Thank you for purchasing the MEAP for *NativeScript in Action*. We’re proud and excited to be a part of this project, and hope you can join us in our enthusiasm surrounding NativeScript. This is a book for desktop, web, or mobile software developers with approximately 1-2 years of experience and an intermediate understanding of XML, JavaScript, and CSS. To get the most out of this book, you should have an active interest in mobile development or want to learn more about getting started in mobile development.

Chapter 1 begins with a brief overview of NativeScript, how it works (at a high-level), how it compares to other mobile development frameworks, and why you may choose to use NativeScript. Because we’ve targeted the book towards a broader audience, we purposefully do not dive deep into the internal workings of NativeScript. Although the internals of NativeScript are certainly interesting (especially if you are the type of developer who loves to know what is going on behind the scenes), they could distract you and take your focus away from what we believe our missions to be: to empower you to build amazing apps with NativeScript.

Chapters 2 and 3 dive a little deeper into NativeScript, exploring how NativeScript apps are created, built, and run using the NativeScript command line interface (CLI) tools. We also explore how NativeScript uses a variety of conventions to make mobile development easier.

Parts two and three of the book take you through building three different mobile apps, which are representative apps a new mobile developer may build to showcase their knowledge. By the end of this book, you will be able to write a multi-page mobile app with complex user interface designs and use data binding with remote web service calls to create dynamic content pages within your mobile apps. You will also learn how to write a single app with dynamic user interface layouts that can be deployed to devices with varying sizes, screen resolutions, and platforms (such as an iPad, iPhone, Android phone, and an Android tablet).

On a final note, thank you. This is a book for you and others looking for a different and potentially better way of developing cross-platform (yet native!) mobile apps. We hope you take advantage of the Author Online forum, where you can send us direct feedback. We’ll be reading your comments and responding regularly. Your feedback and engagement are incredibly important to us and will make NativeScript in Action a better book.

Thanks again!
—Mike and Nick Branstein

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders. https://forums.manning.com/forums/nativescript-in-action
brief contents

PART 1: THE BASICS
  1 Why NativeScript
  2 Your first app
  3 Anatomy of a NativeScript app

PART 2: STRUCTURING YOUR APP
  4 Pages and navigation
  5 Understanding the basics of app layouts
  6 Using advanced layouts
  7 Styling NativeScript apps

PART 3: REFINING YOUR APP
  8 Working with data
  9 Native hardware
  10 Creating professional UIs with themes
  11 Refining the user experience
  12 Deploying an Android app
  13 Preparing an iOS app for distribution
  14 iOS security and building your app with Xcode

PART 4: ANGULAR 2 AND NATIVESCRIPT
  15 Creating a NativeScript App with Angular
  16 Using Angular components and routing
  17 Angular databinding and services

APPENDIXES:
  A Android emulator tips
  B NativeScript CLI quick reference
  C NativeScript conventions
  D Creating custom UI controls

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders. https://forums.manning.com/forums/nativescript-in-action
This chapter covers

- What is NativeScript
- What NativeScript means to the mobile development world
- How NativeScript works

In the early days of mobile apps (pre-iPhone), not much emphasis was placed on methodologies for writing code once and deploying it to multiple platforms. Developers just wanted to get an app out to the Apple or Google Play stores as fast as possible. And if that meant their app didn’t support both platforms, that was a reasonable sacrifice.

Fast forward to today: the mobile world is continually changing, making it more and more difficult to keep up with the latest devices. As developers create app, they need to reach the largest audience possible: focusing on a single platform just isn’t an option anymore. Apps need to be available across platforms and devices. To keep up with ever-changing environment, developers are placing a premium on any technology that enables them to simplify the mobile app development process.

Today, developers have various choices for writing mobile apps that can target multiple platforms from a single code base. NativeScript is one of those choices, but it isn’t the only one. You may have heard of others like PhoneGap, Xamarin, and React Native. Each of these frameworks is capable of writing code once and deploying it to both Android and iOS, but we’re not here to debate the merits of one framework over another. Instead, we want you to learn how to write professional cross-platform mobile apps using skills you likely already have. If you’re a beginner who knows the basics of writing web apps with HTML, JavaScript, and CSS, or seasoned expert, you can write a mobile app with NativeScript.

As you read this book, we’ll show you how to write cross-platform apps from a single code base using the structured approach that NativeScript offers. When you’re finished, you’ll have...
the skills to create your own mobile apps for Android and iOS with your choice of technologies: HTML, JavaScript, and CSS or Angular, TypeScript, and CSS.

