RANDY CONNOLLY
RICARDO HOAR

Fundamentals of
WEB DEVELOPMENT

Third Edition

Fundamentals of Web Development

Third Edition

This page intentionally left blank

Fundamentals of Web Development

Third Edition

Randy Connolly
Mount Royal University, Calgary

Ricardo Hoar
Silicon Hanna Inc.

Content Development: Tracy Johnson
Content Management: Dawn Murrin, Tracy Johnson
Content Production: Carole Snyder
Product Management: Holly Stark
Product Marketing: Wayne Stevens
Rights and Permissions: Anjali Singh

Please contact https://support.pearson.com/getsupport/s/ with any queries on this content

Cover Image by Randy Connolly

Library of Congress Cataloging-in-Publication Data
Names: Connolly, Randy, author. | Hoar, Ricardo, author.
Title: Fundamentals of web development / Randy Connolly, Mount Royal University, Calgary, Ricardo Hoar, Silicon Hanna Inc.
Description: Third edition. | NY, NY : Pearson, 2022. | Includes bibliographical references and index.
Identifiers: LCCN 2020052860 | ISBN 9780135863336 (hardcover) | ISBN 0135863333 (hardcover)
Subjects: LCSH: Web site development.
Classification: LCC TK5105.888 .C658 2022 | DDC 006.7–dc23
LC record available at https://lccn.loc.gov/2020052860

ScoutAutomatedPrintCode

ISBN 10: 0-13-586333-3
ISBN 13: 978-0-13-586333-6

To all whose lives have been afflicted by the COVID-19 pandemic and especially for the loved ones that have been lost to it.

Randy Connolly

To every student working to build a better world.

Ricardo Hoar

Brief Table of Contents

Table of Contents

Preface

Welcome to the *Fundamentals of Web Development*. This textbook is intended to cover the broad range of topics required for modern web development and is suitable for intermediate to upper-level computing students. A significant percentage of the material in this book has also been used by the authors to teach web development principles to first-year computing students and to non-computing students as well.

One of the difficulties that we faced when planning this book is that web development is taught in a wide variety of ways and to a diverse student audience. Some instructors teach a single course that focuses on server-side programming to third-year students; other instructors teach the full gamut of web development across two or more courses, while others might only teach web development indirectly in the context of a networking, HCI, or capstone project course. We have tried to create a textbook that supports learning outcomes in all of these teaching scenarios.

What Is Web Development?

Web development is a term that takes on different meanings depending on the audience and context. In practice, web development requires people with complementary but distinct expertise working together toward a single goal. Whereas a graphic designer might regard web development as the application of good graphic design strategies, a database administrator might regard it as a simple interface to an underlying database. Software engineers and programmers might regard web development as a classic software development task with phases and deliverables, where a system administrator sees a system that has to be secured from attackers. With so many different classes of users and meanings for the term, it's no wonder that web development is often poorly understood. Too often, in an effort to fully cover one aspect of web development, the other principles are ignored altogether, leaving students without a sense of where their skills fit into the big picture.

A true grasp of web development requires an understanding of multiple perspectives. As you will see, the design and layout of a website are closely related to the code and the database. The quality of the graphics is related to the performance and configuration of the server, and the security of the system spans every aspect of development. All of these seemingly independent perspectives are interrelated and,

therefore, a web developer (of any type) should have a foundational understanding of all aspects, even if he/she only possesses expertise in a handful of areas.

What's New in the Third Edition?

The first edition of this title was mainly written in the first half of 2013 and then published in early 2014. The second edition was mainly written in the first half of 2016 and then published in early 2017. This edition was mainly written in the first half of 2020.

The focus of the book has always been on the conceptual and practical fundamentals of web development. As such, many of the topics covered in the book are as important today as they were when we wrote the first edition in 2013. Nonetheless, the field of web development is constantly in flux, which has resulted in many changes in the underlying technologies of web development since the first and second editions were written. The third edition reflects both these recent changes as well as those enduring fundamental aspects of web development.

Over the past decade, the key technology stack within real-world web development has migrated away from back-end technologies such as PHP, JSP, and ASP.NET. While these technologies are still important, the front-end technology of JavaScript has become the focal practice of most web developers today. This edition reflects this transformation in real-world practices.

Some of the key changes in this edition include the following:

- Existing chapters have been revised based on user feedback. Instructor focus groups from 2018 provided helpful information on what content was missing or needed improvement. Emailed suggestions from other instructors and our own student feedback also informed changes to existing content.
- Updated, expanded, or new coverage of a wide-variety of topics that reflect current approaches to web development. Some of these topics include CSS preprocessors, CSS design principles, ES6+ language additions, web and browser APIs, React, Node, TypeScript, SQLite and NoSQL databases, GraphQL, serverless computing, caching, new security vulnerabilities, JWT authentication, DevOps, continuous integration/deployment, and microservice architectures.
- Enhanced coverage of contemporary JavaScript. This edition has five chapters on both front-end and back-end JavaScript (almost 300 pages versus the second edition's three chapters of 170 pages).
- Dedicated chapters on React and Node. Both of these have become an essential skill for contemporary web developers.
- New pedagogical features within most chapters. These include Test Your Knowledge exercises and Essential Solutions boxes. The former are short exercises for student to apply the knowledge in the section(s) they have

just read, while the latter provide quick guidance to the reader on how to accomplish common tasks.

- Updated art style throughout most of the book.
- Most of the end-of-chapter projects have been revised or replaced.

Features of the Book

To help students master the fundamentals of web development, this book has the following features:

- **Covers both the concepts and the practice of the entire scope of web development.** Web development can be a difficult subject to teach because it involves covering a wide range of theoretical material that is technology independent as well as practical material that is very specific to a particular technology. This book comprehensively covers both the conceptual and practical side of the entire gamut of the web development world.
- **Comprehensive coverage of a modern Internet development platform.** In order to create any kind of realistic Internet application, readers require detailed knowledge of and practice with a single specific Internet development platform. This book covers HTML, CSS, JavaScript, and two server-side stacks (PHP and MySQL, as well as Node and MongoDB). The book also covers the key concepts and infrastructures—such as web protocols and architecture, security, hosting provision, and server administration—that are important learning outcomes for any web development course.
- **Focused on the web development reality of today's world and in anticipation of future trends.** The world of web development has changed remarkably in the past decade. Fewer and fewer sites are being created from scratch; instead, many developers make use of existing sophisticated frameworks and environments. This book includes coverage of essential frameworks such as Bootstrap, React, and WordPress.
- **Sophisticated, realistic, and engaging case studies.** Rather than using simplistic "Hello World" style web projects, this book makes extensive use of three case studies: an art store, a travel photo sharing community, a stock trading site, and a movie review site. For all the case studies, supporting material such as the visual design, images, and databases are included. We have found that students are more enthusiastic and thus work significantly harder with attractive and realistic cases.
- **Content presentation suitable for visually oriented learners.** As long-time instructors, the authors are well aware that today's students are often extremely reluctant to read long blocks of text. As a result, we have tried to make the

content visually pleasing and to explain complicated ideas not only through text but also through diagrams.

- **Content that is the result of over 25 years of classroom experience** (in college, university, and adult continuing education settings) teaching web development. The book's content also reflects the authors' deep experience engaging in web development work for a variety of international clients.
- **Additional instructional content available online.** Rather than using long programming listings to teach ideas and techniques, this book uses a combination of illustrations, short color-coded listings, and separate lab exercises. These step-by-step tutorials are not contained within the book, but are available online at **www.funwebdev.com**. Code listings within book as well as starting files for projects are publicly available on GitHub.
- **Complete pedagogical features for the student.** Each chapter includes learning objectives, margin notes, links to step-by-step online labs, advanced tips, keyword highlights, test your knowledge exercises, essential solution boxes, end-of-chapter review questions, and three different case study exercises.

Organization of the Book

The chapters in *Fundamentals of Web Development* can be organized into three large sections.

