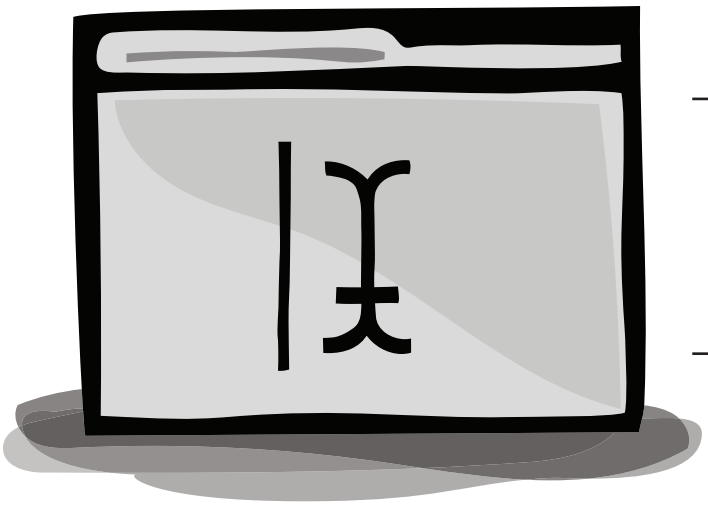


GYMNASIUM

INTRODUCTION TO MODERN WEB DESIGN

Lesson 1 Handout

Getting Your Bearings

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

CORE CONCEPTS

1. This course will teach you how to write solid HTML and understand how CSS works. You'll also learn how to adapt a design to different screen sizes—what's referred to as responsive web design—and discover the basics of working with JavaScript and project in which you create a JavaScript dropdown navigation menu.
2. The role of HTML is to structure or "mark up" content within a document. CSS is a styling language that allows you to apply various enhancements to HTML such as color, typographic effects, and much more. Adding interactivity to a web page is accomplished using JavaScript, a scripting or programming language.
3. The term *front end* typically refers to part of a website's architecture, in particular, the languages HTML, CSS, and JavaScript. The term *back end* typically refers to the programming or database components of a web server such as the database or the content management system. Front-end developers and back-end developers are typically two different people, although the term "full-stack developer" has emerged in recent years as someone who is comfortable in both areas, although these individuals are rare (and sometimes called unicorns!)
4. A typical list of roles that a front-end developer might interact with on a medium-to-large size website project would be content strategists, information architects, UX designers, visual designers, back-end developers, and project managers.
5. There are two main philosophical approaches when it comes to building websites: graceful degradation and progressive enhancement. Graceful degradation in web design generally focuses on providing the best experience for modern browsers first and then going back to older browsers, evaluating what features can "degrade" and then making the necessary fixes so users do not receive major errors. Progressive Enhancement starts with defining the core experience for older browsers first and then adding additional features (or enhancements) for modern browsers. Some of the primary benefits of progressively enhanced websites are increased accessibility, browser support, easier maintenance, and future-proofing your application. This course focuses on a progressive enhancement approach.

ASSIGNMENTS

Your final goal for this course is to create a simple, multi-page website using content from a Wikipedia article (of your choosing) as the source.

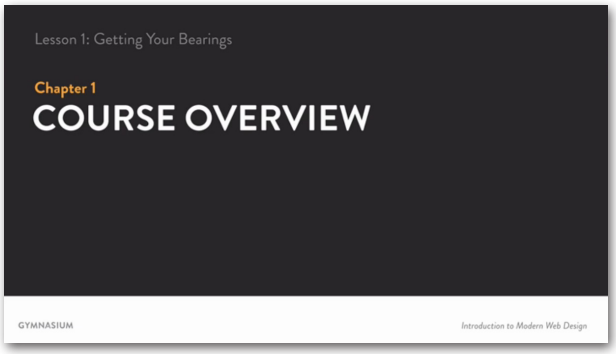
To help you visualize the final project here, be sure to explore http://aarongustafson.github.io/i-heart-cuttlefish/ which is the website the instructor uses throughout this course as an example.

