1. Introduction to Computers and the Internet

> ***People are using the web to build things they have not built or written or drawn or communicated anywhere else.***
>
> **—Tim Berners-Lee**

> ***How wonderful it is that nobody need wait a single moment before starting to improve the world.***
>
> **—Anne Frank**

> ***Man is still the most extraordinary computer of all.***
>
> **—John F. Kennedy**

Objectives

In this chapter you'll learn:

- Computer hardware, software and Internet basics.
- The evolution of the Internet and the World Wide Web.
- How HTML5, CSS3 and JavaScript are improving web-application development.
- The data hierarchy.
- The different types of programming languages.
- Object-technology concepts.
- And you'll see demos of interesting and fun Internet applications you can build with the technologies you'll learn in this book.

Outline

1.1. Introduction

Welcome to the exciting and rapidly evolving world of Internet and web programming! There are more than two billion Internet users worldwide

—that's approximately 30% of the Earth's population.[1] In use today are more than a billion general-purpose computers, and billions more *embedded* computers are used in cell phones, smartphones, tablet computers, home appliances, automobiles and more—and many of these devices are connected to the Internet. According to a study by Cisco Internet Business Solutions Group, there were 12.5 billion Internet-enabled devices in 2010, and the number is predicted to reach 25 billion by 2015 and 50 billion by 2020.[2] The Internet and web programming technologies you'll learn in this book are designed to be *portable*, allowing you to design web pages and applications that run across an enormous range of Internet-enabled devices.

---

[1] www.internetworldstats.com/stats.htm.

---

[2] www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

You'll begin by learning the *client-side programming* technologies used to build web pages and applications that are run on the *client* (i.e., in the browser on the user's device). You'll use HyperText Markup Language 5 (HTML5) and Cascading Style Sheets 3 (CSS3)—the recent releases of HTML and CSS technologies—to add powerful, dynamic and fun features and effects to web pages and web applications, such as audio, video, animation, drawing, image manipulation, designing pages for multiple screen sizes, access to web storage and more.

You'll learn *JavaScript*—the language of choice for implementing the client side of Internet-based applications (we discuss JavaScript in more detail in Section 1.3). Chapters 6–13 present rich coverage of JavaScript and its capabilities. You'll also learn about *jQuery*—the JavaScript library that's dramatically reshaping the world of web development. Throughout the book there's also an emphasis on *Ajax* development, which helps you create better-performing, more usable applications.

Later in the book, you'll learn *server-side programming*—the applications that respond to requests from client-side web browsers, such as searching the Internet, checking your bank-account balance, ordering a book from

Amazon, bidding on an eBay auction and ordering concert tickets. We present condensed treatments of four popular Internet/web programming languages for building the server side of Internet- and web-based client/server applications. Chapters 19–22 and 23–28 present three popular server-side technologies, including PHP, ASP.NET (in both C# and Visual Basic) and JavaServer Faces.

Be sure to read both the Preface and the Before You Begin section to learn about the book's coverage and how to set up your computer to run the hundreds of code examples. The code is available at www.deitel.com/books/iw3htp5 and www.pearsonhighered.com/deitel. Use the source code we provide to *run every program and script* as you study it. Try each example in *multiple browsers*. If you're interested in smartphones and tablet computers, be sure to run the examples in your browsers on iPhones, iPads, Android smartphones and tablets, and others. The technologies covered in this book and browser support for them are evolving rapidly. *Not every feature of every page we build will render properly in every browser*. All seven of the browsers we use are free.

Moore's Law

Every year, you probably expect to pay at least a little more for most products and services. The opposite has been the case in the computer and communications fields, especially with regard to the costs of hardware supporting these technologies. For many decades, hardware costs have fallen rapidly. Every year or two, the capacities of computers have approximately *doubled* inexpensively. This remarkable trend often is called **Moore's Law**, named for the person who identified it, Gordon Moore, co-founder of Intel—the leading manufacturer of the processors in today's computers and embedded systems. Moore's Law and related observations apply especially to the amount of memory that computers have for programs, the amount of secondary storage (such as disk storage) they have to hold programs and data over longer periods of time, and their processor speeds—the speeds at which computers execute their programs (i.e., do their work). Similar growth has occurred in the communications field, in which costs have plummeted as enormous demand for communications bandwidth (i.e., information-carrying capacity) has

attracted intense competition. We know of no other fields in which technology improves so quickly and costs fall so rapidly. Such phenomenal improvement is truly fostering the *Information Revolution*.

1.2. The Internet in Industry and Research

These are exciting times in the computer field. Many of the most influential and successful businesses of the last two decades are technology companies, including Apple, IBM, Hewlett Packard, Dell, Intel, Motorola, Cisco, Microsoft, Google, Amazon, Facebook, Twitter, Groupon, Foursquare, Yahoo!, eBay and many more. These companies are major employers of people who study computer science, information systems or related disciplines. At the time of this writing, Apple was the most valuable company in the world.

In the past, most computer applications ran on computers that were not connected to one another, whereas today's Internet applications can be written to communicate among computers throughout the world.

Figures 1.1–1.4 provide a few examples of how computers and the Internet are being used in industry and research. Figure 1.1 lists two examples of how computers and the Internet are being used to improve health care.

Name	Description
Electronic health records	These might include a patient's medical history, prescriptions, immunizations, lab results, allergies, insurance information and more. Making this information available to health care providers across a secure network improves patient care, reduces the probability of error and increases overall efficiency of the health care system.
Human Genome Project	The Human Genome Project was founded to identify and analyze the 20,000+ genes in human DNA. The project used computer programs to analyze complex genetic data, determine the sequences of the billions of chemical base pairs that make up human DNA and store the information in databases which have been made available over the Internet to researchers in many fields.

Fig. 1.1. Computers and the Internet in health care.

Figure 1.2 provides a sample of some of the exciting ways in which computers and the Internet are being used for social good. In the exercises at

the end of this chapter, you'll be asked to propose other projects that would use computers and the Internet to "make a difference."

Name	Description
AMBER™ Alert	The AMBER (America's Missing: Broadcast Emergency Response) Alert System is used to find abducted children. Law enforcement notifies TV and radio broadcasters and state transportation officials, who then broadcast alerts on TV, radio, computerized highway signs, the Internet and wireless devices. AMBER Alert recently partnered with Facebook, whose users can "Like" AMBER Alert pages by location to receive alerts in their news feeds.
World Community Grid	People worldwide can donate their unused computer processing power by installing a free secure software program that allows the World Community Grid (`www.worldcommunitygrid.org`) to harness unused capacity. This computing power, accessed over the Internet, is used in place of expensive supercomputers to conduct scientific research projects that are making a difference, providing clean water to third-world countries, fighting cancer, growing more nutritious rice for regions fighting hunger and more.
One Laptop Per Child (OLPC)	One Laptop Per Child (`one.laptop.org`) is providing low-power, inexpensive, Internet-enabled laptops to poor children worldwide—enabling learning and reducing the digital divide.

Fig. 1.2. Projects that use computers and the Internet for social good.

We rely on computers and the Internet to communicate, navigate, collaborate and more. Figure 1.3 gives some examples of how computers and the Internet provide the infrastructure for these tasks.

Name	Description
Cloud computing	**Cloud computing** allows you to use software, hardware and information stored in the "cloud"—i.e., accessed on remote computers via the Internet and available on demand—rather than having it stored on your personal computer. Amazon is one of the leading providers of public cloud computing services. You can rent extra storage capacity using the Amazon Simple Storage Service (Amazon S3), or augment processing capabilities with Amazon's EC2 (Amazon Elastic Compute Cloud). These services, allowing you to increase or decrease resources to meet your needs at any given time, are generally more cost effective than purchasing expensive hardware to ensure that you have enough storage and processing power to meet your needs at their peak levels. Business applications (such as CRM software) are often expensive, require significant hardware to run them and knowledgeable support staff to ensure that they're running properly and securely. Using cloud computing services shifts the burden of managing these applications from the business to the service provider, saving businesses money.
GPS	Global Positioning System (GPS) devices use a network of satellites to retrieve location-based information. Multiple satellites send time-stamped signals to the GPS device, which calculates the distance to each satellite based on the time the signal left the satellite and the time the signal arrived. This information is used to determine the exact location of the device. GPS devices can provide step-by-step directions and help you easily find nearby businesses (restaurants, gas stations, etc.) and points of interest. GPS is used in numerous location-based Internet services such as check-in apps to help you find your friends (e.g., Foursquare and Facebook), exercise apps such as RunKeeper that track the time, distance and average speed of your outdoor jog, dating apps that help you find a match nearby and apps that dynamically update changing traffic conditions.
Robots	Robots can be used for day-to-day tasks (e.g., iRobot's Roomba vacuum), entertainment (e.g., robotic pets), military combat, deep sea and space exploration (e.g., NASA's Mars rover) and more. RoboEarth (`www.roboearth.org`) is "a World Wide Web for robots." It allows robots to learn from each other by sharing information and thus improving their abilities to perform tasks, navigate, recognize objects and more.
E-mail, Instant Messaging, Video Chat and FTP	Internet-based servers support all of your online messaging. E-mail messages go through a mail server that also stores the messages. Instant messaging (IM) and Video Chat apps, such as AIM, Skype, Yahoo! Messenger and others allow you to communicate with others in real time by sending your messages and live video through servers. FTP (file transfer protocol) allows you to exchange files between multiple computers (e.g., a client computer such as your desktop and a file server) over the Internet using the TCP/IP protocols for transferring data.

Fig. 1.3. Examples of computers and the Internet in infrastructure.

Figure 1.4 lists a few of the exciting ways in which computers and the Internet are used in entertainment.

Name	Description
iTunes and the App Store	iTunes is Apple's media store where you can buy and download digital music, movies, television shows, e-books, ringtones and apps (for iPhone, iPod and iPad) over the Internet. Apple's iCloud service allows you to store your media purchases "in the cloud" and access them from any iOS (Apple's mobile operating system) device. In June 2011, Apple announced at their World Wide Developer Conference (WWDC) that 15 billion songs had been downloaded through iTunes, making Apple the leading music retailer. As of July 2011, 15 billion apps had been downloaded from the App Store (`www.apple.com/pr/library/2011/07/07Apples-App-Store-Downloads-Top-15-Billion.html`).
Internet TV	Internet TV set-top boxes (such as Apple TV and Google TV) allow you to access an enormous amount of content on demand, such as games, news, movies, television shows and more.
Game programming	Global video game revenues are expected to reach $65 billion in 2011 (`uk.reuters.com/article/2011/06/06/us-videogames-factbox-idUKTRE75552I20110606`). The most sophisticated games can cost as much as $100 million to develop. Activision's *Call of Duty 2: Modern Warfare*, released in 2009, earned $310 million in just one day in North America and the U.K. (`news.cnet.com/8301-13772_3-10396593-52.html?tag=mncol;txt`)! Online *social gaming*, which enables users worldwide to compete with one another over the Internet, is growing rapidly. Zynga—creator of popular online games such as *Farmville* and *Mafia Wars*—was founded in 2007 and already has over 265 million monthly users. To accommodate the growth in traffic, Zynga is adding nearly 1,000 servers each week (`techcrunch.com/2010/09/22/zynga-moves-1-petabyte-of-data-daily-adds-1000-servers-a-week/`)!