- **Foundational client-side knowledge (Chapters 1–10).** These first chapters cover the foundational knowledge needed by any front-end web developer. This includes a broad introduction to web development (Chapter 1), how the web works (Chapter 2), HTML (Chapters 3 and 5), CSS (Chapters 4 and 7), web media (Chapter 6), and JavaScript (Chapters 8–11).
- **Essential server-side development (Chapters 12–16).** Despite the increasing importance of JavaScript-based development, learning server-side development is still the essential skill taught in most web development courses. The two most popular server-side environments are covered in Chapter 12 (PHP) and Chapter 13 (Node). Database-driven web development is covered in Chapter 14, while state management is covered in Chapter 15.
- **Specialized topics (Chapters 16–18).** Contemporary web development has become a very complex field, and different instructors will likely have different interest areas beyond the foundational topics. As such, our book provides specialized chapters that cover a variety of different interest areas. Chapter 16 covers the vital topic of web security. Chapter 17 focuses on the web server with topics such as DevOps, hosting options, and server configuration. Finally, Chapter 18 covers search, social media integration, content management, advertising, and support tools and analytics.

Pathways through this Book

There are many approaches to teach web development and our book is intended to work with most of these approaches. It should be noted that this book has more material than can be plausibly covered in a single semester course. This is by design as it allows different instructors to chart their own unique way through the diverse topics that make up contemporary web development.

We do have some suggested pathways through the materials (though you are welcome to chart your own course), which you can see illustrated in the pathway diagrams.

- **All the web in a single course**. Many computing programs only have space for a single course on web development. This is typically an intermediate or upper-level course in which students will be expected to do a certain amount of learning on their own. In this case, we recommend covering Chapters 1–5, 8, 9, 12 or 13, 14, and 16.
- **Client-focused course for introductory students**. Some computing programs have a web course with minimal programming that may be open to non-major students or which acts as an introductory course to web development for major students. For such a course, we recommend covering Chapters 1–7. You can use Chapters 8 and 9 to introduce client-side scripting if desired.
- **Front end focused course for intermediate students**. For courses that wish to focus on front-end development, you could cover Chapters 1–11 as well as Chapter 13 and parts of Chapter 14.
- **Infrastructure-focused course**. In some computing programs the emphasis is less on the particulars of web programming and more on integrating web technologies into the overall computing infrastructure within an organization. Such a course might cover Chapters 1, 2, 3, 4, 8, 13, 16, and 17 with an option to include some topics from Chapters 6, 14, 15, and 18.

For the Instructor

Web development courses have been called "unteachable" and indeed teaching web development has many challenges. We believe that using our book will make teaching web development significantly less challenging.

The following instructor resources are available at **www.pearsonhighered.com/cs-resources/**:

- Attractive and comprehensive PowerPoint presentations (one for each chapter).
- Images and databases for all the case studies.
- Solutions to end-of-chapter projects.
- Additional questions in exam bank.

Many of the code listings and examples used in the book are available on GitHub (**github.com/funwebdev-3rd-ed**).

For the Student

There are a variety of student resources available on GitHub (**github.com/funwebdev-3rd-ed**), the publisher's resource site (**www.pearsonhighered.com/cs-resources/**), and the book's website (**www.funwebdev.com**). These include:

- All code listings organized by chapter.
- Starting files, images, and database scripts for all end-of-chapter projects.
- Starting files and solutions to all Test Your Knowledge exercises.
- Instructions for lab exercises for each chapter.
- Starting files for all labs.
- Video lectures for a selection of chapter topics.

Why This Book?

The ACM computing curricula for computer science, information systems, information technology, and computing engineering all recommend at least a single course devoted to web development. As a consequence, almost every postsecondary computing program offers at least one course on web development.

Despite this universality, we could not find a suitable textbook for these courses that addressed both the theoretical underpinnings of the web together with modern web development practices. Complaints about this lack of breadth and depth have been well documented in published accounts in the computing education research literature. Although there are a number of introductory textbooks devoted to HTML and CSS, and, of course, an incredibly large number of trade books focused on specific web technologies, many of these are largely unsuitable for computing major students. Rather than illustrating how to create simple pages using HTML and JavaScript with very basic server-side capabilities, we believed that instructors increasingly need a textbook that guides students through the development of realistic, enterprise-quality web applications using contemporary Internet development platforms and frameworks.

This book is intended to fill this need. It covers the required ACM web development topics in a modern manner that is closely aligned with contemporary best practices in the real world of web development. It is based on our experience teaching a variety of different web development courses since 1997, our working professionally in the web development industry, our research in published accounts in the computing education literature, and in our corresponding with colleagues across the world. We hope that you find that this book does indeed satisfy your requirements for a web development textbook!

Acknowledgments

A book of this scale and scope incurs many debts of gratitude. We are first and foremost exceptionally grateful to Matt Goldstein, formerly the Acquisitions Editor at Pearson for the first two editions, championed the book and guided the overall process of bringing the book to market. Tracy Johnson, the Content Development Manager for Computer Science, navigated this edition through the complexities of the new electronic-first approach to textbook publishing. Louise Capulli was once again the very capable Project Manager who facilitated communication between the often finicky authors and the production team. Carole Synder from Pearson also contributed throughout the writing and production process. We would like to thank Pradeep Subramani and his team at Integra Software Services for the work they did on the postproduction side. We would also like to thank Rose Kernan, proofreader, who made sure that the words and illustrations actually work to tell a story that makes sense.

Reviewers help ensure that a textbook reflects more than just the authors' perspective. For this edition, the book was immeasurably improved by our talented and close-eyed reviewer, Jordan Pratt of Mount Royal University. A variety of very helpful students provided inspirational feedback on labs and lecture material. Some of these include: Farsos Bulsara, Raj Dutta, Hamid Hemani, Peter Huang, Jason Hutson, Andrews Juchem, Sarfaraz Kermali, Shuntian Li, Robert Martin, Brett Miller, Peter Morrison, and Renato Niro. Indeed, to be honest, we should list all of our students over the past five years here, as they have improved our insight and acted as non-voluntary guinea pigs in the evolution of our thinking on teaching web development.

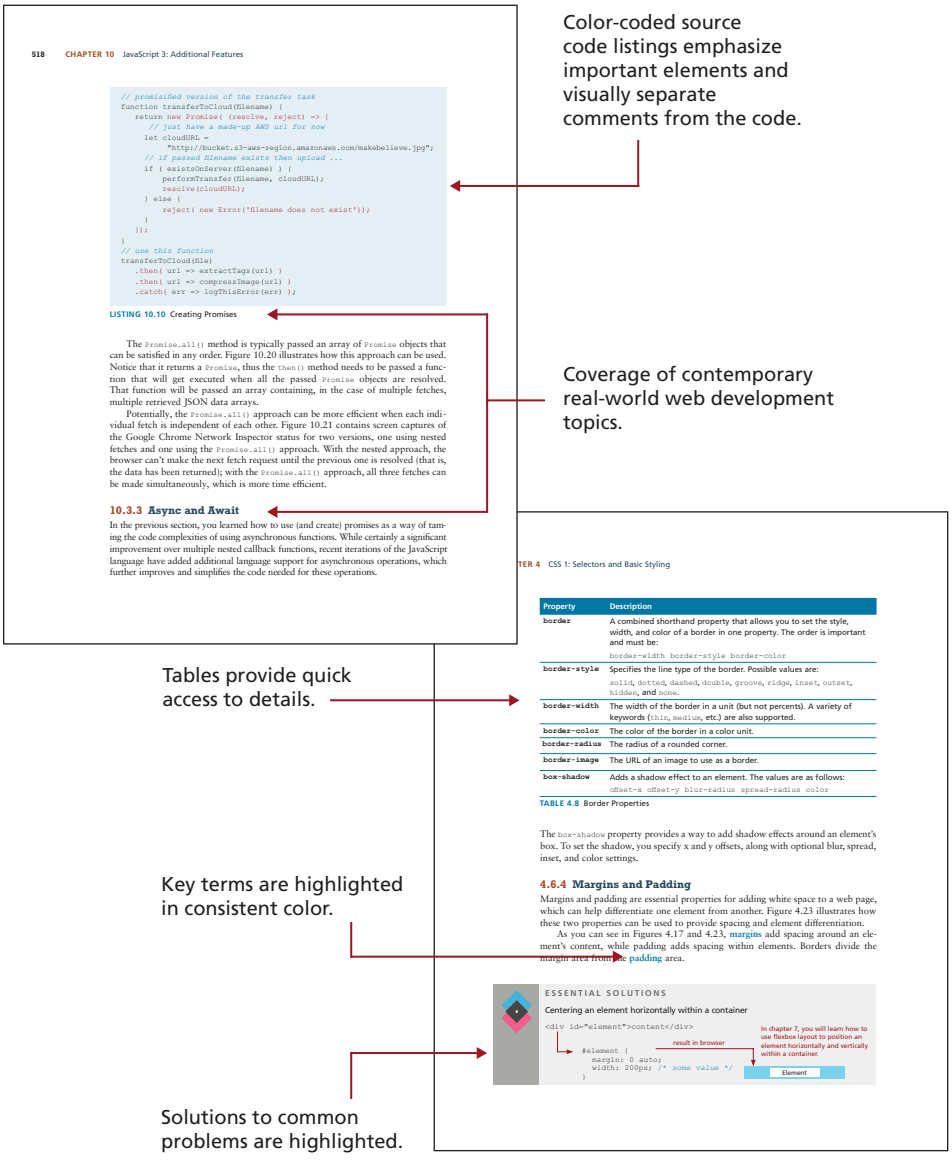
There are many others who helped guide our thinking, provided suggestions, or made our administrative and teaching duties somewhat less onerous. While we cannot thank everyone, Randy Connolly is especially grateful to Brigitte Jellinek for inviting him to spend a semester in 2017 at Salzburg University of Applied Sciences, as it provided early inspiration for many of the changes made in this edition. We would also like to express our gratitude to all the instructors who took the time to email us about the first two editions. Their praise, suggestions for improvements, or their admonition for mistakes or omissions was always very welcome and hopefully resulted in a better third edition.