Each lesson will have one or more assignments to help you structure your work. After you complete an assignment you should post a link to your work on the Gymnasium forum for feedback.The assignments for this lesson are as follows:

- You will need a way to host your HTML, CSS, and JavaScript code online. We recommend using GitHub and specifically GitHub Pages to do this. First, sign up for an account at GitHub.com if you don't have one already. Then go to pages.github.com, and follow the tutorial on how to use GitHub Pages for hosting (you will want to follow the instructions for a Project Site.)
- The next assignment is to decide what subject you want to use for your website. In order to do this, go to Wikipedia and find an article with enough content to fill at least five pages and have a handful of images. Feel free to use the instructor's Wikipedia article on cuttlefish as a reference, but we highly encourage you to take the time and choose a subject of your liking, you will be working with the material for 6 lessons, so it should be material you are interested in!

INTRODUCTION

(Note: This is an edited transcript of the Modern Web Design 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.)



Welcome to Introduction to Modern Web Design, an online course developed by Aquent. My name is Aaron Gustafson, and every day I work to make the web a better place for users and developers. I've been working on the web in varying roles from junior web designer to manager of all things front end for over two decades. Toward the beginning of my career, I even worked as an independent contractor through Aquent.

Over the years, I've had the great pleasure of working with a broad range of clients and brands, including *The New York Times*, Guinness, Nestle, and American Idol. I'm currently a web standards and accessibility advocate at Microsoft, and I want to thank you for joining me today.

Now, the purpose of this course is to get you up to speed with what it means to be a modern front-end web developer. So, throughout this course we're going to be talking about what a web developer does, how to mark up content using HTML, how to design pages using CSS, and how to use JavaScript to enhance web pages.

By the end of this course, I would expect that you'll know how to write solid HTML, you'll understand how CSS works and the options it provides, you'll know how to adapt a design to different screen sizes—what's referred to as responsive web design—and you'll know a little bit about JavaScript and the basics of how to read and write it. And, perhaps most importantly, you'll understand what goes into building a website and how to work as part of a web design team. Now, I'm sure you know what they say about making assumptions, but I have a couple about you.

One, you aren't afraid to look at code. Two, you've surfed the web before. Three, you have a text editor on your computer. (We'll talk about what a text editor is in a minute.)

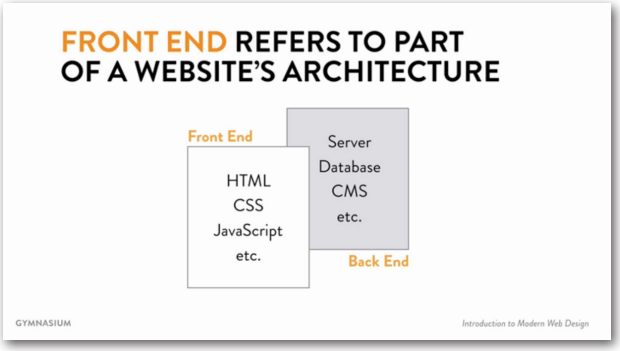
And four, you know what a web browser is and have one. Typically, that would be a prerequisite for number two, but I just wanted to put that out there as a separate assumption. Throughout this course, we'll be working on a small website borrowing content from a Wikipedia article that interests you.

I'll get into the specifics of that down the road, but for right now if you wanted a take a peek at the site that I've put together as part of this course, you can look at it at aarongustufson.github.io/iheartcuttlefish. And you can view it either on your desktop browser or on your mobile device. I hope you'll join me in the second video of this series, where we'll discuss what it means to be a front-end developer.

WHAT IS A FRONT-END DEVELOPER?

I'm glad you decided to stick around. As I mentioned in the introduction, this course will cover modern web design from the perspective of a front-end developer. So the first thing I should probably cover is what the heck is a front-end developer? But before I get there, what the heck is front end?

