

GYMNASIUM

---

# CODING FOR DESIGNERS

---

*Lesson 2 Handout*

*"Hello World", Coding Pages, And Fixing Problems*

# ABOUT THIS HANDOUT

This handout includes the following:

- A list of the core concepts covered in this lesson.
- The assignment(s) for this lesson.
- A list of readings and resources for this lesson including books, articles and websites mentioned in the videos by the instructor, plus bonus readings and resources hand-picked by the instructor.
- A transcript of the lecture videos for this lesson

## ASSIGNMENTS

I've put up some web pages, but they're not looking right. I think they have syntax problems. Can you fix them for me, please?

Open the JS Bin links below. The errors in these pages are all syntax errors, and most are due to misplaced symbols and characters. When you start editing the HTML of these pages, the system will automatically create a new copy just for you. (The address bar will change to reflect your new version.) When you're done, copy-and-paste the web address of your edit and post it in the forum.

By the way, the original (broken) version stays intact, so if you need to start over from scratch, just click the links below again.

- <http://jsbin.com/osuhab/1/edit>
- <http://jsbin.com/iyuyac/1/edit>
- <http://jsbin.com/owukuc/1/edit>
- <http://jsbin.com/adarum/1/edit>

Note: These pages have HTML “comments” in their code. They look like this:

```
<!-- This is a comment. -->
```

The `<!--` and `-->` tags are HTML tags—just strange-looking ones—and they hide their contents from being displayed on the web page. I'm using them to leave notes to you inside the code of the page.

## EXTRA CREDIT

I've placed the bare-bones text for a cocktail recipe onto a web page, and I need you to format it using HTML tags.

Here's the recipe (and if you drink alcohol, it's a tasty cocktail!): <http://jsbin.com/eposiv/1/edit>

Because this is an extra credit assignment, there are some new tags for you to play with:

`<h1>`: This isn't new.

`<h2>`: This is new! For subheads. `<h1>` is for the most important headline(s) and `<h2>` is for the second-most important. It's okay to have more than one `<h2>` element on a page.

`<p>`: This isn't new.

`<ul>`: This is new! It stands for "unordered list," which is essentially a bulleted list. It's a double-barreled tag, which means you also need this tag:

`<li>`: "List item." It's used for one of the individual bullets inside the `<ul>`.

## HERE'S HOW IT WORKS:

`<ul>` `<!-- This is the start of an unordered, or bulleted, list -->`

`<li>Milk</li>` `<!-- Each item in the list needs its own set of <li> tags -->`

`<li>Eggs</li>`

`<li>Butter</li>`

`</ul>` `<!-- This is the end of the list. -->`

It's a tricky one, but give it a try!

# RESOURCES

## HTML REFERENCES

- [HTML tags, by category](#)
- [HTML tags, alphabetically](#)
- [HTML tutorials](#)

## CSS REFERENCES

- [CSS properties, by category](#)
- [CSS properties, alphabetically](#)

# INTRODUCTION

*(Note: This is an edited transcript of the Coding for Designers lecture videos. Some students work better with written material than by watching videos alone, so we're offering this to you as an optional, helpful resource. Some elements of the instruction, like live coding, can't be recreated in a document like this one.)*

Hi, and welcome back to Coding for Designers, an online course developed by Aquent. Coding for Designers, or How to Be the Kind of Web Designer That Developers Love. This is lesson two. I'm glad you're here.

Today we're going to jump straight into code. And if this is your first time doing HTML and CSS, fantastic. I'm glad to have you.

## TODAY'S BIG IDEAS

**BIG IDEA NUMBER ONE:** The trinity of web design—the three tools that web designers who code their own pages need to know.

**BIG IDEA NUMBER TWO:** “Hello world,” or how to get yourself into the family of web programmers, the family that you're about to join.

**BIG IDEA NUMBER THREE:** When you're creating webpages, especially when you're a beginner, concentrate on the words and images first, then layer in the design. I'll explain how, and we're going to do it together.

**BIG IDEA NUMBER FOUR:** Computers are dumb. They're stupid. They're arbitrary and capricious. We have to cater to them and do our code in a way that's suitable to them, even if it doesn't make a ton of sense to us. We'll talk about what that means for you as a new coder, how to work around it, and how to make life easier for yourself.

## BIG IDEA #1

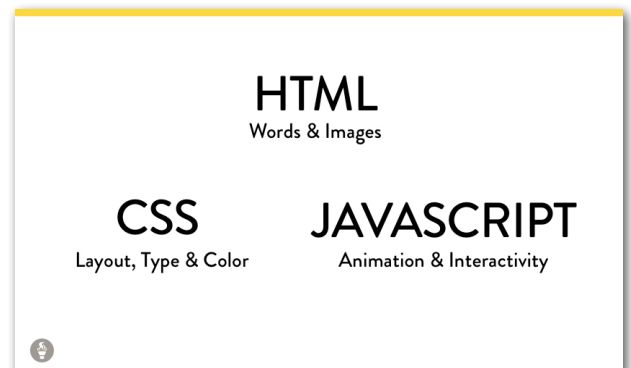
### THE TRINITY OF WEB DESIGN

Let's begin with Big Idea #1, The Trinity of Web Design: the three tools that designers and developers who work in web pages use to create those pages. You've probably heard of these tools. They are HTML, CSS, and JavaScript.

But what do these things actually do? What's the difference between them? Well, I'm glad you asked.

HTML is the basic bone structure of the page. It's responsible for the words and the pictures that actually appear on the page—the content.

CSS is responsible for something different. It's layout, typog-

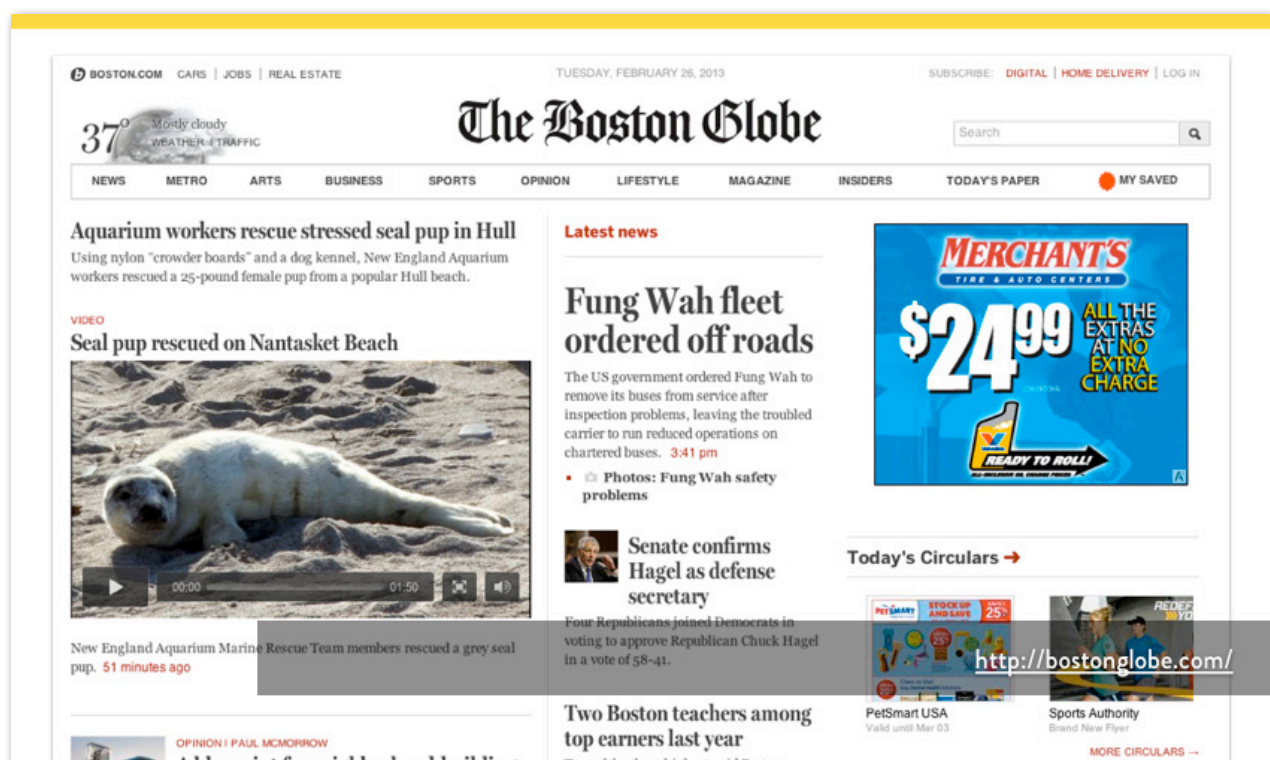


raphy, and color. That's right: CSS is responsible for visual design. It's what lays out your page. And therefore, as designers, it's what we're going to be spending a lot of time in.

JavaScript is something different, and interesting. It's used for making things animate across the page and for making them more interactive. So JavaScript is sort of a different beast. It's used to make more complex web pages and more complex interactions on web pages.

In this class, we're not going to touch JavaScript at all. We're just going to stick with HTML and CSS. In future classes, and if you're interested, I'd encourage you to go into JavaScript. It's fascinating. And there's a ton that you can do.

But let me explain how HTML, CSS, and JavaScript might work on a typical web page. Let me take this one. It's the Boston Globe.



It's an old screenshot. Doesn't really matter what day it's from. It still works for our purposes.

It's got a bunch of headlines. And for simplicity's sake, let me take one of these headlines, like this one right here. The "Fung Wah fleet ordered off roads."

If I blow it up, I see that there's a couple of different things going on here. I've got a headline. I've got a deck here. I've got a timestamp and a link to photos and stuff.



Now, HTML is responsible for these words appearing on the page. It's also responsible for the fact that if you click on the headline, it'll take you to the full story where you can read more. And in fact, if I look at this page in raw HTML with all the visual design and the CSS stripped away, this is what it looks like. And you can see, it's the same headline and the same deck. This is what raw HTML looks like:

## Latest news

### [Fung Wah fleet ordered off roads](#)

The US government ordered Fung Wah to remove its buses from service after inspection problems, leaving the troubled carrier to run reduced operations on chartered buses. 3:41 pm

- [Photos: Fung Wah safety problems](#)



### [Senate confirms Hagel as defense secretary](#)

Four Republicans joined Democrats in voting to approve Republican Chuck Hagel in a vote of 58-41.