Fig. 1.4. Examples of computers and the Internet in entertainment.

1.3. HTML5, CSS3, JavaScript, Canvas and jQuery

You'll be learning the latest versions of several key client-side, web-application development technologies in this book. This section provides a brief overview of each.

HTML5

Chapters 2–3 introduce HTML (HyperText Markup Language)—a special type of computer language called a *markup language* designed to specify the *content* and *structure* of web pages (also called documents) in a portable manner. HTML5, now under development, is the emerging version of HTML. HTML enables you to create content that will render appropriately across the extraordinary range of devices connected to the Internet—including smartphones, tablet computers, notebook computers, desktop

computers, special-purpose devices such as large-screen displays at concert arenas and sports stadiums, and more.

You'll learn the basics of HTML5, then cover more sophisticated techniques such as creating tables, creating forms for collecting user input and using new features in HTML5, including page-structure elements that enable you to give meaning to the parts of a page (e.g., headers, navigation areas, footers, sections, figures, figure captions and more).

A "stricter" version of HTML called *XHTML (Extensible HyperText Markup Language)*, which is based on XML (eXtensible Markup Language, introduced in Chapter 15), is still used frequently today. Many of the server-side technologies we cover later in the book produce web pages as XHTML documents, by default, but the trend is clearly to HTML5.

Cascading Style Sheets (CSS)

Although HTML5 provides some capabilities for controlling a document's presentation, *it's better not to mix presentation with content*. HTML5 should be used only to specify a document's structure and content.

Chapters 4–5 use **Cascading Style Sheets** (**CSS**) to specify the *presentation*, or styling, of elements on a web page (e.g., fonts, spacing, sizes, colors, positioning). CSS was designed to style portable web pages *independently* of their content and structure. By separating page styling from page content and structure, you can easily change the look and feel of the pages on an *entire* website, or a portion of a website, simply by swapping out one style sheet for another. CSS3 is the current version of CSS under development. Chapter 5 introduces many new features in CSS3.

JavaScript

JavaScript is a language that helps you build *dynamic* web pages (i.e., pages that can be modified "on the fly" in response to *events*, such as user input, time changes and more) and computer applications. It enables you to do the client-side programming of web applications. In addition, there are now several projects dedicated to *server-side* JavaScript, including

CommonJS (www.commonjs.org), Node.js (`nodejs.org`) and Jaxer (`jaxer.org`).

JavaScript was created by Netscape, the company that built the first wildly successful web browser. Both Netscape and Microsoft have been instrumental in the standardization of JavaScript by ECMA International (formerly the European Computer Manufacturers Association) as ECMAScript. ECMAScript 5, the latest version of the standard, corresponds to the version of JavaScript we use in this book.

The JavaScript chapters of the book are more than just an introduction to the language. They also present computer-programming fundamentals, including control structures, functions, arrays, recursion, strings and objects. You'll see that JavaScript is a portable scripting language and that programs written in JavaScript can run in web browsers across a wide range of devices.

Web Browsers and Web-Browser Portability

Ensuring a consistent look and feel on client-side browsers is one of the great challenges of developing web-based applications. Currently, a standard does not exist to which software vendors must adhere when creating web browsers. Although browsers share a common set of features, each browser might render pages differently. Browsers are available in many versions and on many different platforms (Microsoft Windows, Apple Macintosh, Linux, UNIX, etc.). Vendors add features to each new version that sometimes result in cross-platform incompatibility issues. It's difficult to develop web pages that render correctly on all versions of each browser.

All of the code examples in the book were tested in the five most popular desktop browsers and the two most popular mobile browsers (Fig. 1.5). Support for HTML5, CSS3 and JavaScript features varies by browser. The *HTML5 Test* website (http://html5test.com/) scores each browser based on its support for the latest features of these evolving standards. Figure 1.5 lists the five desktop browsers we use in reverse order of their HTML5 Test scores from most compliant to least compliant at the time of this writing. Internet Explorer 10 (IE10) is expected to have a much higher

compliance rating than IE9. You can also check sites such as http://caniuse.com/ for a list of features covered by each browser.

PORTABILITY TIP 1.1

The web is populated with many different browsers, including many older, less-capable versions, which makes it difficult for authors and web-application developers to create universal solutions. The W3C is working toward the goal of a universal client-side platform (http://www.w3.org/2006/webapi/admin/charter).

Browser	Approximate market share as of August 2011 (http://gs.statcounter.com)	Score out of 450 from html5test.com
Desktop browsers	***Market share***	
Google Chrome 13	17%	330
Mozilla Firefox 6	27%	298
Apple Safari 5.1	7%	293
Opera 11.5	2%	286
Internet Explorer 9	40%	141
Mobile browsers	***Mobile market share***	
iPhone	15% (of mobile browsers)	217
Android	18% (of mobile browsers)	184

Fig. 1.5. HTML5 Test scores for the browsers used to test the examples.

jQuery

jQuery (`jQuery.org`) is currently the most popular of hundreds of *JavaScript libraries*.[3] jQuery simplifies JavaScript programming by making it easier to manipulate a web page's elements and interact with servers in a portable manner across various web browsers. It provides a library of custom graphical user interface (GUI) controls (beyond the basic GUI controls provided by HTML5) that can be used to enhance the look and feel of your web pages.

[3] www.activoinc.com/blog/2008/11/03/jquery-emerges-as-most-popular-javascript-library-for-web-development/.

Validating Your HTML5, CSS3 and JavaScript Code

As you'll see, JavaScript programs typically have HTML5 and CSS3 portions as well. You must use proper HTML5, CSS3 and JavaScript syntax to ensure that browsers process your documents properly. Figure 1.6 lists the validators we used to validate the code in this book. Where possible, we eliminated validation errors.

Technology	Validator URL
HTML5	`http://validator.w3.org/` `http://html5.validator.nu/`
CSS3	`http://jigsaw.w3.org/css-validator/`
JavaScript	`http://www.javascriptlint.com/` `http://www.jslint.com/`

Fig. 1.6. HTML5, CSS3 and JavaScript validators.

1.4. Demos

Browse the web pages in Fig. 1.7 to get a sense of some of the things you'll be able to create using the technologies you'll learn in this book, including HTML5, CSS3, JavaScript, canvas and jQuery. Many of these sites provide links to the corresponding source code, or you can view the page's source code in your browser.

URL	Description
`https://developer.mozilla.org/en-US/demos/`	Mozilla's DemoStudio contains numerous HTML5, canvas, CSS3 and JavaScript demos that use audio, video, animation and more.
`http://js-fireworks.appspot.com/`	Enter your name or message, and this JavaScript animation then writes it using a fireworks effect over the London skyline.
`http://9elements.com/io/projects/html5/canvas/`	Uses HTML5 canvas and audio elements to create interesting effects, and ties in tweets that include the words "HTML5" and "love" (click anywhere on the screen to see the next tweet).
`http://www.zachstronaut.com/lab/text-shadow-box/text-shadow-`	Animated demo of the CSS3 text-shadow effect. Use the mouse to shine a light on the text

box.html	and dynamically change the direction and size of the shadow.
http://clublime.com/lab/html5/sphere/	Uses an HTML5 canvas to create a sphere that rotates and changes direction as you move the mouse cursor.
http://spielzeugz.de/html5/liquid-particles.html	The Liquid Particles demo uses an HTML5 canvas. Move the mouse around the screen and the "particles" (dots or letters) follow.
http://www.paulbrunt.co.uk/bert/	Bert's Breakdown is a fun video game built using an HTML5 canvas.
http://www.openrise.com/lab/FlowerPower/	Canvas app that allows you to draw flowers on the page, adjust their colors, change the shapes of the petals and more.
http://alteredqualia.com/canvasmol/	Uses canvas to display a 3D molecule that can be viewed from any desired angle (0–360 degrees).
http://pasjans-online.pl/	The game of Solitaire built using HTML5.
http://andrew-hoyer.com/experiments/cloth/	Uses canvas to simulate of the movement of a piece of cloth. Click and drag the mouse to move the fabric.
http://www.paulrhayes.com/experiments/cube-3d/	CSS3 demo allows you to use the mouse to tilt and rotate the 3D cube. Includes a tutorial.
http://www.effectgames.com/demos/canvascycle/	Animated waterfall provides a nice demo of using color in HTML5 canvas.
http://macek.github.com/google_pacman/	The Google PAC-MAN® game (a Google Doodle) built in HTML5.
http://www.benjoffe.com/code/games/torus/	A 3D game similar to Tetris® built with JavaScript and canvas.
http://code.almeros.com/code-examples/water-effect-canvas/	Uses canvas and JavaScript to create a water rippling effect. Hover the cursor over the canvas to see the effect. The site includes a tutorial.
http://jqueryui.com/demos/	Numerous jQuery demos, including animations, transitions, color, interactions and more.
http://lab.smashup.it/flip/	Demonstrates a flip box using jQuery.
http://tutorialzine.com/2010/09/html5-canvas-slideshow-jquery/	Slideshow built with HTML5 canvas and jQuery (includes a tutorial).
http://css-tricks.com/examples/Circulate/	Learn how to create an animated circulation effect using jQuery.
http://demo.tutorialzine.com/2010/02/photo-shoot-css-jquery/demo.html	Uses jQuery and CSS to create a photoshoot effect, allowing you to focus on an area of the page and snap a picture (includes a tutorial).

Fig. 1.7. HTML5, CSS3, JavaScript, canvas and jQuery demos.

1.5. Evolution of the Internet and World Wide Web

The Internet—a global network of computers—was made possible by the *convergence of computing and communications technologies*. In the late 1960s, ARPA (the Advanced Research Projects Agency) rolled out blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research institutions. They were to be connected

with communications lines operating at a then-stunning 56 Kbps (i.e., 56,000 bits per second)—this at a time when most people (of the few who could) were connecting over telephone lines to computers at a rate of 110 bits per second. A **bit** (short for "binary digit") is the smallest data item in a computer; it can assume the value 0 or 1.