We are very appreciative of those who donated photos for the Travel case study used throughout the book: Robert Boschman, Alexander Connolly, Norman Connolly,

Mark Eagles, Sonya Flessati, Emily Girard, Mike Gouthro, Jordan Kidney, Roy Kuhnlein, and Jocelyn Sealy. For this edition, our Art case study was able to take advantage of the public-spirited and generous open content policies of the Rijksmuseum, the J. Paul Getty Museum, and the National Gallery of Art (Washington, DC).

From the early inception of the book in May of 2012 all the way to the conclusion of this edition in the late months of 2020, Dr. Janet Miller provided incredible and overwhelming encouragement, understanding, and feedback for which Randy Connolly will be always grateful. Joanne Hoar, holding a M.Sc. in computer science, has always been an inspiration for Ricardo Hoar, so he apologizes profusely for the systemic racism and sexism among computer science faculty that has excluded her, a brilliant programmer, from gainful employment in academia. Finally, we want to thank our children, Alexander Connolly, Benjamin Connolly, Mark Miller, Hann Miller, Archimedes Hoar, Curia Hoar, and Hypatia Hoar, who saw less of their fathers during this time but were always on our minds.

Visual Walkthrough



Color-coded source code listings emphasize important elements and visually separate comments from the code.

Coverage of contemporary real-world web development topics.

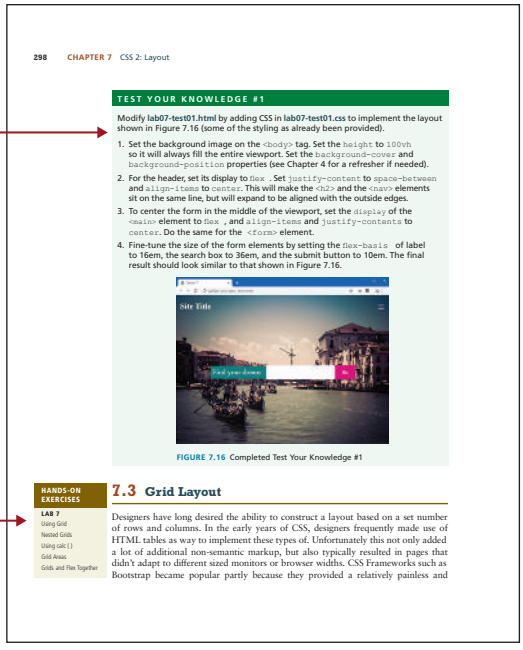
Tables provide quick access to details.

Key terms are highlighted in consistent color.

Solutions to common problems are highlighted.

Test Your Knowledge sections provide opportunities for readers to apply their knowledge.

Separate hands-on lab exercises (available online) give readers opportunity to practically apply concepts and techniques covered in the text.



298 CHAPTER 7 CSS 2: Layout

TEST YOUR KNOWLEDGE #1

Modify **lab07-test01.html** by adding CSS in **lab07-test01.css** to implement the layout shown in Figure 7.16 (some of the styling as already been provided).

1. Set the background image on the `<body>` tag. Set the `height` to `100vh` so it will always fill the entire viewport. Set the `background-cover` and `background-position` properties (see Chapter 4 for a refresher if needed).
2. For the header, set its display to `flex` . Set `justify-content` to `space-between` and `align-items` to `center`. This will make the `<h2>` and the `<nav>` elements sit on the same line, but will expand to be aligned with the outside edges.
3. To center the form in the middle of the viewport, set the `display` of the `<main>` element to `flex` , and `align-items` and `justify-contents` to `center`. Do the same for the `<form>` element.
4. Fine-tune the size of the form elements by setting the `flex-basis` of label to 16em, the search box to 36em, and the submit button to 10em. The final result should look similar to that shown in Figure 7.16.

FIGURE 7.16 Completed Test Your Knowledge #1

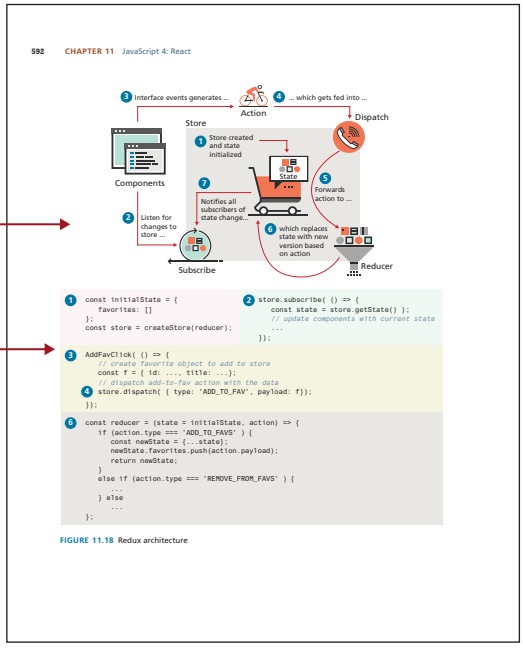
HANDS-ON EXERCISES
LAB 7
Using Grid
Nested Grids
Using calc ( )
Grid Areas
Grids and Flex Together

7.3 Grid Layout

Designers have long desired the ability to construct a layout based on a set number of rows and columns. In the early years of CSS, designers frequently made use of HTML tables as way to implement these types of. Unfortunately this not only added a lot of additional non-semantic markup, but also typically resulted in pages that didn't adapt to different sized monitors or browser widths. CSS Frameworks such as Bootstrap became popular partly because they provided a relatively painless and

Hundreds of illustrations help explain especially complicated processes.

Important algorithms are illustrated visually to help clarify understanding.



592 CHAPTER 11 JavaScript 4: React


```
1 const initialState = {
    favorites: []
  };
  const store = createStore(reducer);

2 store.subscribe( () => {
    const state = store.getState() );
    // update components with current state
    ...
  });

3 AddFavClick( () => {
    // create favorite object to add to store
    const f = { id: ..., title: ...};
    // dispatch add-to-fav action with the data
  4 store.dispatch( { type: 'ADD_TO_FAV', payload: f});
  });

6 const reducer = (state = initialState, action) => {
    if (action.type === 'ADD_TO_FAVS' ) {
      const newState = {...state};
      newState.favorites.push(action.payload);
      return newState;
    }
    else if (action.type === 'REMOVE_FROM_FAVS' ) {
      ...
    } else
      ...
  };
```

FIGURE 11.18 Redux architecture

Note and Pro Tip boxes emphasize important concepts and practical advice.

