

MOBILE-FIRST WEB DESIGN

Build Responsive Websites with Bootstrap

From mobile-first layouts to complete, accessible sites with Bootstrap 5

Grid, utilities, components, Sass, and real projects — with exercises and solutions in every chapter

HTML · CSS · GRID · COMPONENTS · SASS

THENEXBOOKS

Build Responsive Websites with Bootstrap

From Mobile-First Layouts to Complete, Accessible Sites with Bootstrap 5
Grid, Utilities, Components, Sass Customization, and Real Projects

First Edition

THENEXBOOKS

Build Responsive Websites with Bootstrap
From Mobile-First Layouts to Complete, Accessible Sites with Bootstrap 5

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples and exercise solutions may be used freely for learning and in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Examples target Bootstrap 5.3; always consult the official documentation for the version you use.

Bootstrap is an open-source project; the name is used in this book for identification only. This book is independent and is not affiliated with, sponsored, or endorsed by the Bootstrap team. Other product names are trademarks of their respective owners.

Juniper Lane Bakery, the running example in this book, and all people and businesses in the examples are fictional. Any resemblance to real organizations or people is coincidental.

Preface

Most people now meet a website on a phone first. A site that looks polished on a wide monitor but forces visitors to pinch, zoom, and scroll sideways on a small screen loses them in seconds. Building layouts that adapt gracefully to every screen size used to mean writing and debugging a lot of hand-made CSS. Bootstrap changed that: it gives you a tested responsive grid, a large set of utility classes, and ready-made components such as navigation bars, cards, forms, and modals — all designed to work together.

This book teaches you to use Bootstrap well, not just to copy snippets. We start with how responsive design works and how Bootstrap thinks — mobile-first, breakpoints, and utilities — and then build up layouts, content, and components step by step. Later you will learn to make Bootstrap look like your brand instead of like every other Bootstrap site, with CSS variables, color modes, Sass, and the Utility API, and to ship sites that are accessible and fast.

Every chapter ends with graded exercises and worked solutions. By the end you will have built a complete multi-page website for a small business, a personal portfolio, and an admin dashboard — and you will know how to plan, build, troubleshoot, and launch responsive sites of your own.

Who This Book Is For

- **Beginners** who know a little HTML and CSS and want to build professional-looking responsive sites quickly.
- **Junior front-end developers** who use Bootstrap but want to understand the grid, utilities, and customization properly.
- **Designers and back-end developers** who need clean, responsive front ends without writing everything from scratch.
- **Students and freelancers** building sites for courses, clients, or their own portfolio.

What You Will Learn

- How responsive, mobile-first design works and how Bootstrap's breakpoints and utility classes express it.
- The grid system in depth: containers, rows, columns, gutters, nesting, ordering, and alignment.
- Typography, images, spacing, colors, flexbox, and tables with utility classes.
- Components: buttons, cards, navbars, forms with validation, modals, offcanvas, carousels, toasts, and more.
- Customization with CSS variables, light and dark color modes, Sass, the Utility API, icons, and custom fonts.
- Accessibility, performance, deployment, troubleshooting, and three complete projects.

How This Book Is Organized

Part I — Getting Started with Bootstrap (Chapters 1–5) covers responsive design, setup, a CSS refresher, mobile-first thinking, and the grid. **Part II — Layout and Content** (Chapters 6–10) covers advanced grid techniques, typography and images, utilities, flexbox, and tables. **Part III — Components** (Chapters 11–15) covers the components you will use most. **Part IV — Customizing Bootstrap** (Chapters 16–20) covers CSS variables, color modes, Sass, the Utility API, icons and fonts, and the JavaScript API. **Part V — Building Real Pages** (Chapters 21–25) covers landing pages, multi-page sites, accessibility, production, and troubleshooting. **Part VI — Projects and Case Study** (Chapters 26–28) builds three complete sites. The appendices provide a cheat sheet, starter templates and snippets, a launch checklist, and a glossary.

