Build Your First Web App

Hi there! Welcome to HubSpot Academy's Building Your First Web App course. I'm Zoe Sobin, and I'm an Engineering Lead here at HubSpot.

This course is based on a popular in-person workshop we developed a few years ago here at HubSpot. The in-person workshop teaches women in college computer science programs how to build full, working web applications, using the most widely available tools and technologies. Now, we're excited to be bringing this workshop to the web.

This course assumes a working knowledge of basic programming concepts. But we do explain the concepts thoroughly and take things one step at a time, so no matter your skill level, we welcome you to join us for the ride.

During these lessons, there are a number of resources and links we'll mention. You can find each of those in the "Resources" section below. If you'd like to follow along with written instructions as well, you'll find those in the Resources section, too.

Let's start with an introduction to how the internet works and go over vocabulary you'll be hearing in the course.

Okay, so we're going to learn about building a web application, or in other words, a web app. But first, what is a web application? Well, it's a collection of code that's stored on a remote server (meaning, not on your computer, but rather a computer that's located somewhere else) and delivered over the internet through a web browser. Let's cover each of those things in more depth.

Let's say you open a browser, go to Google, and search for "cute cats." When you click the link to a specific URL in the browser, you're making a request. The request gets sent to a server — which is just a fancy name for a computer. Then, the server returns a response — in this case, the code that makes up the website.

The URL is the web address of an online resource, like a website or document. Each URL consists of a protocol, server name, and URL path. The protocol declares how your web browser should communicate with a web server when sending or fetching a web page or document. It would be like us saying "English" before every sentence you spoke. "HTTP," which stands for "Hyper Text Transfer Protocol," is the most common.

"HTTPS" is its security-focused counterpart. The "S" means that the browser is encrypting the data it's sending, so it's much more secure for things like passwords or credit card information. The

server name is a unique identifier – it maps to the IP address of the server. The URL path tells you where, exactly, the file you're looking at is stored on the server.

We'll often talk about "servers" and "clients." The server stores and serves information (like your application). Clients are all the devices requesting that information, like your laptop or phone. The server constantly listens for incoming connections, accepts those connections, parses requests, and sends back responses to the client.

Applications can be served in different environments, which might have different configurations and are used for different purposes. Applications you visit every day are in production environments – they're ready for traffic, secure, and (hopefully) stable versions of the application. But delivering software is not an easy task, so software engineers will often use QA (or quality assurance) environments to test their application on the server before it goes users. But most of the time, when engineers are working on improvements or fixing problems, they use a local environment. This lets them use their personal computer as a server to run applications.

Today, you'll be using mostly the local environment to develop your app, but you'll eventually push your code to a production server. That production server is managed by a company called Heroku. Heroku is a cloud platform, meaning that the server exists in a data center somewhere. Once you push your code, it'll live on that server, and anyone in the world will be able to access your web app via a URL. We chose Heroku because you get a lot of great features for free. Their service is very popular, so there are lots of resources out there to help you if you get stuck.

To get an idea of what kind of code you'll be storing on this server, let's again look at our example web application. What you see here is the front-end of the application. The front-end is the user interface, or everything you can see. The browser interprets the front-end code – made up of HTML, CSS, and JavaScript – and displays it on the screen. The back-end is the code that's invisible to the users. It's the part that stores, transforms, and serves data.

HTML, CSS, and JavaScript – the front-end code we just mentioned – work together in harmony to make beautiful, dynamic web pages. We like to think of them as the "anatomy" of a web page.

Using this example, the HTML is like the skeleton of your application. It makes up the structure of the page by creating a way to put elements like headers, buttons, links, and paragraphs on the page.

JavaScript is like the muscle of your application. It's what makes your page dynamic by creating a way for the page to do things based on user actions.

Finally, CSS is like the skin of your application. Default HTML doesn't look great, but CSS provides a way to make your application beautiful by adding different colors, fonts, and spacing.

These are the three languages we'll focus on in this course.

As we mentioned before, the back-end of an application deals with data. Most applications have some data to store – it could be your Amazon orders, the songs in your favorite playlist, or a list of recipes. This data is stored in a tool called a database, which is an orderly way of storing all your app's data. Here's an example of a database for a music playlist – you can see that each column stores a type of information, and each row stores a song in the playlist.

To reach that data, we use API, which stands for Application Programming Interface. APIs give different access points to different pieces of data. With APIs, you can incorporate data from an outside service, or retrieve and modify data in your database.

We can make these API requests using JavaScript. For example, if we wanted to get some data from an API, we can make a GET request using http (the same protocol that allows us to retrieve a web page). The server will then send back a response in the form of JSON – or JavaScript Object Notation – an organized format that allows you to see values mapped to types of information, like in the weather information here.

If this seems overwhelming, don't worry. We'll dive into these concepts in more depth later.

Finally, let's go over some of the tools you'll be using to create your web app. Node.js is an open-source server environment. This means that you'll be able to use JavaScript to run your application on a server. While you can do this through many other programming languages, the advantage of using Node is that you don't have to learn an additional programming language – it's all written in JavaScript.

You'll also be using a framework called Express.js. What's a framework? Well, think about it as a group of common code pieces and popular solutions that can be reused across different applications. It can also include recommended solutions or best practices. Express provides a set of features for creating web and mobile applications. It provides lots of tools for creating pages and letting users move throughout your app, saving you the trouble of having to write it yourself.

We chose Node and Express for this workshop because they're quick to set up and very popular. In fact, Node.js's package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

In this workshop, we'll be starting from a basic web application rather than building a Node and Express app from scratch. But if you'd like to dive into the nitty-gritty of how these technologies work, a Google search will turn up lots of resources.

You'll also be using tools called git and GitHub.

Git is a free and open source distributed version control system. This means that git makes it easy to track changes to files. For example, when you edit a file, git can help you determine exactly what changed, who changed it, when the change occurred, and why. The versions created in git with every code change makes it easy for multiple people to work together on the same code base. It's also useful for tracking progress over time by saving "checkpoints" in your code.

You can work with git using your computer's command line, but git also has visual interfaces. GitHub, a web-based git repository hosting service, is the most popular. Since GitHub is a web application, you can use it to easily share your code, like if you want to someday show a recruiter your portfolio. You can also use GitHub to find and contribute to other developers' projects, including thousands of open source projects.

Along with GitHub, you'll use your computer's terminal to interact with your app. The terminal is a very basic way to communicate with applications – you use simple lines of text to run specific commands. You'll be using the terminal today to run and deploy your app, as well as to use git.

Finally, you'll use an editor when you're writing or modifying your application's code. There are many code editors out there. In this course, we'll be using Visual Studio Code, but feel free to use whatever editor you like best.

With this background, you're finally ready to get started building your basic web app! Here's what to expect in the rest of this course:
First, you'll build your web app.
Next, You'll build out additional functionality for your app. For this stage, we'll take a deeper dive into what Node and Express do. You'll add pages to your app and learn how to add CSS to make your app more aesthetically pleasing.

Then you'll make your app more dynamic by integrating it with an external API.
And finally, you'll set up a database. You'll send and store user-generated data, as well as fetch and display the contents of your database in your app. Once you complete the first lesson in this course, you don't need to do the other lessons in order. Only do the ones you find most useful.

Good luck building your first web app!