Tools Insight sections introduce many of the most essential tools used in web development.

Tangential material has been moved into Dive Deeper sections, thereby keeping the main text more focused.

Extended Example sections provide detailed guidance in the application of a chapter's content.

374 CHAPTER 8 JavaScript 1: Language Fundamentals

Probably the most common postloop operation is to increment a counter variable, as shown in Figure 8.11. An alternative way to increment this counter is to use `i+=1` instead of `i++`.

There are two additional, more specialized, variations of the basic `for` loop. There is a `for...in` loop and in ES6 and beyond, a `for...of` loop. The `for...in` loop is used for iterating through enumerable properties of an object, while the more useful `for...of` loop is used to iterate through iterable objects, and will be demonstrated in the next section on arrays.

NOTE

Infinite while loops can happen if you are not careful, and since the scripts are executing on the client computer, it can appear to them that the browser is "locked" while endlessly caught in a loop, processing. Some browsers will even try to terminate scripts that execute for too long a time to mitigate this unpleasantness.

DIVE DEEPER: ERRORS USING TRY AND CATCH

When the browser's JavaScript engine encounters a runtime error, it will throw an **exception**. These exceptions interrupt the regular, sequential execution of the program and can stop the JavaScript engine altogether. However, you can optionally catch these errors (and thus prevent the disruption) using the **try . . . catch block** as shown below.

```
try {
   nonexistantfunction("hello");
}
catch(err) {
   alert ("An exception was caught:" + err);
}
```

in JavaScript errors, it messages. The `throw` built-in exceptions as

470 CHAPTER 9 JavaScript 2: Using JavaScript

TOOLS INSIGHT

JavaScript has become one of the most important programming languages in the world. As a result, there has been tremendous growth in the availability of tools to help with different aspects of JavaScript development. We could quite easily fill an entire chapter of this book examining just a small subset of these tools![4] In this Tools Insight section, we are going to look at just two JavaScript tools; subsequent JavaScript chapters will include additional Tools Insight sections that will introduce others.

The first, and most important, JavaScript tool is one that you have using, namely, your browser. All modern browsers now include sophisticate and profiling tools. Just as the authors' grandparents used to regale us in c with stories of walking miles to school in the snow going uphill there a authors sometimes tell our students what it used to be like in the late 19 ming in JavaScript without having access to any type of debugger. Now th ship! Thankfully in today's more civilized and developed world, you can add step through code line by line, and inspect variables all within the con browser, as shown in Figure 9.22.

Contemporary browsers provide additional tools that are essential f JavaScript development. As more and more functionality has migrated fro to the client, it has become increasingly important to assess the performar JavaScript code. Figure 9.23 illustrates the Profile view of a page's JavaS mance. It allows a developer to pinpoint time-consuming functions or visu mance as timeline charts.

FIGURE 9.22 Debugging within the FireFox browser

9.4 Event Types 453

EXTENDED EXAMPLE

Now that we have covered the basics of working with events and the DOM, we are going to put this knowledge to work in an extended example. In the **example.html** page, an image is displayed with some related text as well as a Hide button. Using some CSS filters and transitions along with some JavaScript event handling, the example will fade the text in and out of visibility when the user clicks on the button. Also, the example will apply or remove a grayscale filter to the image when the user moves the mouse in or out of the image.

When Hide button is clicked, the text fades to transparent

The label for the button is also changed

When text is transparent, the element for that text is hidden, thus removing the extra space for the hidden element

If the user mouses over the image, then the grayscale filter is applied to the image

If the user mouses out of the image, then the grayscale filter is removed from the image

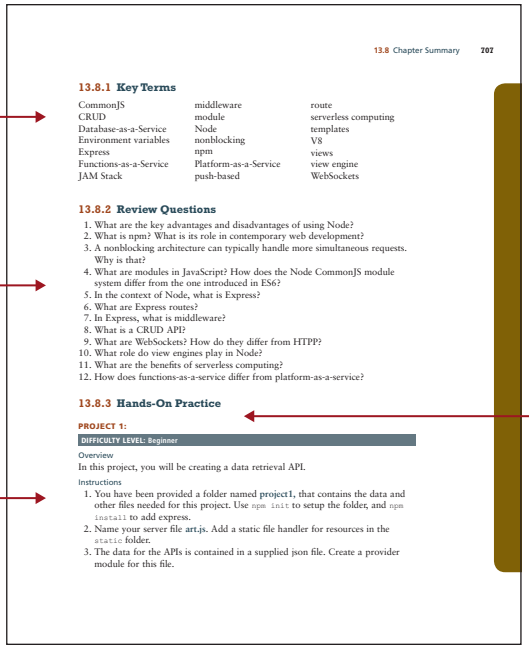
(continued)

Key terms appear again at end of chapter.

Review questions at end of chapter provide opportunity for self-testing.

Projects contain step-by-step instructions of varying difficulty.

Each chapter ends with three projects that allow the reader to practice the material covered in the chapter within a realistic context.



13.8 Chapter Summary 707

13.8.1 Key Terms

CommonJS	middleware	route
CRUD	module	serverless computing
Database-as-a-Service	Node	templates
Environment variables	nonblocking	V8
Express	npm	views
Functions-as-a-Service	Platform-as-a-Service	view engine
JAM Stack	push-based	WebSockets

13.8.2 Review Questions

1. What are the key advantages and disadvantages of using Node?
2. What is npm? What is its role in contemporary web development?
3. A nonblocking architecture can typically handle more simultaneous requests. Why is that?
4. What are modules in JavaScript? How does the Node CommonJS module system differ from the one introduced in ES6?
5. In the context of Node, what is Express?
6. What are Express routes?
7. In Express, what is middleware?
8. What is a CRUD API?
9. What are WebSockets? How do they differ from HTPP?
10. What role do view engines play in Node?
11. What are the benefits of serverless computing?
12. How does functions-as-a-service differ from platform-as-a-service?

13.8.3 Hands-On Practice

PROJECT 1:

DIFFICULTY LEVEL: Beginner

Overview

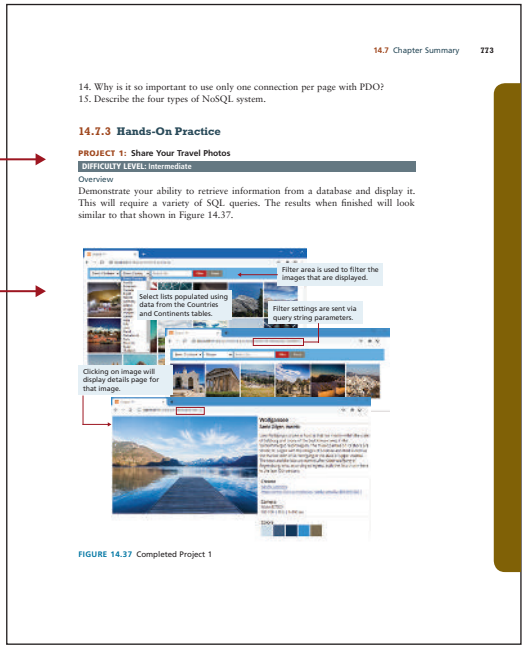
In this project, you will be creating a data retrieval API.

Instructions

1. You have been provided a folder named **project1**, that contains the data and other files needed for this project. Use `npm init` to setup the folder, and `npm install` to add express.
2. Name your server file **art.js**. Add a static file handler for resources in the `static` folder.
3. The data for the APIs is contained in a supplied json file. Create a provider module for this file.

Attractive and realistic case studies help engage the readers' interest.

All images, starting files, database scripts, and other material for each of the end of chapter projects are available for download.