There was great excitement. Researchers at Harvard talked about communicating with the powerful Univac computer at the University of Utah to handle the intensive calculations related to their computer graphics research. Many other intriguing possibilities were raised. Academic research was about to take a giant leap forward. ARPA proceeded to implement the **ARPANET**, which eventually evolved into today's **Internet**.

Things worked out differently from what was originally planned. Rather than enabling researchers to share each other's computers, it rapidly became clear that communicating quickly and easily via electronic mail was the key early benefit of the ARPANET. This is true even today on the Internet, which facilitates communications of all kinds among the world's Internet users.

Packet Switching

One of the primary goals for ARPANET was to allow *multiple* users to send and receive information simultaneously over the *same* communications paths (e.g., phone lines). The network operated with a technique called **packet switching**, in which digital data was sent in small bundles called **packets**. The packets contained *address*, *error-control* and *sequencing* information. The address information allowed packets to be *routed* to their destinations. The sequencing information helped in reassembling the packets—which, because of complex routing mechanisms, could actually arrive out of order—into their original order for presentation to the recipient. Packets from different senders were intermixed on the same lines to efficiently use the available bandwidth. This packet-switching technique greatly reduced transmission costs, as compared with the cost of dedicated communications lines.

The network was designed to operate without centralized control. If a portion of the network failed, the remaining working portions would still

route packets from senders to receivers over alternative paths for reliability.

TCP/IP

The protocol (i.e., set of rules) for communicating over the ARPANET became known as **TCP**—the **Transmission Control Protocol**. TCP ensured that messages were properly routed from sender to receiver and that they arrived intact.

As the Internet evolved, organizations worldwide were implementing their own networks for both intraorganization (i.e., within the organization) and interorganization (i.e., between organizations) communications. A wide variety of networking hardware and software appeared. One challenge was to get these different networks to communicate. ARPA accomplished this with the development of **IP**—the **Internet Protocol**, truly creating a **network of networks,** the current architecture of the Internet. The combined set of protocols is now commonly called **TCP/IP**. Each computer on the Internet has a unique **IP address**. The current IP standard, Internet Protocol version 4 (IPv4), has been in use since 1984 and will soon run out of possible addresses. The next-generation Internet Protocol, **IPv6**, is just starting to be deployed. It features enhanced security and a new addressing scheme, hugely expanding the number of IP addresses available so that we will not run out of IP addresses in the forseeable future.

Explosive Growth

Initially, Internet use was limited to universities and research institutions; then the military began using it intensively. Eventually, the government decided to allow access to the Internet for commercial purposes. The research and military communities were concerned that response times would become poor as the Internet became saturated with users.

In fact, the opposite has occurred. Businesses realized that they could tune their operations and offer new and better services to their clients, so they started spending vast amounts of money to develop and enhance the Internet. This generated fierce competition among communications carri-

ers and hardware and software suppliers to meet this demand. The result is that **bandwidth** (i.e., the information-carrying capacity) on the Internet's is increasing rapidly as costs dramatically decline.

World Wide Web, HTML, HTTP

The **World Wide Web** allows computer users to execute web-based applications and to locate and view multimedia-based documents on almost any subject over the Internet. The web is a relatively recent creation. In 1989, **Tim Berners-Lee** of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via hyperlinked text documents. Berners-Lee called his invention the **HyperText Markup Language** (**HTML**). He also wrote communication protocols to form the backbone of his new information system, which he called the World Wide Web. In particular, he wrote the **Hypertext Transfer Protocol** (**HTTP**)—a communications protocol used to send information over the web. The **URL (Uniform Resource Locator)** specifies the address (i.e., location) of the web page displayed in the browser window. Each web page on the Internet is associated with a unique URL. URLs usually begin with `http://`.

HTTPS

URLs of websites that handle private information, such as credit card numbers, often begin with `https://`, the abbreviation for **Hypertext Transfer Protocol Secure (HTTPS)**. HTTPS is the standard for transferring encrypted data on the web. It combines HTTP with the Secure Sockets Layer (SSL) and the more recent Transport Layer Security (TLS) cryptographic schemes for securing communications and identification information over the web. Although there are many benefits to using HTTPS, there are a few drawbacks, most notably some performance issues because encryption and decryption consume significant computer processing resources.

Mosaic, Netscape, Emergence of Web 2.0

Web use exploded with the availability in 1993 of the Mosaic browser, which featured a user-friendly graphical interface. Marc Andreessen,

whose team at the National Center for Supercomputing Applications (NCSA) developed Mosaic, went on to found Netscape, the company that many people credit with igniting the explosive Internet economy of the late 1990s. But the "dot com" economic bust brought hard times in the early 2000s. The resurgence that began in 2004 or so has been named **Web 2.0**. Google is widely regarded as the signature company of Web 2.0. Some other companies with Web 2.0 characteristics are YouTube (video sharing), Facebook (social networking), Twitter (microblogging), Groupon (social commerce), Foursquare (mobile check-in), Salesforce (business software offered as online services "in the cloud"), Craigslist (mostly free classified listings), Flickr (photo sharing), Skype (Internet telephony and video calling and conferencing, now owned by Microsoft) and Wikipedia (a free online encyclopedia).

1.6. Web Basics

In this section, we discuss the fundamentals of web-based interactions between a client web browser and a web server. In its simplest form, a *web page* is nothing more than an HTML (HyperText Markup Language) document (with the extension `.html` or `.htm`) that describes to a web browser the document's content and structure.

Hyperlinks

HTML documents normally contain **hyperlinks**, which, when clicked, load a specified web document. Both images and text may be hyperlinked. When the mouse pointer hovers over a hyperlink, the default arrow pointer changes into a hand with the index finger pointing upward. Often hyperlinked text appears underlined and in a different color from regular text in a web page.

Originally employed as a publishing tool for scientific research, hyperlinks are widely used to reference sources, or sites that have more information on a particular topic. The paths created by hyperlinking create the effect of the "web."

When the user clicks a hyperlink, a **web server** locates the requested web page and sends it to the user's web browser. Similarly, the user can type

the *address of a web page* into the browser's *address field* and press *Enter* to view the specified page.

Hyperlinks can reference other web pages, e-mail addresses, files and more. If a hyperlink's URL is in the form `mailto:` *emailAddress*, clicking the link loads your default e-mail program and opens a **message window** addressed to the specified e-mail address. If a hyperlink references a file that the browser is incapable of displaying, the browser prepares to **download** the file, and generally prompts the user for information about how the file should be stored. When a file is downloaded, it's copied onto the user's computer. Programs, documents, images, sound and video files are all examples of downloadable files.

URIs and URLs

URIs (Uniform Resource Identifiers) identify resources on the Internet. URIs that start with `http://` are called *URLs (Uniform Resource Locators)*. Common URLs refer to files, directories or server-side code that performs tasks such as database lookups, Internet searches and business-application processing. If you know the URL of a publicly available resource anywhere on the web, you can enter that URL into a web browser's address field and the browser can access that resource.

Parts of a URL

A URL contains information that directs a browser to the resource that the user wishes to access. Web servers make such resources available to web clients. Popular web servers include Apache's HTTP Server and Microsoft's Internet Information Services (IIS).

Let's examine the components of the URL

http://www.deitel.com/books/downloads.html

The text `http://` indicates that the HyperText Transfer Protocol (HTTP) should be used to obtain the resource. Next in the URL is the server's fully qualified **hostname** (for example, www.deitel.com)—the name of the web-server computer on which the resource resides. This computer is re-

ferred to as the **host**, because it houses and maintains resources. The hostname www.deitel.com is translated into an **IP (Internet Protocol) address**—a numerical value that uniquely identifies the server on the Internet. An Internet **Domain Name System (DNS) server** maintains a database of hostnames and their corresponding IP addresses and performs the translations automatically.

The remainder of the URL (`/books/downloads.html`) specifies the resource's location (`/books`) and name (`downloads.html`) on the web server. The location could represent an actual directory on the web server's file system. For *security* reasons, however, the location is typically a *virtual directory*. The web server translates the virtual directory into a real location on the server, thus hiding the resource's true location.

Making a Request and Receiving a Response

When given a web page URL, a web browser uses HTTP to request the web page found at that address. Figure 1.8 shows a web browser sending a request to a web server.

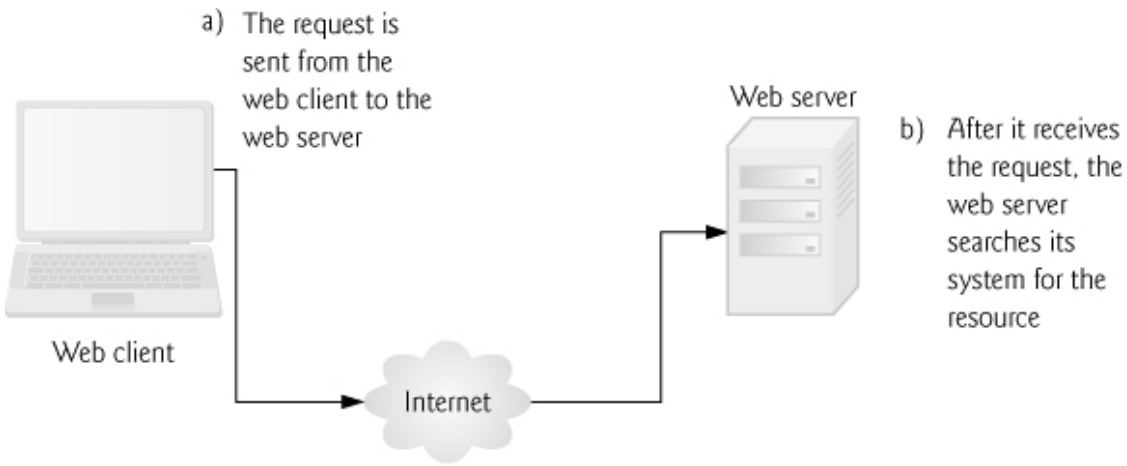

Fig. 1.8. Client requesting a resource from a web server.

In Fig. 1.8, the web browser sends an HTTP request to the server. The request (in its simplest form) is

GET /books/downloads.html HTTP/1.1

The word **GET** is an **HTTP method** indicating that the client wishes to obtain a resource from the server. The remainder of the request provides

the path name of the resource (e.g., an HTML5 document) and the protocol's name and version number (`HTTP/1.1`). The client's request also contains some required and optional headers.

