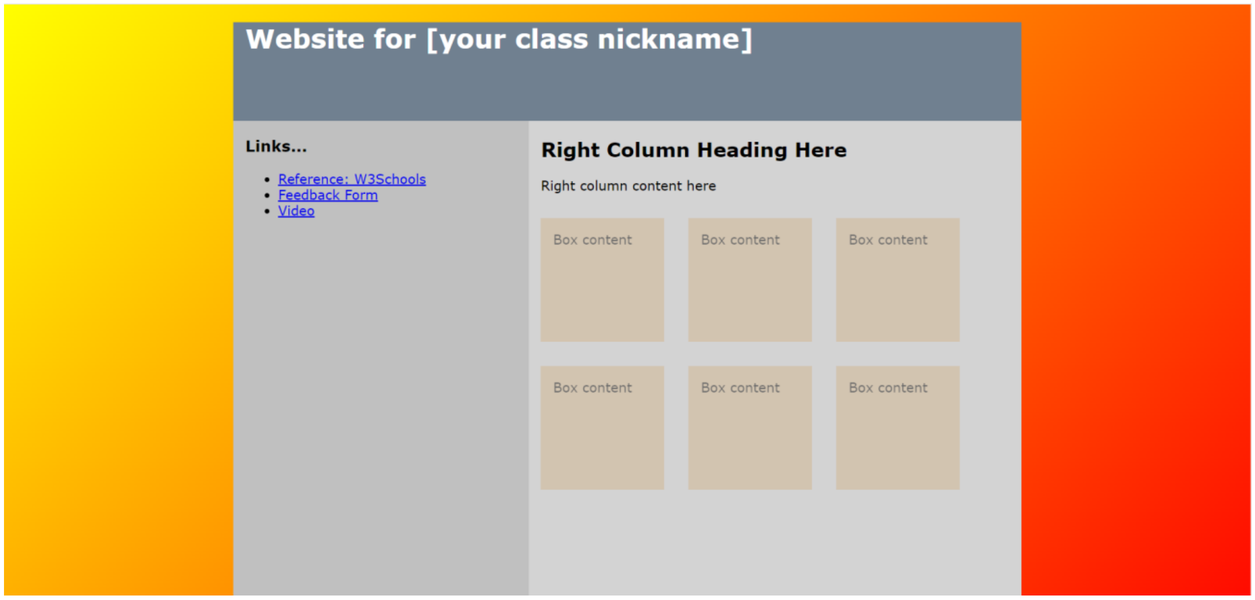


Figure 16 Gradient with -45 Degree

Radial Gradients for Dramatic Effect

Radial gradients blend from the inside of a circle or ellipse to the outside. In the example below, I drew on a technique to add a border radius to the boxes in our template, and then I applied a radial gradient.

```css
.box
{
height: 100px;
width: 100px;float: left;
margin: 15px;
padding: 25px;
opacity:.5;
border-radius: 75px;
}
```

The border-radius attribute is a CSS3 style property that creates rounded corners. The length of the radius defines how rounded the corner is. In the example above, the total width and height of our box remains at 150 pixels (100 pixels plus 25 pixels padding on all sides). By defining a border radius that's half the total width of the box, we convert the box into a circle. So, the boxes (now circles) each have a radial gradient, and the background still has the angled gradient (45 deg).