14.7 Chapter Summary 773

14. Why is it so important to use only one connection per page with PDO?
15. Describe the four types of NoSQL system.

14.7.3 Hands-On Practice

PROJECT 1: Share Your Travel Photos

DIFFICULTY LEVEL: Intermediate

Overview

Demonstrate your ability to retrieve information from a database and display it. This will require a variety of SQL queries. The results when finished will look similar to that shown in Figure 14.37.

FIGURE 14.37 Completed Project 1

Introduction to Web Development

1

CHAPTER OBJECTIVES

In this chapter you will learn . . .

- About web development in general
- The history of the Internet and World Wide Web
- Fundamental concepts that form the foundation of the Internet
- About the hardware and software that support the Internet
- The range of careers and companies in web development

This chapter introduces the World Wide Web (WWW). It begins with an answer to the broad question, what is web development. It then progresses from that large question to a brief history of the Internet. It also provides an overview of key Internet technologies and ideas that make web development possible. To truly understand these concepts in depth, one would normally take courses in computer science or information technology (IT) covering networking principles. If you find some of these topics too in-depth or advanced, you may decide to skip over some of the details here and return to them later.

1.1 A Complicated Ecosystem

You may remember from your primary school science class that nature can be characterized as an ecosystem, a complex system of interrelationships between living and nonliving elements of the environment. As visualized in Figure 1.1, web development can also be understood as an ecosystem, one that builds on existing technologies (URL, DNS, and Internet), and contributes new protocols and standards (HTTP, HTML, and JavaScript) that facilitate client-server interactions. As this ecosystem matures, new client and server technologies, frameworks, and platforms continue to be developed in support of the web (PHP, Node, React etc.). The rich web development ecosystem has created entirely new areas of interest for both research and businesses including search engines, social networks, ecommerce, content management systems, and more.

Just as you don't need to know everything about worms, trees, birds, amphibians, and dirt to be a biologist, you don't necessarily need to understand every concept in Figure 1.1 in complete depth in order to be successful as a web developer. Nonetheless, it is important to see how this complicated network of concepts and technologies defines the scope of modern web development, and how concepts from each chapter fit into the bigger picture.

In Figure 1.1, web development is visualized as a series of related platforms. The two teal platforms represent the topics typically understood to constitute web development.

There are two distinct development platforms in the diagram which represents the fact that there are two distinct forms of development: front end and back end. The term **front end** refers to those technologies that run in the browser: in this diagram, they are HTML, CSS, JavaScript, and a wide-range of front-end oriented frameworks such as React; much of this book is focused on these technologies. The term **back end** refers to those technologies that run on the server. The book focuses on two of the most popular back-end development technologies—PHP and Node—and covers a variety of other back-end-related topics such as APIs, databases, and a variety of server-based development tools.

The platform at the top of the diagram contains a variety of topics that are typically dependent upon first having knowledge of the development technologies. These "advanced" topics are typically an important part of "real" web development; however, not all developers require expertise in all of these topics.

At the bottom of the diagram are two white platforms that represent the infrastructural topics of web development. These include the servers and networking topics that constitute the infrastructure of the web. To fully learn about the infrastructure of the web is beyond the scope of this book; nonetheless, it is important for any contemporary web developer to have some understanding of the basics of this infrastructure.

FIGURE 1.1 The web development ecosystem

Finally, the light-blue platform just below the back-end platform represents a variety of foundational topics that are important for anyone who works within the programming or the infrastructural side of the web. This includes the key protocols and standards of the web, such as HTTP and DNS, as well as the vital topic of security.

The textbook also covers the topics of these different platforms but focuses especially on the front-end and back-end development topics, since most entry-level web development positions require proficiency with these topics.

It is the perspective of the book, however, that web development is more than just markup and programming. In recent years, knowledge of the infrastructure upon which the web is built has become increasingly important for practicing web developers. For this reason, this chapter (and the next) journeys into the basement of foundational protocols, hardware infrastructure, and key terminology.

The last third of the book corresponds to some of the topics covered in the top platform. If you are taking a single course in web development, you might not have time to cover these more "advanced" topics. Yet, as far as real-world web development, they are just as important as the more recognizable ones on the explicitly development-focused platforms. We would encourage all of our readers to ascend to the upper-platform topics during their journey to become a web developer with this book. But before we go there, it is now time to begin with the foundational knowledge and learn more about web development in general.

1.2 Definitions and History

The World Wide Web (WWW or simply the web) is certainly what most people think of when they see the word "Internet." But the web is only a subset of the Internet, as illustrated in Figure 1.2. While this book is focused on the web, part of this chapter is also devoted to a broad understanding of that larger circle labeled the "Internet."

1.2.1 A Short History of the Internet

The history of telecommunication and data transport is a long one. There is a strategic advantage in being able to send a message as quickly as possible (or at least, more quickly than your competition). The Internet is not alone in providing instantaneous digital communication. Earlier technologies such as the radio, the telegraph, and the telephone provided the same speed of communication, albeit in an analog form.

Telephone networks in particular provide a good starting place to learn about modern digital communications. In the telephone networks of the past, calls were routed through operators who physically connected the caller and the receiver by connecting a wire to a switchboard to complete a circuit. These operators were around in some areas for almost a century before being replaced with automatic mechanical switches that did the same job: physically connect caller and receiver.

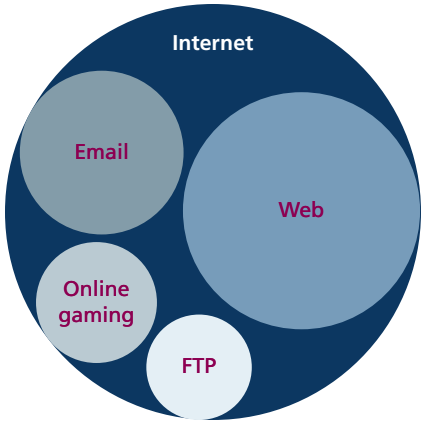


FIGURE 1.2 The web as a subset of the Internet

One of the weaknesses of having a physical connection is that you must establish a link and maintain a dedicated circuit for the duration of the call. This type of network connection is sometimes referred to as **circuit switching** and is shown in Figure 1.3.

The problem with circuit switching is that it can be difficult to have multiple conversations simultaneously (which a computer might want to do). It also requires more **bandwidth**, since even the silences are transmitted (that is, unused capacity in the network is not being used efficiently).

Bandwidth is a measurement of how much data can (maximally) be transmitted along a communication channel. Normally measured in bits per second (bps), this measurement differs according to the type of Internet access technology you are using. A dial-up 56-Kbps modem has far less bandwidth than a 10-Gbps fiber optic connection.

In the 1960s, as researchers explored digital communications and began to construct the first networks, the research network ARPANET was created. ARPANET did