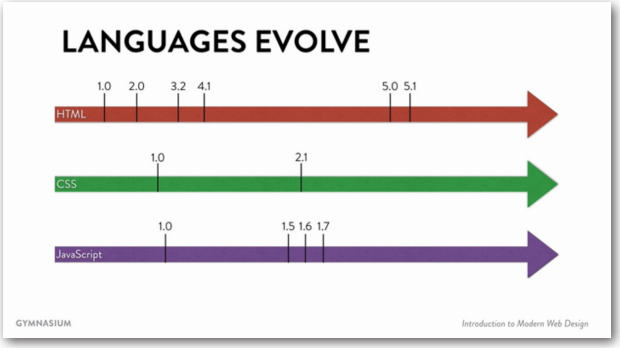
So when we use the term front end, we're referring to part of a website's architecture. And front end technology is stuff like HTML, CSS, and JavaScript. And we use the term front end to separate it from back end, which is the stuff that makes the web page. In all cases, the website is being served by a server, but sometimes we've got things like databases which can store your content in them. Perhaps they're storing them as part of a content management system or CMS. That sort of stuff, the heavy-duty programming, that's considered the back end, the stuff that happens on the server. The front end is what we deliver to the browser, and it's what the browser renders and interprets.



Front end can also refer to the people whose focus it is. So sometimes you'll hear people say, oh, I work on the front end or I work on the back end or they might say I'm a front-end developer or I'm the back-end developer.

Some people do work on both ends of the system, especially at smaller companies and startups. You'll sometimes hear them referred to as full-stack developers, because they understand all of the layers that come together to create a website.

And those people are kind of rare, which is why they've been given the moniker "unicorns" by some people. They understand design, UX, front-end coding, back-end development. And they're highly sought after at a lot of companies. If you can master all of these areas, you can pretty much name your price when it comes to doing web development.

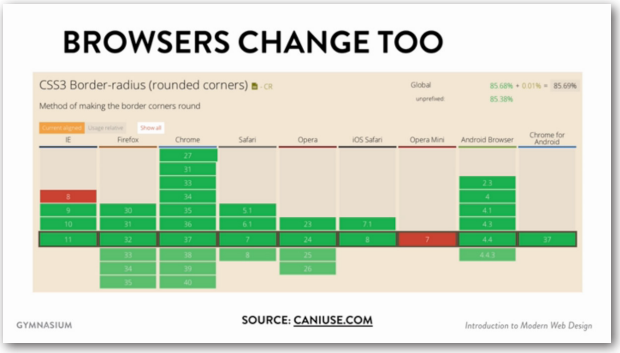


Now, the front-end languages, as I've mentioned before, are HTML, which is used to mark up your content; styling, which is accomplished via CSS; and your programming on the front end is done using JavaScript.

Now, it's important to realize that HTML, CSS, and JavaScript actually evolve over time. Each language evolves at its own pace and so different versions come out at different times. And they're not always in sync. This can be a little confusing and can make web design a little bit challenging, trying to track which version different browsers are handling. And in many cases, browsers change over time as well. They won't implement a particular spec, but then later on they will.

As you can see, IE8 all the way over on the left here in red, doesn't support the CSS3 border radius property. All of these other browsers that are in green do. You'll notice that IE9, IE10, and IE11, which come after IE8 in that column, also support border radius. This information is all stored on a really useful website called caniuse.com, which helps you to track which options are available to you in CSS, HTML, and JavaScript and which browsers support them.

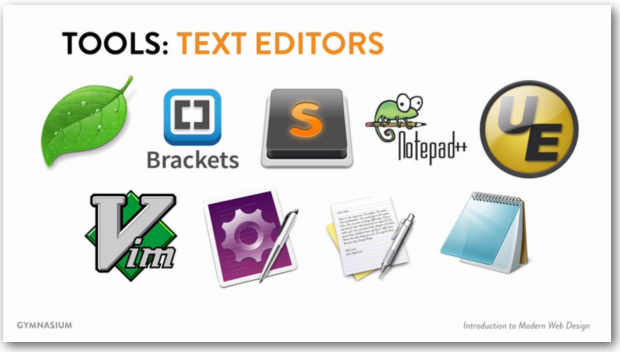
There's also a really neat view where you can actually see the amount of market share a given browser has in order to best determine whether or not a particular feature is available.