Any server that understands HTTP (version 1.1) can translate this request and respond appropriately. Figure 1.9 shows the web server responding to a request.

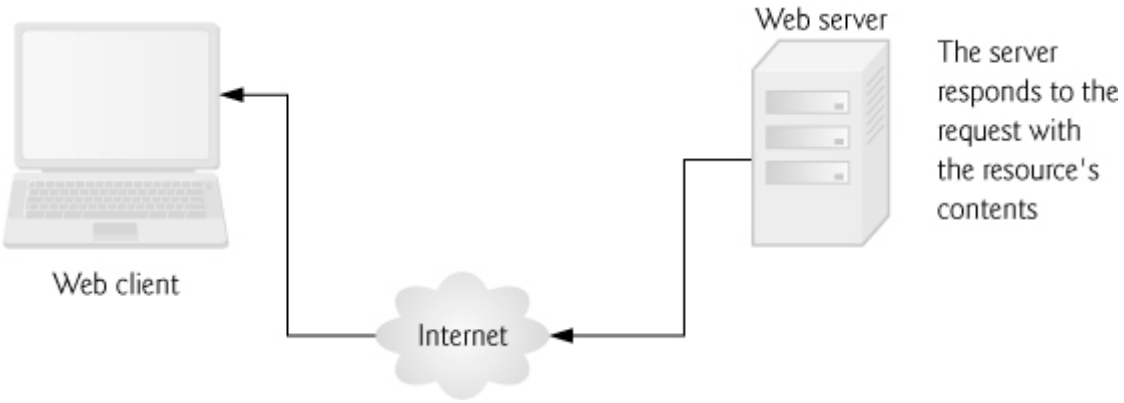

Fig. 1.9. Client receiving a response from the web server.

The server first sends a line of text that indicates the HTTP version, followed by a numeric code and a phrase describing the status of the transaction. For example,

HTTP/1.1 200 OK

indicates success, whereas

HTTP/1.1 404 Not found

informs the client that the web server could not locate the requested resource. A complete list of numeric codes indicating the status of an HTTP transaction can be found at www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

HTTP Headers

Next, the server sends one or more **HTTP headers**, which provide additional information about the data that will be sent. In this case, the server

is sending an HTML5 text document, so one HTTP header for this example would read:

Content-type: text/html

The information provided in this header specifies the **Multipurpose Internet Mail Extensions (MIME) type** of the content that the server is transmitting to the browser. The MIME standard specifies data formats, which programs can use to interpret data correctly. For example, the MIME type `text/plain` indicates that the sent information is text that can be displayed directly. Similarly, the MIME type `image/jpeg` indicates that the content is a JPEG image. When the browser receives this MIME type, it attempts to display the image.

The header or set of headers is followed by a blank line, which indicates to the client browser that the server is finished sending HTTP headers. Finally, the server sends the contents of the requested document (`downloads.html`). The client-side browser then renders (or displays) the document, which may involve additional HTTP requests to obtain associated CSS and images.

HTTP `get` and `post` Requests

The two most common **HTTP request types** (also known as **request methods**) are `get` and `post` . A `get` request typically gets (or retrieves) information from a server, such as an HTML document, an image or search results based on a user-submitted search term. A `post` request typically posts (or sends) data to a server. Common uses of `post` requests are to send form data or documents to a server.

An HTTP request often posts data to a **server-side form handler** that processes the data. For example, when a user performs a search or participates in a web-based survey, the web server receives the information specified in the HTML form as part of the request. `Get` requests and `post` requests can both be used to send data to a web server, but each request type sends the information differently.

A `get` request appends data to the URL, e.g., www.google.com/search?q=deitel. In this case `search` is the name of Google's server-side form handler, `q` is the `name` of a variable in Google's search form and `deitel` is the search term. The `?` in the preceding URL separates the **query string** from the rest of the URL in a request. A *name/value* pair is passed to the server with the *name* and the *value* separated by an equals sign (=). If more than one *name/value* pair is submitted, each pair is separated by an ampersand (&). The server uses data passed in a query string to retrieve an appropriate resource from the server. The server then sends a response to the client. A `get` request may be initiated by submitting an HTML form whose `method` attribute is set to `"get"`, or by typing the URL (possibly containing a query string) directly into the browser's address bar. We discuss HTML forms in Chapters 2–3.

A `post` request sends form data as part of the HTTP message, not as part of the URL. A `get` request typically limits the query string (i.e., everything to the right of the `?`) to a specific number of characters, so it's often necessary to send large amounts of information using the `post` method. The `post` method is also sometimes preferred because it hides the submitted data from the user by embedding it in an HTTP message. If a form submits several hidden input values along with user-submitted data, the `post` method might generate a URL like www.searchengine.com/search. The form data still reaches the server and is processed in a similar fashion to a `get` request, but the user does not see the exact information sent.

SOFTWARE ENGINEERING OBSERVATION 1.1

The data sent in a `post` request is not part of the URL, and the user can't see the data by default. However, tools are available that expose this data, so you should not assume that the data is secure just because a `post` request is used.

Client-Side Caching

Browsers often **cache** (save on disk) recently viewed web pages for quick reloading. If there are no changes between the version stored in the cache and the current version on the web, this speeds up your browsing experience. An HTTP response can indicate the length of time for which the content remains "fresh." If this amount of time has not been reached, the browser can avoid another request to the server. If not, the browser loads the document from the cache. Similarly, there's also the "not modified" HTTP response, indicating that the file content has not changed since it was last requested (which is information that's send in the request). Browsers typically do not cache the server's response to a `post` request, because the next `post` might not return the same result. For example, in a survey, many users could visit the same web page and answer a question. The survey results could then be displayed for the user. Each new answer would change the survey results.

1.7. Multitier Application Architecture

Web-based applications are often **multitier applications** (sometimes referred to as **n-tier applications**) that divide functionality into separate **tiers** (i.e., logical groupings of functionality). Although tiers can be located on the same computer, the tiers of web-based applications often reside on separate computers. Figure 1.10 presents the basic structure of a **three-tier web-based application**.

The **bottom tier** (also called the data tier or the information tier) maintains the application's data. This tier typically stores data in a relational database management system (RDBMS). We discuss RDBMSs in Chapter 18. For example, Amazon might have an inventory information database containing product descriptions, prices and quantities in stock. Another database might contain customer information, such as user names, billing addresses and credit card numbers. These may reside on one or more computers, which together comprise the application's data.

The **middle tier** implements business logic, controller logic and presentation logic to control interactions between the application's clients and its data. The middle tier acts as an intermediary between data in the information tier and the application's clients. The middle-tier **controller logic**

processes client requests (such as requests to view a product catalog) and retrieves data from the database. The middle-tier **presentation logic** then processes data from the information tier and presents the content to the client. Web applications typically present data to clients as HTML documents.

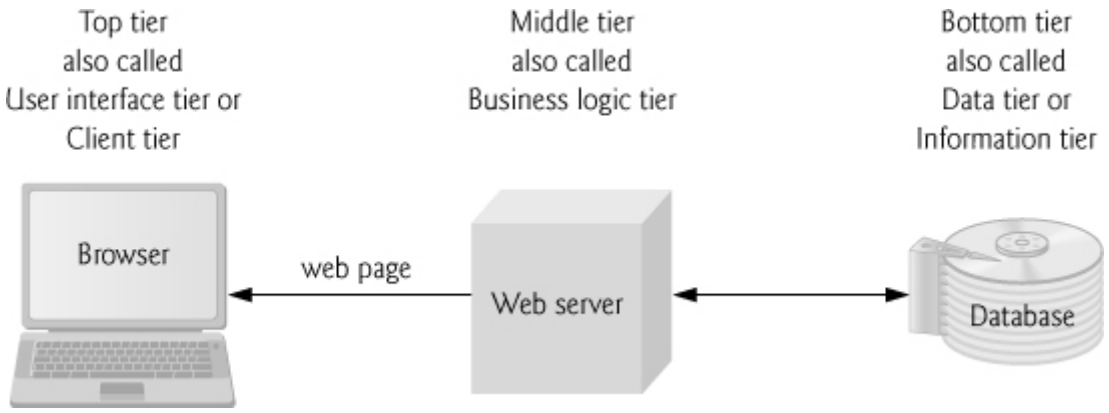

Fig. 1.10. Three-tier architecture.

Business logic in the middle tier enforces **business rules** and ensures that data is reliable before the application updates a database or presents data to users. Business rules dictate how clients access data and how applications process data. For example, a business rule in the middle tier of a retail store's web-based application might ensure that all product quantities remain positive. A client request to set a negative quantity in the bottom tier's product information database would be rejected by the middle tier's business logic.

The **top tier**, or client tier, is the application's user interface, which gathers input and displays output. Users interact directly with the application through the user interface, which is typically a web browser or a mobile device. In response to user actions (e.g., clicking a hyperlink), the client tier interacts with the middle tier to make requests and to retrieve data from the information tier. The client tier then displays the data retrieved for the user.

1.8. Client-Side Scripting versus Server-Side Scripting

Client-side scripting with JavaScript can be used to validate user input, to interact with the browser, to enhance web pages, and to add client/server communication between a browser and a web server.

Client-side scripting does have limitations, such as browser dependency; the browser or **scripting host** must support the scripting language and capabilities. Scripts are restricted from arbitrarily accessing the local hardware and file system for security reasons. Another issue is that client-side scripts can be viewed by the client by using the browser's source-viewing capability. Sensitive information, such as passwords or other personally identifiable data, should not be on the client. All client-side data validation should be mirrored on the server. Also, placing certain operations in JavaScript on the client can open web applications to security issues.

Programmers have more flexibility with **server-side scripts**, which often generate custom responses for clients. For example, a client might connect to an airline's web server and request a list of flights from Boston to San Francisco between April 19 and May 5. The server queries the database, dynamically generates an HTML document containing the flight list and sends the document to the client. This technology allows clients to obtain the most current flight information from the database by connecting to an airline's web server.

Server-side scripting languages have a wider range of programmatic capabilities than their client-side equivalents. Server-side scripts also have access to server-side software that extends server functionality—Microsoft web servers use **ISAPI (Internet Server Application Program Interface) extensions** and Apache HTTP Servers use **modules**. Components and modules range from programming-language support to counting the number of web-page hits. We discuss some of these components and modules in subsequent chapters.