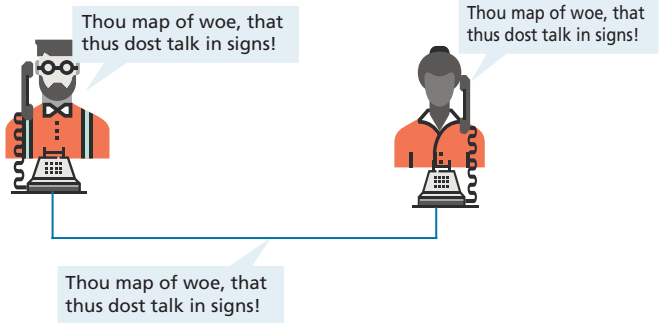


FIGURE 1.3 Telephone network as example of circuit switching

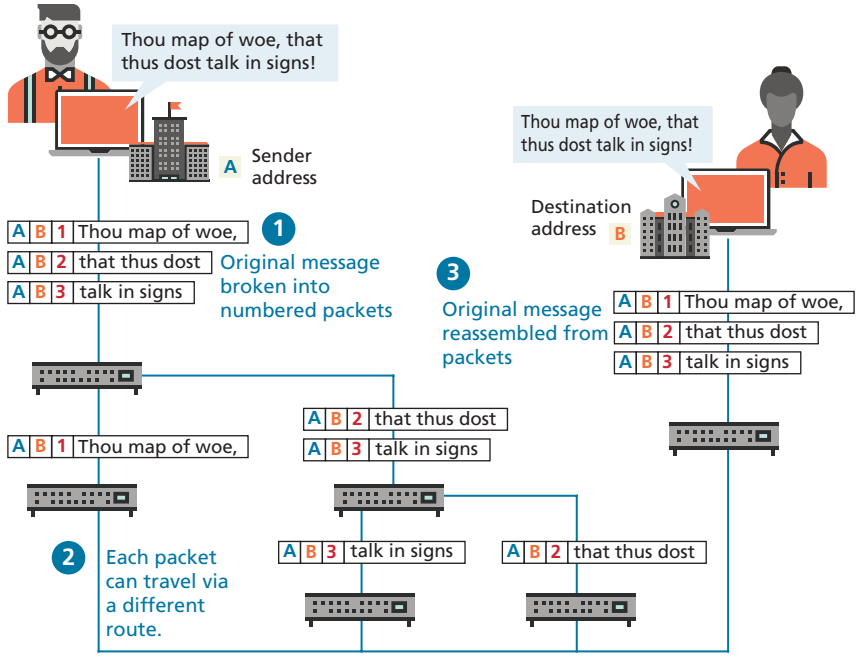


FIGURE 1.4 Internet network as example of packet switching

not use circuit switching but instead used an alternative communications method called **packet switching**. A packet-switched network does not require a continuous connection. Instead, it splits the messages into smaller chunks called **packets** and routes them to the appropriate place based on the destination address. The packets can take different routes to the destination, as shown in Figure 1.4. This may seem a more complicated and inefficient approach than circuit switching but is in fact more robust (it is not reliant on a single pathway that may fail) and a more efficient use of network resources (since a circuit can communicate data from multiple connections).

This early ARPANET network was funded and controlled by the United States government and was used exclusively for academic and scientific purposes. The early network started small, with just a handful of connected university campuses and research institutions and companies in 1969, and grew to a few hundred by the early 1980s.

At the same time, alternative networks were created like X.25 in 1974, which allowed (and encouraged) business use. USENET, built in 1979, had fewer restrictions still, and as a result grew quickly to 550 connected machines by 1981. Although there was growth in these various networks, the inability for them to communicate with each other was a real limitation. To promote the growth and unification of the disparate networks, a suite of **protocols** was invented to unify the networks. A protocol is the name given to a formal set of publicly available rules

that manage data exchange between two points. Communications protocols allow any two computers to talk to one another, so long as they implement the protocol.

By 1981, protocols for the Internet were published and ready for use.[1,2] New networks built in the United States began to adopt the **TCP/IP (Transmission Control Protocol/Internet Protocol)** communication model (discussed in the next section), while older networks were transitioned over to it.

Any organization, private or public, could potentially connect to this new network so long as they adopted the TCP/IP protocol. On January 1, 1983, TCP/IP was adopted across all of ARPANET, marking the end of the research network that spawned the Internet.[3] Over the next two decades, TCP/IP networking was adopted across the globe.

1.2.2 The Birth of the Web

The next decade saw an explosion in the number of users, but the Internet of the late 1980s and the very early 1990s did not resemble the Internet we know today. During these early years, email and text-based systems were the extent of the Internet experience.

This transition from the old terminal and text-only Internet of the 1980s to the Internet of today is due to the invention and massive growth of the web. This invention is usually attributed to the British Tim Berners-Lee (now Sir Tim Berners-Lee), who, along with the Belgian Robert Cailliau, published a proposal in 1990 for a hypertext system while both were working at CERN (European Organization for Nuclear Research) in Switzerland. Shortly thereafter Berners-Lee developed the main features of the web.[4]

This early web incorporated the following essential elements that are still the core features of the web today:

- A Uniform Resource Locator (URL) to uniquely identify a resource on the WWW.
- The Hypertext Transfer Protocol (HTTP) to describe how requests and responses operate.
- A software program (later called web server software) that can respond to HTTP requests.
- Hypertext Markup Language (HTML) to publish documents.
- A program (later called a browser) that can make HTTP requests to URLs and that can display the HTML it receives.

URLs and the HTTP are covered in this chapter. This chapter will also provide a little bit of insight into the nature of web server software; HTML will require several chapters to cover in this book. Chapter 17 will examine the inner workings of server software in more detail.

So while the essential outline of today's web was in place in the early 1990s, the web as we know it did not really begin until **Mosaic**, the first popular graphical

browser application, was developed at the National Center for Supercomputing Applications at the University of Illinois Urbana-Champaign and released in early 1993 by Eric Bina and Marc Andreessen (who was a computer science undergraduate student at the time). Andreessen later moved to California and cofounded Netscape Communications, which released **Netscape Navigator** in late 1994. Navigator quickly became the principal web browser, a position it held until the end of the 1990s, when Microsoft's Internet Explorer (first released in 1995) became the market leader, a position it would hold for over a decade.

Also in late 1994, Berners-Lee helped found the **World Wide Web Consortium (W3C)**, which would soon become the international standards organization that would oversee the growth of the web. This growth was very much facilitated by the decision of CERN to not patent the work and ideas done by its employee and instead leave the web protocols and code-base royalty free.

NOTE

The **Request for Comments (RFC)** archive lists all of the Internet and WWW protocols, concepts, and standards. It started out as an unofficial repository for ARPANET information and eventually became the de facto official record. Even today new standards are published there.

1.2.3 Web Applications in Comparison to Desktop Applications

The user experience for a website is unlike the user experience for traditional desktop software. The location of data storage, limitations with the user interface, and limited access to operating system features are just some of the distinctions. However, as web applications have become more and more sophisticated, the differences in the user experience between desktop applications and web applications are becoming more and more blurred.

There are a variety of advantages and disadvantages to web-based applications in comparison to desktop applications. Some of the advantages of web applications include the following:

- They can be accessed from any Internet-enabled computer.
- They can be used with different operating systems and browser applications.
- They are easier to roll out program updates since only software on the server needs to be updated as opposed to every computer in the organization using the software.
- They have a centralized storage on the server, which means fewer security concerns about local storage (which is important for sensitive information such as health care data).

Unfortunately, in the world of IT, for every advantage, there is often a corresponding disadvantage; this is also true of web applications. Some of these disadvantages include the following:

- Requirement to have an active Internet connection (the Internet is not always available everywhere at all times).
- Security concerns about sensitive private data being transmitted over the Internet.
- Concerns over the storage, licensing, and use of uploaded data.
- Problems with certain websites not having an identical appearance across all browsers.
- Restrictions on access to operating system resources can prevent additional software from being installed and hardware from being accessed (like Adobe Flash on iOS).
- In addition, clients or their IT staff may have additional plugins added to their browsers, which provide added control over their browsing experience, but which might interfere with JavaScript, cookies, or advertisements.

We will continually try to address these challenges throughout the book.

DIVE DEEPER

One of the more common terms you might encounter in web development is the term "**intranet**" (with an "**a**"), which refers to an internal network using Internet protocols that is local to an organization or business. Intranet resources are often private, meaning that only employees (or authorized external parties such as customers or suppliers) have access to those resources. Thus, "**Internet**" (with an "**e**") is a broader term that encompasses both private (intranet) and public networked resources.

Intranets are typically protected from unauthorized external access via security features, such as firewalls or private IP ranges, as shown in Figure 1.5. Because intranets are private, search engines, such as Google, have limited or no access to content within them.

Due to this private nature, it is difficult to accurately gauge, for instance, how many web pages exist within intranets and what technologies are more common in them. Some especially expansive estimates guess that almost half of all web resources are hidden in private intranets.

Being aware of intranets is also important when one considers the job market and market usage of different web technologies. If one focuses just on the public Internet, it will appear that React, PHP, MySQL, and Node are the most commonly used web development stack. But when one adds in the private world of corporate intranets, other technologies such as ASP.NET, JSP, SharePoint, Oracle, SAP, and IBM WebSphere are just as important.