The CSS is what's responsible for the typography. And here's what the CSS for this particular example looks like, or at least a big part of it. It says the font-family, the typeface, is Miller Headline Bold for this headline here, that the size is 36 pixels. That's "px." That's a unit of typographic measurement on the web. It's more frequently used than points.

And that the line-height, or the leading, is "36px" also. So it's 36 pixels solid Miller Headline Bold. The deck underneath it is Georgia, which is a very nice serif typeface that reads well on monitors. It's 13 pixels on 18 pixels. That's the spec.

This is all done in CSS. And this is actual CSS code you're seeing here in front of you.

So that's HTML and CSS.

JavaScript, the third pillar, which we're not going to focus on in this class, what that's responsible for, say, when you roll over the nav bar on the Boston Globe homepage, you see a dropdown that animates down. Well, JavaScript makes that dropdown happen. The headlines and the images here in the dropdowns, that's HTML and CSS. But the fact that the dropdown drops down, that's JavaScript.

## CSS

```
font-family: "Miller Headline Bold";  
font-size: 36px;  
line-height: 36px;
```

```
font-family: Georgia;  
font-size: 13px;  
line-height: 18px;
```

### Fung Wah fleet ordered off roads

The US government ordered Fung Wah to remove its buses from service after inspection problems, leaving the troubled carrier to run reduced operations on chartered buses.  
3:41 pm  
• [Photos: Fung Wah safety problems](#)

Lesson 1 of 6: Coding for Designers

In this class, we're focusing on HTML and CSS. But I'll warn you that these two things are very different. They're two completely different programming languages. They have different syntax. They have different rules. They have different structure. It's frustrating for the new learner because instead of just learning one language, you've got to learn two.

Why do you have to learn two? Well, here's why. It's because HTML actually came from the desire of researchers to share academic papers, academic papers like this one. This is an example paper.

And this paper, like all academic papers, has a headline. It's got some paragraphs of text. It's got some subheads. And it's got a bibliography or references.

Now, in the '80s, when HTML was first developed, a big driving force was to make references in a bibliography clickable so that you could see the actual paper that folks were referring to. This was cool, groundbreaking stuff in the early '80s. The only problem was they were using it on computers that were ancient to us.

So they didn't care about graphic design. In fact, they couldn't even do anything with graphic design. HTML—which developed in the early '80s—predates CSS, which cares about graphic design.

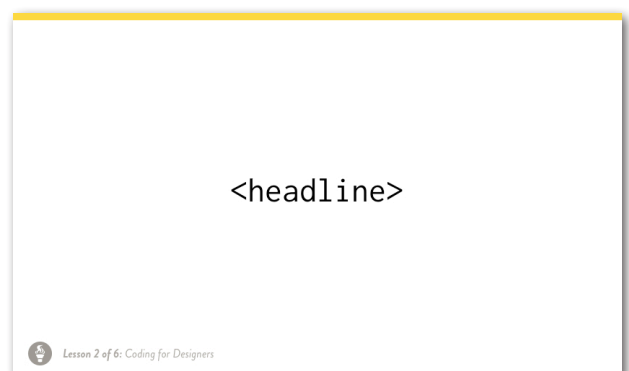
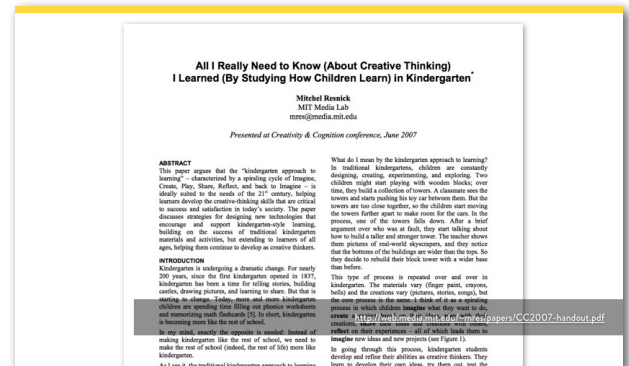
They came at different times. And so they're different languages. But you've got to learn both.

All right, so let's start with HTML and talk about how HTML actually works. Let's say you've got a title or a headline. We need to mark out that it's a headline or a title. But we need to do it using only the keys on the keyboard.

What I mean by that is: HTML started in the early '80s. This was before they had mice, for goodness' sakes. They didn't have dropdown menus. They didn't have Microsoft Word. They don't have any of that stuff.

So they needed to do it in an incredibly simple way. And here's what they came up with—this. It's the word “headline,” in this particular example, surrounded by a less-than sign and a greater-than sign. Yep, those brackets on the end are just the less-than and greater-than sign. That's all they are.

They're called corner brackets. That's we call them in fancy web speak. So they put the word “headline,” in this example, surrounded by two corner brackets, a left corner bracket and a right corner bracket, with the intention that when the computer sees this, it doesn't actually display this to the public. This is hidden.



But what it means is the thing next to it is actually the headline. So display “Hamlet,” but display it as a headline. This was a great idea. But it has a flaw.

The flaw is, what if there’s another line of text? This is ambiguous. Is “Hamlet” the headline? Or is “Hamlet by William Shakespeare” the headline? It’s unclear to the computer. And the computer does not like ambiguity.

So the folks who made HTML came up with a solution. And that is a matching marker. It’s the same thing as headline with two corner brackets. But it includes a slash right before the word “headline.”

That’s a regular slash, now. That’s the kind of slash that you’d use if you were writing a date, for instance. It’s right there on your keyboard. It’s just a regular, plain-old slash.

And so they took this, now, pair of markers. And they surround the thing that you’re trying to mark out with the markers. So this is saying, “Hamlet,” and only the word “Hamlet,” is the headline. This is how it might be rendered on a final web page.

So this is how HTML works and how it looks. The only difference in this example and real life is that it’s actually not called “headline.” I just made that simpler for us for learning. In real life, it’s called “h1.” They abbreviated it.

But this is real, genuine HTML. And h1 is a headline or a header. That’s all there is to it.

Now, this h1 thing, we call that a tag. That’s what the “h1” with the corner brackets is. It’s a tag.

This one right here, the matching pair, that’s also a tag. The first one is the opening tag. The second one is called the closing tag.

If we’re going into vocabulary. I might as well tell you that the whole thing, together, is called an element. So the whole thing, including the tags and the words in the middle, is the element. And this is an h1 element.

This, for instance, is a paragraph element. It has an opening paragraph tag and a closing paragraph tag. It’s abbreviated to “p.” That’s what HTML uses.

```
<headline> Hamlet
```

Lesson 2 of 6: Coding for Designers

```
<headline>Hamlet</headline>
```

Lesson 2 of 6: Coding for Designers

**OPENING TAG**      **CLOSING TAG**

↓                      ↓

```
<h1>Hamlet</h1>
```

└── **ELEMENT** ─┘

Lesson 2 of 6: Coding for Designers

```
<p>To be, or not to be: that is the question:  
Whether 'tis nobler in the mind to suffer the  
slings and arrows of outrageous fortune, or to  
take arms against a sea of troubles, and by  
opposing end them?</p>
```

Lesson 2 of 6: Coding for Designers



It doesn't—you don't type out the word "paragraph." You just use the letter "p." And that signifies that the stuff in between those two tags is a paragraph of text.

Now, let's add CSS to this. HTML is responsible for words and images. CSS is responsible for layout, typography, and color. Let's layer some CSS on this bad boy.

So we've got this h1 of "Hamlet." If we were to put some CSS on it, here's how it might look. We could change the typeface. It's called "font-face" in CSS. We could change the typeface to Helvetica, and it would look like this:.

If we—we can change the weight of the typeface. We can change it to Bold. And it might look something like this.

We can change the color of that particular typeface. This is using an RGB triplet, the same kind that you would find in Photoshop. And it changes it to this sort of green color specified in Red, Green, and Blue.

We can even do things like transform it to uppercase. CSS has the ability, just like you would find in InDesign, or in Illustrator, or Photoshop, to be able to make something all caps. This is actual, genuine CSS that you're seeing here.

This is how you might style typography on a web page. This is how it works. And this is what we're going to continue to go through today and for the rest of this class.

So the holy trinity of web design—HTML, CSS, and JavaScript. And JavaScript, of course, we're not covering in this class, but HTML and CSS, we certainly are. And that's the Trinity of Web Design.

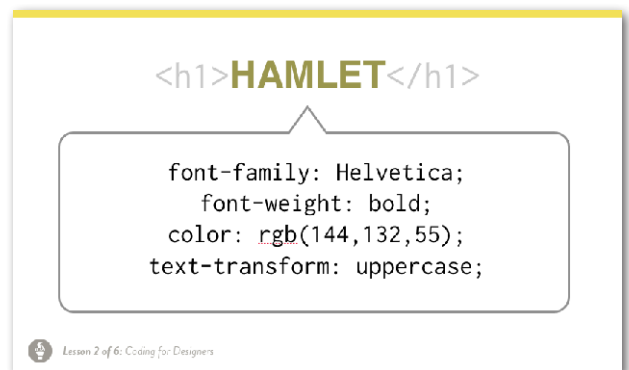
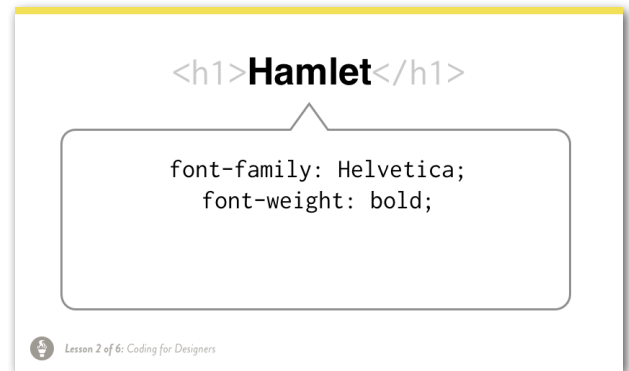
## BIG IDEA #2

### "HELLO, WORLD"

That brings us to big idea number two, "Hello, world." Since at least 1974, computer programmers, when they're learning a new language, have a tradition. The tradition is that the first program that they write is one that simply writes, "Hello world."

