

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.



Pearson

**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

Copyright © 2022 by Pearson Education, Inc. or its affiliates, 221 River Street, Hoboken, NJ 07030.  
All Rights Reserved. Manufactured in the United States of America. This publication is protected by copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights and Permissions department, please visit [www.pearsoned.com/permissions/](http://www.pearsoned.com/permissions/).

Attributions of third-party content appear on the appropriate page within the text or on pages 1029–1032, which constitute an extension of this copyright page.

PEARSON, ALWAYS LEARNING, and REVEL are exclusive trademarks owned by Pearson Education, Inc. or its affiliates in the U.S. and/or other countries.

Unless otherwise indicated herein, any third-party trademarks, logos, or icons that may appear in this work are the property of their respective owners, and any references to third-party trademarks, logos, icons, or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of Pearson's products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc., or its affiliates, authors, licensees, or distributors.

**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

<b>Chapter 1</b>	<b>Introduction to Web Development</b>	1
<b>Chapter 2</b>	<b>How the Web Works</b>	42
<b>Chapter 3</b>	<b>HTML 1: Introduction</b>	73
<b>Chapter 4</b>	<b>CSS 1: Selectors and Basic Styling</b>	122
<b>Chapter 5</b>	<b>HTML 2: Tables and Forms</b>	189
<b>Chapter 6</b>	<b>Web Media</b>	240
<b>Chapter 7</b>	<b>CSS 2: Layout</b>	282
<b>Chapter 8</b>	<b>JavaScript 1: Language Fundamentals</b>	348
<b>Chapter 9</b>	<b>JavaScript 2: Using JavaScript</b>	418
<b>Chapter 10</b>	<b>JavaScript 3: Additional Features</b>	480
<b>Chapter 11</b>	<b>JavaScript 4: React</b>	545

**Chapter 12 Server-Side Development 1: PHP** 603

**Chapter 13 Server-Side Development 2: Node.js** 673

**Chapter 14 Working with Databases** 711

**Chapter 15 Managing State** 778

**Chapter 16 Security** 813

**Chapter 17 DevOps and Hosting** 880

**Chapter 18 Tools and Traffic** 932

# Table of Contents

*Preface* xxix

*Acknowledgments* xxxv

## **Chapter 1 Introduction to Web Development 1**

---

### **1.1 A Complicated Ecosystem 2**

#### **1.2 Definitions and History 4**

A Short History of the Internet 4

The Birth of the Web 7

Web Applications in Comparison to Desktop Applications 8

From Static to Dynamic (and Back to Static) 10

#### **1.3 The Client-Server Model 15**

The Client 17

The Server 17

Server Types 17

Real-World Server Installations 19

Cloud Servers 23

#### **1.4 Where Is the Internet? 24**

From the Computer to Outside the Home 25

From the Home to the Ocean's Edge 26

How the Internet Is Organized Today 28

#### **1.5 Working in Web Development 31**

Roles and Skills 32

Types of Web Development Companies 36

#### **1.6 Chapter Summary 40**

Key Terms 40

Review Questions 41

References 41

## Chapter 2 How the Web Works 42

---

### 2.1 Internet Protocols 43

- A Layered Architecture 43
- Link Layer 43
- Internet Layer 44
- Transport Layer 47
- Application Layer 48

### 2.2 Domain Name System 49

- Name Levels 51
- Name Registration 53
- Address Resolution 55

### 2.3 Uniform Resource Locators 58

- Protocol 58
- Domain 58
- Port 58
- Path 59
- Query String 59
- Fragment 59

### 2.4 Hypertext Transfer Protocol 60

- Headers 61
- Request Methods 62
- Response Codes 64

### 2.5 Web Browsers 64

- Fetching a Web Page 65
- Browser Rendering 65
- Browser Caching 67
- Browser Features 68
- Browser Extensions 68

### 2.6 Web Servers 69

- Operating Systems 69
- Web Server Software 70
- Database Software 70
- Scripting Software 70

**2.7 Chapter Summary** 71

Key Terms 71

Review Questions 72

References 72

**Chapter 3 HTML 1: Introduction** 73

---

**3.1 What Is HTML and Where Did It Come From?** 74

XHTML 76

HTML5 78

**3.2 HTML Syntax** 79

Elements and Attributes 79

Nesting HTML Elements 80

**3.3 Semantic Markup** 81**3.4 Structure of HTML Documents** 84

DOCTYPE 85

Head and Body 85

**3.5 Quick Tour of HTML Elements** 87

Headings 87

Paragraphs and Divisions 91

Links 92

URL Relative Referencing 92

Inline Text Elements 95

Images 95

Character Entities 98

Lists 99

**3.6 HTML5 Semantic Structure Elements** 102

Header and Footer 103

Navigation 104

Main 105

Articles and Sections 106

Figure and Figure Captions 106

Aside	108
Details and Summary	109
Additional Semantic Elements	110
<b>3.7 Chapter Summary</b>	116
Key Terms	116
Review Questions	116
Hands-On Projects	117
<b>Chapter 4 CSS 1: Selectors and Basic Styling</b>	122
<hr/>	
<b>4.1 What Is CSS?</b>	123
Benefits of CSS	123
CSS Versions	123
Browser Adoption	124
<b>4.2 CSS Syntax</b>	125
Selectors	126
Properties	126
Values	127
<b>4.3 Location of Styles</b>	130
Inline Styles	130
Embedded Style Sheet	131
External Style Sheet	131
<b>4.4 Selectors</b>	132
Element Selectors	133
Class Selectors	133
Id Selectors	135
Attribute Selectors	136
Pseudo-Element and Pseudo-Class Selectors	136
Contextual Selectors	139
<b>4.5 The Cascade: How Styles Interact</b>	142
Inheritance	143
Specificity	145
Location	146