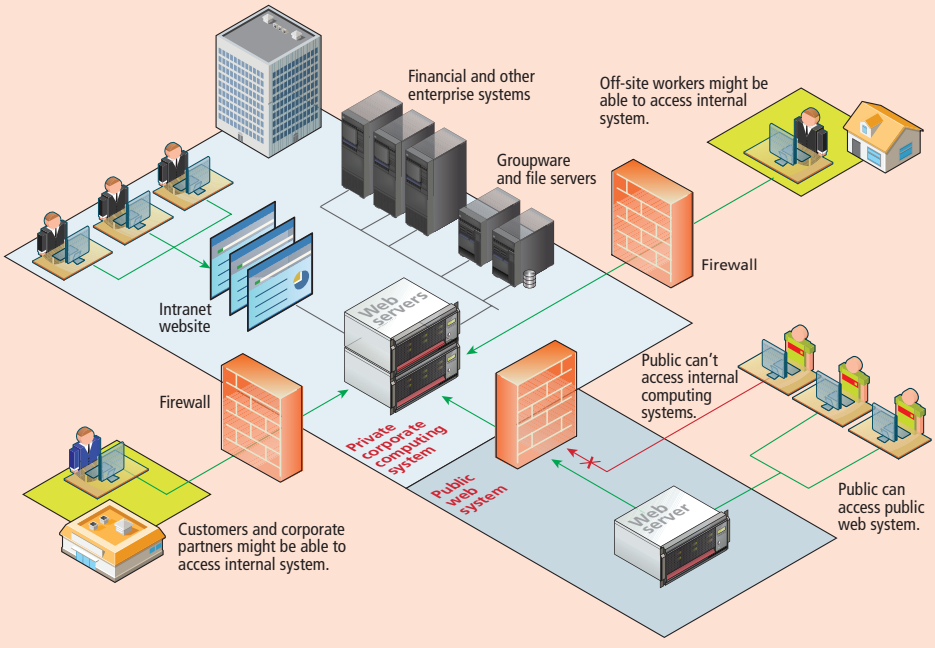


FIGURE 1.5 Intranet versus Internet

1.2.4 From Static to Dynamic (and Back to Static)

In the earliest days of the web, a **webmaster** (the term popular in the 1990s for the person who was responsible for creating and supporting a website) would publish web pages and periodically update them. Users could read the pages but could not provide feedback. The early days of the web included many encyclopedic, collection-style sites with lots of content to read (and animated icons to watch).

In those early days, the skills needed to create a website were pretty basic: one needed knowledge of HTML and perhaps familiarity with editing and creating images. This type of website was commonly referred to as a **static website**, in that it consists only of HTML pages that look identical for all users at all times. Figure 1.6 illustrates a simplified representation of the interaction between a user and a static website (it is referred to in the caption as "first generation" to differentiate it from the contemporary version of static sites).

Within a few years of the invention of the web, sites began to get more complicated as more and more sites began to use programs running on web servers to generate content dynamically. These server-based programs would read content from databases, interface with existing enterprise computer systems, communicate with financial institutions, and then output HTML that would be sent back to the users' browsers. This type of website is called a **dynamic server-side website** because

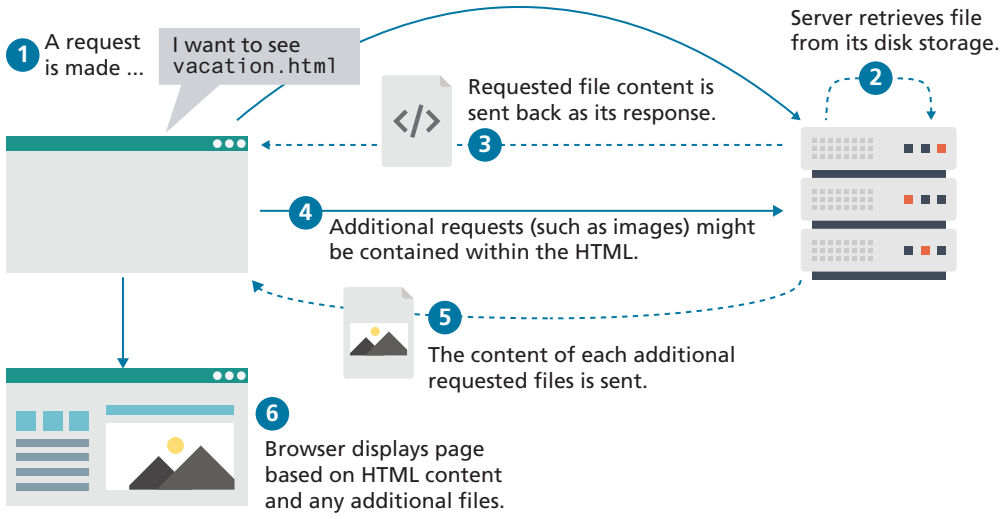


FIGURE 1.6 Static website (first generation)

the page content is being created dynamically by a program running on the server; this page content can vary from user to user. Figure 1.7 illustrates a very simplified representation of the interaction between a user and a dynamic website. The diagram also illustrates a conceptual division within web development that emerged as a consequence: the distinction between the front end and the back end. In the first decade of the 2000s, almost all of the focus in web development circles was on the back-end.

So while knowledge of HTML was still necessary for the creation of these dynamic websites, it became necessary to have programming knowledge as well. Moreover, by the late 1990s, additional knowledge and skills were becoming necessary, such as CSS, usability, and security.

By the end of the 2000s, a new buzzword entered the computer lexicon: **Web 2.0**. This term had two meanings, one for users and one for developers. For the users, Web 2.0 referred to an interactive experience where users could contribute *and* consume web content, thus creating a more user-driven web experience. Some of the most popular websites today fall into this category: Facebook, YouTube, and Wikipedia. This shift to allow feedback from the user, such as comments on a story, threads in a message board, or a profile on a social networking site has revolutionized what it means to use a web application.

For software developers, Web 2.0 also referred to a change in the paradigm of how dynamic websites are created. Programming logic, which previously existed only on the server, began to migrate more and more to the browser, which required learning JavaScript, a sometimes tricky programming language that runs in the browser. While programs running on servers were still necessary, the back end became "thinner" in

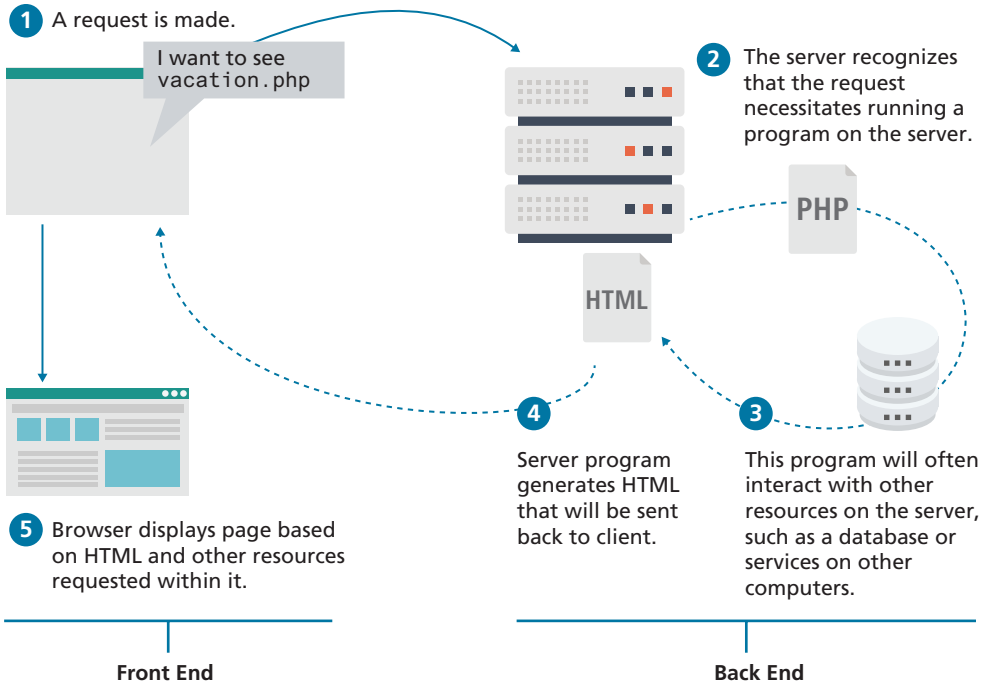


FIGURE 1.7 Dynamic Server-Side website

DIVE DEEPER

Why are programs necessary?

