

# UCCD2323 Front-End Web Development



## Chapter 1 Client-side Programming on Interface Part 1.

# Topic Outline

- **INTRODUCTION**
- **W3C HTML VALIDATION SERVICE**
- **HEADING**
- **LINKING**
- **IMAGES**
- **VOID ELEMENT**
- **SPECIAL CHARACTERS AND HORIZONTAL RULES**

## Objectives

- At the end of this lecture, students will be able to:
1. Understand important components of HTML5 documents
  2. Use HTML5 to create web pages
  3. Add images to web pages
  4. Create and use hyperlinks to help users navigate web pages

# Introduction: What is HTML

HTML specifies the structure and content of documents that are displayed in web browsers.

HTML stands for Hypertext Markup Language.

A markup language is a collection of markup tags.

Other markup languages include SGML, XML, XHTML.

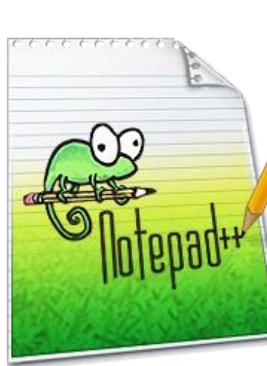
Version of HTML from HTML 1 to HTML 5

*dynamic*

# Introduction: Tools

To create the HTML document, we use text editor by typing the HTML markup text and saving it with the .html or .htm file name extension.

- Computers called **web servers** store HTML5 documents.  
*(User that)*
- **Clients** (such as web browsers running on your local computer or smartphone) request specific resources such as HTML5 documents from web servers.



# Introduction: HTML Validation Service

a webpage

Validation services (e.g., [validator.w3.org/#validate-by-upload](http://validator.w3.org/#validate-by-upload)) ensure that an HTML5 document is syntactically correct.

information page on the W3C QA Website.' Handwritten red text 'check here valid' is written vertically along the left side of the screenshot."/>

W3C® Markup Validation Service  
Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI   Validate by File Upload   Validate by Direct Input

Validate by File Upload

Upload a document for validation:

File:  Choose File No file chosen

► More Options

Check

Note: file upload may not work with Internet Explorer on some versions of Windows XP Service Pack 2, see our [information page](#) on the W3C QA Website.

check here valid

# Introduction: Basic Structure

- ~~Start Tags and End Tags~~
  - HTML5 documents *delimit* most elements with a start tag and end tag.
  - A **start tag** consists of the element name in angle brackets  
For example, `<html>`
  - An **end tag** consists of the element name preceded by a forward slash (/) in angle brackets  
For example, `</html>`
- There are several so-called “**void elements**” that do not have end tags.
- Many **start tags have attributes** that provide additional information about an element, which browsers use to determine how to process the element.
- Each **attribute** has a **name** and a **value** separated by an equals sign (=).

# Introduction: Basic Structure

- An **attribute** should always have:
  - A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
  - The attribute name, followed by an equals sign.
  - Opening and closing quote marks wrapped around the attribute value.



## Software Engineering Observation 2.1

Although not required in HTML5, enclosing attribute values in either single or double quotes is recommended.

- **Document Type Declaration**

- ▶ The **document type declaration** (DOCTYPE) is required in HTML5 documents so that browsers render the page in standards mode.
- ▶ Some browsers operate in quirks mode to maintain backward compatibility with web pages that are not up-to-date with the latest standards.

# Introduction: Basic Structure

- **Comments**
  - ▶ Insert comments in your HTML5 markup to improve readability and describe the content of a document.
  - ▶ The browser ignores comments when your document is rendered.
  - ▶ **Comments start with <!-- and end with -->**

## **HTML, HEAD and BODY Elements**

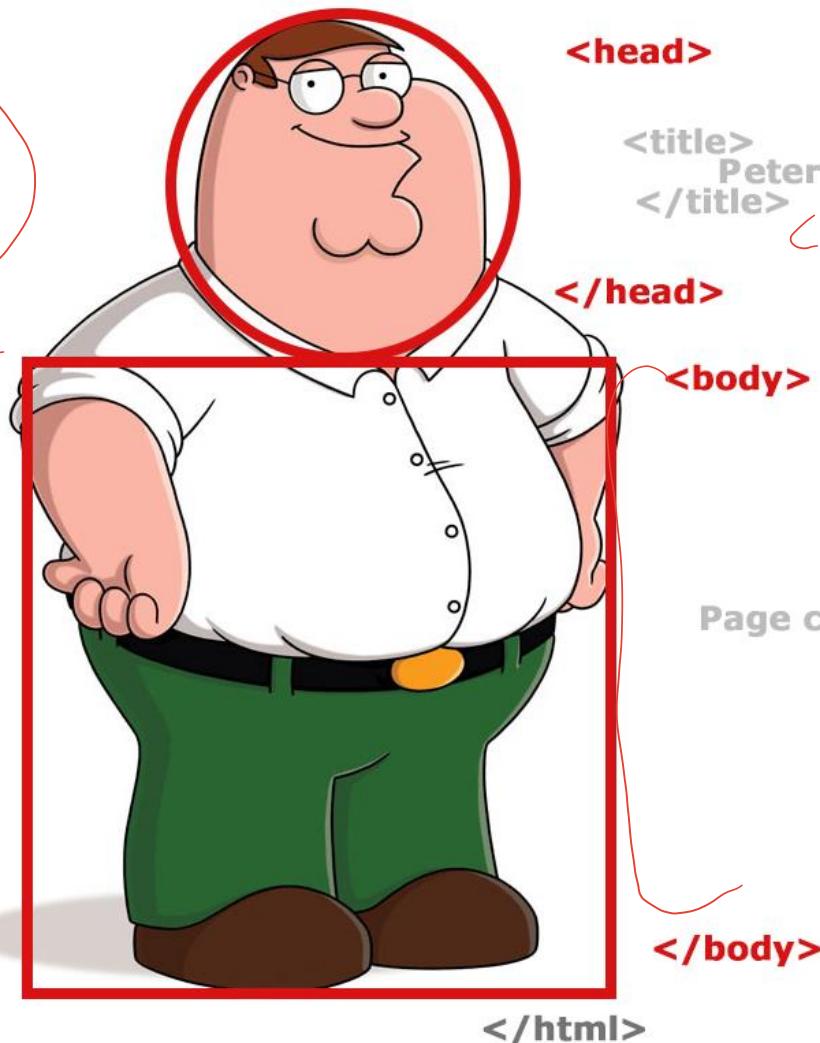
- ▶ HTML5 markup contains text (and images, graphics, animations, audios and videos) that represents the content of a document and elements that specify a document's *structure* and *meaning*.

The `html` element *encloses* the head section (represented by the `head` element) and the body section (represented by the `body` element).  
The **head section** contains information about the HTML5 document, such as the character set (UTF-8, the most popular character-encoding scheme for the web) that the page uses—which helps the browser determine how to render the content—and the **title**.  
The **head section** also can contain special document-formatting instructions called **CSS3 style sheets** and client-side programs called scripts for creating dynamic web pages.  
The **body section** contains the page's content, which the browser displays when the user visits the web page.

# Introduction: Basic Structure

```
<html>  
<head> ... </head>  
<body> ... </body>  
</html>
```

Format got mark



<html>

<head>

<title> Peter Griffin's HTML page  
</title>

*CSS style*

</head>

<body>

Page content goes here..

</body>

</html>

I belong in the body!

>/body> LOGIC OF THE PAGE

P H H  
B B B  
P

# Introduction: Basic Structure

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.1: main.html -->
4 <!-- First HTML5 example. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Welcome</title>
9   </head>
10
11   <body>
12     <p>Welcome to HTML5!</p>
13   </body>
14 </html>
```



## Good Programming Practice 2.1

Although HTML5 element and attribute names are case insensitive (you can use uppercase and lowercase letters), it's a good practice to use only lowercase letters.



## Good Programming Practice 2.2

Indenting nested elements emphasizes a document's structure and promotes readability. We use three spaces for each level of indentation.

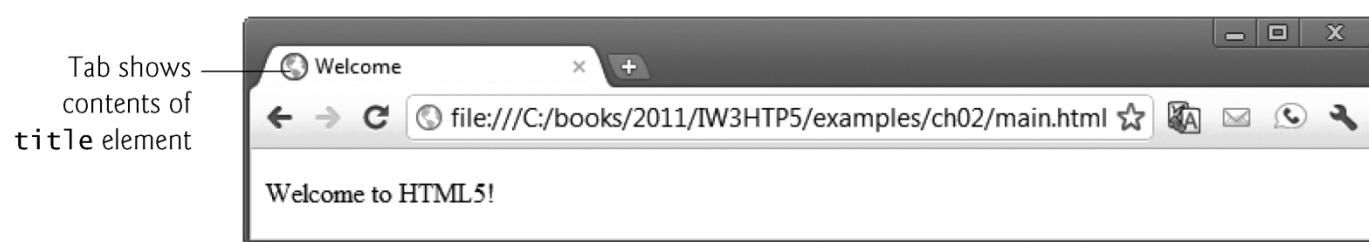


Fig. 2.1 | First HTML5 example.

# Title Element

- The title element is called a **nested element**, because it's enclosed in the head element's start and end tags.
  - ▶ The head element is also a nested element, because it's enclosed in the html element's start and end tags.
  - ▶ The title element describes the web page.

Titles usually appear in the title bar at the top of the browser window, in the browser tab on which the page is displayed, and also as the text identifying a page when users add the page to their list of Favorites or Bookmarks, enabling them to return to their favorite sites. Search engines use the title for indexing purposes and when displaying results

# Paragraph Element

- **Paragraph Element (*< p > ... </ p >*)**

All text placed between the `<p>` and `</p>` tags forms one paragraph.

## Preformatted Text Element (*< pre > ... </ pre >*)

**<pre> element** represents preformatted text which is to be presented exactly as written in the HTML file

*closely important*

**<pre> Preformatted  
Output </pre>**

# Heading Element

- HTML5 provides six heading elements (h1 through h6) for specifying the *relative importance* of information

Heading element **h1** is considered the most significant heading and is rendered in the **largest font**.

Each successive heading element (i.e., **h2, h3, h4, h5, h6**) is rendered in a **progressively smaller font**.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.2: heading.html -->
4 <!-- Heading elements h1 through h6. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Headings</title>
9   </head>
10
11  <body>
12    <h1>Level 1 Heading</h1>
13    <h2>Level 2 heading</h2>
14    <h3>Level 3 heading</h3>
15    <h4>Level 4 heading</h4>
16    <h5>Level 5 heading</h5>
17    <h6>Level 6 heading</h6>
18  </body>
19 </html>
```

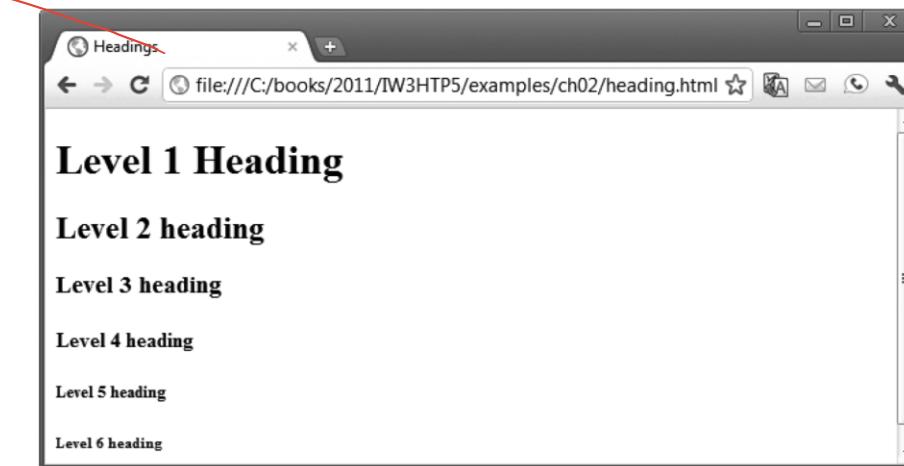


Fig. 2.2 | Heading elements h1 through h6. (Part 2 of 2.)