<b>4.6 The Box Model</b>	149
Block versus Inline Elements	149
Background	153
Borders and Box Shadow	155
Margins and Padding	156
Box Dimensions	159
<b>4.7 CSS Text Styling</b>	165
Font Family	165
Font Sizes	167
Font Weight	171
Paragraph Properties	172
<b>4.8 CSS Frameworks and Variables</b>	174
What is a CSS Framework?	175
CSS Variables	181
<b>4.9 Chapter Summary</b>	183
Key Terms	183
Review Questions	183
Hands-On Practice	184
References	188
<b>Chapter 5 HTML 2: Tables and Forms</b>	189
<hr/>	
<b>5.1 HTML Tables</b>	190
Basic Table Structure	190
Spanning Rows and Columns	191
Additional Table Elements	191
Using Tables for Layout	194
<b>5.2 Styling Tables</b>	195
Table Borders	195
Boxes and Zebras	197
<b>5.3 Introducing Forms</b>	199
Form Structure	199
How Forms Work	200

Query Strings	201
The <form> Element	202
<b>5.4 Form Control Elements</b>	204
Text Input Controls	204
Choice Controls	205
Button Controls	209
Specialized Controls	209
Date and Time Controls	213
<b>5.5 Table and Form Accessibility</b>	215
Accessible Tables	216
Accessible Forms	217
<b>5.6 Styling and Designing Forms</b>	218
Styling Form Elements	219
Form Design	220
<b>5.7 Validating User Input</b>	222
Types of Input Validation	222
Notifying the User	223
How to Reduce Validation Errors	224
Where to Perform Validation	227
<b>5.8 Chapter Summary</b>	234
Key Terms	234
Review Questions	234
Hands-On Practice	235
<b>Chapter 6 Web Media</b>	240

---

<b>6.1 Representing Digital Images</b>	241
Image Types	241
Color Models	242
<b>6.2 Image Concepts</b>	250
Color Depth	250
Image Size	251
Display Resolution	254

**6.3 File Formats** 258

JPEG 258

GIF 259

PNG 264

SVG 264

Other Formats 265

**6.4 Audio and Video** 268

Media Concepts 268

Browser Video Support 269

Browser Audio Support 271

**6.5 Working with Color** 273

Picking Colors 274

Define Shades 275

**6.6 Chapter Summary** 277

Key Terms 277

Review Questions 277

Hands-On Practice 278

---

**Chapter 7 CSS 2: Layout** 282**7.1 Older Approaches to CSS Layout** 283

Floating Elements 283

Positioning Elements 284

Overlapping and Hiding Elements 288

**7.2 Flexbox Layout** 292

Flex Containers and Flex Items 293

Use Cases for Flexbox 294

**7.3 Grid Layout** 298

Specifying the Grid Structure 299

Explicit Grid Placement 300

Cell Properties 302

Nested Grids 302

Grid Areas 306

Grid and Flexbox Together 306

**7.4 Responsive Design** 310

Setting Viewports 313

Media Queries 314

Scaling Images 318

**7.5 CSS Effects** 321

Transforms 322

Filters 324

Transitions 324

Animations 329

**7.6 CSS Preprocessors** 332

The Basics of Sass 333

Mixins and Functions 335

Modules 336

**7.7 Chapter Summary** 340

Key Terms 340

Review Questions 340

Hands-On Practice 341

References 347

---

**Chapter 8 JavaScript 1: Language Fundamentals** 348**8.1 What Is JavaScript and What Can It Do?** 349

Client-Side Scripting 350

JavaScript's History 352

JavaScript and Web 2.0 353

JavaScript in Contemporary Software Development 354

**8.2 Where Does JavaScript Go?** 356

Inline JavaScript 356

Embedded JavaScript 356

External JavaScript 358

Users without JavaScript 359

**8.3 Variables and Data Types** 359

JavaScript Output 362

Data Types 364

Built-In Objects	366
Concatenation	368
<b>8.4 Conditionals</b>	369
Truthy and Falsy	371
<b>8.5 Loops</b>	372
While and do . . . while Loops	373
For Loops	373
<b>8.6 Arrays</b>	375
Iterating an array using for . . . of	378
Array Destructuring	378
<b>8.7 Objects</b>	380
Object Creation Using Object Literal Notation	380
Object Creation Using Object Constructor	381
Object Destructuring	382
JSON	385
<b>8.8 Functions</b>	388
Function Declarations vs. Function Expressions	388
Nested Functions	391
Hoisting in JavaScript	392
Callback Functions	394
Objects and Functions Together	396
Function Constructors	397
Arrow Syntax	399
<b>8.9 Scope and Closures in JavaScript</b>	403
Scope in JavaScript	403
Closures in JavaScript	408
<b>8.10 Chapter Summary</b>	411
Key Terms	412
Review Questions	412
Hands-On Practice	413
References	417

## Chapter 9 JavaScript 2: Using JavaScript 418

---

### 9.1 The Document Object Model (DOM) 419

Nodes and NodeLists 420  
Document Object 420  
Selection Methods 422  
Element Node Object 424

### 9.2 Modifying the DOM 427

Changing an Element's Style 427  
InnerHTML vs textContent vs DOM Manipulation 429  
DOM Manipulation Methods 430  
DOM Timing 433

### 9.3 Events 436

Implementing an Event Handler 436  
Page Loading and the DOM 439  
Event Object 440  
Event Propagation 440  
Event Delegation 444  
Using the Dataset Property 446

### 9.4 Event Types 448

Mouse Events 448  
Keyboard Events 448  
Form Events 450  
Media Events 451  
Frame Events 451

### 9.5 Forms in JavaScript 456

Responding to Form Movement Events 458  
Responding to Form Changes Events 458  
Validating a Submitted Form 458  
Submitting Forms 462