**NOTE** If you’re not familiar with Angular or TypeScript, that’s ok. The last 3 chapters of this book are dedicated to teaching you what they are and how they can be used to create mobile apps.

We’ve worked with a lot of developers learning NativeScript for the first time, and many of them want to jump straight into NativeScript with Angular. If that sounds just like you, go for it, but proceed with caution. If you’re not familiar with Angular, learning both NativeScript and Angular at the same time can be confusing because the lines between what’s NativeScript and what’s Angular will be blurry. So, we recommend you learn about plain-old-vanilla NativeScript first. Follow along with our exercises in the first 3 parts of this book, then jump into NativeScript with Angular.

Before we get ahead of ourselves, let’s back up and look at NativeScript in more detail.

### 1.1 Introducing NativeScript

NativeScript is an open-source framework for building cross-platform mobile apps for iOS and Android, created and maintained by Telerik. NativeScript differs from other mobile frameworks in many ways, the largest being that it is a cross-platform framework that can create native mobile apps with a single code base. Additionally, NativeScript offers a lot of features that make it easy to get started and leverage skills you may already have:

- Leverages your existing knowledge of HTML, JavaScript, and CSS (you don’t have to know Objective C, Swift, or Java)
- All your code is written once
- Access to native platform APIs for Android and iOS
- An opinionated way to create apps that helps structure your code base
- Natively integrates with Angular (but doesn’t have to)

Sometimes learning a new language is a barrier to entry into a new world. When creating NativeScript apps, you’ll leverage your existing knowledge of HTML applications so you can quickly create an app targeting multiple platforms (Android and iOS). Because you already have these skills, you’ll find that creating NativeScript apps can be quick. And even better, you won’t have to learn Objective C, Swift, or Java.

#### 1.1.1 How NativeScript apps are written

NativeScript apps are written in a combination of JavaScript, XML, and CSS, as shown in figure 1.1.
Figure 1.1 JavaScript, CSS, and XML combine to create a NativeScript app.

When you write NativeScript apps, your code has 3 parts: JavaScript, XML, and CSS. The JavaScript component runs business logic, accesses data, or controls the flow of the app. The XML portion defines the user interface (UI), and CSS is used to style the UI, much like an HTML application.

The structure and code of NativeScript apps closely resemble HTML applications, but this is where the similarities end. NativeScript is unique in the cross-platform mobile app space because it allows you to write your UI (XML) code once. When run, the UI code renders native UI elements in the app. For example, on iOS UI elements are rendered as native iOS buttons, dropdowns, lists, and so on. Likewise, on Android UI elements are rendered as native Android components.

Figure 2.1 shows the native rendering of an iOS button, written in NativeScript.

You’ll notice that it looks just like an iOS button. And, that’s because it is an iOS button. All NativeScript UI elements are native iOS and Android UI elements.

In other cross-platform frameworks, you may have to spend time writing specific view code for specific platforms. But, the ability to write your UI code once and have it render as native UI components is a feature that sets NativeScript apart from other frameworks.

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders. https://forums.manning.com/forums/nativescript-in-action
Another unique feature of NativeScript is that you have access to Native APIs.

**NOTE** Yes, several of the previously mentioned frameworks also let you access Native APIs. But, as you’ll learn throughout this book, the way NativeScript runs and accesses the Native APIs of Android and iOS is much different. Even though you write NativeScript code in JavaScript, you have access to every native API function, feature, and hardware component device your app runs on.

As you continue to learn about NativeScript, you’ll see how NativeScript executes all your code as native code running on the device. This allows you to take advantage of the performance gains of writing native code without having to learn or write Objective C, Swift, or Java!

Now that you have a high-level familiarity with NativeScript, let’s look at what you’ll learn in this book.

### 1.2 What you’ll learn in this book

At this point, you’re beginning to get an idea of the technologies you’ll use to write NativeScript apps (HTML, JavaScript, and CSS). Not a whole lot, right? If you already have these skills, you may be wondering why you should keep reading. In this book, we’ll teach you how to take these skills and apply them to create professional-looking mobile apps.

What do we mean by professional? Professional can mean different things to different people. You might think just showing up to work on time is professional, while your friend may think professional means wearing a suit to an interview.

**NOTE** To us, professional means creating a single, maintainable code base for your app so it can continue to grow over time.

Creating a professional app is also about using NativeScript's features so your app looks and feels native on the platform that it is running on.