1.9. World Wide Web Consortium (W3C)

In October 1994, Tim Berners-Lee founded an organization—the **World Wide Web Consortium** (**W3C**)—devoted to developing nonproprietary, interoperable technologies for the World Wide Web. One of the W3C's primary goals is to make the web universally *accessible*—regardless of disability, language or culture. The W3C home page (www.w3.org) provides extensive resources on Internet and web technologies.

The W3C is also a standards organization. Web technologies standardized by the W3C are called **Recommendations**. Current and forthcoming W3C Recommendations include the HyperText Markup Language 5 (HTML5), Cascading Style Sheets 3 (CSS3) and the Extensible Markup Language (XML). A recommendation is not an actual software product but a document that specifies a technology's role, syntax rules and so forth.

1.10. Web 2.0: Going Social

In 2003 there was a noticeable shift in how people and businesses were using the web and developing web-based applications. The term **Web 2.0** was coined by **Dale Dougherty** of **O'Reilly Media**[4] in 2003 to describe this trend. Generally, Web 2.0 companies use the web as a platform to create collaborative, community-based sites (e.g., social networking sites, blogs, wikis).

---

[4] T. O'Reilly, "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1>.

Web 1.0 versus Web 2.0

Web 1.0 (the state of the web through the 1990s and early 2000s) was focused on a relatively small number of companies and advertisers producing content for users to access (some people called it the "brochure web"). Web 2.0 *involves* the users—not only do they often create content, but they help organize it, share it, remix it, critique it, update it, etc. One way to look at Web 1.0 is as a *lecture*, a small number of professors informing a large audience of students. In comparison, Web 2.0 is a *conversation*, with everyone having the opportunity to speak and share views. Companies that understand Web 2.0 realize that their products and services are conversations as well.

Architecture of Participation

Web 2.0 is providing new opportunities and connecting people and content in unique ways. Web 2.0 embraces an **architecture of participation**—a design that encourages user interaction and community contributions. You, the user, are the most important aspect of Web 2.0—so important, in fact, that in 2006, *TIME* magazine's "Person of the Year" was "you."[5] The article recognized the social phenomenon of Web 2.0—the shift away from a *powerful few* to an *empowered many*. Several popular blogs now compete with traditional media powerhouses, and many Web 2.0 companies are built almost entirely on user-generated content. For websites like Facebook®, Twitter™, YouTube, eBay® and Wikipedia®, users create the content, while the companies provide the platforms on which to enter, manipulate and share the information. These companies *trust their users*—without such trust, users cannot make significant contributions to the sites.

---

[5] L. Grossman, "TIME's Person of the Year: You." *TIME*, December 2006 <http://www.time.com/time/magazine/article/0,9171,1569514,00.html>.

The architecture of participation has influenced software development as well. Opensource software is available for anyone to use and modify with few or no restrictions (we'll say more about open source in Section 1.12). Using **collective intelligence**—the concept that a large diverse group of people will create smart ideas—communities collaborate to develop software that many people believe is better and more robust than proprietary software. Rich Internet Applications (RIAs) are being developed using technologies (such as Ajax, which we discuss throughout the book) that have the look and feel of desktop software, enhancing a user's overall experience.

Search Engines and Social Media

Search engines, including Google™, Microsoft Bing™, and many more, have become essential to sifting through the massive amount of content on the web. Social bookmarking sites such as del.icio.us allow users to share their favorite sites with others. Social media sites such as Digg™ enable the community to decide which news articles are the most signifi-

cant. The way we find the information on these sites is also changing—people are **tagging** (i.e., labeling) web content by subject or keyword in a way that helps anyone locate information more effectively.

Semantic Web

In the future, computers will learn to understand the meaning of the data on the web—the beginnings of the **Semantic Web** are already appearing. Continual improvements in hardware, software and communications technologies will enable exciting new types of applications.

These topics and more are covered in our online e-book, Dive Into® Web 2.0 (available at http://www.deitel.com/diveintoweb20/). The e-book highlights the major characteristics and technologies of Web 2.0, providing examples of popular Web 2.0 companies and Web 2.0 Internet business and monetization models. We discuss user-generated content, blogging, content networks, social networking, location-based services and more. In the subsequent chapters of this book, you'll learn key software technologies for building web-based applications.

Google

In 1996, Stanford computer science Ph.D. candidates Larry Page and Sergey Brin began collaborating on a new search engine. In 1997, they chose the name Google—a play on the mathematical term *googol*, a quantity represented by the number "one" followed by 100 "zeros" (or 10^{100})—a staggeringly large number. Google's ability to return extremely accurate search results quickly helped it become the most widely used search engine and one of the most popular websites in the world.

Google continues to be an innovator in search technologies. For example, Google Goggles is a fascinating mobile app (available on Android and iPhone) that allows you to perform a Google search using a photo rather than entering text. You simply take a picture of a landmark, book (covers or barcodes), logo, art or wine bottle label, and Google Goggles scans the photo and returns search results. You can also take a picture of text (for example, a restaurant menu or a sign) and Google Goggles will translate it for you.

Web Services and Mashups

We include in this book a substantial treatment of web services (Chapters 22, 25 and 28) and introduce the applications-development methodology of *mashups*, in which you can rapidly develop powerful and intriguing applications by combining (often free) complementary web services and other forms of information feeds (Fig. 1.11). One of the first mashups was www.housingmaps.com, which combines the real estate listings provided by www.craigslist.org with the mapping capabilities of Google Maps to offer maps that show the locations of apartments for rent in a given area.

Web services source	How it's used
Google Maps	Mapping services
Facebook	Social networking
Foursquare	Mobile check-in
LinkedIn	Social networking for business
YouTube	Video search
Twitter	Microblogging
Groupon	Social commerce
Netflix	Movie rentals
eBay	Internet auctions
Wikipedia	Collaborative encyclopedia
PayPal	Payments
Last.fm	Internet radio
Amazon eCommerce	Shopping for books and more
Salesforce.com	Customer Relationship Management (CRM)
Skype	Internet telephony
Microsoft Bing	Search
Flickr	Photo sharing
Zillow	Real estate pricing
Yahoo Search	Search
WeatherBug	Weather

Fig. 1.11. Some popular web services that you can use to build web applications *(*www.programmableweb.com/apis/directory/1?sort=mashups*)*.

Web services, inexpensive computers, abundant high-speed Internet access, open source software and many other elements have inspired new, exciting, *lightweight business models* that people can launch with only a

small investment. Some types of websites with rich and robust functionality that might have required hundreds of thousands or even millions of dollars to build in the 1990s can now be built for nominal sums.

Ajax

Ajax is one of the premier Web 2.0 software technologies (Fig. 1.12). Ajax helps Internet-based applications perform like desktop applications—a difficult task, given that such applications suffer transmission delays as data is shuttled back and forth between your computer and servers on the Internet.

Chapter	Ajax coverage
Chapter 1	This chapter introduces Ajax.
Chapters 2–14	These chapters cover several key technologies used in Ajax web applications, including HTML5, CSS3, JavaScript, JavaScript event handling, the Document Object Model (DOM) and dynamic manipulation of an HTML5 document—known as dynamic HTML.
Chapter 15	Web applications use XML extensively to represent structured data. This chapter introduces XML, XML-related technologies and key JavaScript capabilities for loading and manipulating XML documents programmatically.
Chapter 16	This chapter uses the technologies presented in Chapters 2–15 to build Ajax-enabled web applications. We use both XML and JSON (JavaScript Object Notation) to send/receive data between the client and the server. The chapter begins by building basic Ajax applications using JavaScript and the browser's `XMLHttpRequest` object. We then build an Ajax application using the jQuery JavaScript libraries.
Chapters 21, 24 and 27	These chapters use Ajax in Microsoft's ASP.NET with C# and in ASP.NET with Visual Basic, and in JavaServer Faces (JSF), respectively, to implement Ajax applications that use Ajax for form validation and partial-page updates.

Fig. 1.12. Ajax coverage in *Internet & World Wide Web How to Program, 5/e*.

Social Applications

Over the last several years, there's been a tremendous increase in the number of social applications on the web. Even though the computer industry is mature, these sites were still able to become phenomenally successful in a relatively short period. Figure 1.13 discusses a few of the social applications that are making an impact.

Company	Description
Facebook	Facebook was launched in 2004 and is already worth an estimated $100 billion. By January 2011, Facebook was the most active site on the Internet with more than 750 million users who were spending 700 billion minutes on Facebook per month (`www.facebook.com/press/info.php?statistics`). At its current growth rate (about 5% per month), Facebook will reach one billion users in 2012, out of two billion Internet users! The activity on the site makes it extremely attractive for application developers. Each day, over 20 million applications are installed by Facebook users (`www.facebook.com/press/info.php?statistics`).
Twitter	Twitter (founded in 2006) has revolutionized *microblogging*. Users post tweets—messages up to 140 characters long. Approximately 140 million tweets are posted per day. You can follow the tweets of friends, celebrities, businesses, government representatives (including Barack Obama, who has 10 million followers), and so on, or you can follow tweets by subject to track news, trends and more. At the time of this writing, Lady Gaga had the most followers (over 13 million). Twitter has become the point of origin for many breaking news stories worldwide.
Groupon	Groupon, a *social commerce* site, was launched in 2008. By August 2011 the company was valued as high as $25 billion, making it the fastest growing company ever! Groupon offers daily deals in each market for restaurants, retailers, services, attractions and more. Deals are activated only after a minimum number of people sign up to buy the product or service. If you sign up for a deal and it has yet to meet the minimum, you might be inclined to tell others about the deal via e-mail, Facebook, Twitter, etc. One of the most successful national Groupon deals to date was a certificate for $50 worth of merchandise from a major retailer for $25. More than 620,000 vouchers were sold in one day (`www.huffingtonpost.com/2011/06/30/the-most-successful-group_n_887711.html`)!
Foursquare	Foursquare, launched in 2009, is a mobile *check-in* application that allows you to notify your friends of your whereabouts. You can download the app to your smartphone and link it to your Facebook and Twitter accounts so your friends can follow you from multiple platforms. If you do not have a smartphone, you can check in by text message. Foursquare uses GPS to determine your location. Businesses use Foursquare to send offers to users in the area. Launched in March 2009, Foursquare already has over 10 million users worldwide (`foursquare.com/about`).
Skype	Skype (founded in 2003) allows you to make mostly free voice and video calls over the Internet using a technology called *VoIP (Voice over IP*; IP stands for "Internet Protocol"*)*. The company was recently sold to Microsoft for $8.5 billion

recently sold to Microsoft for $8.5 billion.