Fig. 2.2 | Heading elements h1 through h6. (Part 1 of 2.)

## Portability Tip 2.1

The text size used to display each heading element can vary between browsers. In Chapter 4, we use CSS to control the text size and other text properties.



## Look-and-Feel Observation 2.1

Placing a heading at the top of each page helps viewers understand the purpose of the page. Headers also help create an outline for a document and are indexed by search engines.

# Linking

- A hyperlink references or links to other resources, such as HTML5 documents and images.
- Web browsers typically underline text hyperlinks and color them blue by default.

```
1 <!DOCTYPE html>
2
3  <!-- Fig. 2.3: links.html --&gt;
4  &lt;!-- Linking to other web pages. --&gt;
5 &lt;html&gt;
6   &lt;head&gt;
7     &lt;meta charset = "utf-8"&gt;
8     &lt;title&gt;Links&lt;/title&gt;
9   &lt;/head&gt;
10
11  &lt;body&gt;
12    &lt;h1&gt;Here are my favorite sites:&lt;/h1&gt;
13    &lt;p&gt;&lt;strong&gt;Click a name to visit that site.&lt;/strong&gt;&lt;/p&gt;
14
15    &lt;!-- create four text hyperlinks --&gt;
16    &lt;p&gt;&lt;a href = "http://www.facebook.com"&gt;Facebook&lt;/a&gt;&lt;/p&gt;
17    &lt;p&gt;&lt;a href = "http://www.twitter.com"&gt;Twitter&lt;/a&gt;&lt;/p&gt;
18    &lt;p&gt;&lt;a href = "http://www.foursquare.com"&gt;Foursquare&lt;/a&gt;&lt;/p&gt;
19    &lt;p&gt;&lt;a href = "http://www.google.com"&gt;Google&lt;/a&gt;&lt;/p&gt;
20
21  &lt;/body&gt;</pre>

hyper ref = "link"


```

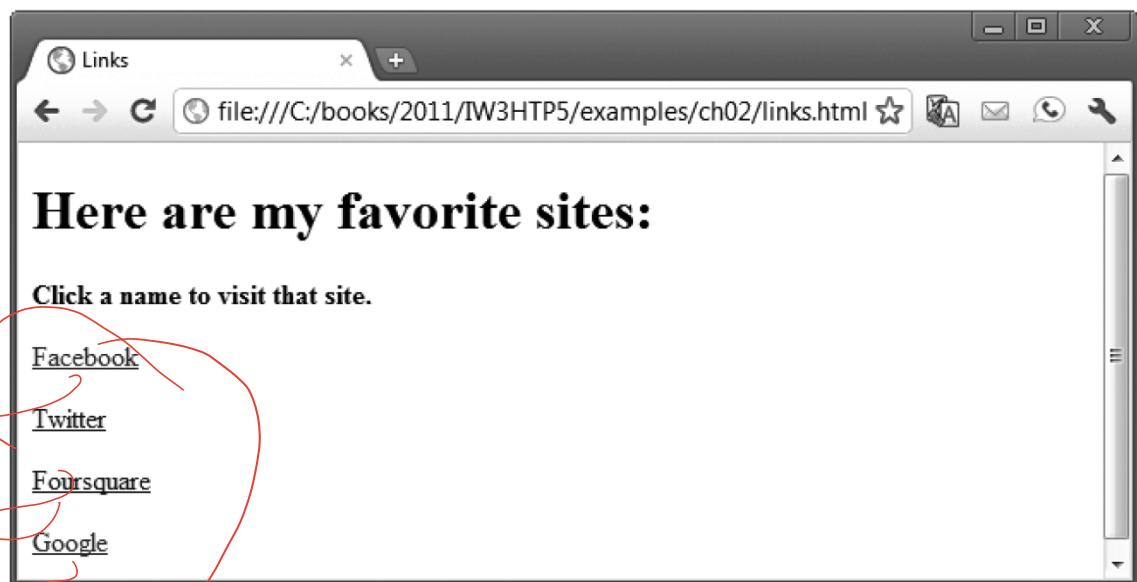


Fig. 2.3 | Linking to other web pages. (Part 1 of 2.)

# Linking

- The **strong element** indicates that the content has high importance. Browsers typically render such text in a bold font.
- ▶ Links are created using the **a (anchor) element**.
- ▶ Attribute **href (hypertext reference)** specifies a resource's location, such as
  - a web page or location within a web page
  - a file
  - an e-mail address
- ▶ When a URL does not indicate a specific document on the website, the web server returns a default web page. This page is often called **index.html**, but most web servers can be configured to use any file as the default web page for the site.
- ▶ If the web server cannot locate a requested document, it returns an **error indication** to the web browser (known as a **404 error**), and the browser displays a web page containing an error message.



```
<a href ="http://www.facebook.com">Facebook</a>
```

# Linking

- **Hyperlinking to an E-Mail Address**

Anchors can link to an e-mail address using a **mailto: URL**

When a user clicks this type of anchored link, most browsers launch the default e-mail program (e.g., Mozilla Thunderbird, Microsoft Outlook or Apple Mail) to enable the user to write an e-mail message to the linked address.

<a href = "mailto:salsalbilal@utar.edu.my">UTAR EMAIL</a>

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.4: contact.html -->
4 <!-- Linking to an e-mail address. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Contact Page</title>
9   </head>
10
11  <body>
12    <p>
13      To write to <a href = "mailto:deitel@deitel.com">
14        Deitel & Associates, Inc.</a>, click the link and your default
15        email client will open an email message and address it to us.
16    </p>
17  </body>
18 </html>
```

Fig. 2.4 | Linking to an e-mail address. (Part 1 of 3.)

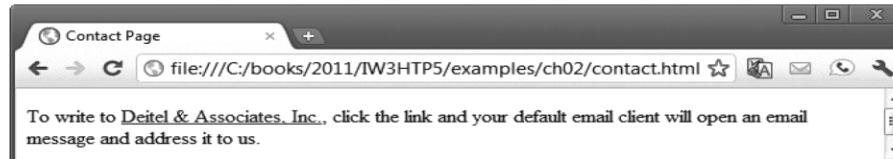


Fig. 2.4 | Linking to an e-mail address. (Part 2 of 3.)

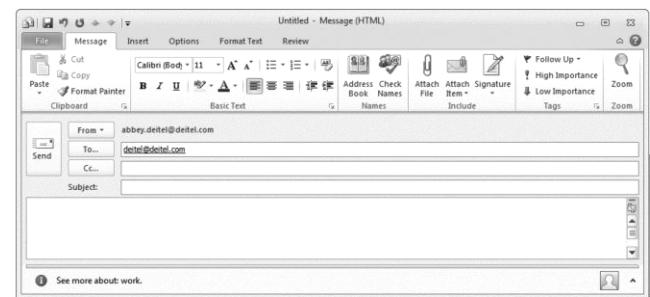


Fig. 2.4 | Linking to an e-mail address. (Part 3 of 3.)

# Image

standard 68 ✓

- The most popular image formats used by web developers today are **PNG (Portable Network Graphics)** and **JPEG (Joint Photographic Experts Group)**.
- Users can create images using specialized software, such as Adobe Photoshop Express ([www.photoshop.com](http://www.photoshop.com)), G.I.M.P. ([www.gimp.org](http://www.gimp.org)), Inkscape ([www.inkscape.org](http://www.inkscape.org)) and many more.  
*trace*
- Images may also be acquired from various websites, many of which offer **royalty-free images**.

Image-sharing site	URL
Flickr®	<a href="http://www.flickr.com">www.flickr.com</a>
Photobucket	<a href="http://photobucket.com">photobucket.com</a>
Fotki™	<a href="http://www.fotki.com">www.fotki.com</a>
deviantART	<a href="http://www.deviantart.com">www.deviantart.com</a>
Picasa™	<a href="http://picasa.google.com">picasa.google.com</a>
TinyPic®	<a href="http://tinypic.com">tinypic.com</a>
ImageShack	<a href="http://www.imageshack.us">www.imageshack.us</a>
FreeDigitalPhotos.net	<a href="http://www.freedigitalphotos.net">www.freedigitalphotos.net</a>
Open Stock Photography	<a href="http://www.openstockphotography.org">www.openstockphotography.org</a>
Open Clip Art Library	<a href="http://www.openclipart.org">www.openclipart.org</a>

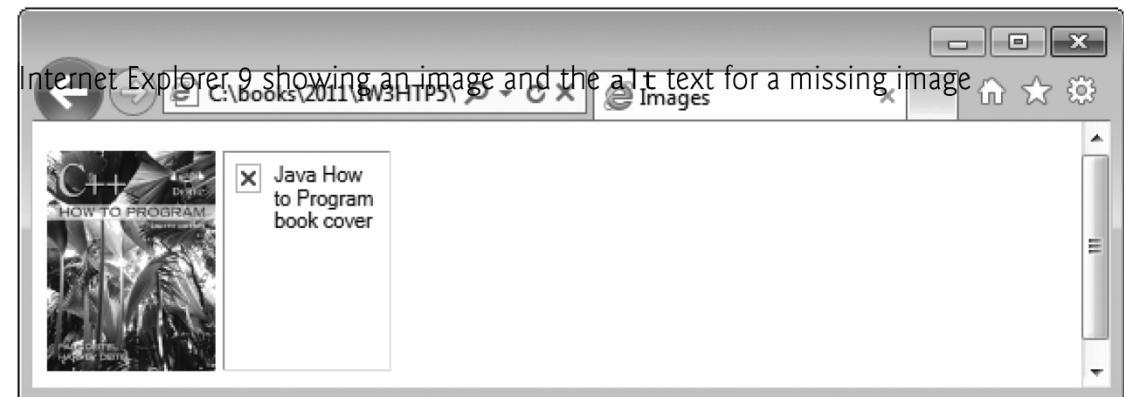
Fig. 2.5 | Popular image-sharing sites.

need  
reduce  
blur

Pixel jber

# Image

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.6: picture.html -->
4 <!-- Including images in HTML5 files. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Images</title>
9   </head>
10
11  <body>    //     ( /
12    <p>
13      <img src = "cpphttp.png" width = "92" height = "120"
14        alt = "C++ How to Program book cover">
15      <img src = "jhttp.png" width = "92" height = "120"
16        alt = "Java How to Program book cover">
17    </p>
18  </body>
19 </html>
```



**Fig. 2.6 |** Including images in HTML5 files. (Part 1 of 2.)

# Image

- The img element's **src attribute** specifies an image's location
- Every img element must have an **alt attribute**, which contains text that is displayed if the client cannot render the image

The alt attribute makes web pages more accessible to users with disabilities, especially vision impairments

Width and height are optional attributes

If omitted, the browser uses the image's actual width and height

Images are **measured in pixels**

Fotokon better

<img src = "cpphttp.png" width = "92" height = "120"  
alt = "book cover" />

36 59 476

# Image

- ***alt Attribute***
  - ▶ A browser may not be able to render an image.
  - ▶ Every img element in an HTML5 document **must have** an alt attribute.
  - ▶ If a browser **cannot render an image**, the browser **displays the alt attribute's value**.
  - ▶ The alt attribute is also important for accessibility—speech synthesizer software can speak the alt attribute's value so that a visually impaired user can understand what the browser is displaying. *For this reason, the alt attribute should describe the image's contents.*

the sources 分享  
FY project

Tag (whole also other)