### 9.6 Regular Expressions 463

Regular Expression Syntax 463  
Extended Example 465

<b>9.7 Chapter Summary</b>	472
Key Terms	472
Review Questions	473
Hands-On Practice	473
References	479
<hr/>	
<b>Chapter 10 JavaScript 3: Additional Features</b>	480
<hr/>	
<b>10.1 Array Functions</b>	481
forEach	481
Find, Filter, Map, and Reduce	482
Sort	484
<b>10.2 Prototypes, Classes, and Modules</b>	485
Using Prototypes	487
Classes	491
Modules	493
<b>10.3 Asynchronous Coding with JavaScript</b>	499
Fetching Data from a Web API	503
Promises	514
Async and Await	518
<b>10.4 Using Browser APIs</b>	524
Web Storage API	524
Web Speech API	526
GeoLocation	527
<b>10.5 Using External APIs</b>	529
Google Maps	529
Charting with Plotly.js	531
<b>10.6 Chapter Summary</b>	539
Key Terms	539
Review Questions	539
Hands-On Practice	540
References	544

---

## Chapter 11 JavaScript 4: React 545

---

### 11.1 JavaScript Front-End Frameworks 546

Why Do We Need Frameworks? 546  
React, Angular, and Vue 547

### 11.2 Introducing React 551

React Components 553

### 11.3 Props, State, Behavior, and Forms 557

Props 557  
State 561  
Behaviors 563  
Forms in React 568  
Component Data Flow 570

### 11.4 React Build Approach 577

Build Tools 577  
Create React App 579  
Other React Build Approaches 582

### 11.5 React Lifecycle 582

Fetching Data 583

### 11.6 Extending React 584

Routing 584  
CSS in React 587  
Other Approaches to State 588

### 11.7 Chapter Summary 596

Key Terms 597

Review Questions 597

Hands-On Practice 597

References 602

---

## Chapter 12 Server-Side Development I: PHP 603

---

### 12.1 What Is Server-Side Development? 604

Front End versus Back End 604  
Common Server-Side Technologies 605

**12.2 PHP Language Fundamentals** 611

- PHP Tags 611
- Variables and Data Types 613
- Writing to Output 614
- Concatenation 615

**12.3 Program Control** 620

- if...else 620
- switch...case 621
- while and do...while 622
- for 623
- Alternate Syntax for Control Structures 624
- Include Files 624

**12.4 Functions** 627

- Function Syntax 627
- Invoking a Function 628
- Parameters 629
- Variable Scope within Functions 632

**12.5 Arrays** 635

- Defining and Accessing an Array 635
- Multidimensional Arrays 636
- Iterating through an Array 639
- Adding and Deleting Elements 640

**12.6 Classes and Objects** 643

- Terminology 643
- Defining Classes 644
- Instantiating Objects 644
- Properties 645
- Constructors 645
- Method 646
- Visibility 648
- Static Members 649
- Inheritance 651

<b>12.7 \$_GET and \$_POST Superglobal Arrays</b>	652
Superglobal Arrays	652
Determining If Any Data Sent	655
Accessing Form Array Data	658
Using Query Strings in Hyperlinks	659
Sanitizing Query Strings	660
<b>12.8 Working with the HTTP Header</b>	664
Redirecting Using Location Header	664
Setting the Content-Type Header	664
<b>12.9 Chapter Summary</b>	666
Key Terms	667
Review Questions	667
Hands on Practice	667
Reference	672

## **Chapter 13 Server-Side Development 2: Node.js** 673

---

<b>13.1 Introducing Node.js</b>	674
Node Advantages	674
Node Disadvantages	679
<b>13.2 First Steps with Node</b>	682
Simple Node Application	682
Adding Express	685
Environment Variables	686
<b>13.3 Creating an API in Node</b>	687
Simple API	687
Adding Routes	689
Separating Functionality into Modules	690
<b>13.4 Creating a CRUD API</b>	692
Passing Data to an API	694
API Testing Tools	695
<b>13.5 Working with Web Sockets</b>	696
<b>13.6 View Engines</b>	700

**13.7 Serverless Approaches** 702

- What Is Serverless? 702
- Benefits of Serverless Computing 704
- Serverless Technologies 704

**13.8 Chapter Summary** 706

- Key Terms 707
- Review Questions 707
- Hands-On Practice 707
- References 710

---

**Chapter 14 Working with Databases** 711**14.1 Databases and Web Development** 712

- The Role of Databases in Web Development 712

**14.2 Managing Databases** 715

- Command-Line Interface 716
- phpMyAdmin 716
- MySQL Workbench 718
- SQLite Tools 719
- MongoDB Tools 719

**14.3 SQL** 720

- Database Design 720
- SELECT Statement 724
- INSERT, UPDATE, and DELETE Statements 727
- Transactions 727
- Data Definition Statements 731
- Database Indexes and Efficiency 732

**14.4 Working with SQL in PHP** 733

- Connecting to a Database 734
- Handling Connection Errors 737
- Executing the Query 738
- Processing the Query Results 739

Freeing Resources and Closing Connection	743
Working with Parameters	744
Using Transactions	747
Designing Data Access	751
<b>14.5 NoSQL Databases</b>	754
Why (and Why Not) Choose NoSQL?	756
Types of NoSQL Systems	757
<b>14.6 Working with MongoDB in Node</b>	761
MongoDB Features	761
MongoDB Data Model	762
Working with the MongoDB Shell	764
Accessing MongoDB Data in Node.js	764
<b>14.7 Chapter Summary</b>	771
Key Terms	772
Review Questions	772
Hands-On Practice	773
References	777

## **Chapter 15 Managing State** 778

---

<b>15.1 The Problem of State in Web Applications</b>	779
<b>15.2 Passing Information in HTTP</b>	781
Passing Information via the URL	781
Passing Information via HTTP Header	782
<b>15.3 Cookies</b>	785
How Do Cookies Work?	786
Using Cookies in PHP	787
Using Cookies in Node and Express	789
Persistent Cookie Best Practices	789
<b>15.4 Session State</b>	792
How Does Session State Work?	793
Session Storage and Configuration	794
Session State in PHP	796
Session State in Node	798