The Running Example: Juniper Lane Bakery

Throughout the book we build the website for **Juniper Lane Bakery**, a fictional neighborhood bakery and café. Its menu, events, pre-order form, gallery, and staff dashboard give every chapter a practical job, and in the final chapter the pieces come together as a complete site ready to launch.

Conventions

Code appears in `monospace`. Examples use Bootstrap 5.3 loaded from a CDN unless a chapter says otherwise; file captions such as `juniper-lane/menu.html` show where code belongs. Exercises are labeled **[Easy]**, **[Medium]**, or **[Challenge]**.

> **TIP**
> A recommended practice, or a "Try It" prompt to pause and experiment.

> **NOTE**
> Background information or a useful aside.

> **WARNING**
> A common mistake that breaks layouts or behavior.

Contents

Preface

PART I — GETTING STARTED WITH BOOTSTRAP

1 What Bootstrap Is and Why Use It: Responsive Web Design in a Nutshell

2 Setting Up: CDN, npm, Project Structure, and Your First Page

3 HTML and CSS Refresher for Bootstrap: The Box Model, Classes, and the Viewport

4 How Bootstrap Thinks: Mobile-First, Breakpoints, and Utility Classes

5 Containers and the Grid System: Rows, Columns, and Gutters

PART II — LAYOUT AND CONTENT

6 Advanced Grid Techniques: Nesting, Ordering, Offsets, and Auto-Layout

7 Typography, Images, and Figures

8 Spacing, Sizing, Colors, and Borders with Utilities

9 Flexbox Utilities, Alignment, and Positioning

10 Tables and Responsive Data Display

PART III — COMPONENTS

11 Buttons, Badges, Alerts, and Cards

12 Navigation: Navbars, Navs, Tabs, Breadcrumbs, and Pagination

13 Forms: Controls, Layout, and Validation

14 Interactive Components I: Modals, Offcanvas, Dropdowns, Collapse, and Accordions

15 Interactive Components II: Carousels, Tooltips, Popovers, Toasts, Spinners, and Progress

PART IV — CUSTOMIZING BOOTSTRAP

16 CSS Variables and Color Modes: Light and Dark Themes

17 Customizing with Sass: Variables, Maps, and Importing Only What You Need

18 The Utility API: Creating and Changing Utilities

19 Bootstrap Icons, Custom Fonts, and Brand Styling

20 JavaScript Behaviors: Data Attributes, the JavaScript API, and Events

PART V — BUILDING REAL PAGES

21 Designing a Landing Page: Hero, Features, Pricing, and Calls to Action

22 Multi-Page Sites: Shared Layout, Navigation, and Footer Patterns

23 Accessibility in Bootstrap Sites

24 Performance and Production: Optimizing, Trimming CSS, and Deploying

25 Common Layout Patterns and Troubleshooting Responsive Problems

PART VI — PROJECTS AND CASE STUDY

26 Project: A Personal Portfolio Site

27 Project: An Admin Dashboard

28 Case Study: Building the Juniper Lane Bakery Website

APPENDICES

A Bootstrap Cheat Sheet

B Starter Templates and Component Snippets

C Responsive Launch Checklist

D Glossary

PART I

Getting Started with Bootstrap

Responsive design, setting up Bootstrap, just enough HTML and CSS, mobile-first thinking, and the grid system.

1 · What Bootstrap Is and Why Use It
2 · Setting Up
3 · HTML and CSS Refresher for Bootstrap
4 · How Bootstrap Thinks
5 · Containers and the Grid System

CHAPTER 1 · PART I — GETTING STARTED WITH BOOTSTRAP

What Bootstrap Is and Why Use It: Responsive Web Design in a Nutshell

Before we write a single class name, we need a clear picture of the problem Bootstrap solves. In this chapter you will learn the three ideas behind every responsive website, what a CSS framework adds on top of plain CSS, what is inside Bootstrap, when it is the right tool, and what the finished Juniper Lane Bakery site will look like by the end of the book.