Here's an example of one like that. This is in the computer programming language called C. And the only thing that it does is it prints the line "hello, world".

Well, now it's your turn. You're about to be inducted into the family of computer programmers, because you're going to make your own hello world program in HTML.



Now, if we're writing HTML, remember that HTML is just text. And so we need a piece of software, a text editor, that can manipulate text. And there are lots of them.

Here are two: TextEdit and Notepad. They're included free on your computer if you've got a Mac or a PC. And they work perfectly fine for making HTML.

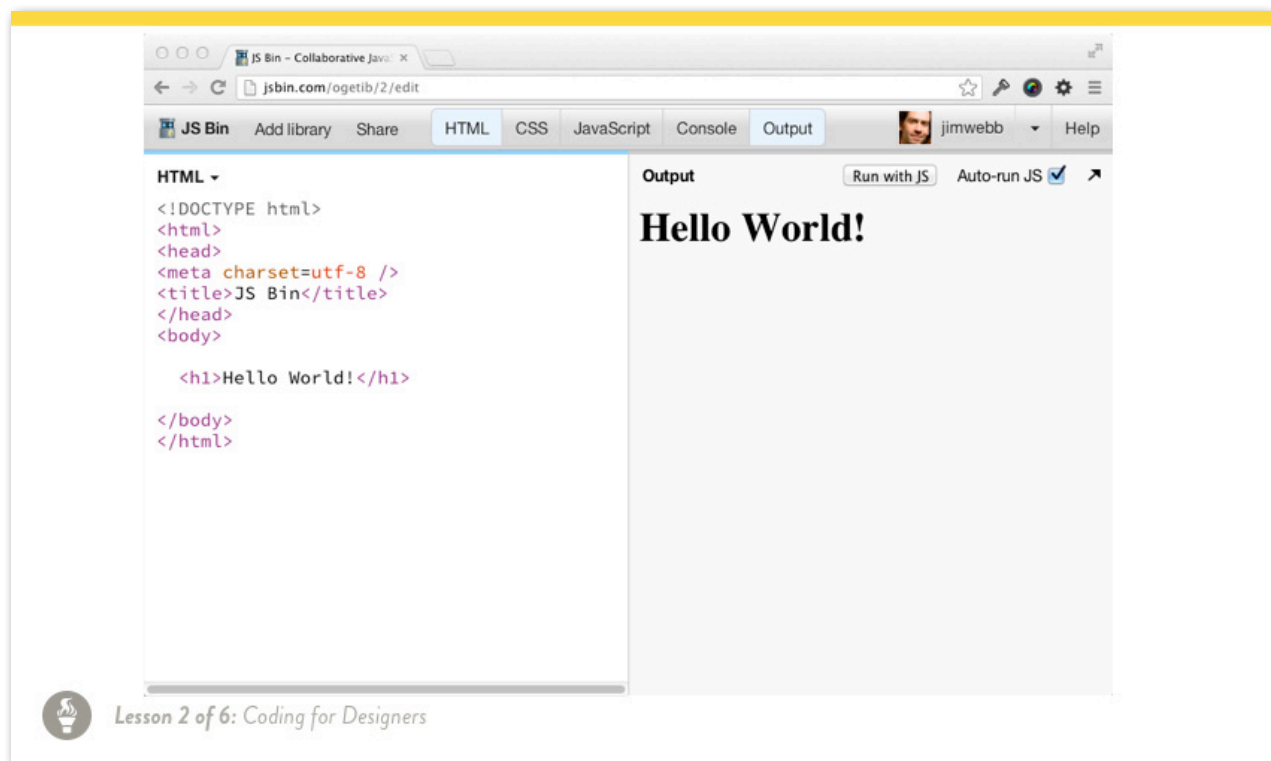
There are also a bunch of commercial programs which are specifically suited for writing code. And they do things that plain old text editors can't. But underneath, they're just text editors. That's all they are. And that's all you need for writing HTML.

However, for this class, I'm going to ask all of you to use the same piece of software so that we can all be consistent. And that piece of software is online. I want you to use an online text editor.

I have chosen one for you. The one I want you to use is called JSBin. It's a website. It's at <http://jsbin.com>. It's free. You don't have to register. They're not collecting information. You don't have to pay anything. It's completely free and open.

JSBin was originally created for sharing JavaScript, "JS." But we're not going to use it for that. We're going to use it for HTML and CSS.

Here's why I've chosen this tool. Here's what it looks like. it's a two-pane model here. This is a web page. This is actually jsbin.com. And there's two panes.



On the left hand side is the place where you can type in code. And on the right hand side is how that code would appear to the public, how it would appear to your mom or dad if they were viewing this web page, or to a customer if they were viewing the website. So you've got this split-screen layout happening. When you type something on the left, it affects what's on the right. And it does it live, too.

So if I come in here and I type, “How Are You?”, you can see it update immediately and automatically on the right hand side with how the web page would actually look. Pretty cool. Very cool for learning how to code web pages.

What’s cool on top of that is that JSBin automatically creates web addresses when you start editing pages. And that web address you can share with other people. For instance, this is the one that it created for me when I started typing to create the materials for this class lesson.

And you could even go to this address right now. You could type this in as you see it on the screen. And you would see this page as I prepared it for this lesson. Pretty cool, huh?

So let’s do it. We’re going to load up JSBin. I’m going to show you what it looks like. And then I’m going to ask you to do it on your own.

To load up JSBin, all you need is a web browser. I use Chrome. And go to JSBin.com.

*(Note: this section relies extensively on live coding. Following along with the video is recommended.)*

When you get there, it’s going to look something like this. There’s code on the left hand side. And it’s a bunch of gibberish. And on the right hand side is their welcome screen.

Well, the code on the left is generating what’s on the right. And in fact, you can go into this code and start typing. And it will automatically make a copy of their home page. You’re not defacing their actual home page. But it makes you a copy. And you can type whatever you want in this copy.

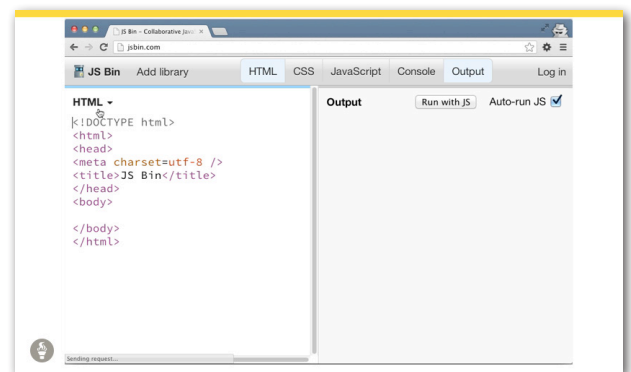
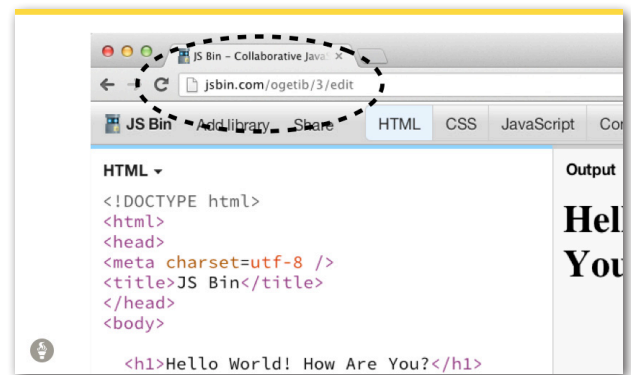
You can see how this web page is built. You can delete things and see what happens. You can totally wreck and destroy this page, and have fun doing it, and see what happens. When you’re done having fun, go to JSBin and click New.

And you’re going to get a page that looks like this. This is a blank page. And this is the page that we’ll use to start typing. You’ll notice a couple of things about this page. First of all, there’s a code here on the left. But there’s nothing on the right. The part below Output is completely blank.

You’ll also notice that there’s coloration in this code here. There’s some purple and orange. That’s JSBin doing that for you. You don’t change the colors in this code. It’s doing the color for you.

I’d like you to go ahead and do this now. Go to JSBin.com and create a new page. Press pause on this video. I’ll be waiting for you here when you get back.

You’ve done that. Now, when you go on this blank page, take care to notice this row of buttons up here, the HTML, CSS, JavaScript, this row right here. If you would, just have HTML and output switched on. The other buttons—you can play with them, go ahead—the other buttons open up new panes where you can type in stuff.



We're only going to use HTML and Output. Even when we start working in CSS, we're going to use the HTML pane to do it. So just make sure that only these two buttons are on.

Now, let me direct you back to the code. You see that the output here is completely blank on the right hand side. But we still have code on the left. And that code looks like this.

It's code that JSBin, this website, put in there for us. You can go ahead and start typing around in there and see what happens if you want. You can play. So this code right here, it's not showing anything visually. The output is blank. So what's that code doing?

Well, let's look at it briefly. Well, remember I told you that HTML is composed of tags like this. They're opening tags, and they're closing tags, and they come in pairs. Well, looking at this code, do you see any pairs of opening tags and closing tags?

I do. Here's one, the "title" tag. We don't know what it's doing. We're going to know in a second. But even if you don't know what it's doing, you can see that there's a tag here, a title tag and a closing title tag with the words "JS Bin" between them.

Do you see another tag? Here's another tag. Almost the very first one, the second tag in the document, is "HTML". It's the HTML tag. Now, do you see the matching pair for that HTML tag, because remember I've told you that tags generally have matching pairs? Yep, it's down here at the bottom.

So there can be a matching pair of tags that have tags inside of them. So the head tag is inside the HTML element. The body tag down below is also inside the HTML element. In fact, all of these tags except for that first one, the weird one that's gray—all of these tags are inside that HTML element.

This is completely normal and natural. HTML is built like Russian dolls. You can nest tags inside other tags, or inside other elements. So back here on our code, I want to take your attention to a special set of tags. This set of tags is very important.

That's the body tag. The body tag and the closing body tag are very, very important for us designers. Why? Because the content of your page goes in between those tags. That's what people see. This other stuff here, it has a purpose, and we're going to talk about that purpose. But for right now, I want you to focus between what's in between the body tag and the closing body tag.