While you’re learning to create professional apps, you’ll discover how NativeScript apps are structured and how to access native hardware components such as the camera, GPS, and location services.

But, before we jump straight into code, it's important that we put NativeScript into perspective, so you understand how it works.

### 1.3 What NativeScript means to mobile development

Think back 15 years (if you can), when you were carrying around a Windows 6 mobile phone or geeking-out over the latest Samsung Blackjack: this was before Android and iOS. There were just fewer platforms and devices back then. Today, new devices come out monthly. And because of this increasing rate and variety, the development community has begun to look for more efficient ways to write mobile apps that target all the platforms.
1.3.1 Different types of mobile apps

Mobile apps fall into one of four major categories: native, hybrid, cross compiled, and just-in-time (JIT) compiled (table 1.1).

Table 1.1 Different mobile app types and their popular frameworks

<table>
<thead>
<tr>
<th>Mobile App Type</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>Android, iOS</td>
</tr>
<tr>
<td>hybrid</td>
<td>PhoneGap/Cordova</td>
</tr>
<tr>
<td>cross compiled</td>
<td>Xamarin</td>
</tr>
<tr>
<td>JIT compiled</td>
<td>NativeScript</td>
</tr>
</tbody>
</table>

**DEFINITION** Just-in-time (JIT) compiled apps are apps that are compiled at runtime versus being compiled before the execution of the app. For example, in a just-in-time app, your source code is not compiled to native machine code until the absolute last minute, or immediately prior to executing each statement.

Excluding native apps, the other three app types in table 1.1 have the same goal: write your app code once and deploy it to multiple platforms (which is what people mean when they say cross-platform).

Even though the cross-platform frameworks listed above achieve similar results, they do so in a variety of ways. Figure 1.3 shows the differences between the different types of mobile apps and how they run on devices.

![Figure 1.3 How types of mobile apps run on a device](https://forums.manning.com/forums/nativescript-in-action)
In figure 1.3, you can see how each type of mobile app uses a different mechanism to run on a mobile device. Hybrid mobile apps are essentially webpages run inside of a web browser. Cross compiled apps are compiled into, which transforms them into a native app. Lastly, JIT compiled apps (like NativeScript), run inside of virtual machine. For NativeScript apps, your app code runs inside of a JavaScript virtual machine.

**DEFINITION** A JavaScript virtual machine is a piece of software that runs JavaScript code.

If you’re a .NET or Java developer, you’re already familiar with running code in a virtual machine because both .NET and Java run code in a virtualized manner. The way NativeScript works is similar.

### 1.3.2 Why NativeScript is important

Besides JIT compilation, NativeScript has a variety of other differences when compared with other mobile app frameworks. We think the most significant difference is your ability to write truly native apps from a single code base and deploy it to both Android and iOS with no changes.

We’ve worked with other mobile app frameworks in the past, and in our opinion, NativeScript stands apart. In other frameworks, we’ve had to write a lot of shim code. This shim code acts like a piece of wood that’s used to level a stove in your kitchen or to help frame a doorway. To continue the analogy, imagine you’re installing a new door and door frame. Most doors are built to a standard width, height, and depth, and they fit almost right. But in all cases, you add a little shim here and a little shim there to get it to fit just right. This is what it’s like when writing code in other frameworks: you add a bit of UI code to make a button display just right in the Android version of the app, and a little more UI code to make it look just right on iOS.

**NOTE** Hold on, we’re not trying to paint the picture that NativeScript is perfect, because nothing is. But, NativeScript is compelling, and in our opinion, requires the least number of shims. In fact, the shims are so limited, you may never come across them when you’re writing a line-of-business app. And when you do run across them, there’s an extensive community of NativeScript experts ready and willing to help at https://nativescript.org.

**GETTING TO MARKET FAST**

So, what does this all mean: less shim code, write-once, deploy everywhere, and so on. Whether you’re a business, an independent developer, or a casual enthusiast, you don’t want to waste your time. And, these things (less shim code, write-one, and deploy everywhere) means you’ll spend less time developing your app, giving you more time to innovate and release more features in less time.
1.3.3 What types of apps can be built with NativeScript

Now that you know a bit more about how NativeScript works, we think it’s important that you know the type of mobile apps you can write with it. You’ll recall that NativeScript apps run directly on the device and are interpreted by a JavaScript virtual machine running inside of the app. This means NativeScript apps aren’t restricted from accessing native device APIs or hardware, so any app can be written as a NativeScript app.

**WARNING** Hold up. Just because you can doesn’t mean you should.

Let’s start by looking at app types that you shouldn’t create with NativeScript.