The Problem: One Website, Many Screens

A website does not get to choose where it is viewed. The same page might be opened on a small phone held in one hand, a tablet turned sideways on a kitchen counter, a laptop, or a wide desktop monitor. Each of these screens has a different width, a different pixel density, and a different way of interacting: fingers on a touchscreen, a mouse and keyboard, or sometimes a keyboard alone.

In the early days of the web, many sites were designed for one fixed width — say, 960 pixels — and simply centered on the screen. On a desktop that looked fine. On a phone, the browser had two bad choices: shrink the whole page until the text was unreadable, or show a small corner of it and make the visitor scroll sideways. Some companies built a second, separate "mobile site" at a different address, which meant maintaining two sets of pages that slowly drifted apart.

Responsive web design is the approach that replaced both of those compromises. Instead of building separate sites for separate devices, you build *one* page whose layout adapts to the space it is given. On a phone, content stacks in a single column with large, tappable buttons. On a tablet, two columns might sit side by side. On a desktop, a navigation bar stretches across the top and cards line up three or four in a row. The HTML is the same; the CSS decides how it is arranged.

Picture the Juniper Lane Bakery home page, the site we will build throughout this book. Rosa Delgado, the owner and head baker, wants customers to check today's hours on their phone while walking down the street, and she wants the same page to look inviting on the big monitor Sam Whitaker uses in the café office. Responsive design is how one set of files serves both of them.

Responsive Web Design in a Nutshell

Responsive design rests on three ideas that work together. Once you understand them, everything Bootstrap does will feel familiar, because Bootstrap is really a well-organized, well-tested collection of these three ideas.

1. Fluid layouts

A fluid layout sizes its boxes with relative units such as percentages instead of fixed pixel values. If a sidebar is `25%` wide, it is a quarter of the available width whether that width is 360 pixels or 1,400 pixels. The layout stretches and shrinks with the window instead of breaking.

```
/* Fixed: always 960px, too wide for a phone */
.page { width: 960px; }

/* Fluid: fills the space, but never gets absurdly wide */
.page {
  width: 100%;
  max-width: 1140px;
  margin: 0 auto;
}
```

The second rule says "take the full width available, but stop growing at 1,140 pixels, and center yourself." On a phone the page fills the screen edge to edge; on a large monitor it stays at a comfortable reading width with space on both sides. Bootstrap's **containers**, which you will meet properly in Chapter 5, work in a very similar way.

2. Flexible media

Images and videos have their own natural size. A photo that is 1,600 pixels wide will happily overflow a 390-pixel phone screen unless you tell it otherwise. Flexible media means letting images scale down to fit their container:

```
img {
  max-width: 100%;
  height: auto;
}
```

With this rule, an image is never wider than the box that holds it, and its height adjusts to keep the original proportions. If the container is wider than the image, the image stays at its natural size rather than being stretched and blurred. Bootstrap packages exactly this behavior into a class called `.img-fluid`, which we cover in Chapter 7.

3. Media queries

Fluid widths and flexible images get you surprisingly far, but sometimes the layout itself has to change. Three columns that look great on a desktop become three skinny, unreadable strips on a phone. A **media query** is a CSS rule that only applies when a condition is true — most often, when the screen is at least a certain width.

```
/* Phones first: one column */
.menu-item { width: 100%; }
```

```
/* From 768px wide and up: two columns */
@media (min-width: 768px) {
  .menu-item { width: 50%; }
}
```

On a phone, each menu item takes the full width, so items stack one above the other. As soon as the window is 768 pixels wide or more, the second rule kicks in and items sit two per row. The width at which the layout changes is called a **breakpoint**. Bootstrap defines a standard set of breakpoints so you do not have to invent your own for every project; Chapter 4 walks through them in detail, and Chapter 3 shows you how to write media queries by hand so you can see what Bootstrap is automating.