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>JS Bin</title>
</head>
<body>

</body>
</html>
```

Lesson 2 of 6: Coding for Designers

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>JS Bin</title>
</head>
<body>

</body>
</html>
```

Lesson 2 of 6: Coding for Designers

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>JS Bin</title>
</head>
<body>
</body>
</html>
```

← **YOUR PAGE'S CONTENT**

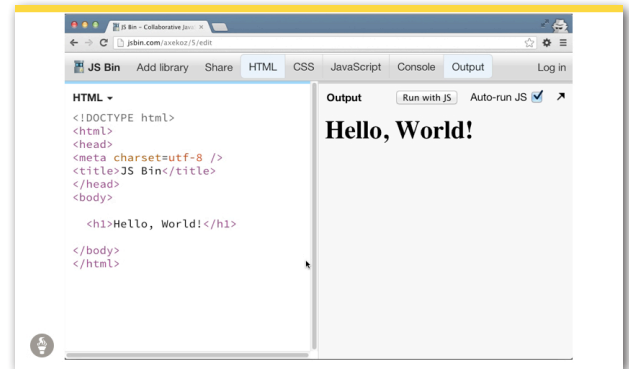
Lesson 2 of 6: Coding for Designers

Let's do that now and create our "hello world." So back here, you can see the code. And I want you to find the opening body tag and the closing body tag. Come in here. I'm going to hit Return a couple of times to give myself a little bit of space. And then when you're ready, start typing.

What are you going to type? Hello, world. And you can see that it updates automatically here. That's what the body represents, the output there. Now let's add our tags, our h1 tags. I'm going to add the opening tag here and the closing tag with the slash here. And you see how the output changed.

The headline now appears bigger and bolder, which it does by default. And it's an h1. Go ahead and do this now. Pause the video. Go do this now. And come back when you're done.

Did you do it? Do you feel like a superhero? Congratulations. You're now inducted into the family of computer programmers as a novice, as a journeyman, and we're excited to have you.



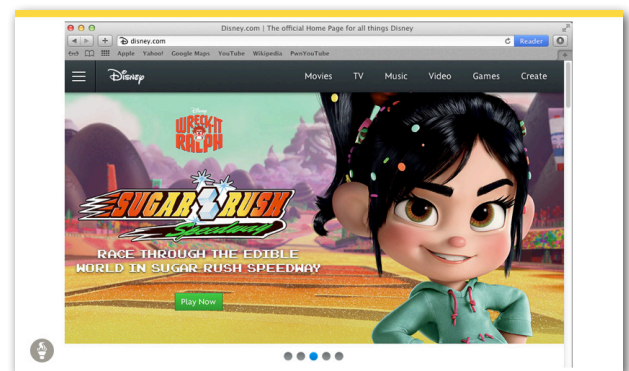
So let's go back to this code. So this is the code that was just created for us by JS Bin. We didn't ask for it. But this was the blank page. What is all this stuff? What's this code doing? Well, let's walk through it. I want you to understand it.

First of all, let's start with that title tag, the thing that we started with. I didn't see any title anywhere on the page, yet this tag exists. So how do we find out what this sucker is actually doing?

Well, here's one easy way. Let's change it and see what happens on the page. So I'm going to come back here. And I'm going to change what's in between these title tags and just type some stuff, testing, and I'm going to see what changes around me.

Nothing's changing in the output. That's not changing. But do you see what is? Yep. This is changing. The title tag is responsible for the title that appears in that tab at the top of your web browser. Pretty cool. And in fact, this is true on web pages all over the internet.

Here, Disney's home page. You see that it's got a title here in the top of the web browser, disney.com, the official home page for all things Disney. That was created by their title tag. Cool.



The title tag is responsible for the page title. And it's not the page title that appears in the body of the page. It appears in the title bar of the browser window.

The body tag is responsible for your page's content. We've already gone through that. That's what appears in the main window of the browser. That's the guts of your page. That's what people see.

Now, there's a bunch of stuff before the body tag, all of this stuff. What's this doing? We know what the title tag is doing.

But generally speaking, these things are other information about the page. They just don't display in the body. For instance, this first one, the DOCTYPE html is telling the computer, hey computer, this text document you're reading, this is HTML. That's the format that it's put in. It's not in some other weird format. It's actually HTML.

Now, this first tag here, by the way—HTML—that's what it's telling the computer—this first tag, the HTML tag, is saying, hey, computer, the HTML starts here. This is where you should start looking for HTML. And the HTML ends down here. That tag should surround all of the content of the HTML page.

Then there's this tag, the head tag. The head tag is interesting, because it seems like the head tag should be for a headline, or for a header. But it's not. In HTML speak, in web page speak, this opening head tag—and you see the closing head tag down there—these guys are for extra information about the page that doesn't appear in the body.

There are two elements here in the head tag. Do you see them, the meta tag and the title tags? The meta tag is responsible for telling the computer what the character set of this particular page is. It's set to "utf-8," which is a general way of saying, this web page could be in any language.

It could be in English. It could be in Spanish. It could be in French. It could also be non-English characters like Japanese characters, or Chinese characters, or Hebrew characters. That's what this is saying. If the page was only Chinese, then, for instance, this might be different. But for us, the utf-8 is a multilingual, multinational, global setting that should be used for all of your web pages. That's why JSBin puts it in here for you by default.

The title tag is a page title. So that appears in the title bar. Now, do you notice how this meta tag here, it does not have a closing tag, but it's got the slash at the end? Yep, some tags don't need closing tags. They're okay by themselves. And the meta tag happens to be one of them.


How would you know which is which? Well, you look it up in a reference. And I'm going to give you some of those references later. For now, I'm telling you the meta tag is a special kind of tag that doesn't require a closing tag.

So back here in our document, the HTML tag surrounds all of this stuff. And because it's hard for me to see what's going on here in this spaghetti mess of code, what I sometimes like to do is to indent stuff that's inside other tags. Here, I've added spaces to all the stuff inside the HTML tag.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8 />
  <title>JS Bin</title>
</head>
<body>

</body>
</html>
```

← **PAGE TITLE  
(FOR THE  
TITLE BAR)**

 Lesson 2 of 6: Coding for Designers



And I'm just hitting the space bar. That's all I'm doing. And here, I'm going to put indent the stuff inside the head tag. And then I'm going to indent the h1 inside the body tag. So it gives me a nice little hierarchical way of seeing what's going on here on the page.

I could even split up the title tag into multiple lines, no problem. Now, this is only for my benefit. I could very easily take all this stuff and put it on one big, ugly line of text without any returns in between them at all. See what I'm doing here? I'm making it look super ugly.

The computer doesn't care, though. See the output of the page? "Hello, world" looks exactly the same. And it's working right? Yep, we know it's working. If I change text, it works.

So what's happening here is that the computer does not care about the space here in my code. But I do. I'm a human. I want to see this stuff. So I'm going to space it all out nice and pretty like this. That's how I'm going to space my code.

You can do yours differently. It's completely fine. It's entirely up to you. And it's just a style thing. But that's how I want to space my code out.

That, my friends, is "Hello, World!" Congratulations. I'm glad you did it. And we're going to move on to bigger and better stuff now. This is Coding for Designers.

## BIG IDEA #3

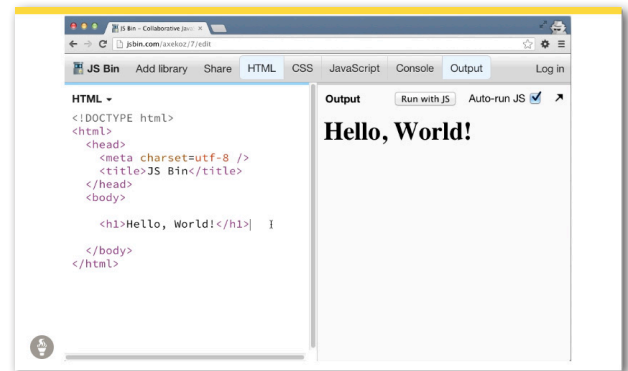
### WORDS AND IMAGES FIRST, THEN ADD DESIGN

*(Note: this section relies extensively on live coding. Following along with the video is recommended.)*

It's time, then, for big idea number three. If you're making web pages, especially if you're new to it, the words and the pictures should come first. And then you layer in the design. We're going to do that together.

Now, let's come back to our "Hello, World" example. We added an h1 element with Hello, World inside the body element. And remember, we're focusing just on what's inside the opening body tag and the closing body tag. That's where the meat of our page is. This h1, it belongs around head-lines. That's one of the tags that we've learned.

Another tag that we've learned but have not yet put into use is the p tag that goes around paragraphs of text. So we're going to add some paragraphs of text to our Hello, World document. I'm going to do it. And then I'm going to ask you to do it for me.



To add paragraphs of text, I'm going to come here to the code. And where am I going to add them? I'm going to add them right here below the headline. I'm going to go find some dummy text. I like fillerati.com. It's got all sorts of authors. I'm going to take three paragraphs of Moby Dick.

I'm going to grab them here, copy them. And then back in my original document, I'm going to paste them into the place that I selected. Boom, they're here below Hello, World. But check it out, it's all coming together as one big paragraph, not three separate paragraphs. Let me add a line break.

It's still one paragraph. That line break didn't take. Why isn't it taking? Well, here's the deal. Wait a second, is this working? Let me just make sure it's working first.

Yeah, okay, it's working. So text is coming through. But it's not coming out as paragraphs. And the reason is, that I haven't added the p tags. The p tags are the only way the computer knows that these are paragraphs. It's not like Microsoft Word.

It's not smart enough to put the paragraphs in for me. It turns out there's reasons for this in HTML. But the long and short of it is, you have to have the p tags. That's what makes paragraphs.