If HTML can be used to describe content on the web, why then are programs necessary? The problem with plain HTML is that it is *static*. That is, it is the same for all users at all times. A program, however, can be used to make a web site *dynamic*: that is, a site that can be be customized for different users and different content.

Furthermore, the sheer volume of content available on a typical website makes static HTML pages impractical. For instance, imagine a web site for displaying user photos. It might contain hundreds, if not thousands, or even millions of user photos. Having a separate HTML page for each photo would be completely unfeasible.

Instead, as shown in Figure 1.8, a single program running on the server can be used to display any of the photos. Typically this might involve a database that keeps a record of every user photo (and likely is used to store each photo as well).

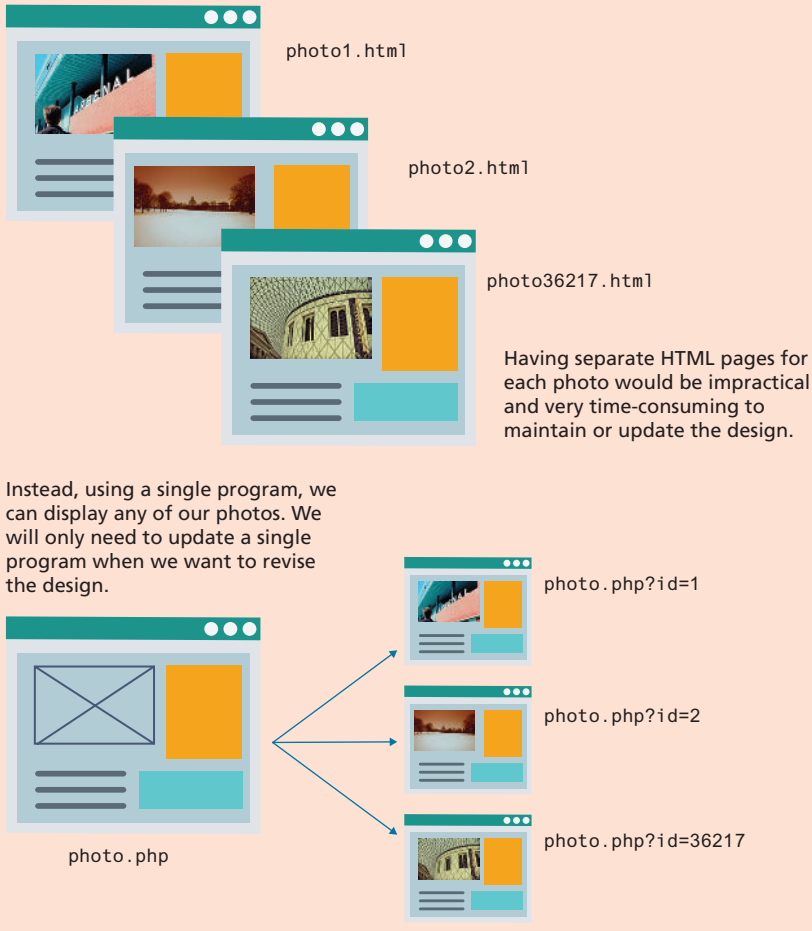


FIGURE 1.8 Why are programs needed

comparison to the front end. By the late 2010s, servers often performed minimal processing outside of authentication and data provision. As shown in Figure 1.9, dynamic websites today are dynamic both on the client and the server-side.

This trend towards thinner and thinner back ends is still continuing. Thanks to innovations in cloud-based services, static websites are back, albeit in a new form. As can be see in Figure 1.10, contemporary static sites make use of two types of servers: static asset servers which do no processing, and third-party cloud services which are consumed by JavaScript. The important point here is that this type of static site doesn't require running or setting up any type of server; instead it makes use of servers configured and operated by third-parties that are providing a wide-range of services, from databases, to authentication, to caching.

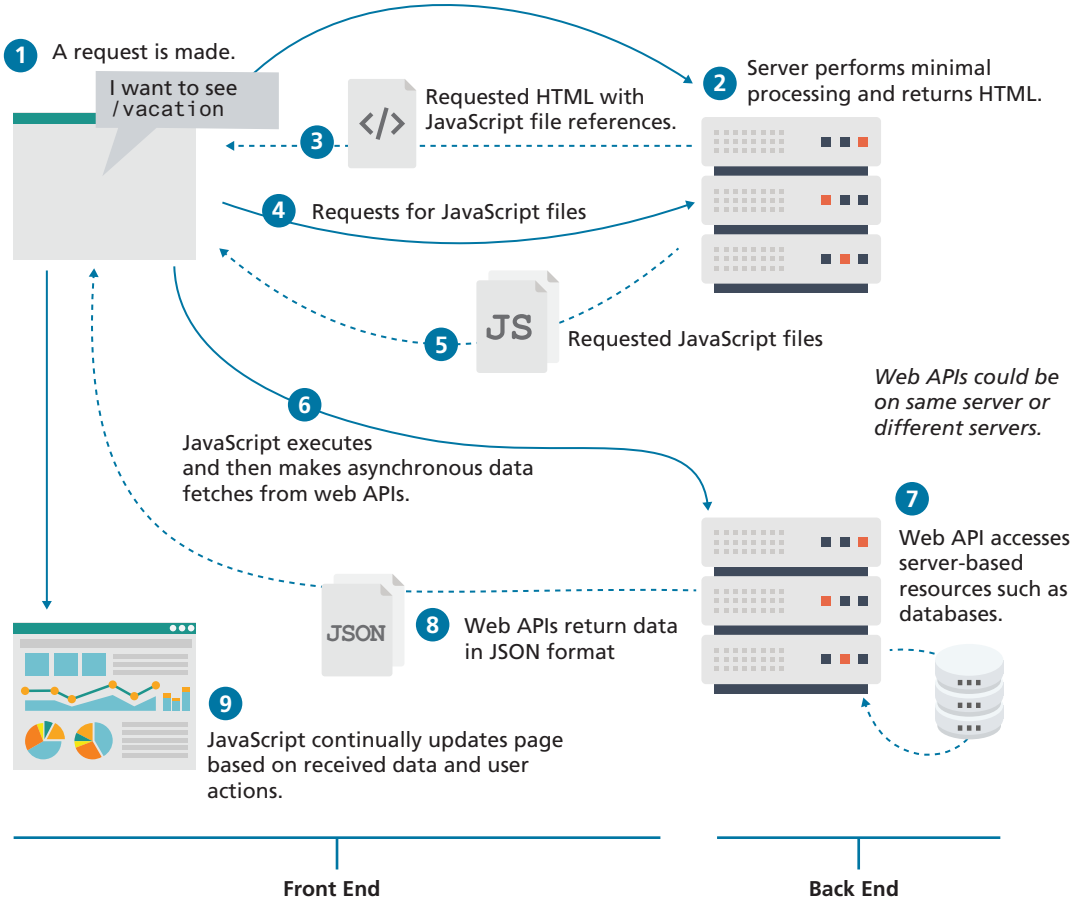


FIGURE 1.9 Dynamic websites today

Web development today is thus significantly more complicated than it was when the first edition of this textbook was written in 2012–2013. Take for instance, the task of uploading a file to a website, which today is a relatively common feature of many websites. Figure 1.11 illustrates the expanding range of processes and technologies that have become part of just this single task. Now expand this single task to dozens of tasks, and you can begin to see that a textbook of this size cannot hope to cover everything you might need to know in web development.

Instead, this book focuses on the fundamentals. Early chapters on HTML and CSS teach layout and structural foundations. The core of the book are its JavaScript chapters, which focus on the fundamentals of the language and its usage within the browser. While back-ends are thinner than they once were, they are still essential to many sites, and cover two key server-side technologies as well as working with