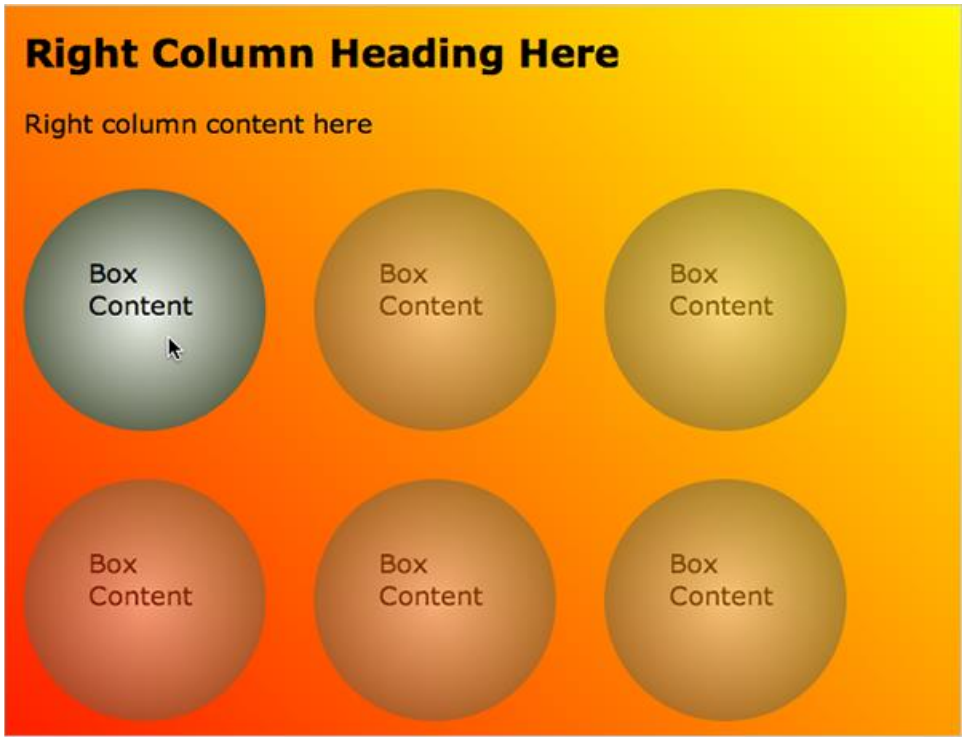


Figure 17 Box Content as radial gradient

Use these gradients for emphasis and impact. If you overuse them, they become distracting. But use them judiciously, and they can add a lot of energy to a Web page. Radial Gradient starts with one color in the middle (white in our example) and radiates out to the end color (dark gray in our example) at the outer edge.

It's possible to hand-code the CSS for radial gradients, but it's even easier to use one of the online tools for generating CSS3. Online CSS3 gradient generators let you define gradients in a WYSIWYG-type interactive environment. (WYSIWYG means What You See Is What You Get. In other words, you can see the results immediately instead of guessing or repeatedly testing your changes.) For instance, at the Ultimate CSS Gradient Generator (https://www.colorzilla.com/gradient-editor/), you can use gradient stops like this one:

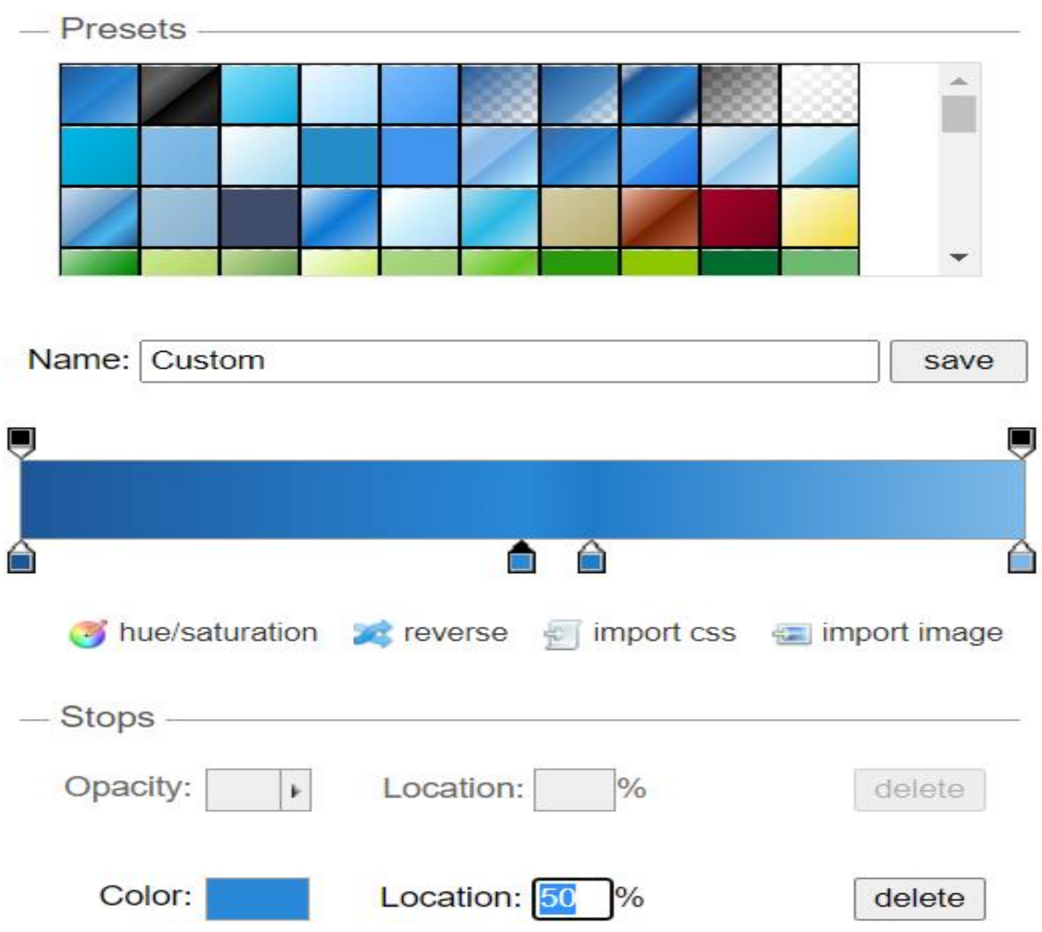
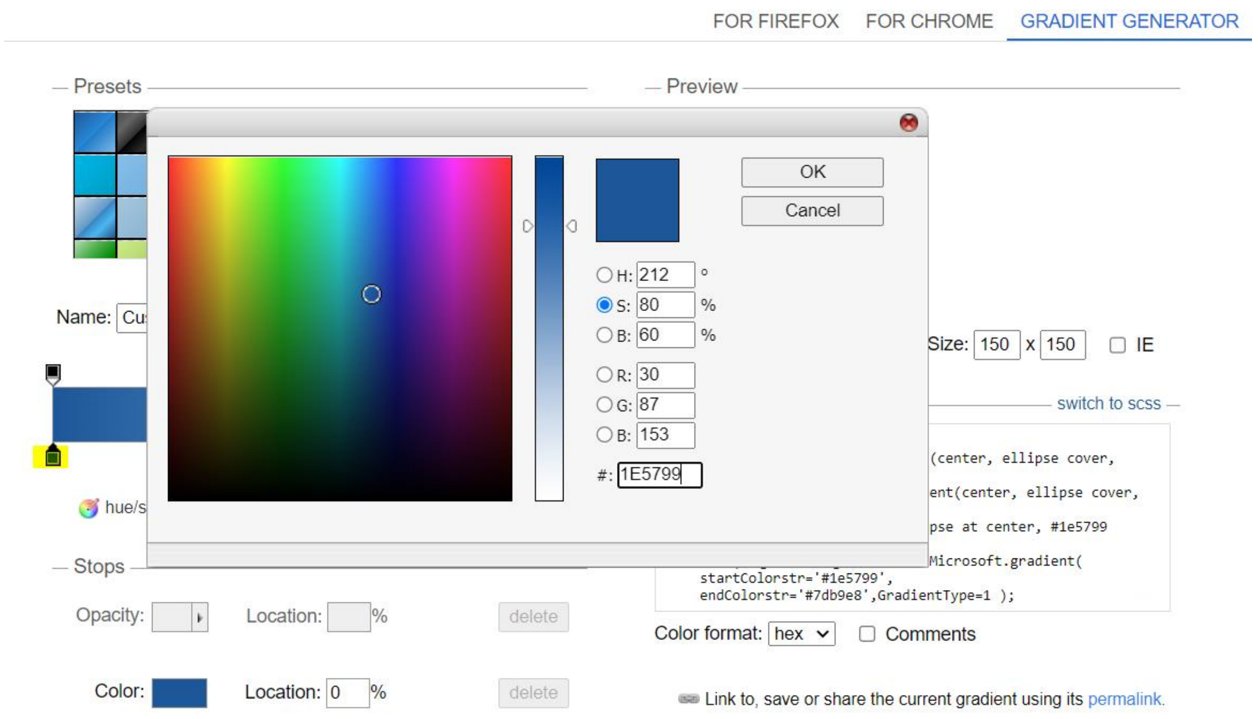