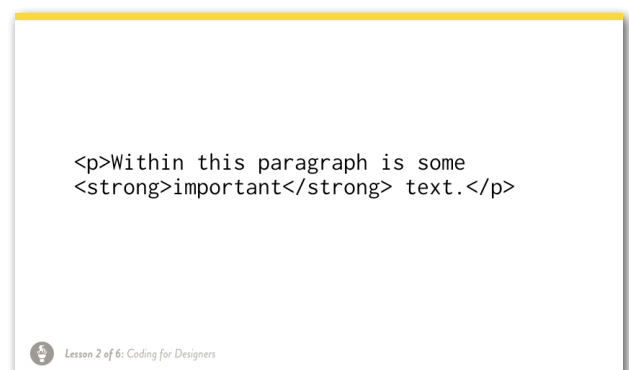
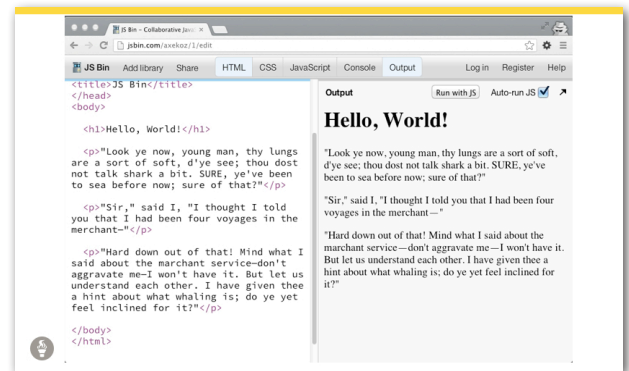
So I'm adding in the p tags and the closing p tags. Once I do that, then now you can see that there are three distinct paragraphs. Without the p tags, they're not at all. You've got to have those p tags. So go ahead and do that now for me please, and add in your own p tags with content that you prefer. I'll be here waiting.

So you've added your p tags. Now, let's say that we wanted to bold or to make something stand out inside one of those paragraphs. Say that we have a paragraph like this. And there's some important text that we want to make stand out.

In Word or in another document program like that, you might hit the Bold button or the Italics button. You can do that in HTML, and it's a tag. It's the strong tag. There are a number of tags, but strong is one of the tags it's used to emphasize a particular piece or area of text.

What it visually does is, it makes it bold. But it does more than that, too, because we can control what its visual effects are. We can make it red or big or small or whatever. But for now, we're just going to make it bold. And let's see how we use a strong tag.

I'm going to come back here to my text, find a piece of text I'd like to make bold. And I'm going to add the opening strong tag. See what happened? It made everything bold.





I'm going to come here and say, "hard down out of that." I'll make that bold or strong. And you see by adding the opening and closing strong tags, I can embolden areas of text. So we have a couple of HTML tags now under our belt.

It's just that we're doing that here in this example. So this is four HTML tags. They're four important good ones. How many are there in total, HTML tags? About 120. That's a lot.

Yes, of course there are. And they're all online. Or many very good ones are online, rather. I've included some here in the classroom. I've included this reference from Mozilla, which are the folks who make Firefox.

[illegible]

CSS Reference	HTML Reference	JavaScript Reference	
	<b>HTML ELEMENTS</b> <ul style="list-style-type: none"><li>• <a href="#">Structural Elements</a></li><li>• <a href="#">Head Elements</a></li><li>• <a href="#">List Elements</a></li><li>• <a href="#">Text Formatting Elements</a></li><li>• <a href="#">Form Elements</a></li><li>• <a href="#">html elements...</a></li></ul>	<b>HTML MICROFORMATS</b> <ul style="list-style-type: none"><li>• <a href="#">hCalendar</a></li><li>• <a href="#">hCard</a></li><li>• <a href="#">hreview</a></li><li>• <a href="#">XFN</a></li><li>• <a href="#">real- Microformats</a></li><li>• <a href="#">html microformats...</a></li></ul>	<b>HTML COMMON ATTRIBUTES</b> <ul style="list-style-type: none"><li>• <a href="#">Core Attributes</a></li><li>• <a href="#">Event Attributes</a></li><li>• <a href="#">Extended Event Attributes</a></li><li>• <a href="#">html common attributes...</a></li></ul>
	<b>HTML CONCEPTS</b> <ul style="list-style-type: none"><li>• <a href="#">Basic Structure of a Web Page</a></li><li>• <a href="#">Doctypes</a></li><li>• <a href="#">HTML and XHTML Syntax</a></li><li>• <a href="#">HTML Versus XHTML</a></li><li>• <a href="#">HTML/XML Accessibility Features</a></li><li>• <a href="#">html concepts...</a></li></ul>	<b>HTML EXTRAS</b> <ul style="list-style-type: none"><li>• <a href="#">Color Names and Related Hex Values</a></li><li>• <a href="#">ISO 2 Letter Language Codes</a></li><li>• <a href="#">MIME Types - Complete List</a></li><li>• <a href="#">html extras...</a></li></ul>	<b>HTML EXAMPLES</b> <ul style="list-style-type: none"><li>• <a href="#">blockquote</a></li><li>• <a href="#">cite</a></li><li>• <a href="#">br</a></li><li>• <a href="#">clear</a></li><li>• <a href="#">div</a></li><li>• <a href="#">html examples...</a></li></ul>

GYMNASIUM Coding for Designers— Lesson 2

So we're going to start working in CSS now. But CSS has a completely different syntax than HTML. It's a different language. And unfortunately, it's apples and oranges.

Let me show you what I mean. In HTML, here's a tag. It's got the left corner bracket or the less than sign, the name of the tag, in this case, h1, and then the right corner bracket. That is the structure of an HTML tag. It's true no matter what the tag is in HTML, whether it's a p tag or a strong tag or a meta tag, like one of those other tags that we saw. CSS is different.

Here's the CSS declaration—font-family, colon, Arial, semi-colon. It's got several parts to it. First, font family, the thing I'm trying to change, the typeface. In this case, I'm setting the typeface to Arial.

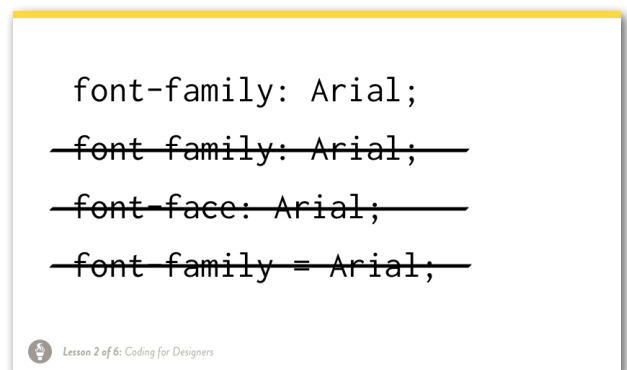
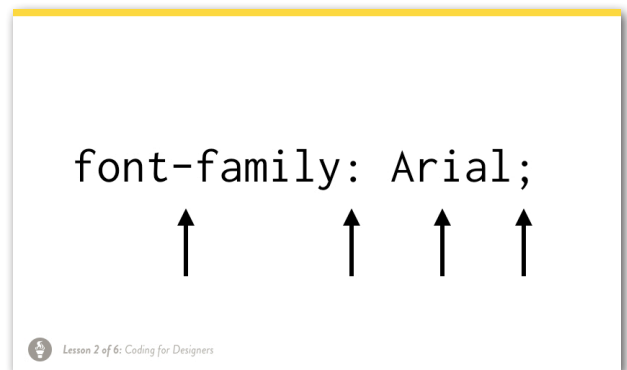
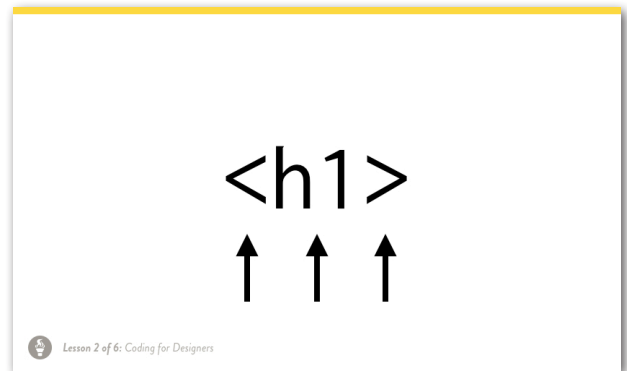
So "font-family" is the thing I'm trying to change. I put a colon right after the thing I'm trying to change, the font family. Then I say what I want to change it to, in this case Arial. And then a semicolon at the very end. This is the structure of a CSS statement. So this is very unusual, right? It's totally different than the HTML we were looking at.

We got semicolons and colons and dashes. We didn't really have that in HTML. Moreover, the syntax here is very exacting. And it has to be precise for CSS to understand what it is that you're trying to accomplish. Font, dash, family, colon, Arial, semicolon. That's the proper declaration for making something Arial, for setting it in Arial.

If you type this in CSS: font, space, family, colon, Arial, it won't work. Believe it or not, the system is so dumb that it will not understand what you're trying to accomplish here. And it won't do it. Likewise, if you say "font-face: Arial." It makes total sense to you and me that you want to change the typeface to Arial. CSS doesn't know that.

You've got to say, "font-family." "Font-face" doesn't do anything. Likewise, if you don't have that colon and you have an equal sign (for instance, font-family equals Arial) it won't work. So none of these work.

The only one that works is the top one, font, dash, family, colon, Arial. Before we actually do this, let me show you another one to show you a little bit more about the syntax of how CSS works. Here's another one: font-size: 24px; — px for pixels.



It's a typographic unit of measurement commonly used with type. There are others, but pixels is way more prevalent than points. You can actually spec in points, but it's weird. And it doesn't behave the way that you might think it behaves or that Illustrator behaves.

Pixels is a much more solid measurement. It's not the only one. There are others. But it's the one that we're starting with.

So, "font-size: 24px;" with a semicolon at the end. You got the thing you're trying to change. You've got that colon. You've got what you're changing it to.

And you've got the semicolon at the end. The colon and the semicolon are consistent with font family.

Likewise, this is the only appropriate way to say this is "font-size:24px;" with a dash. You can't say, font, dash, size 24 space px. It looks totally normal to you and me, right? But the computer won't know what to do with it. It won't make it the right size.

You can't say, font-size: 24pixels. You can't say font-size: 24. You've got to say 24px with no space. These others won't work. Only the top one will work.

I know it's specific and exacting. And once you get the hang of the syntax, you get a better feel for it and it makes sense. But until then, you probably have to look it up each and every time and make sure you get the colons, the spaces, the semicolons, and the dashes exactly correct.