YouTube	YouTube is a video-sharing site that was founded in 2005. Within one year, the company was purchased by Google for $1.65 billion. YouTube now accounts for 8.2% of all Internet traffic (www.engadget.com/2011/05/17/study-finds-netflix-is-the-largest-source-of-internet-traffic-in/). Within one week of the release of Apple's iPhone 3GS—the first iPhone model to offer video—mobile uploads to YouTube grew 400% (www.hypebot.com/hypebot/2009/06/youtube-reports-1700-jump-in-mobile-video.html).

Fig. 1.13. Social applications.

1.11. Data Hierarchy

Data items processed by computers form a **data hierarchy** that becomes larger and more complex in structure as we progress from bits to characters to fields, and so on. Figure 1.14 illustrates a portion of the data hierarchy. Figure 1.15 summarizes the data hierarchy's levels.

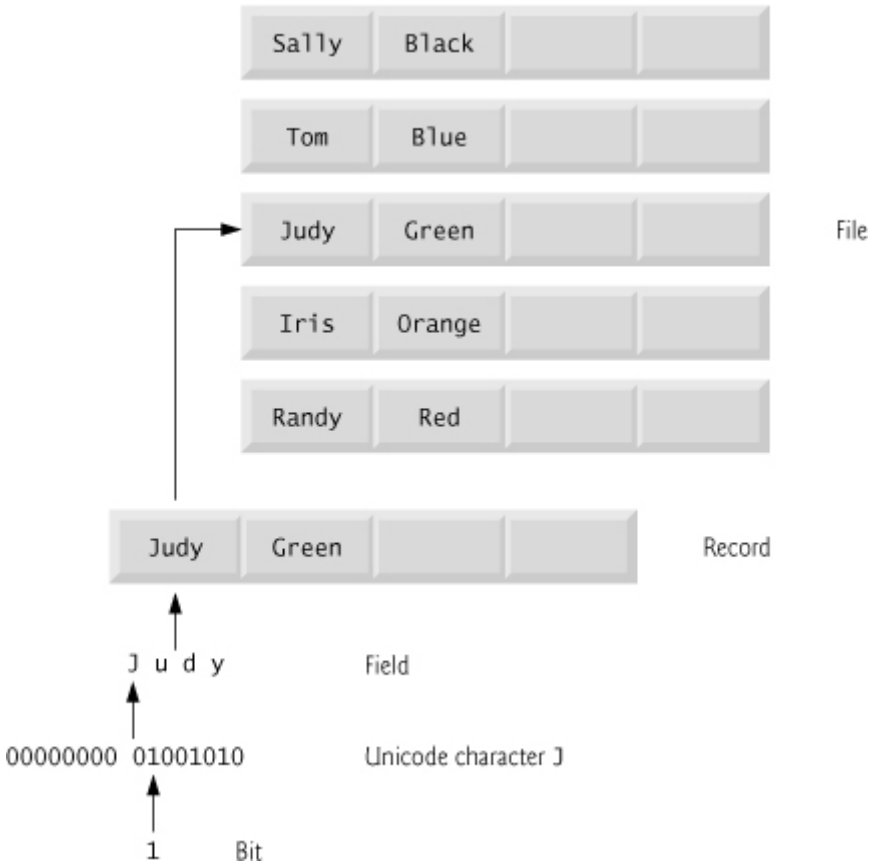

Fig. 1.14. Data hierarchy.

Level	Description
Bits	The smallest data item in a computer can assume the value 0 or the value 1.

	Such a data item is called a **bit** (short for "binary digit"—a digit that can assume one of two values). It's remarkable that the impressive functions performed by computers involve only the simplest manipulations of `0`s and `1`s—*examining a bit's value*, *setting a bit's value* and *reversing a bit's value* (from `1` to `0` or from `0` to `1`).
Characters	It's tedious for people to work with data in the low-level form of bits. Instead, they prefer to work with *decimal digits* (0–9), *letters* (A–Z and a–z), and *special symbols* (e.g., $, @, %, &, *, (,), –, +, ", :, ? and /). Digits, letters and special symbols are known as **characters**. The computer's **character set** is the set of all the characters used to write programs and represent data items. Computers process only `1`s and `0`s, so a computer's character set represents every character as a pattern of `1`s and `0`s. Java uses **Unicode®** characters that are composed of two **bytes**, each composed of eight bits. Unicode contains characters for many of the world's languages. See Appendix F for more information on Unicode. See Appendix D for more information on the **ASCII (American Standard Code for Information Interchange)** character set—the popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.
Fields	Just as characters are composed of bits, **fields** are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters could be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.
Records	Several related fields can be used to compose a **record** (implemented as a `class` in Java). In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses): • Employee identification number (a whole number) • Name (a string of characters) • Address (a string of characters) • Hourly pay rate (a number with a decimal point) • Year-to-date earnings (a number with a decimal point) • Amount of taxes withheld (a number with a decimal point) Thus, a record is a group of related fields. In the preceding example, all the fields belong to the same employee. A company might have many employees and a payroll record for each one.
Files	A **file** is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—any organization of the bytes in a file, such as organizing the data into records, is a view created by the application programmer.] It's not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.
Database	A **database** is an electronic collection of data that's organized for easy access and manipulation. There are various database models. In this book, we introduce relational databases in which data is stored in simple *tables*. A table includes *records* and *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade point average. The data for each student is a record, and the individual pieces of information in each record are the fields. You can search, sort and manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with databases of courses, on-campus housing, meal plans, etc. We discuss databases in Chapter 18 and use them in the server-side programming chapters.

Fig. 1.15. Levels of the data hierarchy.

1.12. Operating Systems

Operating systems are software systems that make using computers more convenient for users, application developers and system administrators. Operating systems provide services that allow each application to execute safely, efficiently and *concurrently* (i.e., in parallel) with other applications. The software that contains the core components of the operating system is called the **kernel**. Popular desktop operating systems include Linux, Windows 7 and Mac OS X. Popular mobile operating systems used in smartphones and tablets include Google's Android, Apple's iOS (for iPhone, iPad and iPod Touch devices), BlackBerry OS and Windows Phone 7.

1.12.1. Desktop and Notebook Operating Systems

In this section we discuss two of the popular desktop operating systems—the proprietary Windows operating system and the open source Linux operating system.

Windows—A Proprietary Operating System

In the mid-1980s, Microsoft developed the **Windows operating system**, consisting of a graphical user interface built on top of DOS—an enormously popular personal-computer operating system of the time that users interacted with by *typing* commands. Windows borrowed from many concepts (such as icons, menus and windows) developed by Xerox PARC and popularized by early Apple Macintosh operating systems. Windows 7 is Microsoft's latest operating system—its features include enhancements to the user interface, faster startup times, further refinement of security features, touch-screen and multitouch support, and more. Windows is a *proprietary* operating system—it's controlled by Microsoft exclusively. Windows is by far the world's most widely used operating system.

Linux—An Open-Source Operating System

The Linux operating system is perhaps the greatest success of the *open-source* movement. **Open-source software** departs from the *proprietary*

software development style that dominated software's early years. With open-source development, individuals and companies *contribute* their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge. Open-source code is often scrutinized by a much larger audience than proprietary software, so errors often get removed faster. Open source also encourages more innovation. Enterprise systems companies, such as IBM, Oracle and many others, have made significant investments in Linux open-source development.

Some key organizations in the open-source community are the Eclipse Foundation (the Eclipse Integrated Development Environment helps programmers conveniently develop software), the Mozilla Foundation (creators of the Firefox web browser), the Apache Software Foundation (creators of the Apache web server used to develop web-based applications) and SourceForge (which provides the tools for managing open-source projects—it has over 306,000 of them under development). Rapid improvements to computing and communications, decreasing costs and open-source software have made it much easier and more economical to create a software-based business now than just a decade ago. A great example is Facebook, which was launched from a college dorm room and built with open-source software.[6]

---

[6] developers.facebook.com/opensource/.

The **Linux** kernel is the core of the most popular open-source, freely distributed, full-featured operating system. It's developed by a loosely organized team of volunteers and is popular in servers, personal computers and embedded systems. Unlike that of proprietary operating systems like Microsoft's Windows and Apple's Mac OS X, Linux source code (the program code) is available to the public for examination and modification and is free to download and install. As a result, Linux users benefit from a community of developers actively debugging and improving the kernel, an absence of licensing fees and restrictions, and the ability to completely customize the operating system to meet specific needs.

A variety of issues—such as Microsoft's market power, the small number of user-friendly Linux applications and the diversity of Linux distributions, such as Red Hat Linux, Ubuntu Linux and many others—have prevented widespread Linux use on desktop computers. But Linux has become extremely popular on servers and in embedded systems, such as Google's Android-based smartphones.

1.12.2. Mobile Operating Systems

Two of the most popular mobile operating systems are Apple's iOS and Google's Android.

iOS for iPhone®, iPad® and iPod Touch®

Apple, founded in 1976 by Steve Jobs and Steve Wozniak, quickly became a leader in personal computing. In 1979, Jobs and several Apple employees visited Xerox PARC (Palo Alto Research Center) to learn about Xerox's desktop computer that featured a graphical user interface (GUI). That GUI served as the inspiration for the Apple Lisa personal computer (designed for business customers) and, more notably, the Apple Macintosh, launched with much fanfare in a memorable Super Bowl ad in 1984. Steve Jobs left Apple in 1985 and founded NeXT Inc.

The Objective-C programming language, created by Brad Cox and Tom Love at Stepstone in the early 1980s, added capabilities for object-oriented programming (OOP) to the C programming language. In 1988, NeXT licensed Objective-C from StepStone and developed an Objective-C compiler and libraries which were used as the platform for the NeXTSTEP operating system's user interface and Interface Builder—used to construct graphical user interfaces. Apple's Mac OS X operating system is a descendant of NeXTSTEP. Apple's proprietary iPhone operating system, **iOS**, is derived from Apple's Mac OS X and is used in the iPhone, iPad and iPod Touch devices.

You can download apps directly onto your iPhone, iPad or iPod device through the App Store. There are over 425,000 apps in the App Store.

Google's Android

Android—the fastest growing mobile and smartphone operating system—is based on the Linux kernel and Java. Experienced Java programmers can quickly dive into Android development. One benefit of developing Android apps is the openness of the platform. The operating system is open source and free.