Figure 18 CSS Gradient Generator

In the above figure you can see that we have 4 stops. Let's take out the 2 middle one. Drag the two middle stops off the bar by clicking on each one and pulling it to the bottom of your screen.

Double-click the first (left) stop. A color box opens. You can choose a color or enter a hexadecimal value from the color scheme we created. After you choose a color, click OK.

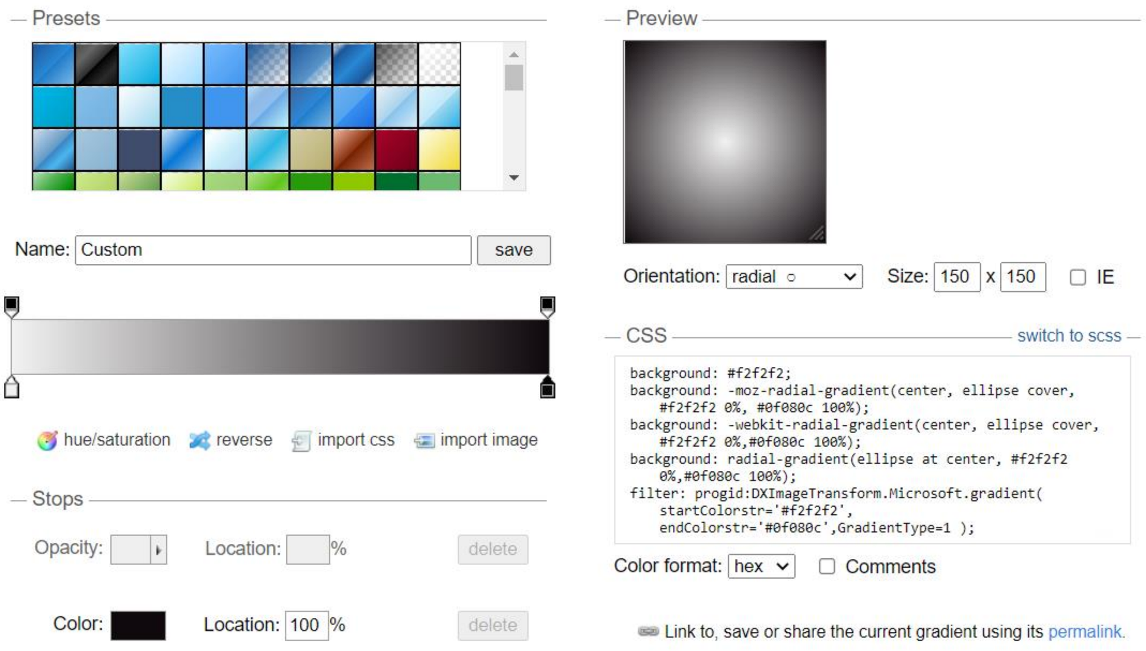


```
(center, ellipse cover,
ent(center, ellipse cover,
pse at center, #1e5799
Microsoft.gradient(
    startColorstr='#1e5799',
    endColorstr='#7db9e8',GradientType=1 );
```

Link to, save or share the current gradient using its permalink.