# Image

- ***Using Images as Hyperlinks***

By using images as hyperlinks, you can create graphical web pages that link to other resources. In the example, we create five different image hyperlinks. Clicking an image in this example takes the user to a corresponding web page—one of the other examples in this chapter.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 2.7: nav.html -->
4  <!-- Images as link anchors. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Navigation Bar</title>
9      </head>
10
11     <body>
12         <p>
13             <a href = "links.html">
14                 <img src = "buttons/links.jpg" width = "65"
15                     height = "50" alt = "Links">
16             </a>
17
18             <a href = "list.html">
19                 <img src = "buttons/list.jpg" width = "65"
20                     height = "50" alt = "List of Features">
21             </a>
22
```

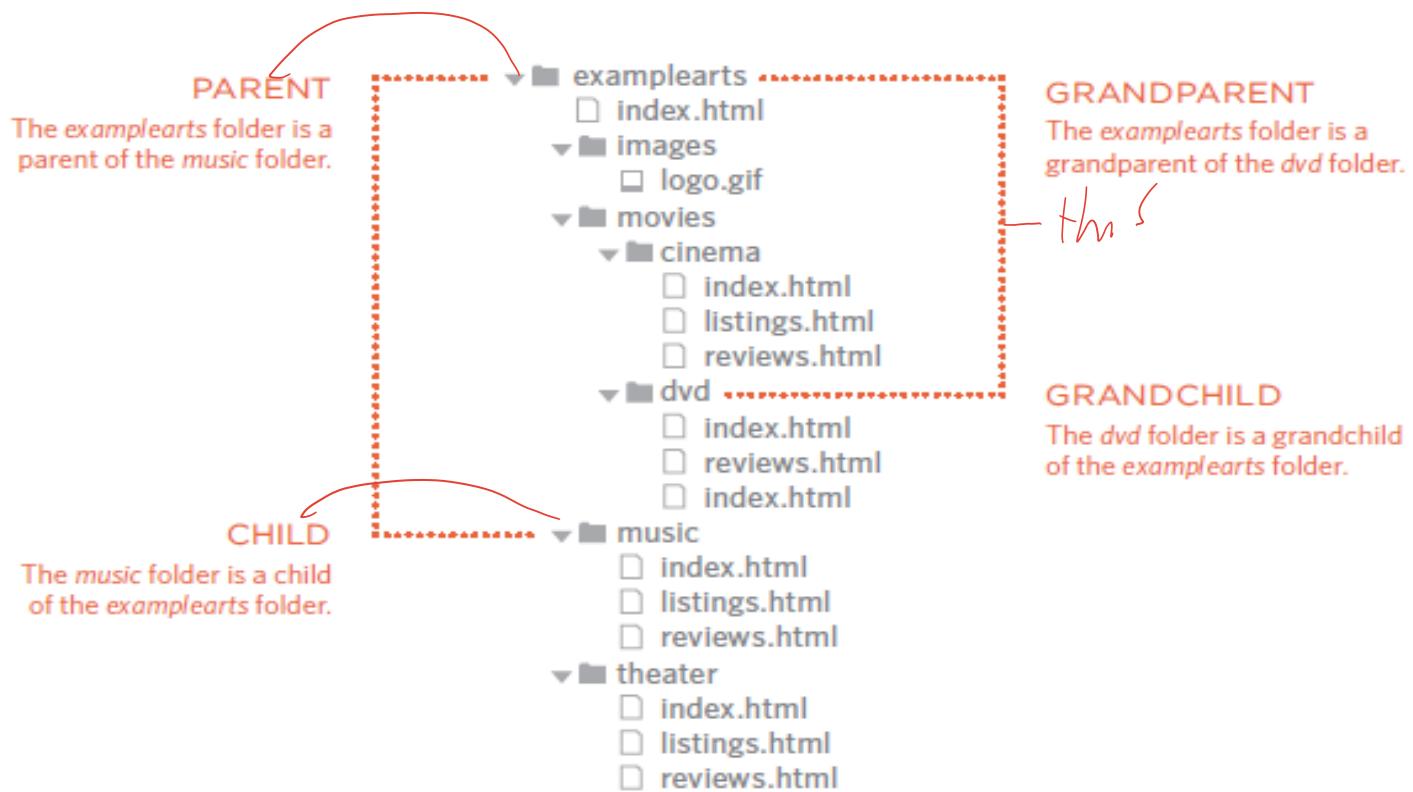
```
23
24             <a href = "contact.html">
25                 <img src = "buttons/contact.jpg" width = "65"
26                     height = "50" alt = "Contact Me">
27             </a>
28
29             <a href = "table1.html">
30                 <img src = "buttons/table.jpg" width = "65"
31                     height = "50" alt = "Tables Page">
32             </a>
33
34             <a href = "form.html">
35                 <img src = "buttons/form.jpg" width = "65"
36                     height = "50" alt = "Feedback Form">
37             </a>
38         </p>
39     </body>
39 </html>
```



# Image

- **Directory Structure**

- On larger websites it's a good idea to **organize** your code by placing the pages for each different section of the site into a new folder. Folders on a website are sometimes referred to as directories.



# Image

- **Relative URLs**
  - Relative URLs can be used when linking to pages within your own website. They provide a shorthand way of telling the browser where to find your files.

RELATIVE LINK TYPE	EXAMPLE (from diagram on previous page)
<b>SAME FOLDER</b> To link to a file in the same folder, just use the file name. (Nothing else is needed.)	To link to music reviews from the music homepage: <code>&lt;a href="reviews.html"&gt;Reviews&lt;/a&gt;</code>
<b>CHILD FOLDER</b> For a child folder, use the name of the child folder, followed by a forward slash, then the file name.	To link to music listings from the homepage: <code>&lt;a href="music/listings.html"&gt;Listings&lt;/a&gt;</code>
<b>GRANDCHILD FOLDER</b> Use the name of the child folder, followed by a forward slash, then the name of the grandchild folder, followed by another forward slash, then the file name.	To link to DVD reviews from the homepage: <code>&lt;a href="movies/dvd/reviews.html"&gt;Reviews&lt;/a&gt;</code>
<b>PARENT FOLDER</b> Use <code>../</code> to indicate the folder above the current one, then follow it with the file name.	To link to the homepage from the music reviews: <code>&lt;a href="../index.html"&gt;Home&lt;/a&gt;</code>
<b>GRANDPARENT FOLDER</b> Repeat the <code>../</code> to indicate that you want to go up two folders (rather than one), then follow it with the file name.	To link to the homepage from the DVD reviews: <code>&lt;a href="....index.html"&gt;Home&lt;/a&gt;</code>

soffer le c tura

# Image

- **Relative URLs – html and image**

- **Common mistake**

```
<a href = " D:\\ links.html">Link</a>
```



## Software Engineering Observation 2.1

Although not required in HTML5, enclosing attribute values in either single or double quotes is recommended.

```
<img src = "D:\\buttons\\links.jpg"  
width = "65" height = "50" alt = "Links" />
```



## Look-and-Feel Observation 2.2

Entering new dimensions for an image that change its width-to-height ratio distorts the appearance of the image. To avoid distortion, if your image is 200 pixels wide and 100 pixels high, for example, any new dimensions should maintain the 2:1 width-to-height ratio.

else keep

# Void Element

- Some HTML5 elements (called **void elements**) contain only attributes and do not mark up text (i.e., text is not placed between a start and an end tag).
  - ▶ You can terminate void elements (such as the img element) by using the forward slash character (/) inside the closing right angle bracket (>) of the start tag.

- ▶ Example:

```
<img src = "jhtp.png" width = "92" height =  
"120" alt = "Java How to Program book cover"/>
```

# Void Element

**meta**

```
<meta charset="utf-8">
```

standard webpage protocol

mark



**img**

```
<img src = "jhttp.png" alt = "Java  
How to Program book cover"/>
```

**input**

```
<input type="submit" value="Ok" />
```

**br**

```
<br />.
```

**hr**

```
<hr />.
```

# Special Characters and Horizontal Rules

- HTML5 provides **character entity references** (in the form &code;) for representing special characters that cannot be rendered otherwise

The code can be:

*part*   *in*,*de*   *webpage*

Word abbreviations

Numbers

Decimal

Hexadecimal

- For an extensive list of character entities, see

<https://html.spec.whatwg.org/multipage/named-characters.html#named-character-references>

<https://dev.w3.org/html5/html-author/charref>

# Special Characters and Horizontal Rules



Symbol	Description	Character entity reference
HTML5 character entities		
&	ampersand	&amp;
,	apostrophe	&apos;
>	greater-than	&gt;
<	less-than	&lt;
"	quote	&quot;
Other common character entities		
non-breaking space		&nbsp;
©	copyright	&copy;
—	em dash	&mdash;
–	en dash	&ndash;
¼	fraction 1/4	&frac14;
½	fraction 1/2	&frac12;



Symbol	Description	Character entity reference
¾	fraction 3/4	&frac34;
...	horizontal ellipsis	&hellip;
®	registered trademark	&reg;
§	section	&sect;
™	trademark	&trade;

Fig. 2.8 | Some common HTML character entity references.

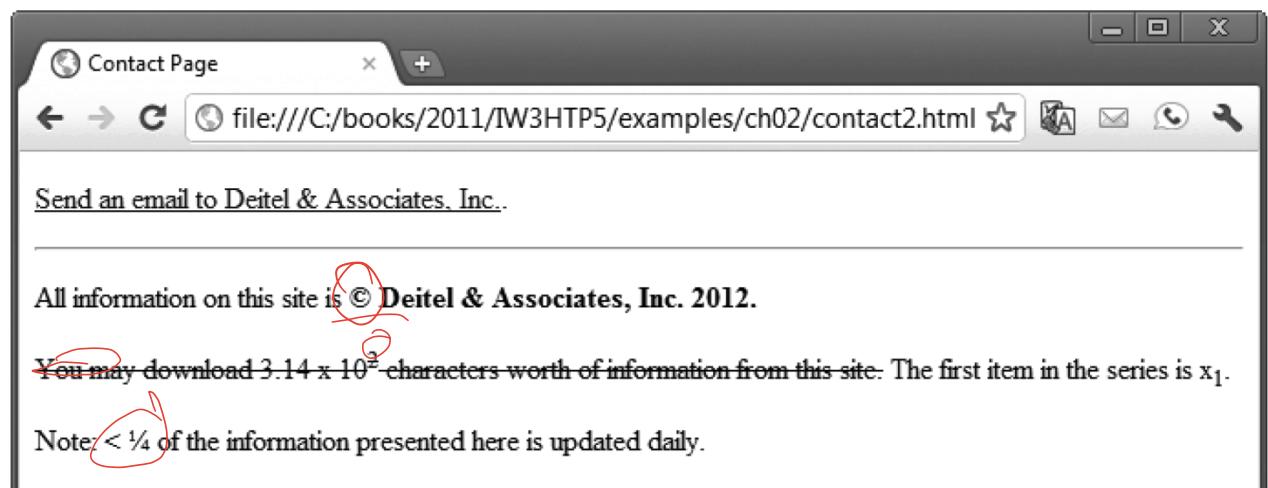
Fig. 2.8 | Some common HTML character entity references.

# Special Characters and Horizontal Rules

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.9: contact2.html -->
4 <!-- Inserting special characters. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Contact Page</title>
9   </head>
10
11  <body>
12    <p>
13      <a href = "mailto:deitel@deitel.com">Send an email to
14        Deitel & Associates, Inc.</a>.
15    </p>
16
17    <hr> <!-- inserts a horizontal rule -->
18
19    <!-- special characters are entered -->
20    <!-- using the form &code; -->
21    <p>All information on this site is <strong>&copy;
22      Deitel & Associates, Inc. 2012.</strong> </p>
23
24    <!-- to strike through text use <del> element -->
```

*Standard alignment  
in every  
Code*

```
25
26   <!-- to subscript text use <sub> element -->
27   <!-- to superscript text use <sup> element -->
28   <!-- these elements are nested inside other elements -->
29   <p><del>You may download  $3.14 \times 10^2$  characters worth of information from this site.</del>
30   The first item in the series is  $x_1$ .
31   <p>Note: <math>\frac{1}{4}</math> of the information
32     presented here is updated daily.</p>
33
34 </body>
</html>
```