So as I mentioned, browsers change over time. So you might have some browsers that get square corners because they don't understand the border radius property. So that would be IE8 here on the left. And then on the right we have IE9, which does understand border radius. So it gets the rounded corners.

Some of you may look at this and think, well, that's not great. We've got two different browsers with two different behaviors but, in reality, it's OK for people to get different experiences based on the browser that they're in. We can't control absolutely everything about the way people experience our websites, and that's OK. We have to come to terms with that. And it's a subject that I'll talk about a bit more when I discuss the philosophy of progressive enhancement.

Now, at their most basic, the tools of a front-end web developer are simply a text editor and a browser. There are numerous text editors out there, everything from Coda and Brackets to Sublime Text, TextMate, Vim, UltraEdit, Notepad, Notepad ++. There's just hundreds of them out there. Which one you want to use is up to you. My personal preference is TextMate. I tend to use that a lot on my Mac, but I also like Sublime Text quite a bit as well. And that one is available on both Macs and Windows machines.



As for browsers, we'll be viewing and testing our work in Google's Chrome browser, but most front-end developers have a handful of browsers on their machine so that they can test their designs and their functionality across the board.

Hopefully this helps you understand what a front-end developer is. In the next chapter, we'll examine where a front-end developer fits into the process of building a website. I'll see you in a minute.

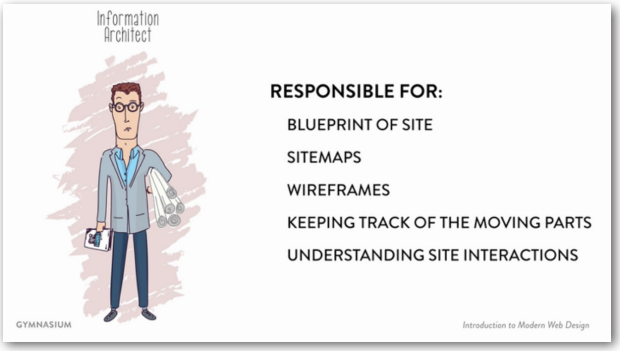
WHERE FRONT-END DEVELOPERS FIT IN

Welcome back. So now that you know what front end is and a little bit more about what a front-end developer does, let's talk about how and where this role fits into the grand scheme of a project.

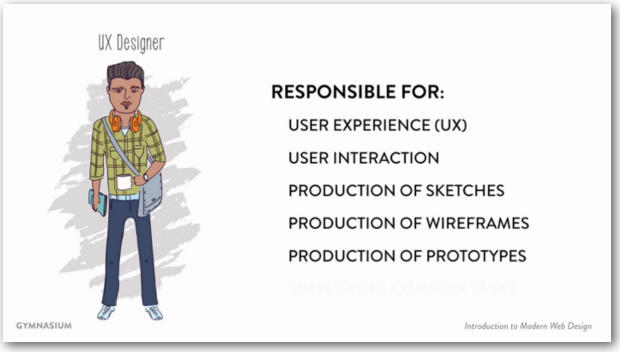
The people in our neighborhood when it comes to web design are content strategists, information architects, UX designers, visual designers, front-end developers—hopefully you in not too much time—back-end developers, and project managers. Let's look a little bit at what each of these roles does.

A content strategist looks at the content of a site—the words, the images, the videos—and acts as an overall editor for the project. She might peruse the content to ensure stylistic consistency from page to page, or she might evaluate the content needs of the site. The content strategist puts together authoring guidelines for future content and also establishes a schedule for when content might need to be updated. In many ways, the content strategist is looking at the big picture in terms of what message the site is trying to get across, and how do we do that most effectively.

The next role I want to talk about is the information architect. The information architect typically works closely with a content strategist. They may even be the same person in some instances. And they work to build the blueprint for the site. Often, they'll create documentation like site maps or wire frames, and they take a look at all of the different moving parts and kind of architect how those different pieces fit together and how they work together.