**15.5 Caching** 799

- Page Output Caching 800
- Application Data Caching 800
- Redis as Caching Service 803

**15.6 Chapter Summary** 808

- Key Terms 808
- Review Questions 808
- Hands-On Practice 808
- References 812

**Chapter 16 Security** 813

---

**16.1 Security Principles** 814

- Information Security 814
- Risk Assessment and Management 815
- Security Policy 818
- Business Continuity 818
- Secure by Design 821
- Social Engineering 823
- Authentication Factors 824

**16.2 Approaches to Web Authentication** 825

- Basic HTTP Authentication 826
- Form-Based Authentication 827
- HTTP Token Authentication 829
- Third-Party Authentication 830

**16.3 Cryptography** 834

- Substitution Ciphers 835
- Public Key Cryptography 838
- Digital Signatures 840

**16.4 Hypertext Transfer Protocol Secure (HTTPS)** 840

- SSL/TLS Handshake 842
- Certificates and Authorities 842
- Migrating to HTTPS 846

**16.5 Security Best Practices** 848

- Credential Storage 849
- Monitor Your Systems 858
- Audit and Attack Thyself 859

**16.6 Common Threat Vectors** 860

- Brute-Force Attacks 860
- SQL Injection 861
- Cross-Site Scripting (XSS) 863
- Cross-Site Request Forgery (CSRF) 868
- Insecure Direct Object Reference 869
- Denial of Service 870
- Security Misconfiguration 871

**16.7 Chapter Summary** 874

- Key Terms 875
- Review Questions 875
- Hands-On Practice 876
- References 878

---

**Chapter 17 DevOps and Hosting** 880**17.1 DevOps: Development and Operations** 881

- Continuous Integration, Delivery, and Deployment 881
- Testing 882
- Infrastructure as Code 885
- Microservice Architecture 886

**17.2 Domain Name Administration** 888

- Registering a Domain Name 888
- Updating the Name Servers 891
- DNS Record Types 891
- Reverse DNS 894

**17.3 Web Server Hosting Options** 895

- Shared Hosting 895
- Dedicated Hosting 898

Collocated Hosting	898
Cloud Hosting	899
<b>17.4 Virtualization</b>	899
Server Virtualization	899
Cloud Virtualization	904
<b>17.5 Linux and Web Server Configuration</b>	905
Configuration	907
Starting and Stopping the Server	907
Connection Management	908
Data Compression	910
Encryption and SSL	911
Managing File Ownership and Permissions	913
<b>17.6 Request and Response Management</b>	914
Managing Multiple Domains on One Web Server	914
Handling Directory Requests	916
Responding to File Requests	917
URL Redirection	918
Managing Access with .htaccess	922
Server Caching	923
<b>17.7 Web Monitoring</b>	925
Internal Monitoring	925
External Monitoring	927
<b>17.8 Chapter Summary</b>	927
Key Terms	927
Review Questions	928
Hands-On Practice	928
References	930
<b>Chapter 18 Tools and Traffic</b>	932

---

<b>18.1 The History and Anatomy of Search Engines</b>	933
Search Engine Overview	933

<b>18.2 Web Crawlers and Scrapers</b>	935
Scrapers	936
<b>18.3 Indexing and Reverse Indexing</b>	938
<b>18.4 PageRank and Result Order</b>	939
<b>18.5 Search Engine Optimization</b>	942
Title	943
Meta Tags	943
URLs	945
Site Design	947
Sitemaps	948
Anchor Text	949
Images	949
Content	950
Black-Hat SEO	950
<b>18.6 Social Networks</b>	955
How Did We Get Here?	956
<b>18.7 Social Network Integration</b>	958
Basic Social Media Presence	959
Facebook's Social Plugins	960
Open Graph	964
Twitter's Widgets	965
Advanced Social Network Integration	969
<b>18.8 Content Management Systems</b>	970
Components of a Managed Website	970
Types of CMS	971
<b>18.9 WordPress Overview</b>	972
Post and Page Management	973
WYSIWYG Editors	975
Template Management	976
Menu Control	977
User Management and Roles	977

User Roles	978
Workflow and Version Control	981
Asset Management	982
Search	983
Upgrades and Updates	983
<b>18.10 WordPress Technical Overview</b>	984
Installation	984
File Structure	984
WordPress Nomenclature	986
WordPress Template Hierarchy	987
<b>18.11 Modifying Themes</b>	988
Changing Theme Files	990
<b>18.12 Web Advertising Fundamentals</b>	991
Web Advertising 101	991
Web Advertising Economy	994
<b>18.13 Support Tools and Analytics</b>	995
Search Engine Webmaster Tools	995
Analytics	996
Third-Party Analytics	999
Performance Tuning and Rating	999
<b>18.14 Chapter Summary</b>	1005
Key Terms	1005
Review Questions	1006
Hands-On Practice	1006
References	1009
<i>Index</i>	1011
<i>Credits</i>	1029

# 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](http://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/](http://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](https://github.com/funwebdev-3rd-ed)).

## For the Student

---

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

518 CHAPTER 10 JavaScript 3: Additional Features

```
// promised version of the transfer task
function transferToCloud(filename) {
  return new Promise((resolve, reject) => {
    // just have a placeholder URL for now
    let cloudURL =
      "http://BucketName.s3-us-west-2.amazonaws.com/makebelieve.jpg";
    if (!existsOnServer(filename)) {
      performTransfer(filename, cloudURL);
      resolve(cloudURL);
    } else {
      reject(new Error('filename does not exist'));
    }
  });
}

// use this function
transferToCloud(filename)
  .then(url => extractTags(url))
  .then(url => compressImage(url))
  .catch(err => logThisError(err));

```

**LISTING 10.10** Creating Promises

The `Promise.all()` method is typically passed an array of `Promise` objects that can be satisfied in any order. Figure 10.20 illustrates how this approach can be used. Notice that it returns a `Promise`, thus the `then()` method needs to be passed a function that will get executed when all the passed `Promise` objects are resolved. That function will be passed an array containing, in the case of multiple fetches, multiple retrieved JSON data arrays.