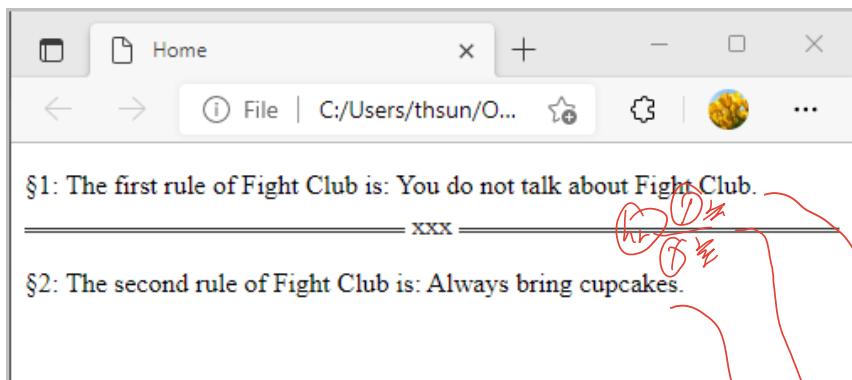


# Special Characters and Horizontal Rules

- A **horizontal rule**, indicated by the `<hr>` tag renders a horizontal line with **extra space above and below it** in most browsers.
  - ▶ The horizontal rule element should be considered a **legacy element** and you should **avoid using it**.
  - ▶ **CSS can be used** to add horizontal rules and other formatting to documents.
  - ▶ Special characters can also be represented as **numeric character references**—decimal or hexadecimal (hex) values representing special characters.  
For example, the & character is represented in decimal and hexadecimal notation as `&#38;` and `&#x26;`, respectively. *S&H*
  - ▶ Hexadecimal numbers are discussed in Appendix E, Number Systems, which is available online at [www.deitel.com/books/iw3htp5/](http://www.deitel.com/books/iw3htp5/).

# <hr>: The Thematic Break (Horizontal Rule) element

- The <hr> HTML element represents a thematic break between paragraph-level elements: for example, a change of scene in a story, or a shift of topic within a section.



```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Home</title>
  <meta charset="utf-8">
  <style>
    hr {
      border: none;
      border-top: 3px double #333;
      color: #333;
      overflow: visible;
      text-align: center;
      height: 5px;
    }

    hr:after {
      background: #fff;
      content: 'xxx';
      padding: 0 4px;
      position: relative;
      top: -13px;
    }
  </style>
</head>

<body>
  <p>§1: The first rule of Fight Club is:  
    You do not talk about Fight Club.</p>
  <hr>
  <p>§2: The second rule of Fight Club is:  
    Always bring cupcakes.</p>
</body>

</html>
```

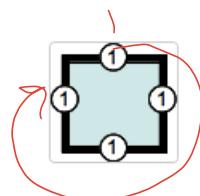
# Edges of a box

- [https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties)

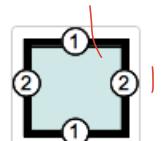
Note: Will be covered in the CSS chapter.

Shorthands handling properties related to edges of a box, like border-style, margin or padding, always use a consistent 1-to-4-value syntax representing those edges:

- 1-value syntax: `border-width: 1em` — The single value represents all edges:



- 2-value syntax: `border-width: 1em 2em` — The first value represents the vertical, that is top and bottom, edges, the second the horizontal ones, that is the left and right ones:

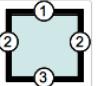


# Edges of a box

- [https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties)

Note: Will be covered in the CSS chapter.

Shorthands handling properties related to edges of a box, like border-style, margin or padding, always use a consistent 1-to-4-value syntax representing those edges:

- **3-value syntax:** `border-width: 1em 2em 3em` — The first value represents the top edge, the second, the horizontal, that is left and right, ones, and the third value the bottom edge:  

- **4-value syntax:** `border-width: 1em 2em 3em 4em` — The four values represent the top, right, bottom and left edges respectively, always in that order, that is clock-wise starting at the top:  


The initial letter of Top-Right-Bottom-Left matches the order of the consonant of the word trouble: TRBL. You can also remember it as the order that the hands would rotate on a clock: 1em starts in the 12 o'clock position, then 2em in the 3 o'clock position, then 3em in the 6 o'clock position, and 4em in the 9 o'clock position.

# HTML Tag

Tag	Description
<u>&lt;b&gt;</u>	Defines bold text
<u>&lt;em&gt;</u>	Defines emphasized text
<u>&lt;i&gt;</u>	Defines a part of text in an alternate voice or mood
<u>&lt;small&gt;</u>	Defines smaller text
<u>&lt;strong&gt;</u>	Defines important text
<u>&lt;sub&gt;</u>	Defines subscripted text
<u>&lt;sup&gt;</u>	Defines superscripted text
<u>&lt;ins&gt;</u>	Defines inserted text
<u>&lt;del&gt;</u>	Defines deleted text

# UCCD2323 Front-End Web Development



## Chapter 1 Client-side Programming on Interface Part 2.

# Topic Outline

- **LISTS**
- **TABLES**
- **INTERNAL LINKING**
- **META ELEMENT**

## Objectives

- At the end of this lecture, students will be able to:
1. **Mark up lists of information**
  2. **Create tables with rows and columns of data**
  3. **Create and use forms to get user input**

# Lists

- **Unordered list element `ul`**
  - Creates a list in which each item in the list begins with a **bullet symbol** (typically a disc)
  - Each entry is an **li (list item)** element.
  - Most web browsers render these elements with a line break and a bullet symbol at the beginning of the line.
- **Ordered-list element `ol`**
  - Creates a list in which each item **begins with a number**.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.10: links2.html -->
4 <!-- Unordered list containing hyperlinks. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Links</title>
9   </head>
10
11   <body>
12     <h1>Here are my favorite sites</h1>
13     <p><strong>Click on a name to go to that page</strong></p>
14
15     <!-- create an unordered list -->
16     <ul>
17       <!-- the list contains four list items -->
18       <li><a href = "http://www.youtube.com">YouTube</a></li>
19       <li><a href = "http://www.wikipedia.org">Wikipedia</a></li>
20       <li><a href = "http://www.amazon.com">Amazon</a></li>
21       <li><a href = "http://www.linkedin.com">LinkedIn</a></li>
22     </ul>
23   </body>
24 </html>
```

Fig. 2.10 | Unordered list containing hyperlinks. (Part 1 of 2.)

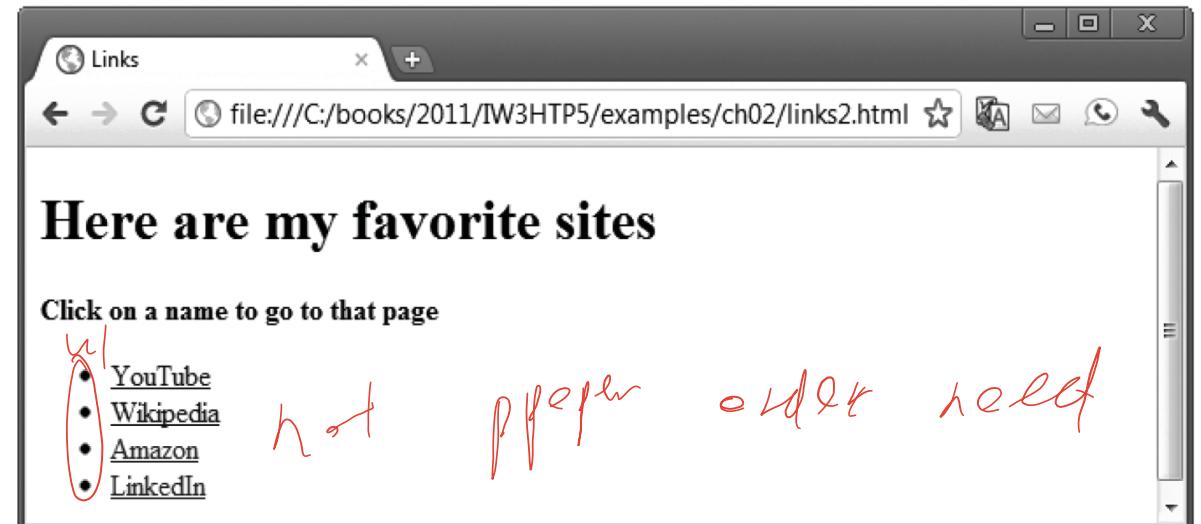


Fig. 2.10 | Unordered list containing hyperlinks. (Part 2 of 2.)

# Lists

- **Nested Lists**

- ▶ Lists may be *nested* to represent *hierarchical* relationships, as in a *multi-level outline*.
- ▶ Figure 2.11 demonstrates nested lists and ordered lists.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.11: list.html -->
4 <!-- Nested lists and ordered lists. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Lists</title>
9   </head>
10
11   <body>
12     <h1>The Best Features of the Internet</h1>
13
```

**Fig. 2.11 |** Nested lists and ordered lists. (Part 1 of 4.)

```
14           <!-- create an unordered list -->
15           <ul>
16             <li>You can meet new people from countries around
17               the world.</li>
18             <li>
19               You have access to new media as it becomes public:
20
21               <!-- this starts a nested unordered list, which uses a -->
22               <!-- different bullet. The list ends when you -->
23               <!-- close the <ul> tag. -->
24               <ul>
25                 <li>New games</li>
26                 <li>New applications
27
28               <!-- nested ordered list -->
29               <ol>
30                 <li>For business</li>
31                 <li>For pleasure</li>
32               </ol>
33             </li> <!-- ends line 27 new applications li-->
34
35             <li>Around the clock news</li>
36             <li>Search engines</li>
37             <li>Shopping</li>
```

**Fig. 2.11 |** Nested lists and ordered lists. (Part 2 of 4.)

# Lists

```
38 <li>Programming  
39  
40     <!-- another nested ordered list -->  
41     <ol>  
42         <li>XML</li>  
43         <li>Java</li>  
44         <li>HTML5</li>  
45         <li>JavaScript</li>  
46         <li>New languages</li>  
47     </ol>  
48     </li> <!-- ends programming li of line 38 -->  
49 </ul> <!-- ends the nested list of line 24 -->  
50 </li>  
51  
52 <li>Links</li>  
53 <li>Keeping in touch with old friends</li>  
54 <li>It's the technology of the future!</li>  
55 </ul> <!-- ends the unordered list of line 15 -->  
56 </body>  
57 </html>
```

Fig. 2.11 | Nested lists and ordered lists. (Part 3 of 4.)