Now, if we're going to actually apply this to something on a page, like to this headline, for instance, how do we do that? How do we say, hey, I want this headline to be Arial 24 pixels? Well, here's how.

We're going to do it this way. We're going to take the opening tag, this opening h1 tag. And we're going to get right after the letters h1 but before that corner bracket. And we're going to split that tag open by hitting the space bar.

When we do that, we're going to add in some text that allows us to apply styles, cascading style sheets, CSS, to this particular thing. The thing that we add is this. We add, style, equal sign, and a set of quotations. That's what we add inside that opening tag.

Then, if that weren't complicated enough, inside that set of quotation marks, here, we're going to put in the CSS that we came up with before—the font-family colon, Arial, semicolon. That goes inside the quotes. So inside the opening tag, we say, style, equals, quotations, font dash family, colon, Arial, semicolon, quotations. That is a lot of characters.

```
font-size: 24px;
```

Lesson 2 of 6: Coding for Designers

```
font-size: 24px;
```

```
font size: 24 px;
```

```
font-size: 24pixels;
```

```
font-size: 24;
```

Lesson 2 of 6: Coding for Designers

```
<h1 style="font-family: Arial;">
Headline</h1>
```

Lesson 2 of 6: Coding for Designers

You'll get the hang of it, I promise. Let's do it together. So I'm coming here, looking for my h1. I'm going to look inside the opening tag, split it open with my space bar, and type style equals in the quotations. Inside the quotes, I use font dash family, colon, Arial, semicolon. And look what happened.

My text magically transformed into Arial. It worked. I changed the headline from Times New Roman, which it was before. And now it's Arial. Pretty cool, huh?

But if I type this a little bit differently, it's not going to work again. So let's see, if I come in here, and let's say I forgot to put in that colon, it went back to Times. It didn't give me an error message. It didn't say anything at all.

It just went back to Times. Put the colon back in, and it's fixed. Font face, back to Times, didn't work. Font by itself doesn't work. Font dash family, that's the only thing that works.

Moreover, if I forget to put the style thing in the quotes, doesn't work. The page still shows, but it doesn't take my declaration that I want it to be Arial instead of Times. It's got to be style equals, the set of quotes, the font family Arial.

Now, you probably can experiment if you haven't already and put in different typefaces, like Times or Impact. What's another typeface we can use here? We can use Georgia. That's a good serif typeface. All these typefaces work.

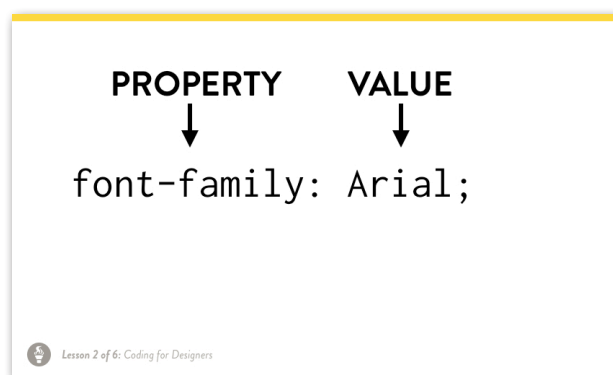
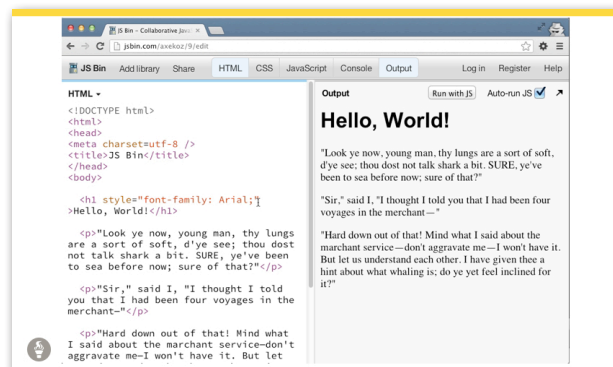
So you might think, oh, I'll use Archer. Archer's that beautiful typeface from Hoefler and Frere-Jones. Well, that doesn't work. It doesn't work, and I'm going to explain to you why it doesn't work.

But Arial does. Arial, Impact, and Georgia are all typefaces that work, because they're typefaces that are installed on the computer. And Archer is not installed on my computer. But I'll get into that in a second.

So to specify a typeface on a web page using CSS, it's this format—font, dash, family, colon, Arial, semicolon. This first part, font, family, that's a CSS property. There are a bunch of properties, like font size, color, font weight. There are a bunch of different properties.

And the thing that I want to change it to, in this case, Arial, that's the value. So here I'm changing the property of font family to the value of Arial. The colon and the semicolon are extremely important. You've got to have both of those for the CSS to work. If you don't, it doesn't.

Font, family, colon, Arial, font size, colon, 24 pixels. So if I want to apply both of these to that headline, to that h1, how do I do it? Well, it's simple. You just string them next to each other—font, family, Arial, semicolon, font size, 24 pixels. And it works. Let's try it.



We're going to come back here to our font, family, Arial. And then inside the quotes, I type, font, dash size, 24px, semicolon. And it worked. It changed it to 24 pixels. It's smaller.

Let me test it and make it real big—124. See? Now we know it's working, because when you change the number there, it changes the size. And we changed it something dramatic so we can see what's happening.

That's how you change type in a web page. Pretty cool, huh? Now, we strung these two together—font, family, Arial and font size 24px, by putting them all on one line. It wrapped to several lines in the little text editor because my window was kind of small. That's fine. There's no problem with that.

But this semicolon here, the semicolon in the middle, the reason it's so important to have semicolons after each of these CSS statements is so that you can put them all on one line. The semicolon is kind of like the period at the end of a sentence. It's saying, okay, hey, web browser, change font family to Arial.

If you don't have the semicolon there, it's like a run-on sentence for the browser. And the browser would interpret this as, font family, okay, font family, I want you change to Arial font size 24px.

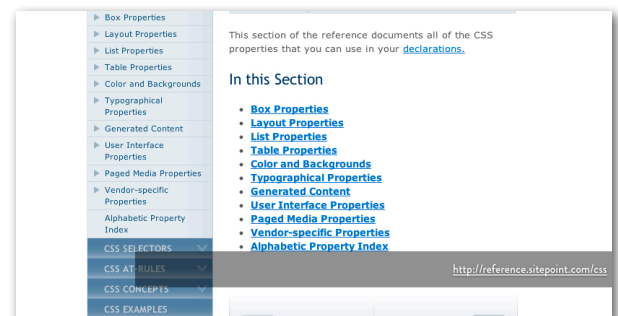
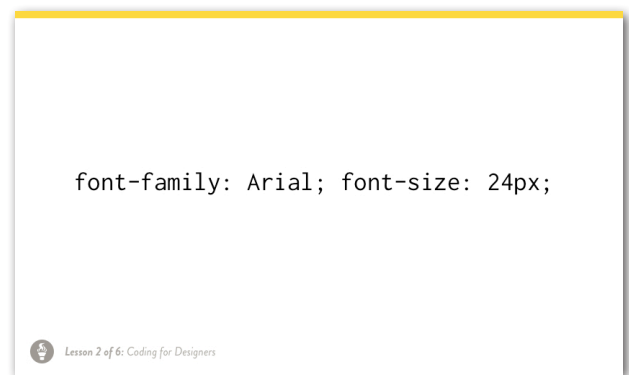
"I have no idea what that means. It doesn't make any sense." That's what the computer will say. And it won't do any of it. That's why you have to have the semicolon.

You might be asking, okay, I want to do more this. I want to look up different things that I can do with CSS. Fantastic. I love it.

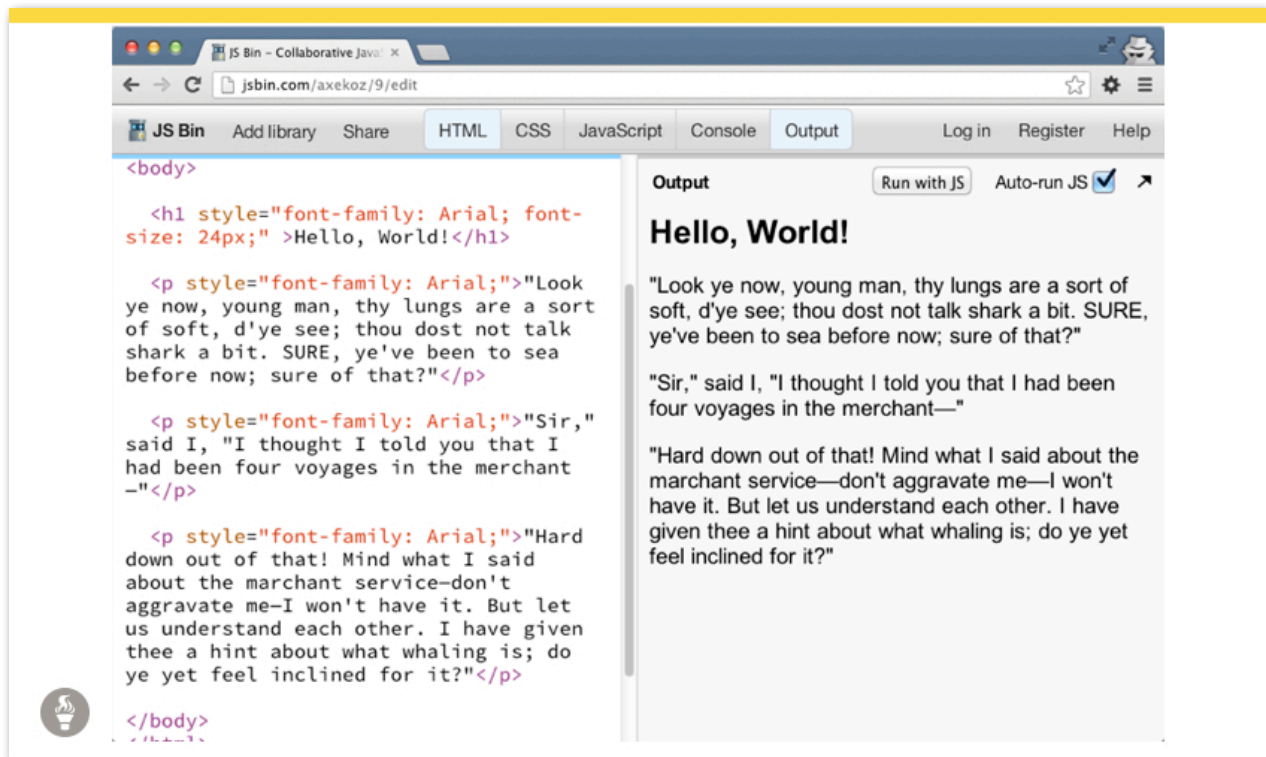
I'm providing for you several references here in the classroom. One of them is another one from SitePoint, and it organizes CSS properties by the kind of thing, like layout and typography, and color, and background, and then another one from Mozilla that organizes them alphabetically. It includes all sorts of esoteric things that folks may not even use.