The `Promise.all()` approach can be more efficient when each individual fetch is independent of each other. Figure 10.21 contains screen captures of the code in Chapter 10 that demonstrates how to make three separate parallelized fetches and one using the `Promise.all()` approach. With the nested approach, the browser can't make the next fetch request until the previous one is resolved (that is, the data has been returned); with the `Promise.all()` approach, all three fetches can be made simultaneously, which is more time efficient.

**10.3.3 Async and Await**

In the previous section, you learned how to use (and create) promises as a way of taming the code complexities of using asynchronous functions. While certainly a significant improvement over using raw callback functions, recent iterations of the JavaScript language have added additional language support for asynchronous operations, which further improves and simplifies the code needed for these operations.

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.

**TER 4 CSS 1: Selectors and Basic Styling**

Property	Description
<code>border</code>	A shorthand property that allows you to set the style, width, and color of a border in one property. The order is important and must be:
<code>border-width</code>	Specifies the line type of the border. Possible values are: solid, dotted, dashed, double, groove, ridge, inset, outset, hidden, and none.
<code>border-style</code>	The width of the border in a unit (but not percent). A variety of keywords (thin, medium, etc.) are also supported.
<code>border-color</code>	The color of the border in a color unit.
<code>border-radius</code>	The radius of a rounded corner.
<code>border-image</code>	The URL of an image to use as a border.
<code>box-shadow</code>	Adds a shadow effect to an element. The values are as follows: offset-x offset-y blur-radius spread-radius color

**TABLE 4.8** Border Properties

The `box-shadow` property provides a way to add shadow effects around an element's box. To set the shadow, you specify x and y offsets, along with optional blur, spread, inset, and color settings.

**4.6.4 Margins and Padding**

Margins and padding are properties for adding white space to a web page, which can help differentiate one element from another. Figure 4.23 illustrates how these two properties can be used to provide spacing and element differentiation.

As you can see in Figures 4.17 and 4.23, margins add spacing around an element's content, while padding adds spacing within elements. Borders divide the margin area from the padding area.



**ESSENTIAL SOLUTIONS**

Centering an element horizontally within a container

```
<div id="element">content</div>
```

```
#element { margin: 0 auto; width: 200px; /* some value */ }
```

result browser

In chapter 7, you will learn how to use floating to position an element horizontally and vertically within a container.

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 has already been provided).

- Set the background image on the `<body>` tag. Set the height to 100vh, so it will always fill the entire viewer. Set the background image and background-color properties (see Chapter 6 for a refresher if needed).
- 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.
- To center the form in the center of the page, first, set the `display` of the `<div>` element to flex and `align-items` and `justify-content` to center.
- Fine-tune the size of the form elements by setting the `fontSize` 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**

1 Using Grid  
Nested Grids  
Using calc()  
Grid Areas  
Grid and Flex Together

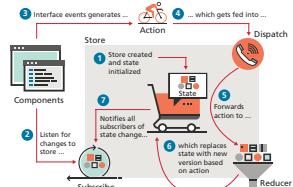
### 7.3 Grid Layout

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 Interface events generates ...
    ② ... which gets fed into ...
    ③ Action
    ④ Store created and state initialized
    ⑤ Dispatch
    ⑥ Notifies all subscribers of state change
    ⑦ Listen for changes to state
    ⑧ Subscribe
    ⑨ Reducer
    ⑩ Forwards action to ...
    ⑪ which replaces current version of state on action
  
```

```

    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 id: ..., title: ...;
      // dispatch add-to-fav action with the data
      store.dispatch({ type: 'ADD_TO_FAV', payload: f });
    };
  
```

```

    4 const reducer = (state = initialState, action) => {
      if (action.type === 'ADD_TO_FAV') {
        const nextState = (...state);
        nextState.favorites.push(action.payload);
        return nextState;
      }
      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.

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<sup>14</sup>! 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.

One of the most common ways to work with JavaScript code is to use a browser, namely, your browser. All modern browsers now include sophisticated and profiling tools. Just as the authors' grandparents used to regale us in 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 1990s. JavaScript, without having to leave the type of developer's browser 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 browser, as shown in Figure 9.22.

JavaScript profilers also provide additional tools that are essential to JavaScript development. As more and more functionality has migrated from the client, it has become increasingly important to assess the performance of JavaScript code. Figure 9.23 illustrates the Profile view of a page's JavaScript console. It allows a developer to pinpoint time-consuming functions or visualize as timeline charts.