Figure 19 Showing color box

In a similar way, redefine the color for the second stop. In the Preview area on the right-hand side of the screen, define the preview area as 150 by 150 to emulate the size of the box we're designing a background for. Set the orientation to radial.



```
background: #f2f2f2;
background: -moz-radial-gradient(center, ellipse cover,
    #f2f2f2 0%, #0f080c 100%);
background: -webkit-radial-gradient(center, ellipse cover,
    #f2f2f2 0%,#0f080c 100%);
background: radial-gradient(ellipse at center, #f2f2f2
    0%,#0f080c 100%);
filter: progid:DXImageTransform.Microsoft.gradient(
    startColorstr='#f2f2f2',
    endColorstr='#0f080c',GradientType=1 );
```

Link to, save or share the current gradient using its permalink.

Figure 20 Change the Orientation to Radial and set the size to 150 by 150

Find the box of generated CSS code on the right-hand side of the page. Copy the code and paste it in .box class style of our style sheet. Now when you test you should be able to see the radial gradient on the box.

Selecting the Web Fonts

Typographers and amateurs have created thousands of fonts. But until recently, Web designers pretty much stuck to fonts that most people have on their computers. The only fonts users could see on a Web page were ones that the user's computer supplied. So, designers who included custom fonts within Web pages were—for the most part—wasting their time.

There are other reasons Web design is more restrained in the use of fonts than print design is. Even cheap home printers, print at 300 dots per inch. But computer monitors have much lower resolutions—typically less than 100 dots per inch. So complex fonts break down in digital display.

Using a variety of fonts in a print project often creates a messy-looking design. That rule applies to multiple fonts on a Web page too. It's rarely appropriate to use more than two fonts within a Web page. Having said all that, fonts are fun. They add variety and distinctiveness to a site.

Let’s look at the example, www.keithrichards.com, a site where the custom font is integral to the experience.