> **TRY IT**
>
> Open any news or shopping site you use regularly in a desktop browser. Slowly drag the right edge of the window to make it narrower. Watch for the moments when the layout suddenly jumps — a menu collapses into a button, a sidebar drops below the main content, or columns stack. Each of those jumps is a media query firing at a breakpoint.

The viewport: the missing piece

There is one more detail that makes all of this work on real phones. Mobile browsers, by default, pretend the screen is much wider than it really is and then zoom out, so that old fixed-width sites remain usable. That behavior would defeat our media queries: a phone would claim to be about 980 pixels wide and never see the one-column layout. The fix is a single line in the page's `<head>`:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

This tells the browser "use the real width of the device and do not zoom out." Every page in this book includes it, and Chapter 3 explains it in more depth. Forgetting it is one of the most common reasons a "responsive" page looks tiny and zoomed out on a phone.

What a CSS Framework Gives You

You could build a responsive site with nothing but hand-written CSS, and in Chapter 3 you will do exactly that for one small section. But as soon as a site grows beyond a page or two, the same problems come up again and again:

- **Layout plumbing.** You need columns that line up, consistent gaps between them, and rules for how they rearrange at each breakpoint.
- **Browser differences.** Different browsers apply slightly different default styles to headings, lists, buttons, and form fields.
- **Repeated patterns.** Almost every site needs a navigation bar, buttons, cards, forms, alerts, and a footer.

- **Interactive pieces.** Menus that collapse on phones, pop-up dialogs, dropdowns, and tooltips all need JavaScript that handles clicks, keyboard use, and screen readers correctly.
- **Consistency.** Without a shared system, one page uses 14-pixel margins, another uses 15, and the site slowly looks untidy.

A **CSS framework** is a pre-written, tested stylesheet (often with some JavaScript) that solves these recurring problems once, so you can spend your time on what makes your site unique. You use it by adding class names to your HTML. Instead of writing a new CSS rule for "a green button with rounded corners and some padding," you write `class="btn btn-success"` and get a button that already looks right, already has hover and focus styles, and already matches every other button on the site.

A framework also gives a team a **shared vocabulary**. If you hand the Juniper Lane project to another developer next year, they will recognize `container`, `row`, and `col-md-6` immediately, because those names are documented and used on a huge number of sites. That is much easier than deciphering someone's personal naming scheme.

The trade-off is that you accept the framework's decisions and its file size. You also need to learn its conventions — which is exactly what this book is for.

What Is Inside Bootstrap

Bootstrap is a free, open-source front-end framework, released under the MIT license. It started as an internal project at Twitter and was released publicly in 2011; it has since grown into one of the most widely used CSS frameworks. This book teaches **Bootstrap 5.3**, and everything here applies to Bootstrap 5.3 or a later 5.x release. Bootstrap 5 dropped its old dependency on jQuery, so all of its JavaScript is plain, modern JavaScript.

It helps to think of Bootstrap as four layers, plus a foundation underneath them.

The foundation: Reboot and CSS variables

Reboot is Bootstrap's set of base styles. It smooths out browser differences, sets `box-sizing: border-box` on every element (explained in Chapter 3), uses the operating system's native font stack for body text, and gives headings, paragraphs, and lists sensible default margins. You get it automatically when you load Bootstrap's CSS. On top of Reboot, Bootstrap 5.3 exposes many of its colors and settings as **CSS custom properties** (variables whose names start with `--bs-`), which makes theming and the new light and dark color modes possible. We explore those in Chapter 16.

Layer 1: The grid system

The grid is Bootstrap's layout engine. It is built from **containers**, **rows**, and **columns**, and it divides each row into 12 equal parts. A column can span any number of those parts, and the number can change at each breakpoint. For example, `col-12 col-md-6 col-lg-4` means "full width on phones, half

width from medium screens up, one third from large screens up." The grid is built on CSS flexbox and has five breakpoints above the smallest screens: sm, md, lg, xl, and xxl. Chapters 5 and 6 cover it thoroughly.