FIGURE 9.22 Debugging within the Firefox browser

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 {
    nonexistentfunction("Hello");
} catch(err) {
    alert("An exception was caught:" + err);
}
```

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

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

9.4 Event Types 483

#### 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

(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.

#### 13.8.1 Key Terms

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

13.8 Chapter Summary 202

#### 13.8.2 Review Questions

- What are the key advantages and disadvantages of using Node?
- What is npm? What is its role in contemporary web development?
- A nonblocking architecture can typically handle more simultaneous requests. Why is that?
- What are modules in JavaScript? How does the Node CommonJS module system differ from the one introduced in ES6?
- In the context of Node, what is Express?
- What is a static file?
- In Express, what is middleware?
- What is a CRUD API?
- What are WebSockets? How do they differ from HTTP?
- What role do view engines play in Node?
- What are the benefits of serverless computing?
- How do 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

- 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.
- Name your server file `art.js`. Add a static file handler for resources in the `static` folder.
- The data for the APIs is contained in a supplied json file. Create a provider module for this file.

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

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.

- Why is it so important to use only one connection per page with PDO?
- 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

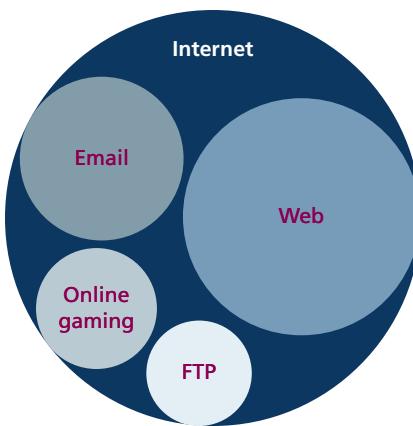
---

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.



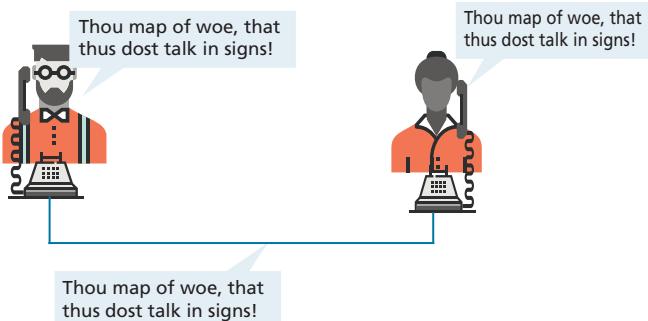
**FIGURE 1.2** The web as a subset of the Internet

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

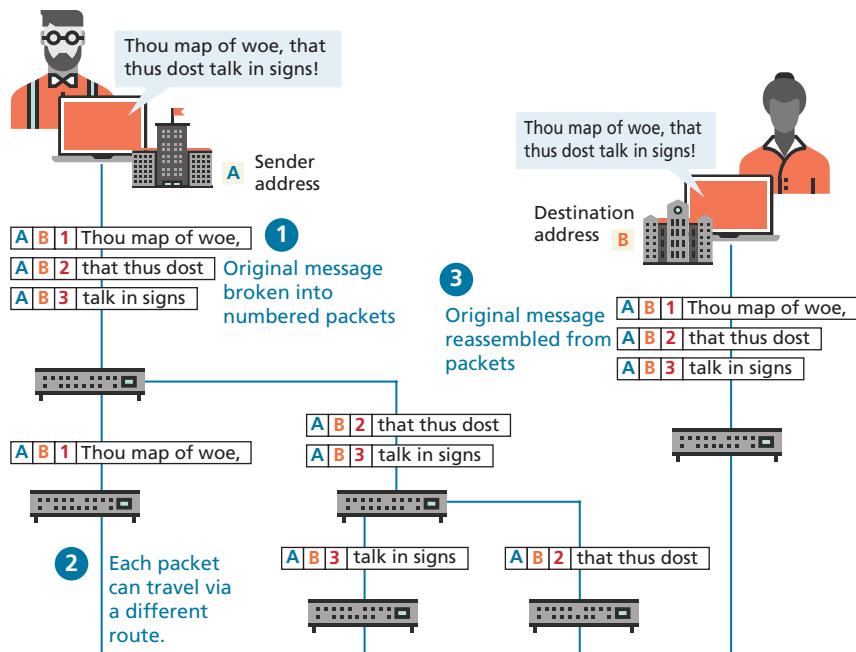