The Android operating system was developed by Android, Inc., which was acquired by Google in 2005. In 2007, the Open Handset Alliance™—a consortium of 34 companies initially and 84 by 2011—was formed to continue developing Android. As of June 2011, more than 500,000 Android smartphones were being activated each day![7] Android smartphones are now outselling iPhones in the United States.[8] The Android operating system is used in numerous smartphones (such as the Motorola Droid, HTC EVO™ 4G, Samsung Vibrant™ and many more), e-reader devices (such as the Barnes and Noble Nook™), tablet computers (such as the Dell Streak and the Samsung Galaxy Tab), in-store touch-screen kiosks, cars, robots, multimedia players and more.

---

[7] news.cnet.com/8301-13506_3-20074956-17/google-500000-android-devices-activated-each-day/.

---

[8] www.pcworld.com/article/196035/android_outsells_the_iphone_no_big_surprise.html.

You can download apps directly onto your Android device through Android Market and other app marketplaces. As of August 2011, there were over 250,000 apps in Google's Android Market.

1.13. Types of Programming Languages

Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation* steps. Any computer can directly understand only its own **machine language**, defined by its hardware design. Machine languages generally consist of numbers (ultimately reduced to 1s and 0s). Such languages are cumbersome for humans.

Programming in machine language—the numbers that computers could directly understand—was simply too slow and tedious for most programmers. Instead, they began using Englishlike abbreviations to represent elementary operations. These abbreviations formed the basis of **assembly languages**. *Translator programs* called **assemblers** were developed to convert assembly-language programs to machine language. Although assembly-language code is clearer to humans, it's incomprehensible to computers until translated to machine language.

To speed the programming process even further, **high-level languages** were developed in which single statements could be written to accomplish substantial tasks. High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical expressions. Translator programs called **compilers** convert high-level language programs into machine language.

The process of compiling a large high-level language program into machine language can take a considerable amount of computer time. **Interpreter** programs were developed to execute high-level language programs directly, although more slowly than compiled programs. In this book we study several key programming languages, including JavaScript and PHP—each of these **scripting languages** is processed by interpreters. Figure 1.16 introduces a number of popular programming languages.

PERFORMANCE TIP 1.1

Interpreters have an advantage over compilers in Internet scripting. An interpreted program can begin executing as soon as it's downloaded to the client's machine, without needing to be compiled before it can execute. On the downside, interpreted scripts generally run slower than compiled code.

Programming language	Description
C	C was implemented in 1972 by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development

	language. Today, most of the code for general-purpose operating systems is written in C or C++.
C++	C++, an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides a number of features that "spruce up" the C language, but more important, it provides capabilities for object-oriented programming.
Objective-C	Objective-C is an object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It has become the key programming language for the Mac OS X operating system and all iOS-based devices (such as iPods, iPhones and iPads).
Visual Basic	Microsoft's Visual Basic language (based on the Basic language developed at Dartmouth College in the 1960s) was introduced in the early 1990s to simplify Microsoft Windows applications development. Its latest versions support object-oriented programming.
Visual C#	Microsoft's three primary object-oriented programming languages are Visual Basic, Visual C++ (based on C++) and C# (based on C++ and Java, and developed for integrating the Internet and the web into computer applications).
Java	Sun Microsystems in 1991 funded an internal corporate research project led by James Gosling, which resulted in the C++-based object-oriented programming language called Java. A key goal of Java is to enable the writing of programs that will run on a great variety of computer systems and computer-controlled devices. This is sometimes called "write once, run anywhere." Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide applications for consumer devices (smartphones, television set-top boxes and more) and for many other purposes.
PHP	PHP—an object-oriented, "open-source" (see Section 1.12) "scripting" language supported by a community of users and developers—is used by numerous websites including Wikipedia and Facebook. PHP is *platform independent*—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports many databases, including MySQL. Two other popular languages similar in concept to PHP are Perl and Python. The term "LAMP" describes four key technologies for building open-source software—Linux (operating system), Apache (web server), MySQL (database) and PHP or Perl or Python (server-side scripting languages).
Python	Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam (CWI), Python draws heavily from Modula-3—a systems-programming language. Python is "extensible"—it can be extended through classes and programming interfaces.
JavaScript	JavaScript—developed by Brendan Eich at Netscape—is the most widely used scripting language. It's primarily used to add programmability to web pages—for example, animations and interactivity with the user. It's provided with all major web browsers.
Ruby on Rails	Ruby—created in the mid-1990s by Yukihiro Matsumoto—is an open-source, object-oriented programming language with a simple syntax that's similar to Python. Ruby on Rails combines the scripting language Ruby with the Rails web-application framework developed by 37Signals. Their book, *Getting Real* (`gettingreal.37signals.com/toc.php`), is a must read for web developers. Many Ruby on Rails developers have reported productivity gains over other languages when developing database-intensive web applications. Ruby on Rails was used to build Twitter's user interface.
Scala	Scala (`www.scala-lang.org/node/273`)—short for "scalable language"—was designed by Martin Odersky, a professor at École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. Released in 2003, Scala uses both the *object-oriented* and *functional programming* paradigms and is designed to inte-

grate with Java. Programming in Scala can significantly reduce the amount of code in your applications. Twitter and Foursquare use Scala.

Fig. 1.16. Popular programming languages.

1.14. Object Technology

Building software quickly, correctly and economically remains an elusive goal at a time when demands for new and more powerful software are soaring. *Objects*, or more precisely the *classes* objects come from, are essentially *reusable* software components. There are date objects, time objects, audio objects, video objects, automobile objects, people objects, etc. Almost any *noun* can be reasonably represented as a software object in terms of *attributes* (e.g., name, color and size) and *behaviors* (e.g., calculating, moving and communicating). Software developers are discovering that using a modular, object-oriented design and implementation approach can make software-development groups much more productive than was possible with earlier techniques—object-oriented programs are often easier to understand, correct and modify.

The Automobile as an Object

Let's begin with a simple analogy. Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal*. What must happen before you can do this? Well, before you can drive a car, someone has to *design* it. A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house. These drawings include the design for an accelerator pedal. The pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal hides the mechanisms that slow the car, and the steering wheel *hides* the mechanisms that turn the car. This enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.

Before you can drive a car, it must be *built* from the engineering drawings that describe it. A completed car has an *actual* accelerator pedal to make the car go faster, but even that's not enough—the car won't acceler-

ate on its own (hopefully!), so the driver must *press* the pedal to accelerate the car.

Methods and Classes

Let's use our car example to introduce some key object-oriented programming concepts. Performing a task in a program requires a **method**. The method houses the program statements that actually perform its tasks. It hides these statements from its user, just as a car's accelerator pedal hides from the driver the mechanisms of making the car go faster. In object-oriented programming languages, we create a program unit called a **class** to house the set of methods that perform the class's tasks. For example, a class that represents a bank account might contain one method to *deposit* money to an account, another to *withdraw* money from an account and a third to *inquire* what the account's current balance is. A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.

Instantiation

Just as someone has to *build a car* from its engineering drawings before you can actually drive a car, you must *build an object* from a class before a program can perform the tasks that the class's methods define. The process of doing this is called *instantiation*. An object is then referred to as an **instance** of its class.

Reuse

Just as a car's engineering drawings can be *reused* many times to build many cars, you can *reuse* a class many times to build many objects. Reuse of existing classes when building new classes and programs saves time and effort. Reuse also helps you build more reliable and effective systems, because existing classes and components often have gone through extensive *testing*, *debugging* and *performance tuning*. Just as the notion of *interchangeable parts* was crucial to the Industrial Revolution, reusable classes are crucial to the software revolution that has been spurred by object technology.

SOFTWARE ENGINEERING OBSERVATION 1.2

Use a building-block approach to creating your programs. Avoid reinventing the wheel—use existing pieces wherever possible. This software reuse is a key benefit of object-oriented programming.

Messages and Method Calls

When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster. Similarly, you *send messages to an object*. Each message is implemented as a **method call** that tells a method of the object to perform its task. For example, a program might call a particular bank-account object's *deposit* method to increase the account's balance.

Attributes and Instance Variables

A car, besides having capabilities to accomplish tasks, also has *attributes*, such as its color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading). Like its capabilities, the car's attributes are represented as part of its design in its engineering diagrams (which, for example, include an odometer and a fuel gauge). As you drive an actual car, these attributes are carried along with the car. Every car maintains its *own* attributes. For example, each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.

An object, similarly, has attributes that it carries along as it's used in a program. These attributes are specified as part of the object's class. For example, a bank-account object has a *balance attribute* that represents the amount of money in the account. Each bank-account object knows the balance in the account it represents, but *not* the balances of the *other* accounts in the bank. Attributes are specified by the class's **instance variables**.

Encapsulation

Classes **encapsulate** (i.e., wrap) attributes and methods into objects—an object's attributes and methods are intimately related. Objects may communicate with one another, but normally they're not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves. This **information hiding** is crucial to good software engineering.

Inheritance

A new class of objects can be created quickly and conveniently by **inheritance**—the new class absorbs the characteristics of an existing class, possibly customizing them and adding unique characteristics of its own. In our car analogy, an object of class "convertible" certainly *is an* object of the more *general* class "automobile," but more *specifically*, the roof can be raised or lowered.

1.15. Keeping Up-to-Date with Information Technologies

This completes our introduction to the Internet and the web. As you work through the book, if you have a question, send an e-mail to deitel@deitel.com and we'll get back to you promptly. We hope you enjoy using *Internet and World Wide Web How to Program, 5/e*. Figure 1.17 lists key technical and business publications that will help you stay up-to-date with the latest news and trends in computer, Internet and web technology. Enjoy!

Publication	URL
ACM TechNews	technews.acm.org/
ACM Transactions on Accessible Computing	www.is.umbc.edu/taccess/index.html
ACM Transactions on Internet Technology	toit.acm.org/
Bloomberg BusinessWeek	www.businessweek.com
CNET	news.cnet.com
Communications of the ACM	cacm.acm.org/
Computer World	www.computerworld.com
Engadget	www.engadget.com
eWeek	www.eweek.com
Fast Company	www.fastcompany.com/
Fortune	money.cnn.com/magazines/fortune/
IEEE Computer	www.computer.org/portal/web/computer
IEEE Internet Computing	www.computer.org/portal/web/internet/home
InfoWorld	www.infoworld.com
Mashable	mashable.com
PCWorld	www.pcworld.com
SD Times	www.sdtimes.com
Slashdot	slashdot.org/
Smarter Technology	www.smartertechnology.com
Technology Review	technologyreview.com
Techcrunch	techcrunch.com
Wired	www.wired.com

Fig. 1.17. Technical and business publications.