Layer 2: Utility classes

Utilities are small, single-purpose classes that each do one thing: `mt-3` adds a top margin, `text-center` centers text, `d-none` hides an element, `bg-light` sets a light background. Many of them have responsive versions, such as `text-md-start` or `d-lg-block`, that only apply from a given breakpoint up. Utilities let you make small adjustments directly in the HTML without writing new CSS for every tweak. Chapters 8 and 9 go through spacing, sizing, colors, borders, flexbox, and positioning utilities.

Layer 3: Components

Components are ready-made interface pieces: buttons, badges, alerts, cards, navbars, navs and tabs, breadcrumbs, pagination, forms, list groups, modals, offcanvas panels, dropdowns, accordions, carousels, tooltips, popovers, toasts, spinners, and progress bars. Each one is a documented pattern of HTML plus a few classes. A card, for instance, is a `div.card` containing a `div.card-body` with a `h5.card-title` and some text. Part III of this book (Chapters 11 to 15) covers components one family at a time.

Layer 4: JavaScript plugins

Some components need behavior, not just styling: a navbar that expands when you tap the menu button, a modal that opens and traps keyboard focus, a carousel that slides. Bootstrap ships these as JavaScript plugins. You usually do not write any JavaScript to use them; you add `data-bs-*` attributes such as `data-bs-toggle="collapse"` to your HTML and the plugin wires everything up. When you need more control, each plugin also has a JavaScript API with methods and events, which Chapter 20 explains.

Bootstrap offers its JavaScript in two main builds. The **bundle** file, `bootstrap.bundle.min.js`, includes Popper, a small positioning library that dropdowns, tooltips, and popovers rely on. The plain `bootstrap.min.js` leaves Popper out. In this book we always use the bundle, so everything works with a single script tag.

Part	What it does	Example classes	Chapters
Reboot	Consistent base styles	(automatic)	3, 7
Grid	Responsive columns	`row`, `col-md-6`	5, 6
Utilities	One-job helpers	`mb-4`, `d-none`	8, 9, 18
Components	Ready-made UI	`card`, `navbar`	11–15
JS plugins	Interactive behavior	`data-bs-toggle`	14, 15, 20

Bootstrap's source is written in **Sass**, a language that compiles to CSS. You do not need Sass to use Bootstrap — the ready-made CSS file works on its own — but if you want to change the brand colors at the source, remove components you do not use, or add your own utilities, Sass is the tool. Part IV of the book covers that, starting with Chapter 17.

> **NOTE**
>
> Bootstrap 5 does not support Internet Explorer. It targets current versions of the major evergreen browsers (Chrome, Edge, Firefox, Safari) on desktop and mobile. If you ever need to support very old browsers, check the "Browsers and devices" page of the official documentation for the version you use.

When Bootstrap Is — and Isn't — a Good Fit

No tool is right for every job. Knowing where Bootstrap shines and where it gets in the way will help you make good decisions for your own projects and explain those decisions to clients.

Bootstrap is a good fit when...

- **You need a solid, responsive site quickly.** A small business site like Juniper Lane, a landing page, a portfolio, or an internal tool can be built in days rather than weeks.
- **You are not a full-time designer.** Bootstrap's defaults are clean and consistent. Back-end developers and students can produce a professional-looking result without deep design skills.
- **You are building dashboards and admin screens.** Tables, forms, cards, navs, and modals are exactly what admin interfaces are made of. The Juniper Lane staff dashboard in Chapter 27 is a classic Bootstrap use case.
- **A team needs shared conventions.** Documented class names mean new team members can contribute quickly.
- **You want accessible components without starting from zero.** Bootstrap's interactive components handle much of the keyboard and ARIA work for you, though you still have responsibilities, which Chapter 23 covers.
- **You are prototyping.** Trying out a layout with real, clickable components is often faster than drawing it.

Bootstrap may not be the best fit when...

- **The design is highly custom.** If a designer has created a very distinctive look that shares little with Bootstrap's components, you may spend more time overriding styles than you save. You can still use just the grid and utilities, though.
- **Every kilobyte matters.** The full CSS and JavaScript files are larger than a tiny hand-written stylesheet. Chapter 17 shows how to import only the parts you use, and Chapter 24 covers trimming unused CSS, which closes much of the gap.