The user experience (UX) designer is focused on how the user interacts with a site. This role can overlap a bit with the information architect and the visual designer, so it may be rolled into one or both sides of that. And depending on the skills this person has, they could be generating drawings, wire frames, graphics, or even full-blown prototypes. Regardless, the point of a UX designer's work is to make complex tasks easier and more intuitive for your users.



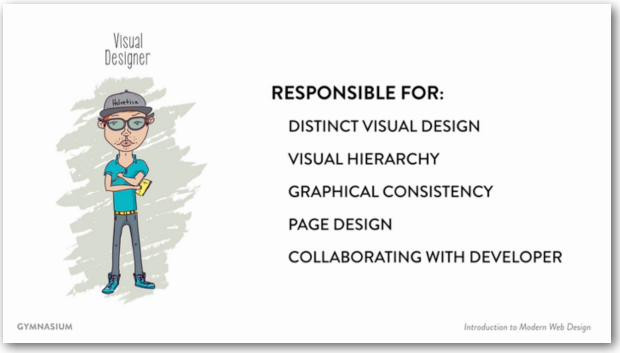
The project's visual designer is tasked with taking the blueprints and the other sorts of artifacts that are created for the site and to then create a distinct visual design for that project. The visual designer establishes visual hierarchy, enforces consistency, and works to guide users naturally through each page, drawing attention where it's needed using tools like proportion, contrast, and white space. A visual designer might produce a series of page designs, or better yet, a design system like you're seeing right here, comprised of headings, paragraph text, and other sorts of content in order to make the front-end developer's job easier.

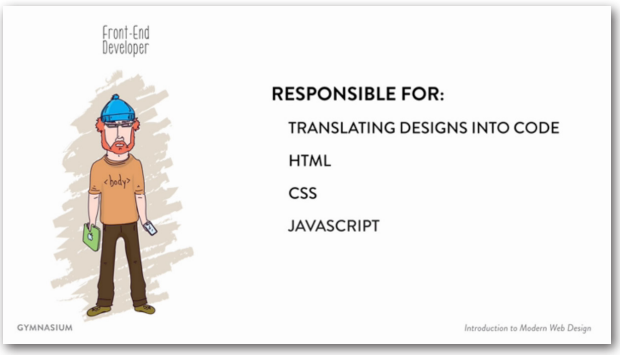
Now, the front-end developer—that's you-—are where the rubber meets the road in terms of producing a website. Your task is to translate the visual and interaction designs into production-level code, which means HTML, CSS, and JavaScript. Don't worry if all of that doesn't make sense to you quite yet. By the end of this course, it absolutely will.

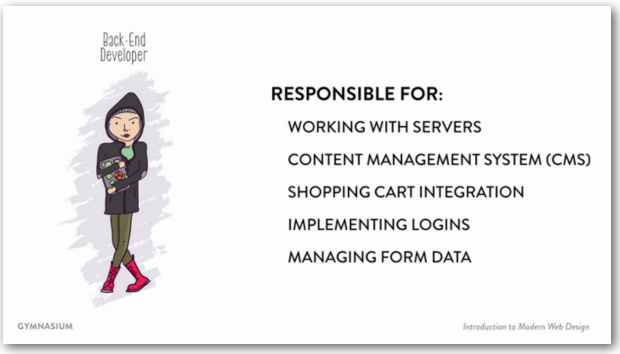
Now, working beside you is the back-end developer. The back-end developer may be the person in charge of the servers, or they might be the person who integrates your HTML into some sort of content management system or other server-side programming language. They'll probably be the person building things like shopping carts, login systems, forums, and the like.

And finally, the glue that holds the whole project together is the project manager. This is the person who is tasked with organizing the project in terms of schedule, team, budget, and all that sort of stuff. And they have the thankless job of managing things like deadlines and project scope. A good project manager will keep a project on track and ensure everyone walks away feeling good about the results, as well as the process.