FIGURE 1.4 Internet network as example of packet switching

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.<sup>1,2</sup> 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.<sup>3</sup> 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.<sup>4</sup>

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 “e”), 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.



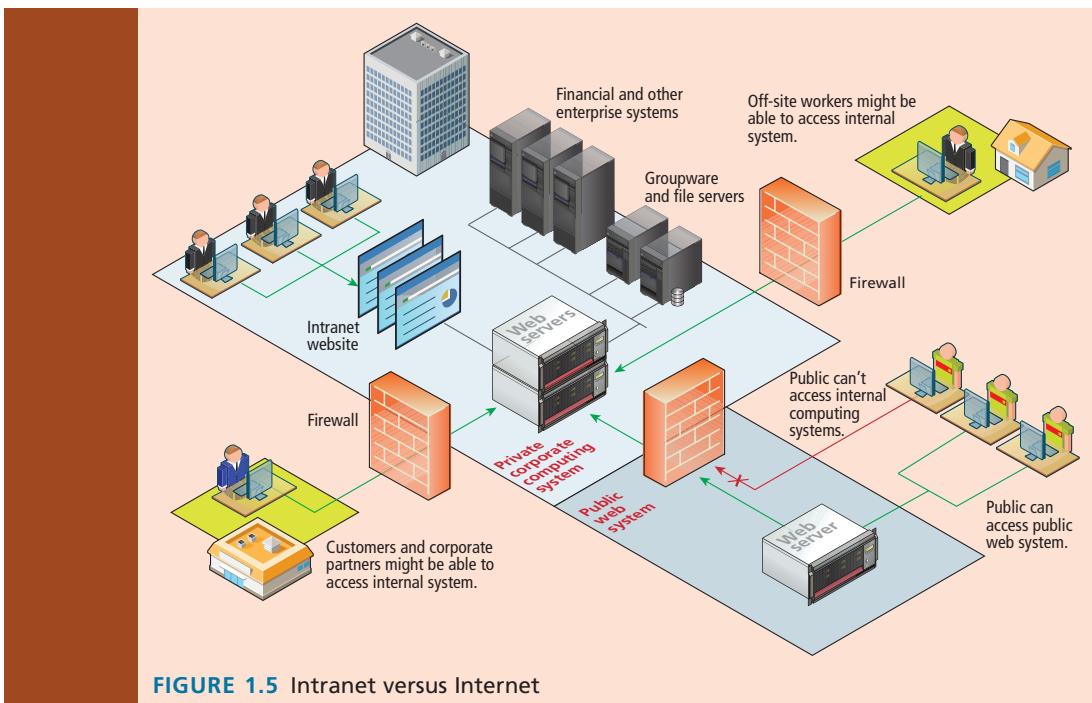


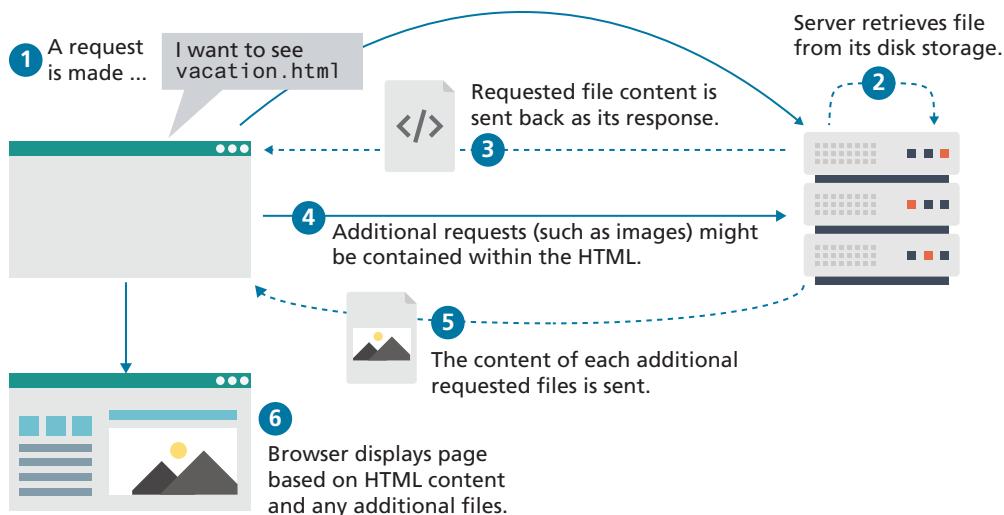
FIGURE 1.5 Intranet versus Internet

#### 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

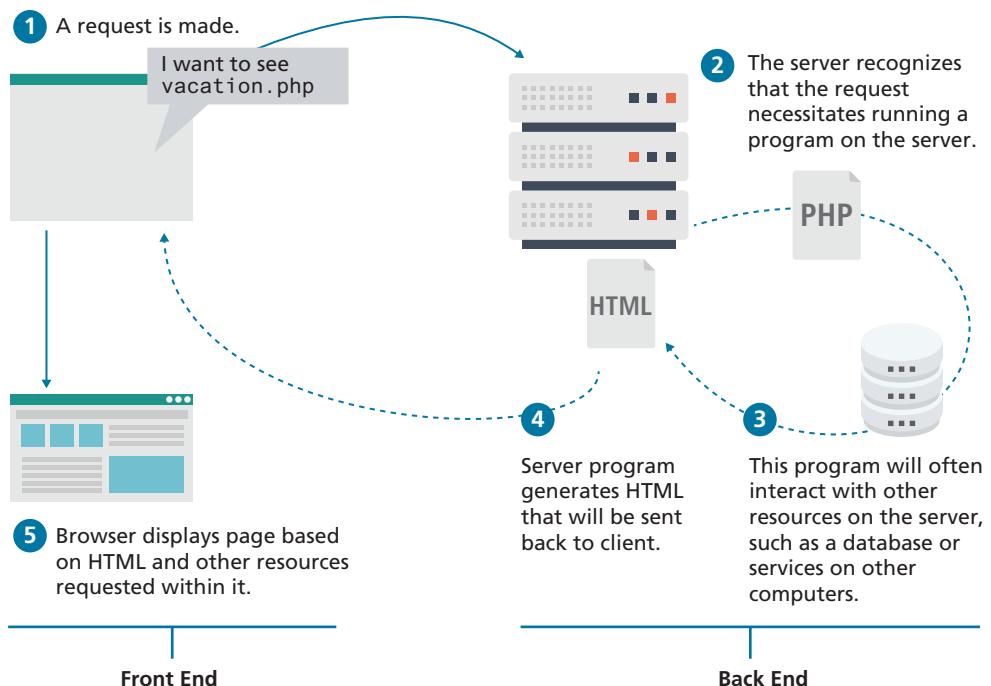


FIGURE 1.7 Dynamic Server-Side website



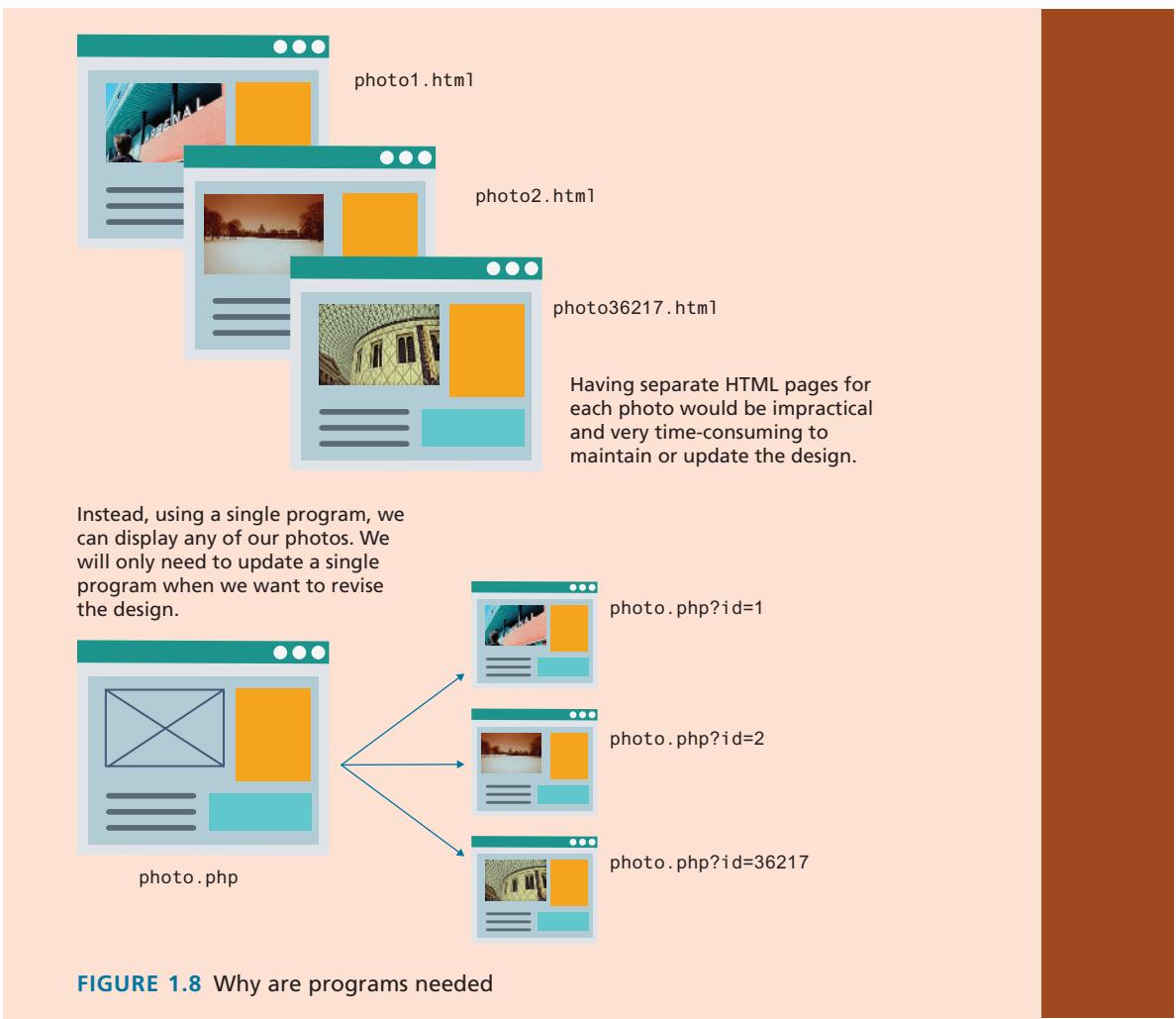
### DIVE DEEPER

**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 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).



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 seen 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.

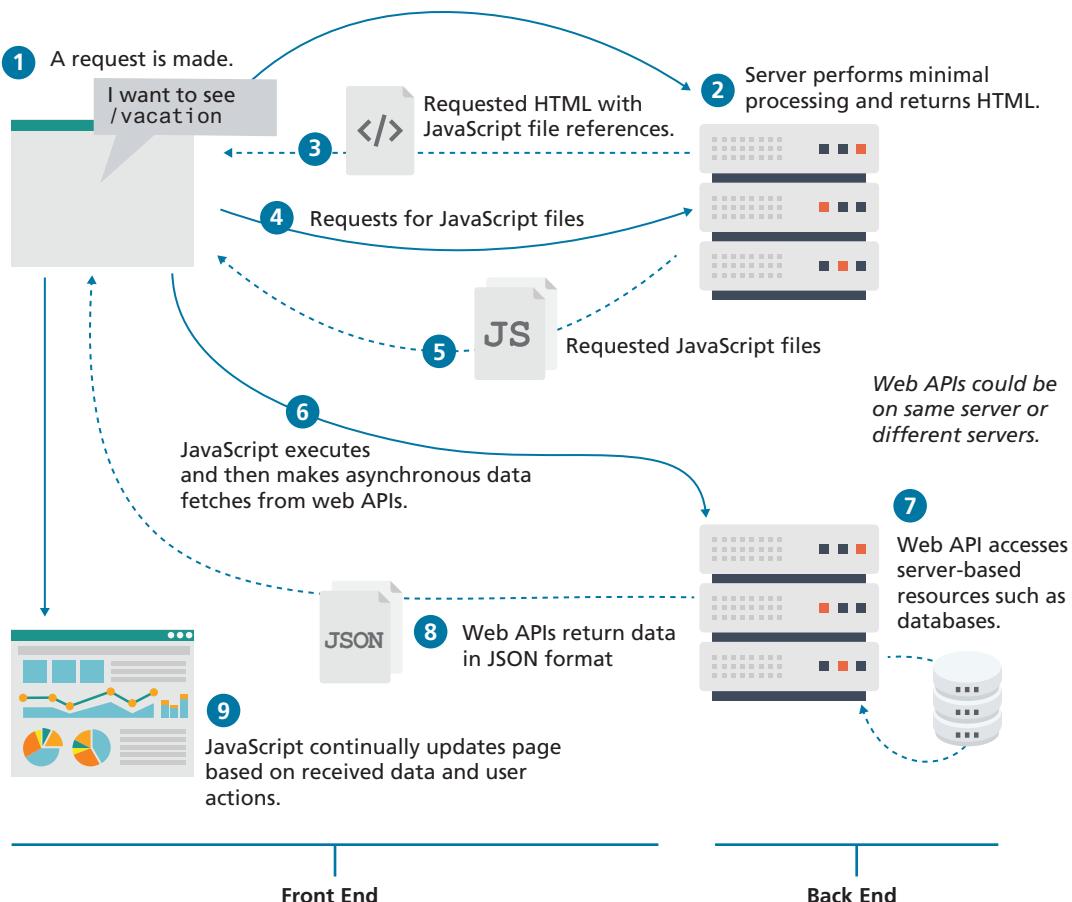


FIGURE 1.9 Dynamic websites today

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