- **Your project already uses a different system.** If a team has standardized on another framework or a component library tied to a JavaScript framework, mixing in Bootstrap usually creates conflicts rather than solving problems.
- **You want to learn CSS layout deeply.** Bootstrap is not a substitute for understanding CSS. That is why Chapter 3 is a refresher and why we explain what each class does, not just which one to use.

> **WARNING**
>
> The most common complaint about Bootstrap sites is that they "all look the same." That is not a limitation of the framework; it is what happens when a site ships with every default untouched. In Part IV you will give Juniper Lane its own colors, fonts, and details so it looks like a bakery, not like a template. Plan for that customization from the start.

A Tour of the Finished Juniper Lane Home Page

Throughout the book we build a complete website for **Juniper Lane Bakery**, a fictional neighborhood bakery and café at 42 Juniper Lane, Maple Bay. You are the web developer. Your clients are Rosa Delgado (owner and head baker), Sam Whitaker (café manager), and Priya Nair (pastry chef). The site lives in a project folder called `juniper-lane/` and will eventually include a home page, a menu, an about page, events, an online pre-order form, a gallery, a contact page, and a staff dashboard. Chapter 28 is a case study that pulls the whole site together.

Let's walk through the finished home page, `index.html`, in words — first on a phone, then on a desktop. We will not explain the code yet; the point is to see what responsive design looks like in practice and to preview the tools you will learn.

On a phone (about 390 pixels wide)

- At the top, a **navbar** shows the bakery name on the left and a menu button (the familiar three-line "hamburger" icon) on the right. Tapping the button slides the links — Home, Menu, About, Events, Gallery, Contact — down into view, followed by an "Order ahead" button. That is Bootstrap's navbar with its collapse plugin.
- Below it, a **hero section** fills the screen: a photo of fresh loaves, the headline "Baked this morning on Juniper Lane," a short sentence, and a full-width "Pre-order for pickup" button that is easy to tap with a thumb.
- Next, three **feature cards** — Breads, Pastries, and Coffee — are stacked one above another, each full width.
- A **Customer favorites** section shows a few favorites with prices, such as the Sourdough Loaf at $7.50 and the Butter Croissant at $3.75.
- An **hours and contact strip** shows the address, phone number (555) 014-2200, and opening hours as a simple stacked list.

- The **footer** stacks its columns of links and ends with a copyright line.



On a desktop (1280 pixels wide)

- The navbar shows all the links in a single horizontal row; the menu button disappears because there is enough room.
- The hero splits into two columns: headline and buttons on the left, photo on the right.
- The three feature cards sit side by side in one row, with even gaps between them.
- The customer favorites and the hours strip sit next to each other.
- The content is centered in a container that stops growing at a comfortable width, with cream background on either side.

Notice what did *not* change: the HTML. The same headings, cards, and links are on both screens. Only the arrangement changed, driven by grid classes like `col-md-4`, display utilities like `d-none d-lg-block`, and the navbar's built-in responsive behavior. By the end of Chapter 5 you will be able to build that home page skeleton yourself.

The site will also carry the bakery's brand: a terracotta primary color (`#9a3412`), a honey accent (`#f59e0b`), a warm cream background (`#fdf8f3`), and dark gray text (`#1f2937`). Headings use Georgia, a serif font, for a friendly, traditional feel, while body text uses Bootstrap's default system fonts. Our custom classes all start with the prefix `jl-`, such as `.jl-hero` and `.jl-card`, so it is always obvious which styles are ours and which come from Bootstrap.

> **TRY IT**
>
> On your phone, open the website of a local café or shop. Then open the same site on a computer. Write down three things that change between the two (for example, the menu, the number of columns, or the size of images) and one thing you think could be improved on the phone. Keep the list; you will recognize the techniques behind each change as you read on.