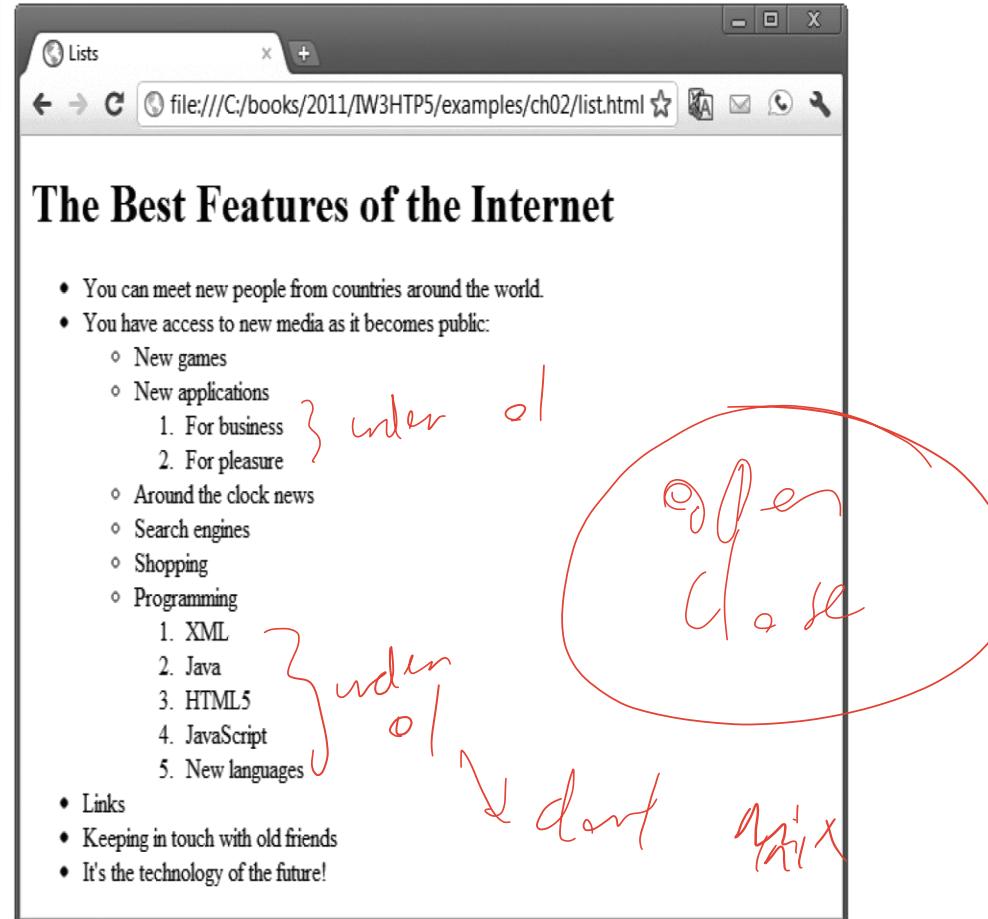


Fig. 2.11 | Nested lists and ordered lists. (Part 4 of 4.)

# Description Lists

- A list of terms/names, with a description of each term/name.

► **<dl>** tag defines the description list.

<dt> defines terms/names

<dd> describes each term/name

Right

d | ~~f~~ | o |

## Example

<dl>

<dt>Milk</dt>

~~Note~~ <dd>A white color cold drink</dd>

<dt>Coca ColaM</dt>

<dd>A dark brown color soft/fizzy drink</dd>

</dl>

M  
A ~ (pt. under)  
E ~  
A ~

# Lists and Attributes

- <ol> Browser Support



The <ol> tag is supported in all major browsers.

All supported ✓

Attribute	Value	Description
<u>compact</u>	compact	Not supported in HTML5. Deprecated in HTML 4.01. Specifies that the list should render smaller than normal
<u>reversed</u> <span style="border: 1px solid green; padding: 2px;">New</span>	reversed	Specifies that the list order should be descending (9,8,7...)
<u>start</u>	number	Specifies the start value of an ordered list
<u>type</u>	1 A a I i	Specifies the kind of marker to use in the list

# Lists and Attributes

- <ul> Browser Support



The <ol> tag is supported in all major browsers.

*ER later update best*

Attribute	Value	Description
<u>compact</u>	compact	Not supported in HTML5. Deprecated in HTML 4.01. Specifies that the list should render smaller than normal
<u>type</u>	disc square circle	Not supported in HTML5. Deprecated in HTML 4.01. Specifies the kind of marker to use in the list

# Tables

- **Tables** are frequently used to organize data into *rows* and *columns*.
- The table element defines an HTML5 table
- The **summary** attribute summarizes the table's contents and is used by speech devices to make the table more accessible to users with visual impairments.
- The **caption** element specifies a table's title.  
\_\_\_\_\_
- It's good practice to include a **general description** of a **table's information** in the table element's **summary attribute**—one of the many HTML5 features that make web pages more accessible to users with disabilities.
- Speech devices use this attribute to make the table more accessible to users with visual impairments.

# Tables

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 2.12: table1.html -->
4  <!-- Creating a basic table. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>A simple HTML5 table</title>
9      </head>
10
11     <body>
12         <!-- the <table> tag opens a table -->
13         <table border = "1"> now ≠ for all table
14
15             <!-- the <caption> tag summarizes the table's -->
16             <!-- contents (this helps visually impaired people) -->
17             <caption><b>Table of Fruits (1st column)</b> and
18                 Their Prices (2nd column)</strong></caption>
19
```

Fig. 2.12 | Creating a basic table. (Part 1 of 4.)

```
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
```

<!-- the <thead> section appears first in the table -->  
<!-- it formats the table header area -->

<thead>

<tr> <!-- <tr> inserts a table row -->  
 <th>Fruit</th> <!-- insert a heading cell -->  
 <th>Price</th>

</tr>  
</thead> *→ for folder one*

<!-- the <tfoot> section appears last in the table -->  
<!-- it formats the table footer -->

<tfoot>

<tr>  
 <th>Total</th>  
 <th>\$3.75</th>

</tr>  
</tfoot>

<!-- all table content is enclosed -->  
<!-- within the <tbody> -->

<tbody> *Content*

<tr>  
 <td>Apple</td> <!-- insert a data cell -->  
 <td>\$0.25</td>

</tr>

Fig. 2.12 | Creating a basic table. (Part 2 of 4.)

# Tables

```
45  
46      <td>Orange</td>  
47      <td>$0.50</td>  
48    </tr>  
49    <tr>  
50      <td>Banana</td>  
51      <td>$1.00</td>  
52    </tr>  
53    <tr>  
54      <td>Pineapple</td>  
55      <td>$2.00</td>  
56    </tr>  
57  </tbody>  
58 </table>  
59 </body>  
60 </html>
```

Fig. 2.12 | Creating a basic table. (Part 3 of 4.)

The screenshot shows a web browser window with the title "A simple HTML5 table". The page contains a table with the following data:

Fruit	Price
Apple	\$0.25
Orange	\$0.50
Banana	\$1.00
Pineapple	\$2.00
Total	\$3.75

Annotations with red arrows point to specific parts of the table:

- Table caption: Points to the title "Table of Fruits (1st column) and Their Prices (2nd column)".
- Table header: Points to the first row of the table.
- Table body: Points to the main data rows of the table.
- Table footer: Points to the last row of the table.
- Table border: Points to the outer border of the table.

Fig. 2.12 | Creating a basic table. (Part 4 of 4.)

# Tables

- A table can be split into three distinct sections:
  - Head (**thead** element)
    - Table titles
    - Column headers
  - Body (**tbody** element)
    - Primary table data
  - Table Foot (**tfoot** element)
    - Calculation results
    - Footnotes
    - Above body section in the code, but displays at the bottom in the page

- Tr Element
  - Defines individual table rows
  - Element th
  - Defines a header cell
- Td Element
  - Contains table data elements

R

~~X~~ C

# Tables

- Using rowspan and colspan with Tables
  - ▶ Figure 2.13 introduces two new attributes that allow you to build more complex tables.
  - ▶ You can merge data cells with the **rowspan** and **colspan** attributes *put part in take*  
The values of these attributes specify the **number of rows or columns** occupied by the cell.  
Can be placed inside any data cell or table header cell.
  - ▶ The **br** element is rendered as a line break in most browsers—any markup or text following a **br** element is rendered on the **next line**.
  - ▶ Like the **img** element, **br** is an example of a **void element**.
  - ▶ Like the **hr** element, **br** is considered a **legacy formatting element** that you should **avoid using**—in general, formatting should be specified using CSS.

# Tables

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 2.13: table2.html -->
4 <!-- Complex HTML5 table. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Tables</title>
9   </head>
10
11 <body>
12   <h1>Table Example: Spanning Rows and Columns</h1>
13
14   <table border = "1">
15     <caption>A more complex sample table</caption>
16
```

Fig. 2.13 | Complex HTML5 table. (Part 1 of 4.)

```
17
18   <thead>
19     <!-- rowspans and colspans merge the specified -->
20     <!-- number of cells vertically or horizontally -->
21     <tr>
22       <!-- merge two rows -->
23       <th rowspan = "2"> A <br> Lh <br> 羚 <br> 羚子
24         <img src = "camel.png" width = "205"
25           height = "167" alt = "Picture of a camel">
26     </th>
27
28     <!-- merge four columns -->
29     <th colspan = "4">
30       <strong>Camelid comparison</strong><br>
31         Approximate as of 10/2011
32       </th>
33     </tr>
34     <tr>
35       <th># of humps</th>
36       <th>Indigenous region</th>
37       <th>Spits?</th>
38       <th>Produces wool?</th>
39     </tr>
</thead>
```

Fig. 2.13 | Complex HTML5 table. (Part 2 of 4.)



# Tables

```
40 <tbody>
41   <tr>
42     <th>Camels (bactrian)</th>
43     <td>2</td>
44     <td>Africa/Asia</td>
45     <td>Yes</td>
46     <td>Yes</td>
47   </tr>
48   <tr>
49     <th>Llamas</th>
50     <td>1</td>
51     <td>Andes Mountains</td>
52     <td>Yes</td>
53     <td>Yes</td>
54   </tr>
55 </tbody>
56 </table>
57 </body>
58 </html>
```

Fig. 2.13 | Complex HTML5 table. (Part 3 of 4.)

The screenshot shows a web browser window titled "Tables". The URL in the address bar is "file:///C:/books/2011/IW3HTP5/examples/ch02/table2.html". The page title is "Table Example: Spanning Rows and Columns". A caption above the table reads "A more complex sample table". The table has a header row with four columns: "# of humps", "Indigenous region", "Spits?", and "Produces wool?". Below the header, there are two data rows. The first data row contains the text "Camels (bactrian)" in the first column, and "2", "Africa/Asia", "Yes", and "Yes" in the subsequent columns respectively. The second data row contains the text "Llamas" in the first column, and "1", "Andes Mountains", "Yes", and "Yes" in the subsequent columns respectively. A red oval highlights the camel silhouette in the first data row.

	# of humps	Indigenous region	Spits?	Produces wool?
Camels (bactrian)	2	Africa/Asia	Yes	Yes
Llamas	1	Andes Mountains	Yes	Yes

Fig. 2.13 | Complex HTML5 table. (Part 4 of 4.)

# Internal Linking

First page to next page link

- The `a` tag can be used to link to another section of the same document by specifying the element's `id` as the link's `href`.
  - To link internally to an element with its `id` attribute set, use the syntax `#id`.

```
I <!DOCTYPE html>
2
3 <!-- Fig. 2.16: internal.html -->
4 <!-- Internal Linking -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Internal Links</title>
9   </head>           go down ↗
10
11  <body>
12    <!-- id attribute creates an internal hyperlink destination -->
13    <h1 id = "features">The Best Features of the Internet</h1>
14
15    <!-- an internal link's address is "#id" -->
16    <p><a href = "#bugs">Go to <em>Favorite Bugs</em></a></p>
17
18    <ul>           ↑ go here if 可能有
19      <li>You can meet people from countries
20        around the world.</li>
21      <li>You have access to new media as it becomes public:
22        <ul>
23          <li>New games</li>
```

```
24 <li>New applications
25   <ul>
26     <li>For Business</li>
27     <li>For Pleasure</li>
28   </ul>
29 </li>
30
31 <li>Around the clock news</li>
32 <li>Search Engines</li>
33 <li>Shopping</li>
34 <li>Programming
35   <ul>
36     <li>HTML5</li>
37     <li>Java</li>
38     <li>Dynamic HTML</li>
39     <li>Scripts</li>
40     <li>New languages</li>
41   </ul>
42 </li>
43 </ul>
44 </li>
```