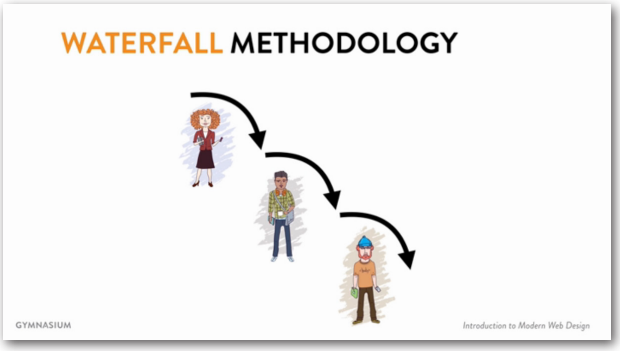
We're going to start digging into HTML in the next lesson. But before we get there, I want to talk a little bit about two ways in which teams work together.



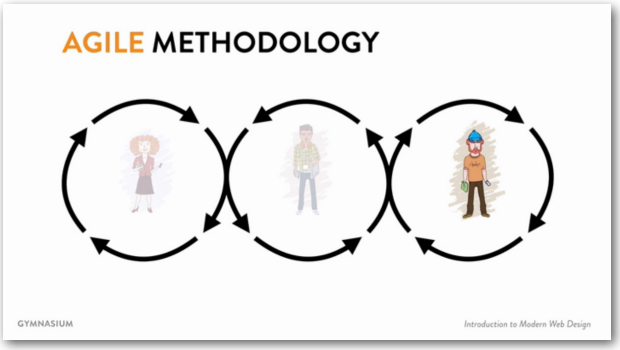




The first methodology I want to touch on is the waterfall methodology. And in the waterfall methodology, one person finishes their part of the project and then hands it over to the next person in the process, and then that person finishes their work and hands it to the next person, and on and so forth. Hence, waterfall. Most large organizations use this process, because they've used it for years. It's very similar to the factory model of an assembly line.



Now, an alternate approach to working together as a team is what's called agile methodology. Basically, what agile means is that you're iterating and working together. So rather than one person doing their task, finishing it, handing it to the next person, and running away, the different people on the team actually work together and iterate on a given page or a given interface or a given problem in order to try and figure out the best solution for it. And that iteration happens over and over and over again.

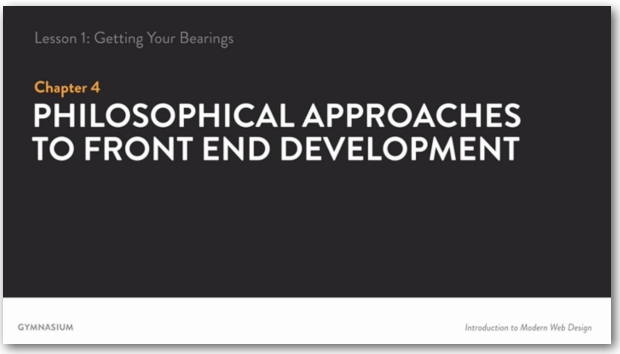


Along with this methodology, you'll often hear terms like "fail fast and fail often," because it's through failing that we learn from our mistakes and then adapt.

And that's it for this chapter. In the next chapter, I'm going to dive into a couple of different philosophical approaches to front-end development. I hope to see you in a few minutes.

PHILOSOPHICAL APPROACHES TO FRONT-END DEVELOPMENT

Welcome back. So far we've learned a bit about what front-end development entails and where it fits into the project. But before we jump in the code, it's worth taking a break to discuss philosophy. The philosophical approaches to front-end web development can really be boiled down into two major camps. The first of those is graceful degradation.