It may not even work in a bunch of browsers. But it's a whole list of every darn thing. You can go nuts with these lists.

I have a quick assignment for you. Let's set the paragraphs on your page in Arial. We've set the headline in Arial; now I want you to set all the paragraphs in Arial. Go ahead and pause the video, and go do that now.



If you're back and you set the paragraphs in Arial, I bet you did it something like this.



See what's going on here? Each of these paragraphs, these p tags, have style added to them. So it's a p tag with style, font family, Arial. And we've added it here to this paragraph, here to this paragraph, and also to the last paragraph, to all three paragraphs. That's one way to set the entire three paragraphs in Arial.

Now, if you're looking at this and saying, wow, that is a really time consuming way. Isn't there a way to set everything at once to Arial, like on the entire page or all the paragraphs no matter what? And yes there is. There absolutely is.

And we'll be doing that as we get more sophisticated in our CSS. That'll be in the next lesson. We'll talk about how to apply styles more globally and to turn them into true style sheets. For now, we're just going to modify one thing at a time to keep things simple while we're learning.

Now, you may be wondering, okay, Arial's great. I want to change to use other typefaces instead of Arial. How do I do that?

Well, we talked a little bit about that and about changing things to, for instance, Georgia, Tahoma, Impact. But Archer didn't work. So if you're using other typefaces, some work and some don't.

What's the difference? Well, you can only use typefaces that are pre-installed. And they have to be pre-installed on the system of the person who's viewing the web page. So it doesn't matter if it's on your system as the designer.



It has to be on the system of the person who's looking at your page, like your Dad or your Mom or your customer. It has to be on their computer, not your computer. And between Macs and PCs, there's a very small list of typefaces that are found on most people's computers. And this is that list.

So you can use any of these typefaces. They're collectively called web-safe fonts, because chances are, it's going to be installed on the computer of the person who's looking at your web page. Now, this is a fairly austere list. There's not a lot of choice here.

There must be some way to add other typefaces. And in fact, there are ways to add other fonts that you own and that are properly licensed, and web fonts that are available on the internet itself.

We're going to go over that in the next lesson, so stay tuned for that. For now, we'll stick with fonts that are pre-installed, which means the fonts in this list.

So the type properties that are available to us in CSS, let me go back over some of the ones that we've already covered. "Font-family: Arial"—this is how you change the typeface. You could do a lot with this. You could change it to Verdana.

You could change it another typeface. And if a typeface has more than one word in it, like Times New Roman, that's three words. or Comic Sans, two words, then you should put quotes around it. And in fact, the real name for Comic Sans on our computers is Comic Sans MS.

So if you want to spec something in Comic Sans, God forbid, you should put single quotes around it, not double quotes, but single quotes. And type in the three words, "Comic Sans MS".

Now, if a computer that's looking at this web page doesn't happen to have a typeface, then something else happens. Let's say you want them to see it in Helvetica.

But Helvetica is not pre-installed on many PCs. It is on some Macs but not on PCs. Well, there's a way you can spec that. You can use font-family and have CSS say hey, if they've got Helvetica, use Helvetica. Set it in Helvetica.

If they don't have Helvetica, move to the next one down this list. So this declaration says, "font-family: Helvetica, Arial". What that's saying is, use Helvetica if you got it. If you don't got it, use Arial. Pretty cool.

ARIAL	TIMES NEW ROMAN
ARIAL BLACK	TREBUCHET MS
COMIC SANS MS	VERDANA
COURIER NEW	SYMBOL
GEORGIA	WEBDINGS
IMPACT	WINGDINGS
TAHOMA	

 Lesson 2 of 6: Coding for Designers

```
font-family: Arial;
font-family: Verdana;
font-family: 'Comic Sans MS';
```

 Lesson 2 of 6: Coding for Designers

```
font-family: Arial;
font-family: Verdana;
font-family: 'Comic Sans MS';
font-family: Helvetica, Arial;
font-family: Helvetica, Arial,
Verdana, sans-serif;
```

 Lesson 2 of 6: Coding for Designers

You can set a list of typefaces. It won't use both. It'll stop at the first one that's usable on the computer that's being used to view the web page.

And you can spec more than two in this list. You could say hey, Helvetica is my first choice. If they don't have Helvetica, set it in Arial. If they don't have Arial for some reason, set it in Verdana. And if they don't have any of those, set it in any sans-serif you can find. That's my last resort. I just want it to be sans-serif.

All of these are valid font family settings. They're valid values for font family.

Let me go to another property that you can experiment with. That's font-size. We know what this is. It's the size of the face, right? Pretty straightforward. The only weirdness here is that if you want it to be 24 pixels, you can't have a space there. It's 24 and px all scrunched together: 24px. No space in the middle.

You can try other settings, of course. 124px if you so desire.

An interesting thing about the web is that generally speaking, you can't use typefaces smaller than about 11 pixels. 10 is about the minimum for legibility—different than in print. You can go down to nine or eight points in print. On the web, that's not really the case.

You should experiment with this using your JSBin "Hello, World" page, and see how small you can get those body paragraphs before they become illegible. But generally speaking, 11, maybe 10 is about the smallest that you can go. That's "font-size."

Let's talk about leading. Leading in the web world is called "line-height." So if you want to change the leading of a paragraph, you change the line-height of that paragraph. And it works like leading otherwise.

This is spec-ing a 38 pixels of leading. This is spec-ing 24 pixels of leading. Interestingly, you can use a different, non-pixel spec for leading. You can say, I want the leading, the line-height, to be 100% of whatever the typeface is.

So essentially, you can mathematically say, I want it to be solid leading. I want the line-height to be 100%. If the typeface is a font-size of 24 pixels, so the line-height would be 24 pixels. You could also say more than 100%.

You can say 120%. So it's a nice little flexible way of being able to spec leading. But it's called line-height.

Next, font-weight. Font-weight: that's how you make something bold. If you don't want to make it bold, you make it normal. See, this is a weird one, too, because why wouldn't you just make something bold by using the strong tag? Well, you could. You could make something bold by using the strong tag.

```
font-size: 24px;  
font-size: 124px;  
font-size: 11px;
```

 Lesson 2 of 6: Coding for Designers

```
font-weight: bold;  
font-weight: normal;
```

 Lesson 2 of 6: Coding for Designers



But it's used in other ways, too. Let's take this example. So if I come to the code and I try to make a paragraph bold by adding in font-weight: bold, it makes the entire paragraph bold. Okay, fair enough.

Not too useful, but you can totally do it. That's fine. I can change it back to normal by typing font-weight normal. That's what it is by default. Now, let's look at this h1.

The h1, let's make this font-weight bold. Hmm, didn't do anything. Add the semicolon, still didn't do anything. That's because it's already bolded by default.

The computer wants to bold headings, h1s, because it figures it's a headline. You probably wanted this thing bold. So it makes the headline bold by default. You can reverse that and make it a normal weight instead of a bold weight. That's a great use for font-weight—pretty cool.

If you want to make something italic, it's not font-weight, it's font-style. So making something italic is font-style: italic. If you want to make it un-italic, it's font-style: normal.

Finally, color. Lots of different ways we can spec color on the web. You could spec it with words, like “black” and “red.” You don't get a lot of control, but it's easy to type in.

You don't get to choose what red it is if you type the word red. But it's nice and red.

Or you can spec it these last two ways, with RGB color triplets, like 255, 204, 0, or a hexadecimal color spec, which is this one down at the bottom that's got the pound sign and ffcc00. How do you find out what those colors are and how to spec them? Easy.

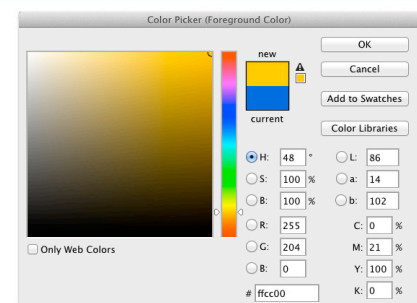
It's this in Photoshop. The color picker in Photoshop and lots of other programs, like Illustrator and InDesign, have places for RGB specs. Here's 255, 204, 0. And then, even down here in Photoshop, you can see the web color picker—#ffcc00. Mathematically, they're the exact same color. So it doesn't make any difference whether you spec it as this RGB color triplet or ffcc00. It makes absolutely no difference. They're the exact same color. And you can get those colors straight out of Photoshop and put them into CSS.

```
font-style: italic;  
font-style: normal;
```

Lesson 2 of 6: Coding for Designers

```
color: black;  
color: red;  
color: rgb(255,204,0);  
color: #ffcc00;
```

Lesson 2 of 6: Coding for Designers



Lesson 2 of 6: Coding for Designers

Now it's time for you to try it. I want you to take "Hello, World," the page that we were just working on, and experiment with each of these properties: font-family, font-size, line-height, font-weight, font-style, and color. These are examples that I have here in gray just to make them easier and more readable. You don't have to use these particular values, these particular examples.