**Fig. 2.16** | Internal hyperlinks to make pages more navigable. (Part 1 of 5.)

**Fig. 2.16** | Internal hyperlinks to make pages more navigable. (Part 2 of 5.)



# Internal Linking

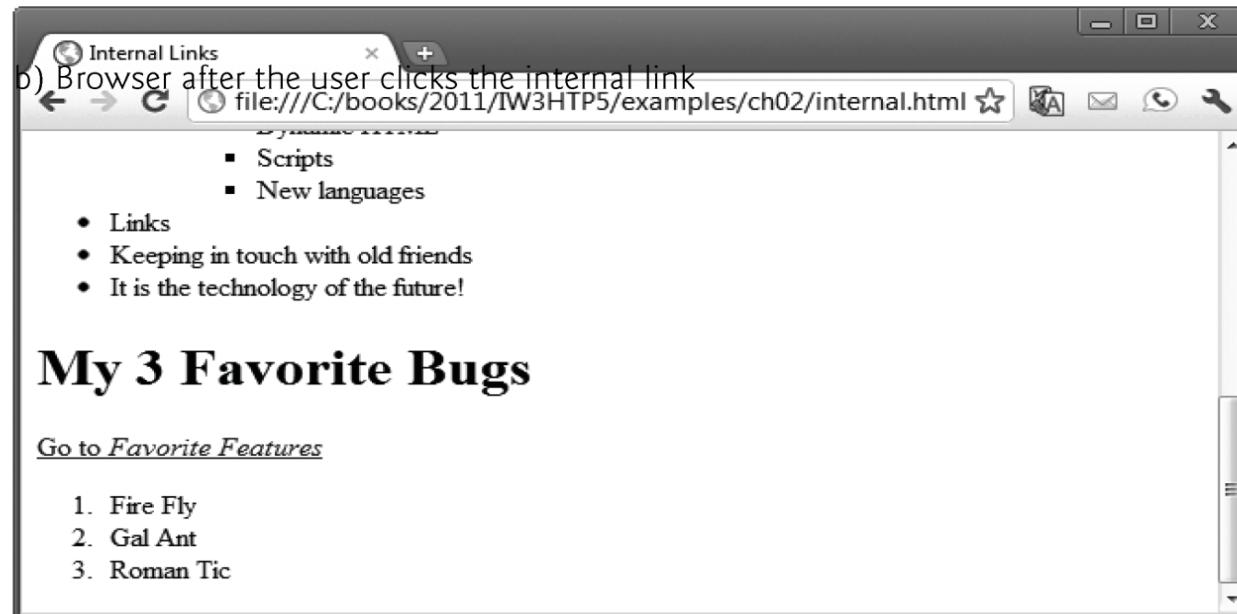
```
46    <li>Links</li>
47    <li>Keeping in touch with old friends</li>
48    <li>It is the technology of the future!</li>
49  </ul>
50
51  <!-- id attribute creates an internal hyperlink destination -->
52  <h1 id = "bugs">My 3 Favorite Bugs</h1>
53  <p>
54    <!-- internal hyperlink to features -->
55    <a href = "#features">Go to <em>Favorite Features</em></a>
56  </p>
57  <ol>
58    <li>Fire Fly</li>
59    <li>Gal Ant</li>
60    <li>Roman Tic</li>
61  </ol>
62  </body>
63 </html>
```

**Fig. 2.16 |** Internal hyperlinks to make pages more navigable. (Part 3 of 5.)



**Fig. 2.16 |** Internal hyperlinks to make pages more navigable. (Part 4 of 5.)

# Internal Linking



**Fig. 2.16** | Internal hyperlinks to make pages more navigable. (Part 5 of 5.)

# Meta Elements

- One way that search engines catalog pages is by reading the meta element's contents.
- The **name** attribute identifies the type of meta element
- The **content** attribute
  - Of a **keywords** meta element: provides search engines with a list of words that describe a page, which are compared with words in search requests
  - Of a **description** meta element: provides a three- to four-line description of a site in sentence form, used by search engines to catalog your site. This text is sometimes displayed as part of the search result

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 2.17: meta.html -->
4  <!-- meta elements provide keywords and a description of a page. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Welcome</title>
9
10         <!-- <meta> tags provide search engines with -->
11         <!-- information used to catalog a site -->
12         <meta name = "keywords" content = "web page, design,
13             HTML5, tutorial, personal, help, index, form,
14             contact, feedback, list, links, deitel">
15         <meta name = "description" content = "This website will
16             help you learn the basics of HTML5 and web page design
17             through the use of interactive examples and
18             instruction.">
19     </head>
```

Fig. 2.17 | meta elements provide keywords and a description of a page. (Part 1 of 3.)

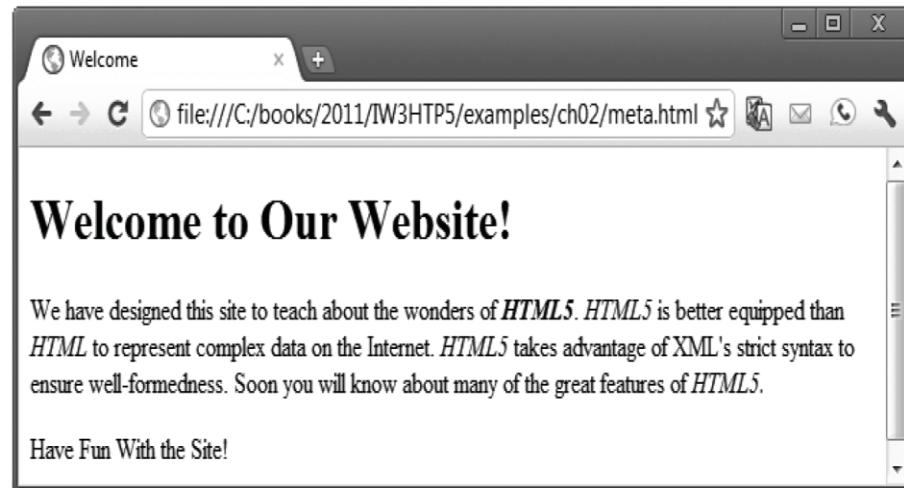
else  
one device  
not support  
meta

(Slanted  
HTML 5)  
meta element  
for all  
device

# Meta Elements

```
20 <body>
21   <h1>Welcome to Our Website!</h1>
22
23   <p>We have designed this site to teach about the wonders
24   of <strong><em>HTML5</em></strong>. <em>HTML5</em> is
25   better equipped than <em>HTML</em> to represent complex
26   data on the Internet. <em>HTML5</em> takes advantage of
27   XML's strict syntax to ensure well-formedness. Soon you
28   will know about many of the great features of
29   <em>HTML5.</em></p>
30
31   <p>Have Fun With the Site!</p>
32 </body>
33 </html>
```

**Fig. 2.17** | meta elements provide keywords and a description of a page. (Part 2 of 3.)



**Fig. 2.17** | meta elements provide keywords and a description of a page. (Part 3 of 3.)



## Software Engineering Observation 2.2

meta elements are not visible to users. They must be placed inside the head section of your HTML5 document; otherwise they will not be read by search engines.

# HTML <button> Tag

```
<!DOCTYPE html>
<html>
<body>

<form method="post">
    First name: <input type="text" name="fname"><br>
    Last name: <input type="text" name="lname"><br>
    <button type="button" value="Add">Add</button>
    <button type="submit" value="Submit">Submit</button>
    <button type="reset" value="Reset">Reset</button>
</form>