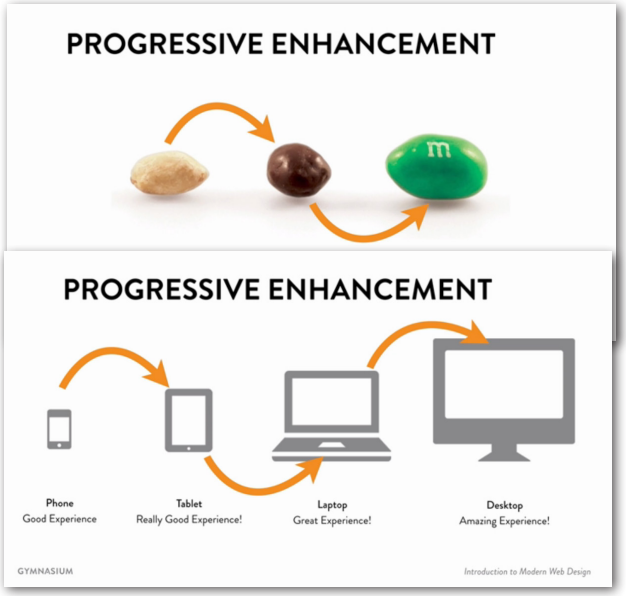
This concept came to web design from the engineering world, and the general idea was that you could provide a degraded or lower quality service, or no service to some people, as long as you don't cause any major errors. Following on that concept, the web design community focused our time and our energy on modern browsers and paid little to no mind to older browsers, often leaving testing on those browsers to the very end of the project and fixing only the most egregious errors. If a browser proved particularly problematic, we might outright block it and present the user with a message telling them they needed to use a different browser.

Trust me when I say you don't want your customers feeling like this little kid. There are so many instances in which a user has no control over the browser they use. It could be because of ignorance or financial means or because their company has a browser lock down and they can only use that corporate standard one. To deny users access to content or the ability to accomplish key tasks, like viewing their bank account simply because their browser is not the one that you want them to have, is just plain rude. In some cases, it might even be grounds for a lawsuit.

Anyway, round about 2003, Steve Champion of the Web Standards Project proposed a new way of approaching web design. And he dubbed it "progressive enhancement." The idea behind progressive enhancement is that you're creating a website that doesn't have any technological restrictions. In other words, you are starting with the baseline and enhancing the experience based on the capabilities of the browser. It's about focusing on the content and the core tasks and then expanding out and adding more enhancements as you have the ability to do so.

If you're a comedy fan like I am, Mitch Hedberg has a great quote, "I like an escalator because an escalator can never break, it can only become stairs." In many ways, progressive enhancement is the same concept. Here we have, again, the square IE8 version and the rounded corner IE9 version, and this would be a progressive enhancement. The IE8 browser doesn't understand rounded corners; IE9 does, so IE9 gets the rounded corners and IE8 doesn't. And that's all right.

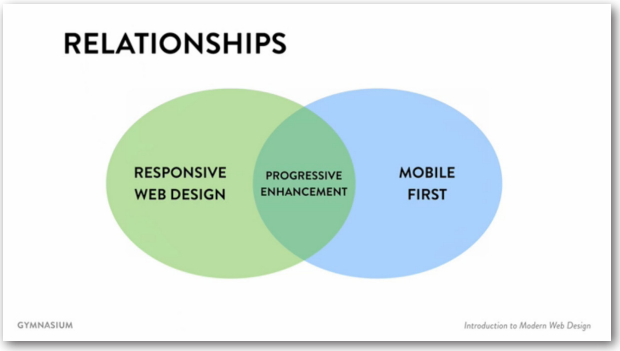
Now, I'm a big proponent of progressive enhancement, and I like to use an analogy of the continuum from a peanut to a peanut M&M. Here on the left we have the peanut, which is a rich and tasty treat. And as long as you don't have a peanut allergy, it's a perfectly acceptable snack. If you wrap it in chocolate, now all of a sudden you have what we refer to—in the States at least—as a goober. And a goober is so much better than just a plain peanut because it's got chocolate on it.



Now, when you add the candy shell to turn it into a peanut M&M, it becomes amazing. But the thing to keep in mind here is that any of these are valid snack options. But in the steps along the way, moving from the peanut to the peanut M&M, we've created better and better experiences for the people who are eating them. In the same way, we should strive to create better and better experiences for people, based on their browser capabilities or device capabilities.