Self-Review Exercises

1.1 Fill in the blanks in each of the following:

a. The company that popularized personal computing was _____.

b. Computers process data under the control of sets of instructions called computer _____.

c. _____ is a type of computer language that uses Englishlike abbreviations for machine-language instructions.

d. _____ languages are most convenient to the programmer for writing programs quickly and easily.

e. The only language a computer can directly understand is that computer's _____.

f. The programs that translate high-level language programs into machine language are called _____.

g. _____, or labeling content, is another key part of the collaborative theme of Web 2.0.

h. With _____ development, individuals and companies contribute their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge.

i. The _____ was the predecessor to the Internet.

j. The information-carrying capacity of a communications medium like the Internet is called _____.

k. The acronym TCP/IP stands for _____.

1.2 Fill in the blanks in each of the following statements.

a. The _____ allows computer users to locate and view multimedia-based documents on almost any subject over the Internet.

b. _____ founded an organization—called the World Wide Web Consortium (W3C)—devoted to developing nonproprietary, interoperable technologies for the World Wide Web.

c. _____ are reusable software components that model items in the real world.

d. _____ is a smartphone operating system based on the Linux kernel and Java.

1.3 Fill in the blanks in each of the following statements (based on Section 1.14):

a. Objects have the property of _____—although objects may know how to communicate with one another across well-defined interfaces, they normally are not allowed to know how other objects are implemented.

b. In object-oriented programming languages, we create _____ to house the set of methods that perform tasks.

c. The process of analyzing and designing a system from an object-oriented point of view is called _____.

d. With _____, new classes of objects are derived by absorbing characteristics of existing classes, then adding unique characteristics of their own.

e. The size, shape, color and weight of an object are considered _____ of the object's class.

1.4 State whether each of the following is *true* or *false*. If the statement is *false*, explain why.

a. HTML5 (HyperText Markup Language 5) is a high-level language designed to specify the content and structure of web pages in a portable manner.

b. Keeping page styling together with the page content and structure enables you to easily change the look and feel of the pages on an entire website, or a portion of a website.

c. A web server maintains a database of hostnames and their corresponding IP addresses, and performs the translations automatically.

1.5 Fill in the blanks in each of the following statements:

a. _____ is a JavaScript library that simplifies JavaScript programming by making it easier to manipulate a web page's elements and interact with servers in a portable manner across various web browsers.

b. _____ is the standard for transferring encrypted data on the web.

c. _____ identify resources on the Internet.

d. An _____ is a numerical value that uniquely identifies the server on the Internet.

e. Browsers often _____ web pages for quick reloading.

f. The _____ operating system is used in the iPhone, iPad and iPod Touch devices.

Answers to Self-Review Exercises

1.1

a. Apple.

b. programs.

c. Assembly language.

d. High-level.

e. machine language.

f. compilers.

g. Tagging.

h. open-source.

i. ARPANET.

j. bandwidth.

k. Transmission Control Protocol/Internet Protocol.

1.2

a. World Wide Web.

b. Tim Berners-Lee.

c. Objects.

d. Android.

1.3

a. information hiding.

b. classes.

c. object-oriented analysis and design (OOAD).

d. inheritance.

e. attributes.

1.4

a. False. HTML is a markup language.

b. False. By separating page styling from page content and structure, you can change the look and feel of the pages on an entire website, or a portion of a website, simply by swapping out one style sheet for another.

c. False. A Domain Name System (DNS) server maintains a database of hostnames and their corresponding IP addresses, and performs the translations automatically.

1.5

a. jQuery.

b. Hypertext Transfer Protocol Secure (HTTPS).

c. URIs (Uniform Resource Identifiers).

d. IP (Internet Protocol) address.

e. cache.

f. iOS.

Exercises

1.6 Fill in the blanks in each of the following statements:

a. The process of instructing the computer to solve a problem is called _____.

b. What type of computer language uses Englishlike abbreviations for machine-language instructions? _____.

c. The level of computer language at which it's most convenient for you to write programs quickly and easily is _____.

d. The only language that a computer directly understands is called that computer's _____.

e. Web 2.0 embraces an _____—a design that encourages user interaction and community contributions.

f. _____ is the concept that a large, diverse group of people will create smart ideas.

1.7 Fill in the blanks in each of the following statements:

a. _____ is now used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices and for many other purposes.

b. _____ initially became widely known as the development language of the UNIX operating system.

c. The Web 2.0 company _____ is the fastest growing company ever.

d. The_____ programming language was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories.

1.8 State whether each of the following is *true* or *false*. If the statement is *false*, explain why.

a. Cascading Style Sheets™ 3 (CSS3) is used to specify the presentation, or styling, of elements on a web page (e.g., fonts, spacing, sizes, colors, positioning).

b. Ensuring a consistent look and feel on client-side browsers is one of the great challenges of developing web-based applications.

c. An `HTTP` request typically posts (or sends) data to a server.

d. Client-side scripts often can access the server's file-directory structure.

1.9 Fill in the blanks in each of the following statements:

a. _____ is the next-generation Internet Protocol that features built-in security and a new addressing scheme, significantly expanding the number of addresses available.

b. HTML documents normally contain _____, which, when clicked, load a specified web document.

c. A _____ contains information that directs a browser to the resource that the user wishes to access; _____ make such resources available to web clients.

d. The two most common HTTP request types are _____ and _____.

e. Web-based applications are multitier applications. The _____ (also called the data tier or the information tier) maintains the application's data and typically stores data in a relational database management system. The _____ implements business logic, controller logic and presentation logic to control interactions between the application's clients and its data. The _____, or client tier, is the application's user interface, which gathers input and displays output.

f. _____, the fastest growing mobile and smartphone operating system. is based on the Linux kernel and Java.

1.10 What is the relationship between JavaScript and ECMAScript?

1.11 Describe the difference between *client-side programming* and *server-side programming*.

1.12 ***(Internet in Industry and Research)*** Figures 1.1–1.4 provide examples of how computers and the Internet are being used in industry and research. Find three additional examples and describe how each is using the Internet and the web.

1.13 ***(Cloud Computing)*** Describe three benefits of the cloud computing model.

1.14 ***(Web Services)*** In Fig. 1.11 we listed several web services that can be used to create your own web applications. Using two different web services—either from the table or that you find online—describe a type of web application that you would like to create. How does it use the content provided by each of the web services?

1.15 ***(Internet Negatives)*** Besides their numerous benefits, the Internet and the web have several downsides, such as privacy issues, identity theft, SPAM and malware. Research some of the negative aspects of the Internet. List five problems and describe what could possibly be done to help solve each.

1.16 ***(Web 2.0)*** In this chapter, we discussed a few popular Web 2.0 businesses, including Facebook, Twitter, Groupon, Foursquare, Skype and YouTube. Identify another Web 2.0 business and describe why it fits the Web 2.0 business model.

1.17 ***(Watch as an Object)*** You're probably wearing on your wrist one of the world's most common types of objects—a watch. Discuss how each of the following terms and concepts applies to the notion of a watch: object, attributes, behaviors, class, inheritance (consider, for example, an alarm

clock), abstraction, modeling, messages, encapsulation, interface and information hiding.

1.18 ***(Privacy)*** Some online e-mail services save all e-mail correspondence for some period of time. Suppose a disgruntled employee were to post all of the e-mail correspondences for millions of people, including yours, on the Internet. Discuss the issues.

1.19 ***(Programmer Responsibility and Liability)*** As a programmer in industry, you may develop software that could affect people's health or even their lives. Suppose a software bug in one of your programs caused a cancer patient to receive an excessive dose during radiation therapy and that the person was severely injured or died. Discuss the issues.

1.20 ***(2010 "Flash Crash")*** An example of the consequences of our excessive dependence on computers was the so-called "flash crash" which occurred on May 6, 2010, when the U.S. stock market fell precipitously in a matter of minutes, wiping out trillions of dollars of investments, and then recovered within minutes. Research online the causes of this crash and discuss the issues it raises.

1.21 ***(Making a Difference Projects)*** The following is a list of just a few worldwide organizations that are working to make a difference. Visit these sites and our Making a Difference Resource Center at www.deitel.com/makingadifference. Prepare a top 10 list of programming projects that you think could indeed "make a difference."

- www.imaginecup.com/

  The *Microsoft Image Cup* is a global competition in which students use technology to try to solve some of the world's most difficult problems, such as environmental sustainability, ending hunger, emergency response, literacy and combating HIV/AIDS. Visit www.imaginecup.com/about for more information about the competition and to learn about the projects developed by previous winners. You can also find several project ideas submitted by worldwide charitable organizations at www.imaginecup.com/students/imagine-cup-solve-this. For ad-

ditional ideas for programming projects that can make a difference, search the web for "making a difference" and visit the following websites:

- www.un.org/millenniumgoals

  The United Nations Millennium Project seeks solutions to major world-wide issues such as environmental sustainability, gender equality, child and maternal health, universal education and more.

- www.ibm.com/smarterplanet/

  The IBM® Smarter Planet website discusses how IBM is using technology to solve issues related to business, cloud computing, education, sustainability and more.

- www.gatesfoundation.org/Pages/home.aspx

  The Bill and Melinda Gates Foundation provides grants to organizations that work to alleviate hunger, poverty and disease in developing countries. In the United States, the foundation focusses on improving public education, particularly for people with few resources.

- www.nethope.org/

  NetHope is a collaboration of humanitarian organizations worldwide working to solve technology problems such as connectivity, emergency response and more.

- www.rainforestfoundation.org/home

  The Rainforest Foundation works to preserve rainforests and to protect the rights of the indigenous people who call the rainforests home. The site includes a list of things you can do to help.

- www.undp.org/

  The United Nations Development Programme (UNDP) seeks solutions to global challenges such as crisis prevention and recovery, energy and the environment and democratic governance.

- www.unido.org

  The United Nations Industrial Development Organization (UNIDO) seeks to reduce poverty, give developing countries the opportunity to participate in global trade, and promote energy efficiency and sustainability.

- www.usaid.gov/

  USAID promotes global democracy, health, economic growth, conflict prevention, humanitarian aid and more.

- www.toyota.com/ideas-for-good/

  Toyota's Ideas for Good website describes several Toyota technologies that are making a difference—including their Advanced Parking Guidance System, Hybrid Synergy Drive®, Solar Powered Ventilation System, T.H.U.M.S. (Total Human Model for Safety) and Touch Tracer Display. You can participate in the Ideas for Good challenge by submitting a short essay or video describing how these technologies can be used for other good purposes.