**GRAPHIC-INTENSIVE GAMES**

Let’s start off being clear: don’t write graphic-intensive games with NativeScript.

Imagine you’re developing the next big mobile game: Floppy Bunny, and Floppy Bunny requires a lot of graphical and computational power to render its intense 3D graphics. While NativeScript is very performant out of the box, there are likely better platforms made for the express purpose of creating highly-performant 3D games.

After all, NativeScript apps run inside of a JavaScript virtual machine, so there’s an extra, albeit small, layer of abstraction between your app and the bare metal. To extract every bit of performance out of a device and make Floppy Bunny an overwhelming success, you should consider writing a native Android or iOS app.

**LINE-OF-BUSINESS AND CONSUMER APPS**

If you’re feeling down because we shattered your hope of writing Floppy Bunny, don’t worry. There are other types of apps that NativeScript is great for!

Unlike our game example, NativeScript is a perfect choice if you’re developing a line-of-business app such as a news feed, companion app for a website, social media app, or even an app to control all the smart devices in your home! In fact, there’s a wide variety of apps already written in NativeScript across dozens of industries. Check out a showcase of these apps at https://www.nativescript.org/showcases.

1.4 How NativeScript works

Writing native mobile apps using JavaScript, XML, and CSS isn’t something you commonly hear about. Instead, you hear about writing native mobile apps in Objective C, Swift, or Java. NativeScript makes it possible to write native mobile apps with several components: the NativeScript runtime, core modules, JavaScript virtual machines, your app code, and the NativeScript command line interface (CLI). Figure 1.5 shows how these components work together to create native Android and iOS projects, which get built into native apps that run on a mobile device.
We know there are a lot of boxes and lines in figure 1.5, and visualizing how these components work together at this point may seem overwhelming. Don’t worry. We will go through each of the items later in this book. For now, let’s get you started by explaining how everything works together at a high level.

Let’s start with something you’ve already learned: your app code is written in JavaScript, CSS, and XML. After you’re written your code, it interacts with the NativeScript runtime and the NativeScript code modules (API modules that you’ll learn about in this book).

Finally, a tool known as the NativeScript CLI, bundles your code, the NativeScript runtime, and NativeScript core modules into a native app that contains a JavaScript virtual machine. This native app then runs on Android and iOS.

**DIVING DEEPER**

That’s it! You just learned how NativeScript apps work at the 10,000-foot level, but let’s dive a little deeper.

After creating your user interface (UI) using XML, you use CSS to style the UI (like CSS is used to style HTML apps). Then, you write JavaScript to augment your UI. Your JavaScript code will contain writing business logic that responds to events (like the app startup event) and interactions (like a button tap or finger swipe). These three pieces (UI written with XML, CSS, and business logic written with JavaScript) combine to create your app code.

By itself, your app code doesn’t have everything it needs to run on a mobile device; you also need the help of three additional components: the NativeScript runtime, core modules, and a JavaScript virtual machine. We’ll explore these components in future chapters, but for...
now, just remember that your app code and these three components form the core of your NativeScript app.

After you’ve developed your app code, it is fed into the NativeScript command line interface (CLI). The CLI is responsible for creating native Android and iOS projects and merging the NativeScript app core into each project. When run, the CLI invokes the native Android or iOS software development kits (SDKs) to build and compile a native app. The compiled app is then deployed (by the CLI) and runs on a physical device, simulator, or emulator.

As you can see, NativeScript’s beauty lies in its universal nature: you don’t have to spend time learning native programming languages like Objective C, Swift, and Java because you can use JavaScript. Furthermore, the platform agnostic commands provided by the NativeScript CLI ensure you don’t have to learn how the native tools and SDKs for Android and iOS work.

## 1.5 Summary

In this chapter, you learned that:

- NativeScript apps are written in JavaScript, XML, and CSS and run in a JavaScript virtual machine.
- Your app code works with the NativeScript runtime, core modules, and a JavaScript virtual machine to create the core of a NativeScript app.
- The NativeScript CLI abstracts away the complexities of native tools and SDKs, providing you with a single platform-agnostic set of commands to build and deploy your app.

**NOTE** Before you continue, you’ll need to get your development environment set up. Please refer to the official NativeScript installation instructions at [http://docs.nativescript.org/start/quick-setup](http://docs.nativescript.org/start/quick-setup).

**TIP** If you are having difficulties getting the Android emulator setup and running, please see the Android Emulator Tips in appendix A.