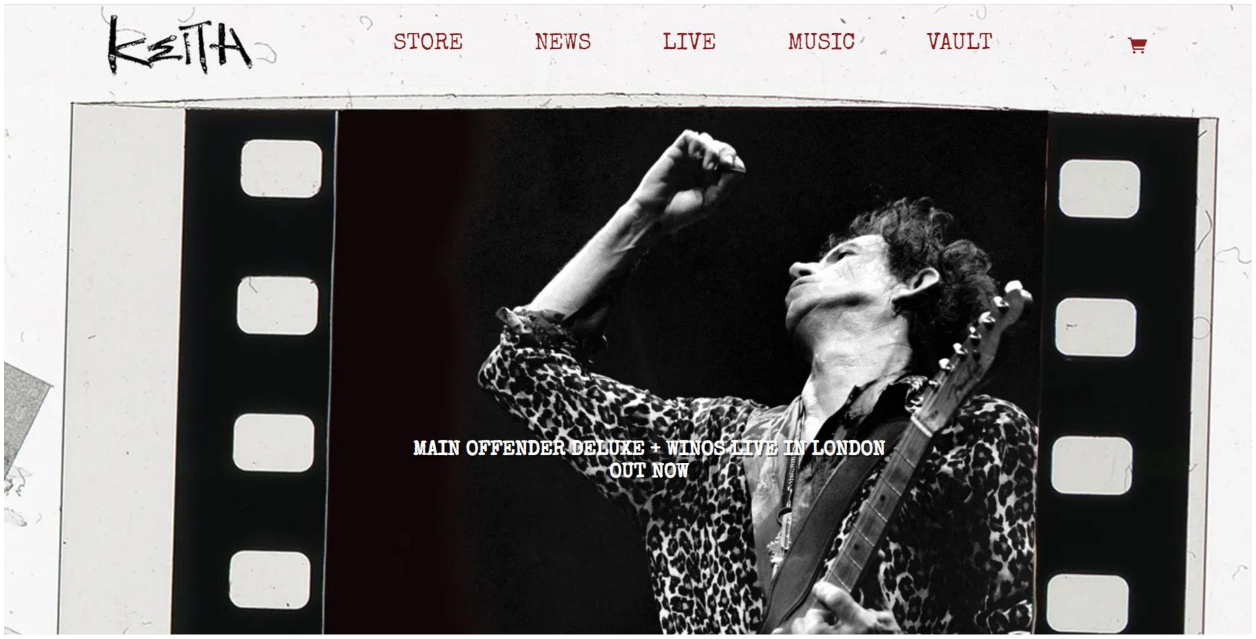


Figure 21 Making effective use of custom fonts

One more thing has kept many designers away from custom fonts: Until recently, commercial enterprises such as Fonts.com and Adobe licensed most catalogs of Web fonts. But Google has made hundreds of fonts available for free, which opens the door to many more people using custom fonts in Web projects. By the way, you may have seen or heard about the @font property in style sheets. Web fonts actually don't require that. They get assigned just like any other font in a style sheet, as values for the font-family property. And Web fonts don't come with color attributes. You assign color to Web fonts just as you would any other font.

Linking to a Font at Google Fonts

There are two steps to using free Web fonts from Google Fonts: defining the font, and applying it in your page. Let's start with defining the font. Since Google Fonts is a cloud-based resource, the interface is constantly evolving, but in your browser, go to Google Fonts (http://www.google.com/webfonts), and follow these basic steps:

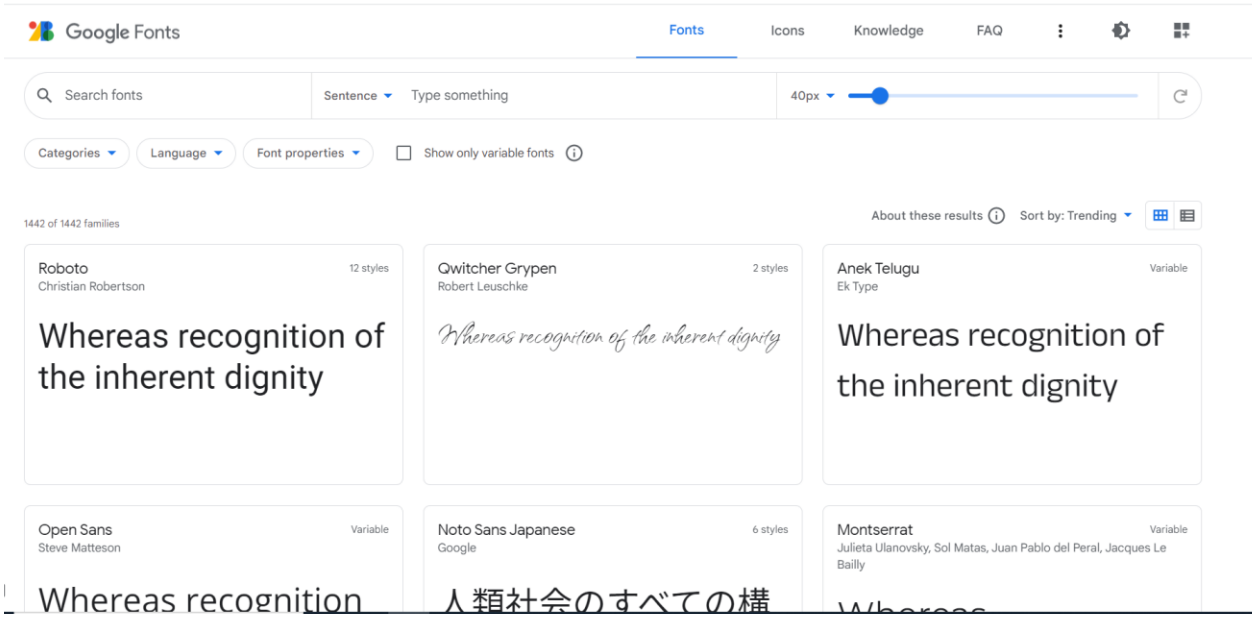


Figure 22 Google font page

1. Notice in the figure below that the site shows the Sentence tab by default. (The sample sentence is about grumpy wizards and an evil queen, which seems odd, but it uses all the letters in the alphabet.) Using this tab is the best way to preview fonts, but there are also options for word, paragraph, or poster. You can also change the preview text and the text size.

