IN THIS CHAPTER

- Learning about web apps
- Figuring out your app's functionality
- Determining your app's data requirements
- Planning your app's workflow
- Visualizing your app's interface

Chapter 1
Planning a Web App

What you can do, or dream you can, begin it,

Boldness has genius, power, and magic in it.

— JOHANN WOLFGANG VON GOETHE

There are many reasons to get and stay interested in web coding and development. Here are a just a few: the challenge of learning something new; the confidence that comes from figuring out hard or complex problems; the satisfactions that inhere from getting code to work; the desire to get a job in web development; the feeling that you're operating right at the leading edge of the modern world. These are all great and motivating reasons to code for the web, but there's another reason to dive deep into CSS and JavaScript and all the rest: as an outlet for your creative side.

Sure, anybody who learns a bit of HTML and a few CSS properties can put up pages of information, but as a full-stack web developer who also knows JavaScript, MySQL, and PHP, you have all the tools you need to create bold and beautiful apps for the web. That's where the real creativity lies: having a vision of something cool, interesting, and fun and then using code to realize that vision for other people to see and use. This minibook helps you unleash the right side of your brain and make your creative vision a reality by showing you a few crucial upgrades to your web coding and development skills and know-how. First up: the all-important planning process.

What Is a Web App?

If you go to the web home for a company called Alphabet (`https://abc.xyz`), you get a general introduction to the company, plus some information for investors, news releases, links to corporate documents such as the company bylaws, and so on. But Alphabet is also the parent company for some of the web's most iconic spots:

- **Google (`www.google.com`):** Search the web.
- **Gmail (`https://mail.google.com`):** Send and receive email messages.
- **Google Maps (`https://maps.google.com`):** Locate and get directions to places using maps.
- **YouTube (`www.youtube.com`):** Play and upload videos.

What's the difference between the parent Alphabet site and these other sites? Lots, of course, but I think two differences are most important:

- Each of the other sites is focused on a single task or topic: searching, emailing, maps, or videos.
- Each of the other sites offers an interface that enables the user to operate the site in some way. For example, Google has a simple search form, whereas Gmail looks like an email inbox and offers commands such as Compose and Reply.

REMEMBER

In other words, the Alphabet home is a basic website that's really just a collection of documents you can navigate, whereas the likes of Google, Gmail, Google Maps, and YouTube are more like the applications you use on your computer. They are, in short, *web apps,* because although they reside on the web and are built using web technologies such as HTML, CSS, JavaScript, MySQL, and PHP, they enable you to perform tasks and create things just like a computer application does.

Fortunately, you don't have to have an idea for the next YouTube or Gmail to get started coding web apps. (Although, hey, if you do, I say go for it!) Web apps can be anything you want, as long as they enable you or your users to do something. If that something happens to be fun, creative, interesting, or useful, congratulations: You've made the world a better place.

Planning Your Web App: The Basics

If you're like me, when you come up with an exciting idea for a web app, the first thing you want to do is open your trusty text editor and start bashing out some code. That's a satisfying way to go, but believe me, that satisfaction dissipates fast when you're forced to go back and redo a bunch of code or restructure your database because, in your haste, you took a wrong turn and ended up at a dead end or too far from your goal.

I plea, then, for just a bit of restraint so that you can spend the first hour or two of your project thinking about what you want to build and laying out the steps required to get there. Think of it like planning a car trip. You know your destination, but it's unlikely you'll want to just get in the car and start driving in the general direction of your goal. You need to plan your route, load up with supplies such as gas, water, and food, gather tools such as a GPS, and so on. To figure out the web-development equivalents of such things, it helps to ask yourself five questions:

REMEMBER

- What is my app's functionality?
- What are my app's data requirements?
- How will my app work?
- How many pages will my app require?
- What will my app's pages look like?

The next few sections go through these questions both in a general way and more specifically with an app idea called FootPower!, which is a simple app for logging and viewing three foot-propelled activities: walking, running, and cycling.

What is my app's functionality?

The first stage in planning any web app is understanding what you want the app to do. You can break this down into two categories:

- **User functions:** These are the tasks that users performs when they operate whatever controls your app provides. The standard four tasks are given by the unfortunately named CRUD acronym: creating, reading, updating, and deleting.
- **App functions:** These are tasks that your app performs outside the interface controls. Examples are creating user accounts, signing users in and out, handling forgotten passwords, and backing up data.

For FootPower!, here's a list of the user functions I'd implement:

- Creating new activities, each of which records activity details such as the type of activity and the activity date, distance, and duration
- Viewing previous activities, with the capability to filter the activities by date and type
- Editing an existing activity
- Deleting an activity

Here are the app functions I'd implement:

- Creating users
- Verifying new users by sending a verification email
- Signing existing users in and out
- Maintaining users' app settings
- Handling forgotten passwords
- Deleting user accounts

What are my app's data requirements?

Web apps don't necessarily have to use a back end. If your web app is a calculator, for example, you'd need to present only the front-end interface to the user; no back-end database or Fetch API calls are required. But if your app requires persistent data — which might be data you supply or data created by each user — you need to store that data in a MySQL database and use Fetch API calls to transfer that data between the browser and the server.


REMEMBER

Before you load up phpMyAdmin, however, you need to sit down and figure out what you want to store in your database. Web app data generally falls into three categories:

- **User data:** If your app has user accounts, you need to store account data such as the username or email address, the password, profile settings, and site preferences.
- **User-generated data:** If your app enables users to create things, you need to save that data so that it can be restored to users the next time they sign in.
- **App data:** If your app presents data to users, you need to store that data in MySQL. You might also want to store behind-the-app-scenes data such as analytics and visitor statistics.