</body>
</html>
```

verb

<button type="submit" value="Add">

<img src = "buttons/contact.jpg" width = "65"  
height = "50" alt = "Contact Me" />

</button>

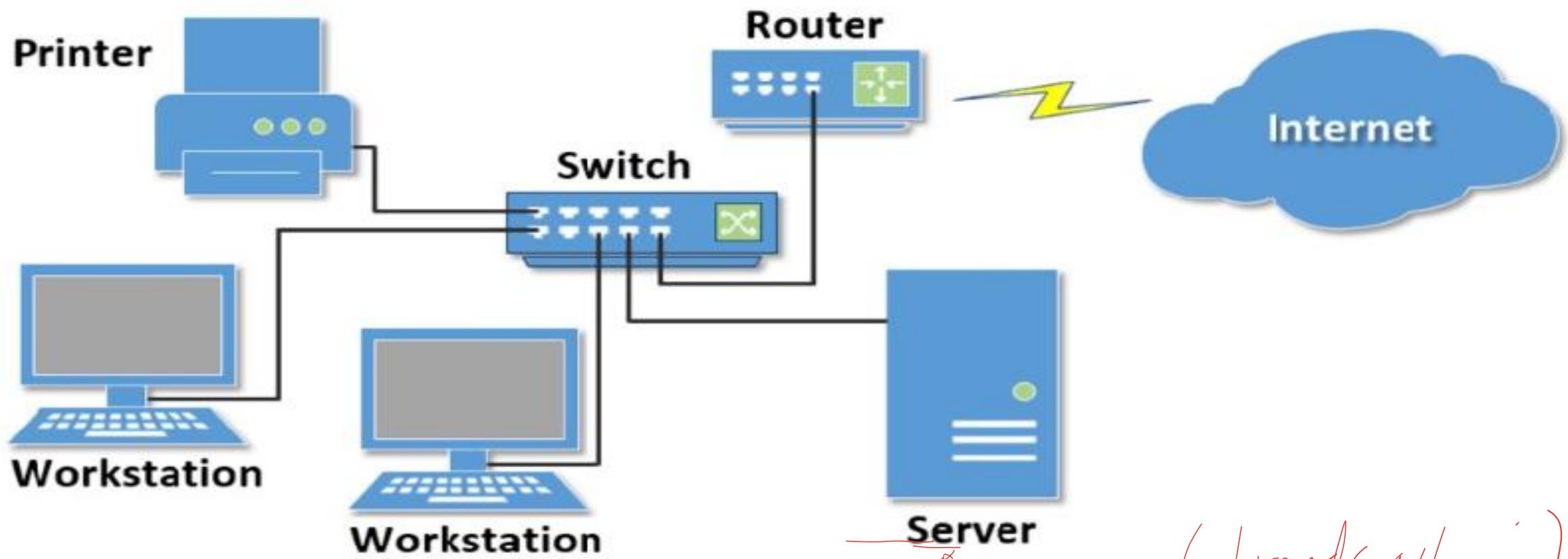
# UCCD2323 Front-End Web Development



## Chapter 1 Client-side Programming on Interface Part 3.

# Network Overview

- Network is a two or more computers connected together for the purpose of communicating and sharing resources.



Azme (broadcast i.h)

# The Client / Server Model

- Client/Server can describe a relationship between two computer programs – the "client" and the "server".

- **Client**

- requests some type of service (such as a file or database access) from the server.

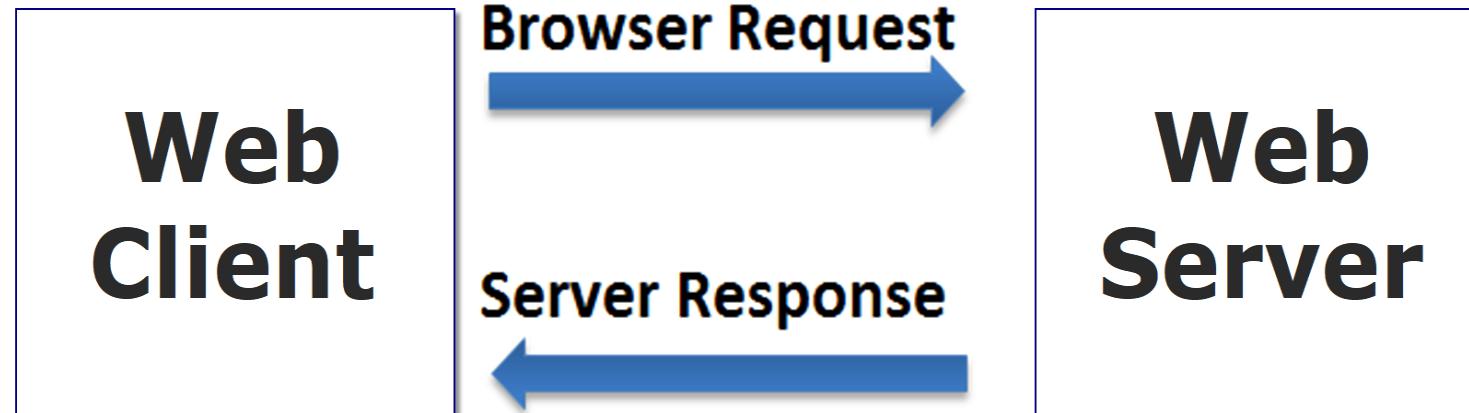
- **Server**

- fulfills the request and transmits the results to the client over a network.

# The Client / Server Model

- The Internet Client/Server Model

- Client: Web Browser
- Server: Web Server



## Internet Protocols

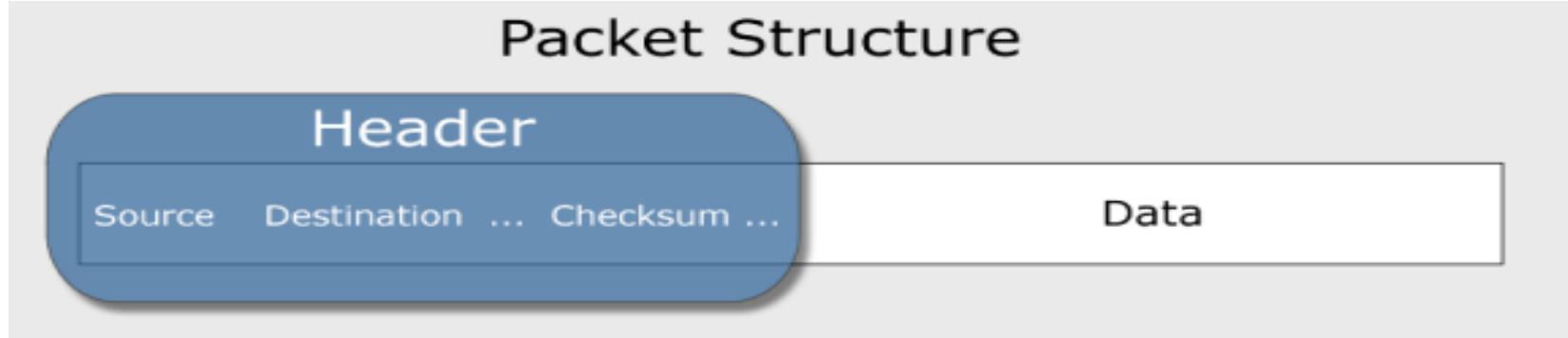
Web / HTTP / TCP / IP

- **Protocols**
  - Rules that describe the methods used for clients and servers to communicate with each other over a network.
- There is no *single* protocol that makes the Internet and Web work.
- A number of protocols with specific functions are needed.

# Common Internet Protocols

← Server Side

- Official Communication Protocol: TCP/IP



- Specialized Protocols:

- File Transfer: FTP
- E-mail: SMTP, POP3, IMAP
- Websites: HTTP, HTTP/2

HTTPS ✓

AK

# HTTP Hypertext Transfer Protocol

- A set of rules for exchanging files such as text, graphic images, sound, video, and other multimedia files on the Web.
- Web browsers send HTTP requests for web pages and their associated files.
- Web servers send HTTP responses back to the web browsers.
- HTTPS Hypertext Transfer Protocol Secure
  - Combines HTTP with a security and encryption protocol

# IP Address

- Each device connected to the Internet has a unique numeric IP address.
- These addresses consist of a set of four groups of numbers, called octets.

216.58.194.46 will get you Google!

- An IP address may correspond to a domain name.

IP

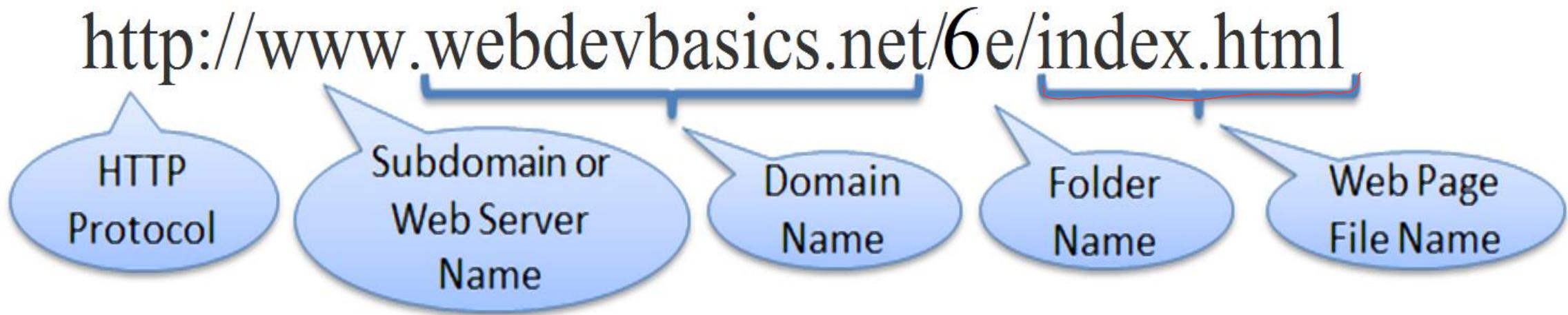
# Domain Name

- Locates an organization or other entity on the Internet
- Domain Name System
  - Divides the Internet into logical groups and understandable names
  - Associates unique computer IP Addresses with the text-based domain names you type into a web browser
    - Browser: <http://google.com>
    - IP Address: 216.58.194.46

# Uniform Resource Indicator (URI)

- (URL) Uniform Resource Locator:

Represents the address of a resource on the Internet.



## Top – Level Domain Name (TLD)

- A top-level domain (TLD) identifies the right-most part of the domain name.
- Some generic TLDs: com, .org, .net, .mil, .gov, .edu, .int, .aero, .asia, .cat, .jobs, .name, .biz, .museum, .info, .coop, .pro, .travel

### NEW gTLDs!

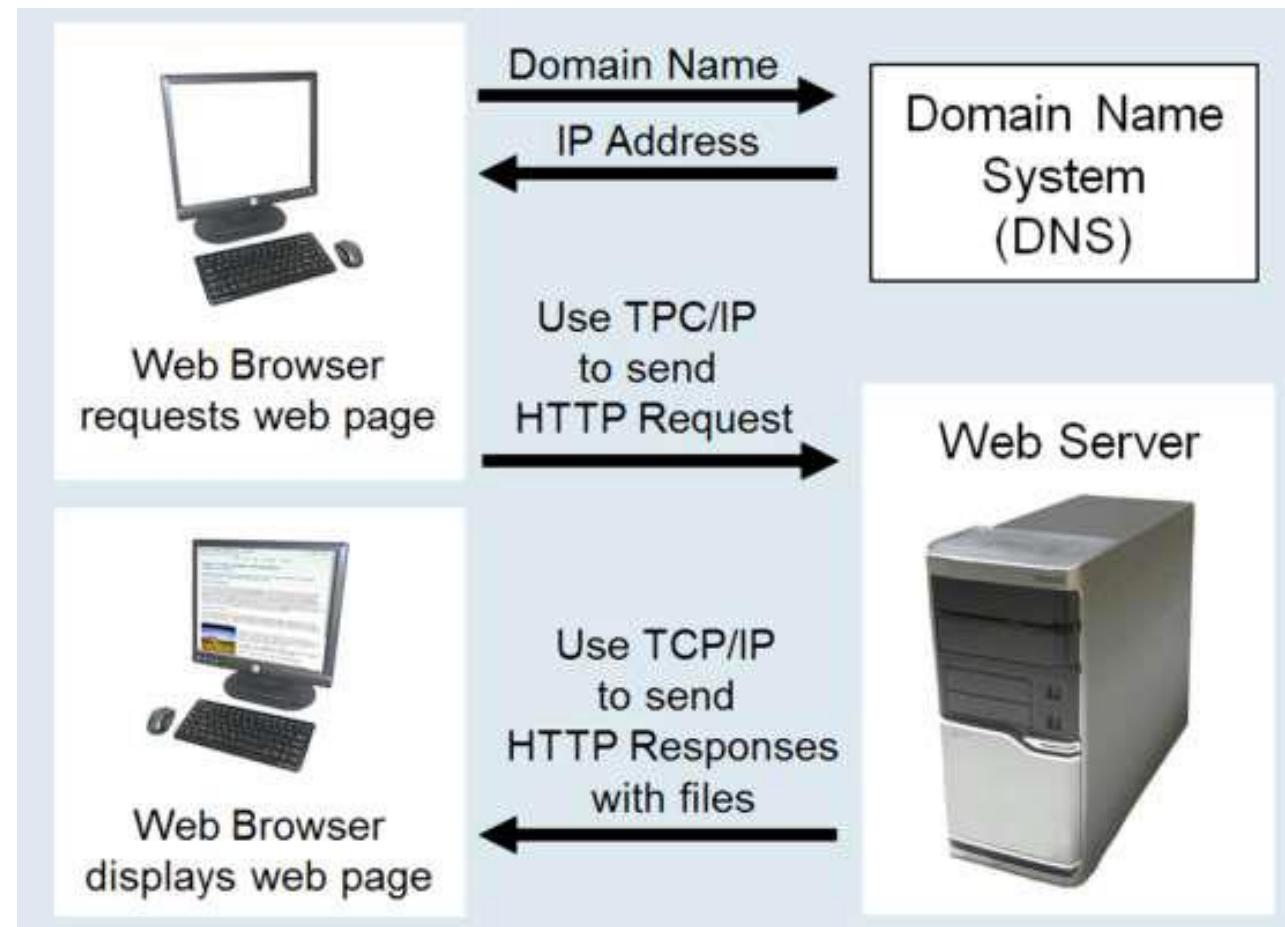
- As of 2020, there were over 1,900 TLDs
  - New gTLDs to become available included .bike, .guru, .holdings, .clothing, .singles, .ventures, and .plumbing.
  - ICANN has set a schedule to periodically launch new gTLDs.
  - List of new gTLDs: <http://newgtlds.icann.org/en/program-status/delegated-strings>

## Country Code TLDs

- Two character codes originally intended to indicate the geographical location (country) of the web site.
- In practice, it is fairly easy to obtain a domain name with a country code TLD that is not local to the registrant.
- Examples:
  - .tv, .ws, .au, .jp, .uk
  - List of TLDs: <http://www.iana.org/domains/root/db>

# Domain Name System

- The Domain Name System (DNS) associates Domain Names with IP addresses.



# Web Hosting

- The term “web hosting” usually refers to the server that host your website
- Data center usually refers to the facility that is used to house the servers.
- Web servers used to host the web sites are housed in data center

## A Web Hosting Server



# Web Server Software

- Web server software allows computers/servers to act as web servers



Server	Developed by	Software license	Last stable version	Latest release date
AOLserver	NaviSoft	Mozilla	4.5.2	2012-09-19
Apache HTTP Server	Apache Software Foundation	Apache	2.4.41	2019-08-14
Apache Tomcat	Apache Software Foundation	Apache	9.0.21	2019-06-07
Boa	Jon Nelson and Larry Doolittle	GNU GPL	0.94.13	2002-07-30 (discontinued)
Caddy	Matt Holt	Apache	1.0.0	2019-04-24
Caudium	The Caudium Group	GNU GPL	1.4.18	2012-02-24
Cherokee HTTP Server	Álvaro López Ortega	GNU GPL	1.2.103	2013-04-21
GlassFish	"Oracle Corporation (initial code from Sun Microsystems)"	Common Development and Distribution License & GNU General Public License	5.1	2019-01-28
Hiawatha	Hugo Leisink	GNU GPLv2	10.9	2019-02-18
HFS	Rejetto	GNU GPL	2.3i	2016-06-14
IBM HTTP Server	IBM	Non-free proprietary	9.0.0	2016-03-11
Internet Information Services	Microsoft	Non-free proprietary	10.0.17763.1	2018-10-02
Jetty	Eclipse Foundation	Apache	9.4.18	2019-04-29
Jexus	Bing Liu	Non-free proprietary	5.5.2	2014-04-27
lighttpd	Jan Kneschke (Incremental)	BSD variant	1.4.54	2019-05-27