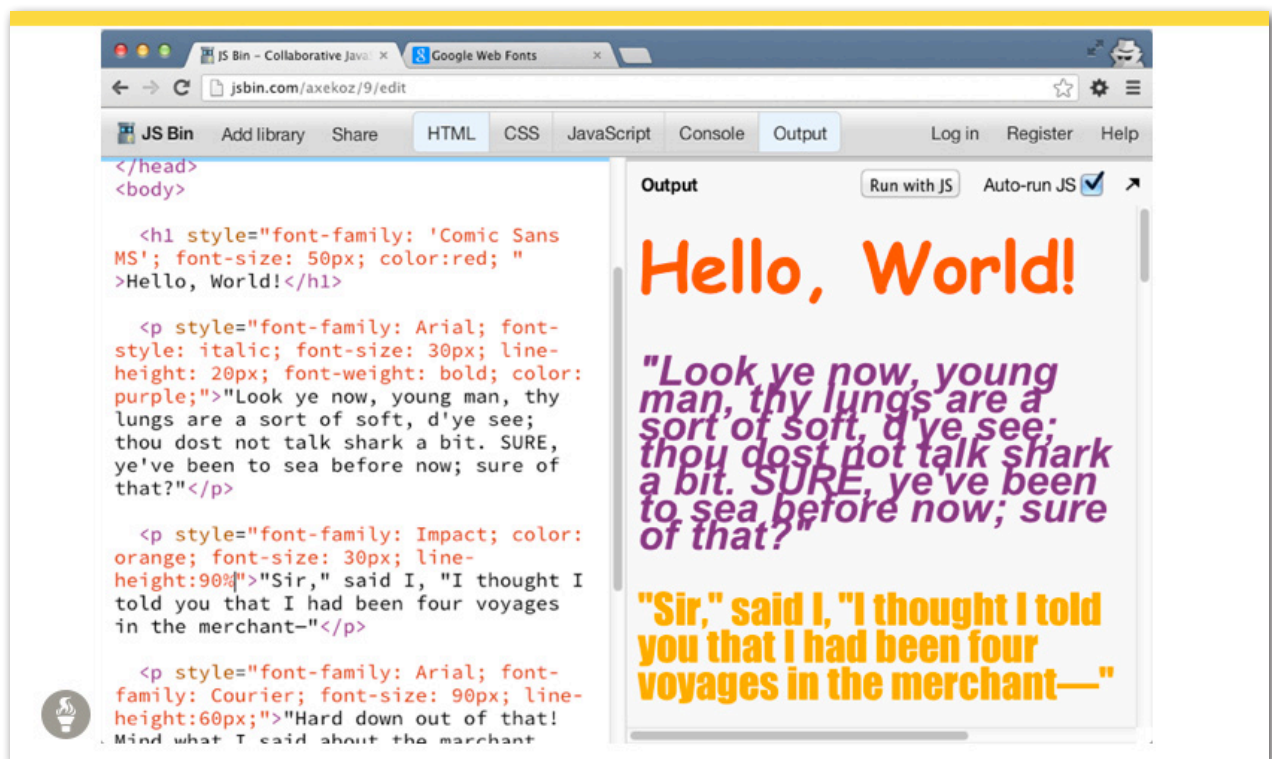
But I want you to experiment. I want you to experiment with both the headline and with each of the three paragraphs, seeing what you can do and spec-ing them in different ways.

Go ahead and pause the video. Go do that. And come on back when you're done.

I hope you had fun experimenting. If you didn't experiment, then you can see what I did with mine, below. I made it look awesome. I am a skilled designer, and this is what I do. Hee hee!

```
font-family: Arial;  
font-size: 24px;  
line-height: 30px;  
font-weight: bold;  
font-style: italic;  
color: red;
```

Lesson 2 of 6: Coding for Designers



I hope you had fun working with yours, too.

That's big idea number three, that when you're creating a web page, especially when you're starting out, you start with the words and images in HTML. And then you add the design in CSS.

## BIG IDEA #4

### COMPUTERS ARE DUMB

And finally, big idea number four, computers are dumb. When coding HTML and CSS, we have to keep track of a lot of different special symbols and characters like these: the corner brackets, quotes, semicolons, colons, equal signs. And if we don't have those in exactly the right place, where HTML and CSS expect them to be, the computer will freak out. Or at least won't do what you expect it to do.

Moreover the rules by which those symbols are required seem arbitrary and capricious until we actually understand better what they are. And small syntax errors can have big visual effects on your web page. Let me show you an example.

Here's the h1 tag, an opening tag that's correctly formed. Here's one with an extra space in the beginning, after the corner bracket but before h1. Here's one with an extra space on the other side. And here's one where I've accidentally forgotten to leave in the right corner bracket.

*(Note: this section relies extensively on live coding. Following along with the video is recommended.)*

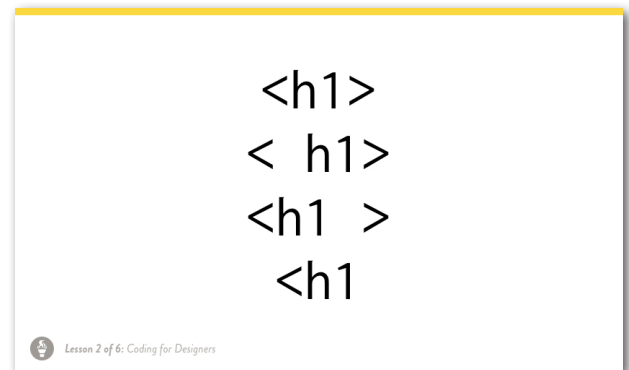
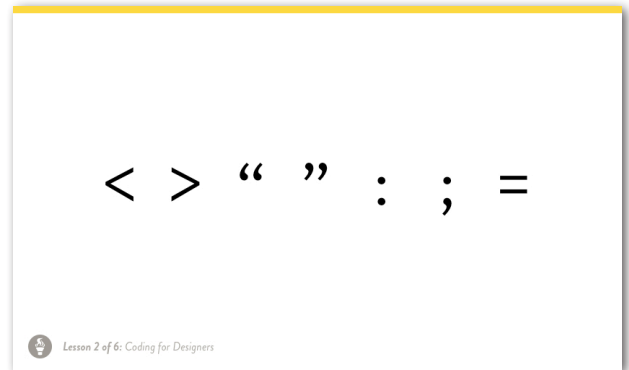
Let's try each of these in our hello world page and see what we can break. Here's hello world. I'm going to go here to my h1. And if it's okay with you I'm going to remove the style, just to make it easier to see what's happening.

So I've lost my CSS here. That's fine, it's just my regular h1. I'll insert an extra space here and see what happens? I totally lost my styling. I can now see the h1 tag but not the closing tag.

I put the space back. It's fixed. I get rid of the space, put it back, and it's broken again. This h1 has disappeared off the page. It's really weird, right?

Now if I fix it, but put a space on the other side, no visual difference. It's fine, no problem. But if I forget that corner bracket, oh the header disappears entirely. It's completely gone off the page. The other paragraphs are okay. But that header is gone.

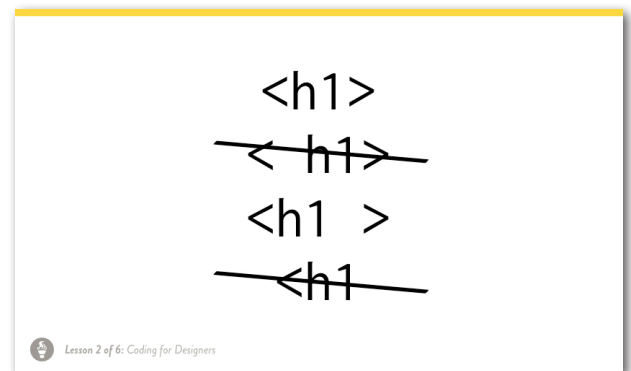
Oh I add a space, look what happens. Some crazy stuff happens. Things are all kinds of crazy. Things got really big. We're not quite sure why they got big because those paragraphs aren't styled in any sort of big way. But they're huge.



We replace that single character and everything snaps immediately back into place. It's crazy, right? How one single character could make such a big visual difference. These are the different h1s we tried. And of all four of these, this one and this one did not work. They broke the page in a significant way. But the other two are okay.

The first one's okay because it's the proper form. And the second one is okay because that extra space is where you might put the word style and that CSS stuff. Remember how we split that opening tag open to put in our CSS? So spaces on that side are okay.

Regardless, small syntax mistakes in your HTML and your CSS can have big visual effects. Thankfully, there's a little bit of help that JS Bin and other text editors like JS Bin provide. They provide it through coloring the code.



So here on my page you can see that the code on the left hand side has a color scheme. There's purple and orange. And the regular text on the page is black. That's put in automatically by JS Bin. I didn't put that in myself. The system puts that in automatically. And that color coding can give you a clue when something is broken.

Watch what happens when I take my first paragraph here, the one that says, 'look ye now young man'. And if I were to take that paragraph and forget to put in the right corner bracket on that p tag, see what happens. The paragraph disappears entirely. It's gone. But on the code side, you see how it all turned red? That's a clue to me. That red text is a clue to me, something ain't right. That text should be black on the code side. But it's not. Something's icky.

And sure enough, if I go in there, I might eventually figure out that I'm missing the corner bracket. I put it back in. And I'm good to go. If I forgot a quote say. Well, that screwed up some stuff too. But you see how everything here now turned a weird color. All three paragraphs on the code side turned a strange color. I can trace back up to where the color started looking strange, and put the quote back.

Now it's not going to fix everything. See if I'm deleting tags here, and some of this stuff should be breaking the page. And in fact, there, that does break the page. I lost my styling there. The color is not changing.

So, in some cases, I've got HTML or CSS mistakes here, but the code color's not changing. So it's not perfect. The color is not an absolute verification that something is right or wrong. But it is a clue, a clue that you can use if something might be broken.

The bottom line is, computers are dumb. We have to be smart enough and detail-oriented enough to make up for their shortcomings.

## CONCLUSION

That concludes today's lesson. Got some homework for you.

First assignment, a short quiz to test your retention of this material and to help reinforce it.

Then, second assignment: I broke some web pages, and I need your help. Can you fix them for me? These pages has syntax errors, and they have problems galore. I need you to go in and fix them. You'll have to sleuth your way around to figure out what's wrong.

If you blow through that assignment, or if you want some extra work, the extra credit assignment is to make a cocktail recipe. I give you the cocktail, you format it into a great looking recipe. We'll show you how in the classroom.

In the next session we'll apply more design to our text. We'll start looking at page layout and we'll get deeper into typography and color.

Again, if you have questions or you'd like to talk to us, we'll see in the forum. Otherwise, see you in the next session. Have a great rest of your day!