How to Use This Book

This is a hands-on book. You will learn fastest by typing the examples yourself, changing them, and watching what happens at different screen sizes.

The structure

- **Part I (Chapters 1–5)** gets you set up and teaches how Bootstrap thinks: mobile-first design, breakpoints, utility classes, and the basic grid.
- **Part II (Chapters 6–10)** covers layout and content: advanced grid techniques, typography and images, spacing and colors, flexbox, and tables.
- **Part III (Chapters 11–15)** works through the components, from buttons and cards to modals and carousels.
- **Part IV (Chapters 16–20)** is about customizing: CSS variables, dark mode, Sass, the Utility API, icons and fonts, and Bootstrap's JavaScript API.
- **Part V (Chapters 21–25)** puts it all together into real pages, with dedicated chapters on accessibility, performance, and troubleshooting.
- **Part VI (Chapters 26–28)** contains three complete projects: a portfolio, an admin dashboard, and the Juniper Lane case study.

If you are new to HTML and CSS, read Parts I and II in order. If you already use Bootstrap, skim Chapters 1 to 4 and dive into whichever part you need, then come back to Part IV — customization is where many experienced users have gaps.

Conventions

- Code blocks that belong to a specific file have a caption above them, such as `juniper-lane/index.html`. Snippets without a caption are fragments you can drop into any page.
- Full-page examples load Bootstrap 5.3.3 from the jsDelivr CDN. Check getbootstrap.com for the newest 5.3.x patch number; any later 5.x release should work with the examples.
- We describe what you should see at each screen size in words, often with a small text diagram.
- **Tip**, **Note**, and **Warning** boxes highlight shortcuts, background details, and common mistakes. **Try It** boxes suggest a quick experiment.
- Every chapter ends with graded Practice Exercises (Easy, Medium, and Challenge), Selected Solutions for some of them, and a Key Takeaways summary.

What you need

You need a modern browser (Chrome, Edge, Firefox, or Safari), a code editor (Visual Studio Code is a popular free choice, but any editor works), and a folder for your project. An internet connection helps because the examples load Bootstrap from a CDN. Later chapters use Node.js and npm for installing Bootstrap locally and compiling Sass; Chapter 2 explains how to set that up. For images, use your own photos and save them under `juniper-lane/assets/img/` — any bread or coffee photo will do.

You can also use an AI assistant such as `<chat-model>` to explain an unfamiliar class or suggest a layout, but always check its suggestions against the official documentation. Assistants sometimes produce class names from older Bootstrap versions or classes that do not exist at all.

> **TIP**
> Keep the official Bootstrap documentation open in a browser tab while you work through this book. We explain concepts in our own words and focus on why things work; the docs are the perfect companion for looking up the full list of options for a component.

A Small Taste of Bootstrap

To finish the chapter, here is a short fragment of the Juniper Lane home page. You do not need to run it yet — Chapter 2 sets up the project — but reading it shows how much the class names say on their own.

juniper-lane/index.html (feature cards section)

```
<section class="container py-5">
  <div class="row g-4">
    <div class="col-12 col-md-4">
      <div class="card h-100">
        <div class="card-body">
          <h2 class="h5 card-title">Breads</h2>
          <p class="card-text">Sourdough and seeded rye, shaped by hand.</p>
        </div>
      </div>
    </div>
    <!-- Pastries and Coffee cards follow the same pattern -->
  </div>
</section>
```

Read it from the outside in. `container` centers the content and limits its width. `py-5` adds generous padding above and below. `row g-4` starts a grid row with a 1.5rem gap between columns. `col-12 col-md-4` makes each card full width on phones and one third of the row from 768 pixels up. `card h-100` creates a card that stretches to the full height of its column, so all three cards line up evenly. And `h5` on the `h2` keeps the correct heading level for accessibility while making it look like a smaller heading. On a phone the three cards stack; from md up they sit three in a row.

Every one of those classes will be second nature to you within a few chapters.