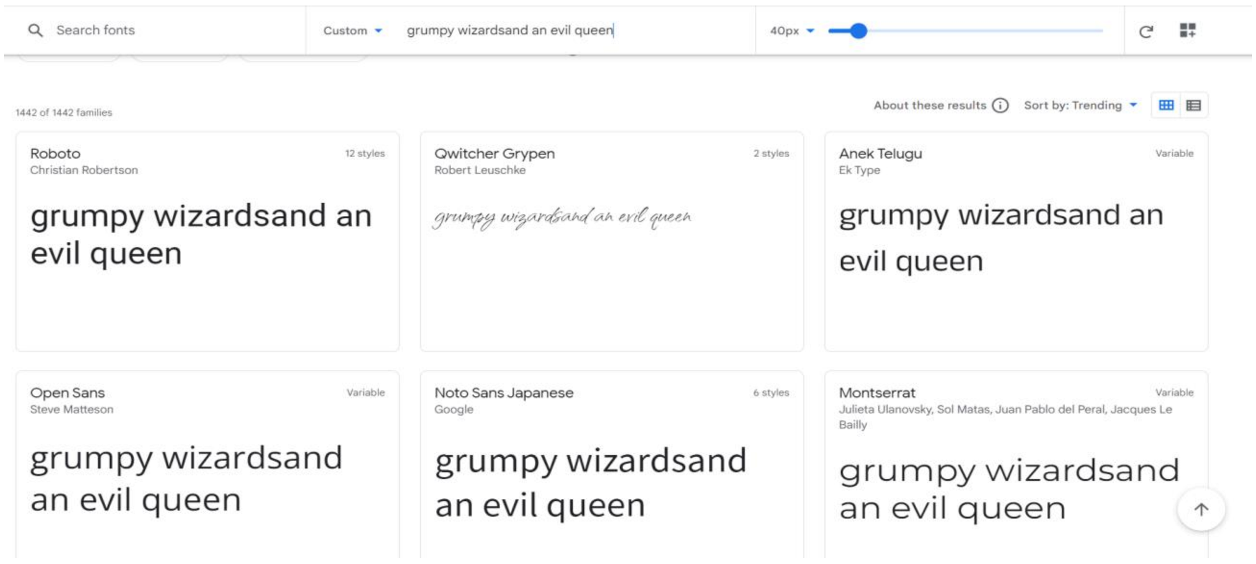
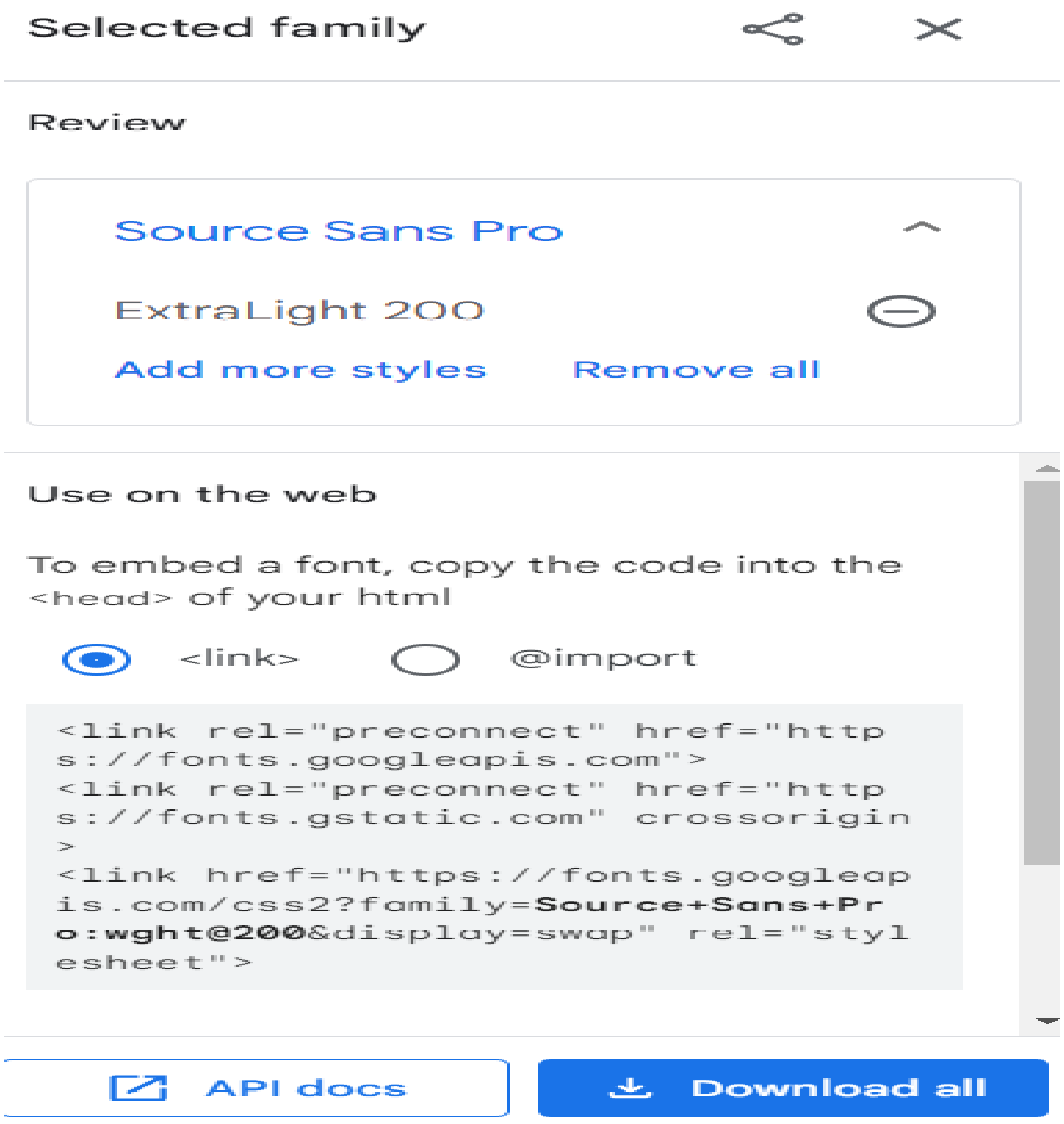

Search fonts
Custom
grumpy wizardsand an evil queen
40px
1442 of 1442 families
About these results
Sort by: Trending
Roboto
Christian Robertson
12 styles
grumpy wizardsand an evil queen
Qwitcher Grypen
Robert Leuschke
2 styles
grumpy wizardsand an evil queen
Anek Telugu
Ek Type
Variable
grumpy wizardsand an evil queen
Open Sans
Steve Matteson
Variable
grumpy wizardsand an evil queen
Noto Sans Japanese
Google
6 styles
grumpy wizardsand an evil queen
Montserrat
Julieta Ulanovsky, Sol Matas, Juan Pablo del Peral, Jacques Le Bailly
Variable
grumpy wizardsand an evil queen

Figure 23 Fonts preview with the custom sentence

2. Scroll down the list of fonts until you find one you like. You've got hundreds to choose from, and many are available in different styles and weights.
3. Click the Add to Collection / + button associated with that font and add the code in the <head> section of HTML.



To embed a font, copy the code into the <head> of your html

[✓ <link>] @import

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Source+Sans+Pro:wght@200&display=swap" rel="stylesheet">
```

Figure 24 Adding custom font in CSS file

4. Scroll it down and add the CSS code in the element where you want to apply the font style.

See you next Week!!