For an app such as FootPower!, the data requirements would fall into two segments:

- The app would have user accounts, so the app would need a MySQL table to store each account's email address, password, verification status, and a few site preferences.
- Users would be recording their foot-propelled movements, so the app would need two tables to store this data:
  - Each user would create a log of their activities, so the app would need a table to record the data for each of these logs, basically just a unique log ID, the ID of the user who owns the log, and the date the log was created.
  - Within each user's log would be the activities themselves, which the app would store in a separate table that includes a unique ID for each activity, the user's log ID, and fields for each chunk of activity data: type, date, distance, and duration.

How will my app work?

REMEMBER

Once you know what you want your app to do and what data your app requires, you're ready to tackle how your app works. This is called the app's *workflow*, and it covers at a high level what the app does and the order in which it does those things. A simple flowchart is usually the way to go here: Just map out what happens from the time users type in your app URL to the time they leave the page.

Figure 1-1 shows the workflow I envision for the FootPower! app.

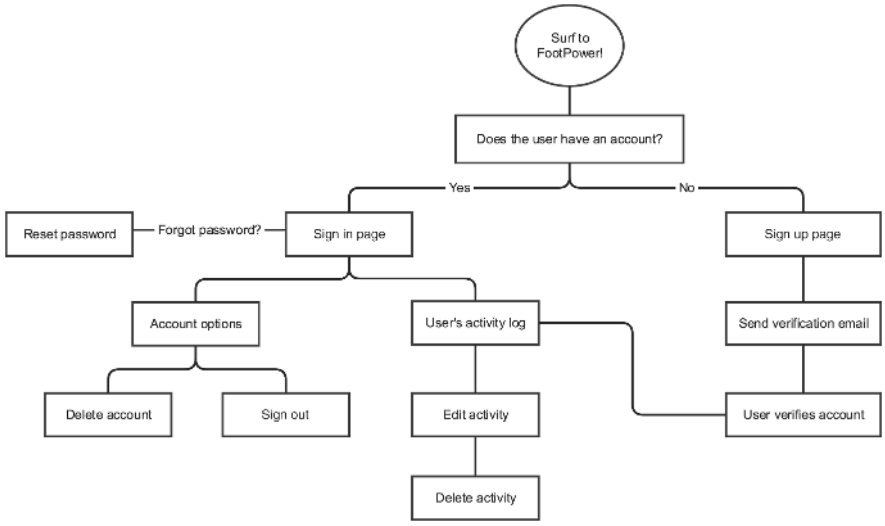

FIGURE 1-1: The workflow for the FootPower! app.

How many pages will my app require?

Your app's workflow should tell you fairly specifically how many pages your app needs. Most web apps are focused on a single set of related tasks, so your users will spend most of their time on the page that provides the app's main interface, usually the home page. However, your app will need other pages to handle tasks such as registering users, signing in users, and displaying account options. Record every page you need; this information will act as an overall to-do list for the front end.

Here's a potential list of pages for the FootPower! app:

- The home page, which would require two versions:
  - The unregistered or signed-out version of the home page, which would serve as a kind of ad for the app
  - The signed-in version, which would show the user's activity log and enable log-based tasks such as creating, filtering, editing, and deleting activities
- A page that enables new users to register
- A page letting new users know that a verification email has been sent
- A sign-in page
- A page that enables the user to edit and delete activities
- A password reset page
- An account options page
- An account delete page

What will my app's pages look like?

Before you start laying down your HTML and CSS code, you need to have a decent sense of what you want your app's pages to look like. Sure, all of that might be in your head, but it really pays in the long run to get those images down on paper with a sketch or two. These sketches don't have to be fancy in the least. Just take a pen, pencil, or your favorite Crayola color and rough out the overall structure.

Simple forms (such as those for signing in or resetting a password) don't require much effort, but for more elaborate pages, such as your app's home page, you need to flesh out the design a bit: header, navigation, main content, sidebar, footer, and so on.

Figure 1-2 shows an example sketch for the FootPower! app's home page.

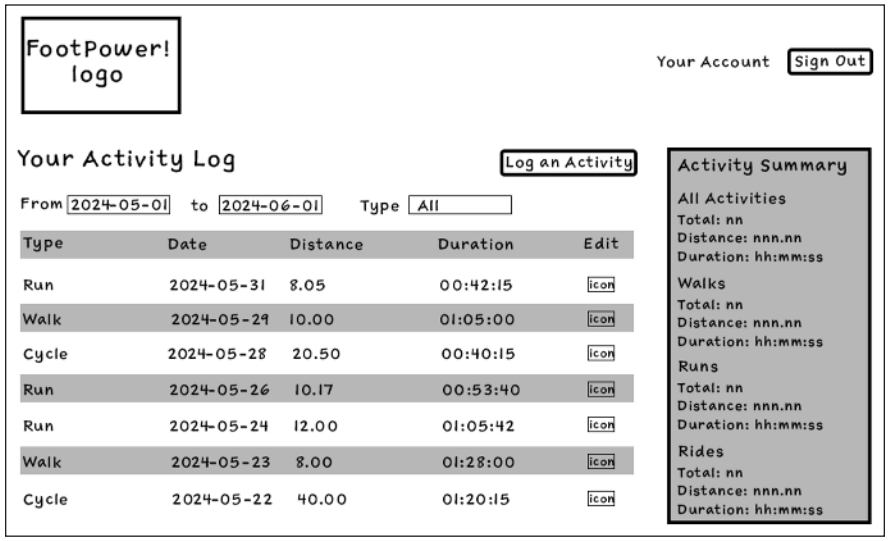

FIGURE 1-2: A sketch of the home page for the FootPower! app.

With your web app plan in place, you're ready to start coding the app. In the next chapter, you learn how to make your web app's layout responsive.