Practice Exercises

1. **[Easy]** In your own words, name the three core ideas of responsive web design and give a one-sentence example of each from a real website you use.
2. **[Easy]** Explain what the viewport meta tag does and what a phone would show if a page left it out.

3. **[Easy]** List the four main layers of Bootstrap described in this chapter and give one example class or attribute from each.
4. **[Easy]** The Juniper Lane brand uses four colors. Write them down with their hex values and say what each is used for.
5. **[Medium]** Write a plain CSS rule set (no Bootstrap) for a `.jl-page` wrapper that fills the screen on phones but never grows wider than 960 pixels, centered on larger screens.
6. **[Medium]** Write CSS so that `.jl-menu-item` elements are full width on phones and three per row from 992 pixels wide and up. Use a media query and percentage widths.
7. **[Medium]** Sam Whitaker asks whether the bakery site should use Bootstrap. Write a short paragraph (four or five sentences) giving your recommendation and two reasons, plus one thing to watch out for.
8. **[Medium]** Read the feature cards fragment from this chapter. Describe in words how the section looks at 390 pixels wide and at 1280 pixels wide.
9. **[Challenge]** Sketch, as an ASCII diagram, the Juniper Lane `menu.html` page on a phone and on a desktop. It should include a navbar, a page heading, the four menu categories (Breads, Pastries, Cakes & Tarts, Drinks) and a footer. Decide how many categories sit side by side on desktop.
10. **[Challenge]** Write a complete, valid HTML page (no Bootstrap yet) with the viewport meta tag, a responsive image rule, and a media query that turns a single-column list of three Juniper Lane events into three columns from 768 pixels up.

Selected Solutions

Exercise 2

The viewport meta tag, `<meta name="viewport" content="width=device-width, initial-scale=1">`, tells a mobile browser to lay the page out at the device's real width and to start at 100% zoom. Without it, most mobile browsers lay the page out as if the screen were roughly 980 pixels wide and then shrink everything to fit, so the text looks tiny and media queries aimed at phones never apply.

Exercise 5

```
.jl-page {
  width: 100%;
  max-width: 960px;
  margin-left: auto;
  margin-right: auto;
}
```

`width: 100%` makes the wrapper fill whatever space it has. `max-width` caps it at 960 pixels, and automatic left and right margins split the leftover space evenly, centering the wrapper. On a phone it is edge to edge; on a large monitor it sits in the middle.

Exercise 6

```
.jl-menu-item {
  width: 100%;
  box-sizing: border-box;
  padding: 1rem;
}

@media (min-width: 992px) {
  .jl-menu {
    display: flex;
    flex-wrap: wrap;
  }
  .jl-menu-item {
    width: 33.333%;
  }
}
```

Below 992 pixels, every item is full width and stacks. From 992 pixels up, the parent becomes a wrapping flex container and each item takes a third of the width, so three sit per row. `box-sizing: border-box` keeps the padding inside the 33.333% so the three items still fit. This is the kind of work Bootstrap's `row` and `col-lg-4` classes do for you.

Exercise 10

scratch/events-plain.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Events | Juniper Lane Bakery</title>
  <style>
    * { box-sizing: border-box; }
    body { margin: 0; font-family: system-ui, sans-serif; color: #1f2937; }
    img { max-width: 100%; height: auto; }
    main { max-width: 1140px; margin: 0 auto; padding: 1rem; }
    .jl-events { list-style: none; margin: 0; padding: 0; }
    .jl-event { padding: 1rem; border: 1px solid #e7d8c9; margin-bottom: 1rem; }
    @media (min-width: 768px) {
      .jl-events { display: flex; gap: 1rem; }
      .jl-event { flex: 1; margin-bottom: 0; }
    }
  </style>
</head>
<body>
  <main>
    <h1>Events at Juniper Lane</h1>
    <ul class="jl-events">
      <li class="jl-event">
        <h2>Saturday Sourdough Class</h2>
        <p>$45 · 10:00–12:00 · max 8 people</p>
```