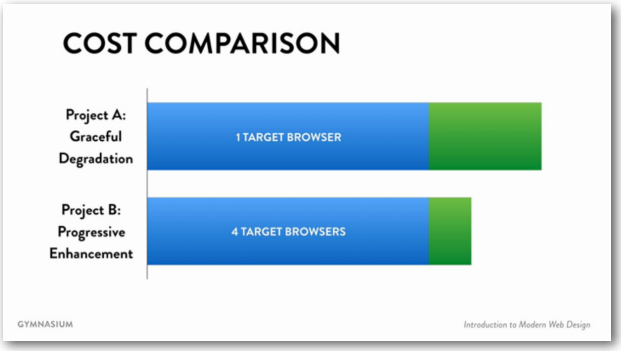
Now, these two philosophies are actually intrinsically related because progressive enhancement is actually a very specialized form of graceful degradation. All progressively enhanced websites will automatically degrade gracefully. The same cannot be said of all sites that gracefully degrade. Not all of them are progressively enhanced.

Andrew Wight tweeted that “progressive enhancement is an escalator that becomes stairs,” whereas, “graceful degradation is building a lift and then having to add stairs when it breaks.” And I think that’s a pretty good upgrade of Mitch Hedberg’s analogy. Now since we’re talking about relationships here, I’ll also bring up that when you’re looking at two concepts of responsive design and mobile-first, where the two of those intersect is actually progressive enhancement as well, because responsive design, approached from a mobile first standpoint, is progressively enhancing the visual design of a site, based on the amount of screen real estate available to the user.



Now, the last thing that I want to share with you with respect to this chapter is a little bit of a story. So here we have two different projects that my company worked on, the first of which we were specifically instructed to use graceful degradation as our philosophical approach. In the second, we were asked to use progressive enhancement as our philosophical approach.

In the first, we were asked to target just one single browser, and that was the Chrome browser, because we were building a Chrome app. In the second project, we were targeting four specific browsers. Now once these projects were completed and had been out on the Internet for awhile, each of these clients came back to us to upgrade the site in order to add more browser support.



In the case of the first project, they wanted to add two new browsers. In the case of the second project, they wanted to add 1400 new browsers. Now, when we went to do this, what we discovered was that with the graceful degradation approach, it actually took us 40 percent of the original budget to add support for two new browsers, because we had built it solely expecting to have that single, one browser.

In the case of Project B, however, we had built with progressive enhancement in mind, so we were automatically able to reach more browsers with less headaches. So we had given them a budget that was about a third of the original project, and we ended up coming in at half of that. So for one-sixth of the cost of the original project, we were able to add 1400 more devices to their support matrix, which is pretty awesome. Progressive enhancement just works.

Now, throughout this course, I’ll be approaching things from a progressive enhancement standpoint because that really is the gold standard for how you should be building on the web today.

Well, this wraps up Lesson One of the course. And so, I've got an assignment for you. In order to ease the process of putting your files online, we're going to use GitHub. And so what I want you to do is actually create a GitHub page and get that all set up in order to be able to post your content and share that in the forum. You can get an account at GitHub.com, and if you go to pages.github.com, they have a complete walk-through on how to use GitHub Pages for hosting.

The other assignment I have for you is to figure out what the subject is that you want to build your site about. And I want you to look specifically on Wikipedia for that content. I want you to look for enough content on a subject to fill at least five pages and have a handful of images. You could do a project on whatever you like. I particularly chose to look up "cuttlefish", and I found a treasure trove of content there about cuttlefish, a bunch of different sections that I could use as pages, and a bunch of different images and even some videos.

As I mentioned earlier in this lesson, my site is up and available at aarongustafson.github.io/i-heart-cuttlefish, and you can view that on your desktop browser, or even on your mobile device. Well, that wraps it for Lesson One. I hope to see you in Lesson Two, where we'll start digging into HTML. See you soon.