LiteSpeed Web Server	LiteSpeed Technologies	Non-free proprietary	5.3.7	2019-03-15
Mongoose	Cesanta Software	GNU GPLv2 / proprietary license	6.14	2019-03-04
Monkey HTTP Server	Monkey Software	Apache	1.6.9	2016-05-04
NaviServer	Various	Mozilla 1.1	4.99.18	2019-02-24
NCSA HTTPd	Robert McCool	Non-free proprietary	1.5.2a	1996-10-08
Nginx	NGINX, Inc.	BSD variant	1.16.1	2019-08-13
OpenLink Virtuoso	OpenLink Software	GNU GPL and proprietary versions	7.2.5.1	2018-10-22
Oracle HTTP Server	Oracle Corporation	Non-free proprietary	12.1.2	2014-09-23
Oracle iPlanet Web Server	Oracle Corporation	BSD	7.0.23	2016-02-12
Oracle WebLogic Server	Oracle Corporation (formerly BEA Systems)	Non-free proprietary	12cR3 (12.1.3)	2014-06-26
Resin Open Source	Caucho Technology	GNU GPLv3 / proprietary license	4.0.57	2018-07-19
Resin Professional	Caucho Technology	Non-free proprietary	4.0.49	2016-10-19
thttpd	Jef Poskanzer for ACME Laboratories	BSD variant	2.27	2014-10-03
TUX web server	Ingo Molnár	GNU GPL	3.2.6.18	2006-09-20
Wakanda Server	Wakanda	GNU AGPLv3 / proprietary license	2.1.0	2017-07-27
WEBrick	Ruby Community	Ruby	1.9.3 p286 (Ruby)	2012-10-12
Xitami	iMatix Corporation	BSD	5.0a0	2009-02-19
Yaws	Claes Wikström	BSD variant	2.0.6	2018-06-26
Zeus Web Server	Zeus Technology	Non-free proprietary	4.3r5	2010-01-13
Zope	Zope Corporation	Zope	2.13.21	2013-07-16
Server	Developed by	Software license	Last stable version	Latest release date

# Operating system support

Server	Windows	Linux	macOS	BSD	Solaris	eComStation	OpenVMS	AIX	IBM i	z/OS	HP-UX
<b>AOLserver</b>	No	Yes	Yes	Yes	Yes	No	No	Unknown	No	Unknown	Unknown
<b>Apache HTTP Server</b>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
<b>Apache Tomcat</b>	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
<b>Boa</b>	Unknown	Yes	No	Yes	Unknown	No	No	Unknown	No	Unknown	Unknown
<b>Caddy</b>	Yes	Yes	Yes	Yes	Yes	No	No	Unknown	Unknown	Unknown	Unknown
<b>Cauchao Resin Server</b>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
<b>Caudium<sup>[16]</sup></b>	No	Yes	Yes	Yes	Yes	No	No	Yes	No	Unknown	Unknown
<b>Cherokee HTTP Server</b>	No <sup>[17]</sup>	Yes	Yes	Yes	Yes	No	No	Unknown	No	Unknown	Unknown
<b>HFS</b>	Yes	No	No	No	No	No	No	No	No	No	No
<b>Hiawatha</b>	with Cygwin <sup>[18]</sup>	Yes <sup>[18]</sup>	Yes <sup>[18]</sup>	Yes <sup>[18]</sup>	Yes <sup>[18]</sup>	No	No	No	No	No	No
<b>IBM HTTP Server</b>	Yes	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes
<b>Internet Information Services</b>	Yes	No	No	No	No	No	No	No	No	No	No
<b>Jetty (Java)</b>	Yes	Yes	Yes	Yes	Yes	Yes	No	Unknown	No	Yes	Unknown
<b>Jexus</b>	No	Yes	No	Yes	Unknown	No	No	No	No	No	No
<b>lighttpd</b>	Yes (Cygwin)	Yes	Yes	Yes	Yes	No	No	Yes	No	No	Yes
<b>LiteSpeed Web Server</b>	No	Yes	Yes	Yes	Yes	No	No	Unknown	No	Unknown	Unknown
<b>Mongoose</b>	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes
<b>Monkey HTTP Server</b>	No	Yes	Yes	No	No	No	No	No	No	No	No
<b>NaviServer</b>	Yes	Yes	Yes	Yes	Yes	Unknown	Unknown	Yes	No	Unknown	Unknown
<b>NCSA HTTPd</b>	Unknown	Yes	No	Yes	Yes	Yes	No	Unknown	No	Unknown	Yes
<b>nginx</b>	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No	Yes
<b>OpenLink Virtuoso</b>	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No	Yes
<b>Oracle HTTP Server</b>	Yes	Yes	No	Unknown	Yes	No	No	Yes	No	Unknown	Unknown
<b>Oracle iPlanet Web Server</b>	Yes	Yes	No	No	Yes	No	No	Yes	No	No	Yes
<b>thttpd</b>	Yes (Cygwin)	Yes	Yes	Yes	Yes	No	No	Unknown	No	Unknown	Unknown
<b>TUX web server</b>	No	Yes	No	No	No	No	No	No	No	No	No
<b>Wakanda Server</b>	Yes	Yes (via libbsd)	Yes	Yes	Unknown	No	No	No	No	No	No
<b>Xitami</b>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Unknown	Yes
<b>Yaws</b>	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Unknown	Unknown

# A Data Center



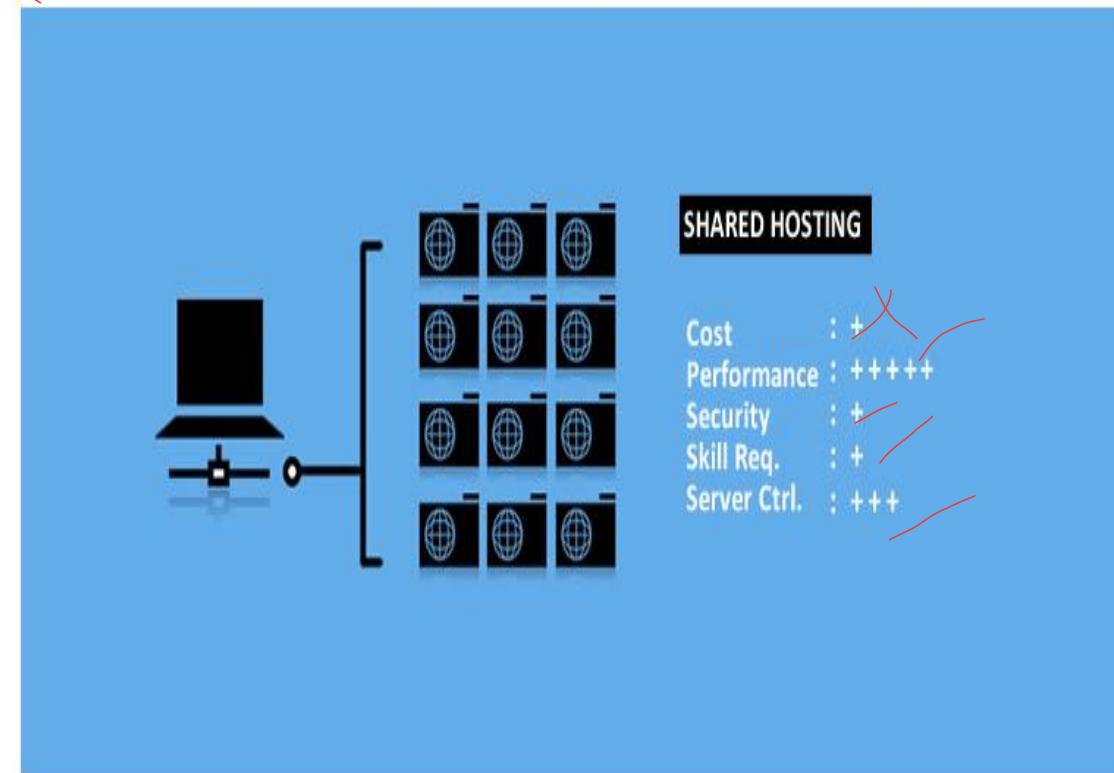
## Types of Web Hosting

- Shared Hosting
- Virtual Private Server (VPS) Hosting
- Dedicated Server Hosting
- Cloud Hosting

X pndme

# Shared Hosting

- The same server is used to host many web sites
- This is analogous to an apartment that has many units in it.
- Share a common pool of server resources, such as RAM and the CPU
- **Advantages**
  - extremely low cost
  - requires minimum technical knowledge

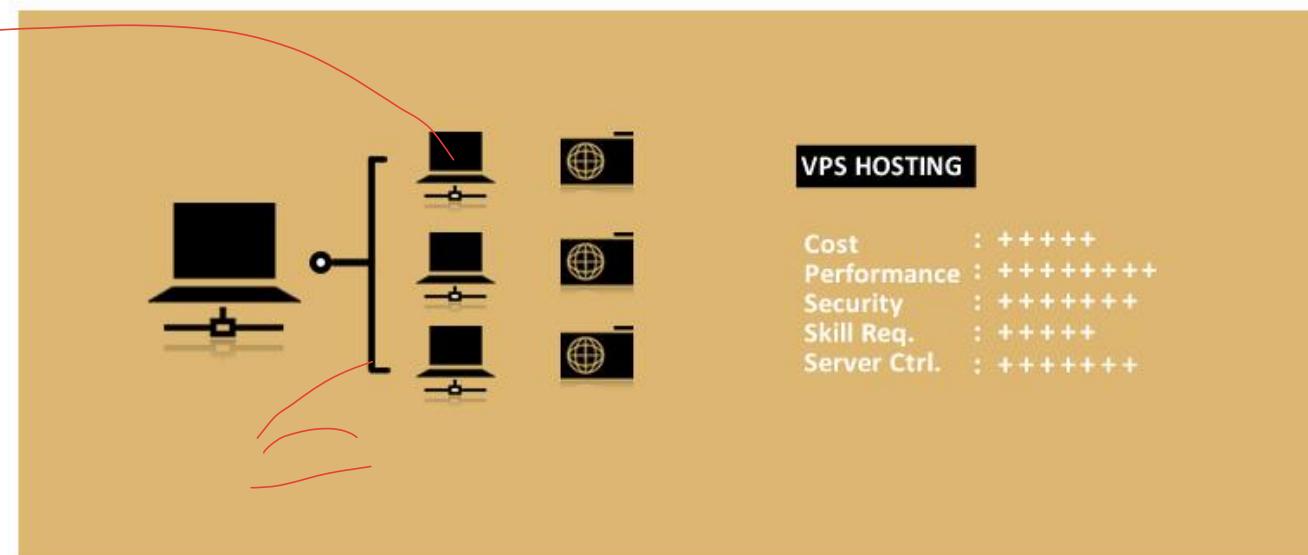


# Shared Hosting

- **Disadvantages**
  - No root access
  - limited ability to handle high traffic levels or spikes
  - site performance can be affected by other sites on the same server
- slow traffic  
hack  
hang

# Virtual Private Server (VPS) Hosting

- Hypervisor software such as (Hyper-V, VMware vSphere or Virtual Box) runs in the server.
- Virtual machines are used for web hosting. Each website is like hosted on their own dedicated server
- **Advantages:**
  - The users may have root access to their own virtual space
  - Better secured hosting environment



# Virtual Private Server (VPS) Hosting

- **Disadvantages:**
  - Limited ability to handle high traffic levels or spikes, your site performance can still be somewhat affected by other sites on the server.

# Dedicated Server Hosting

- An entire server is dedicated to host only your website(s).

High server

- **Advantage**

- Offers the maximum control over the web server

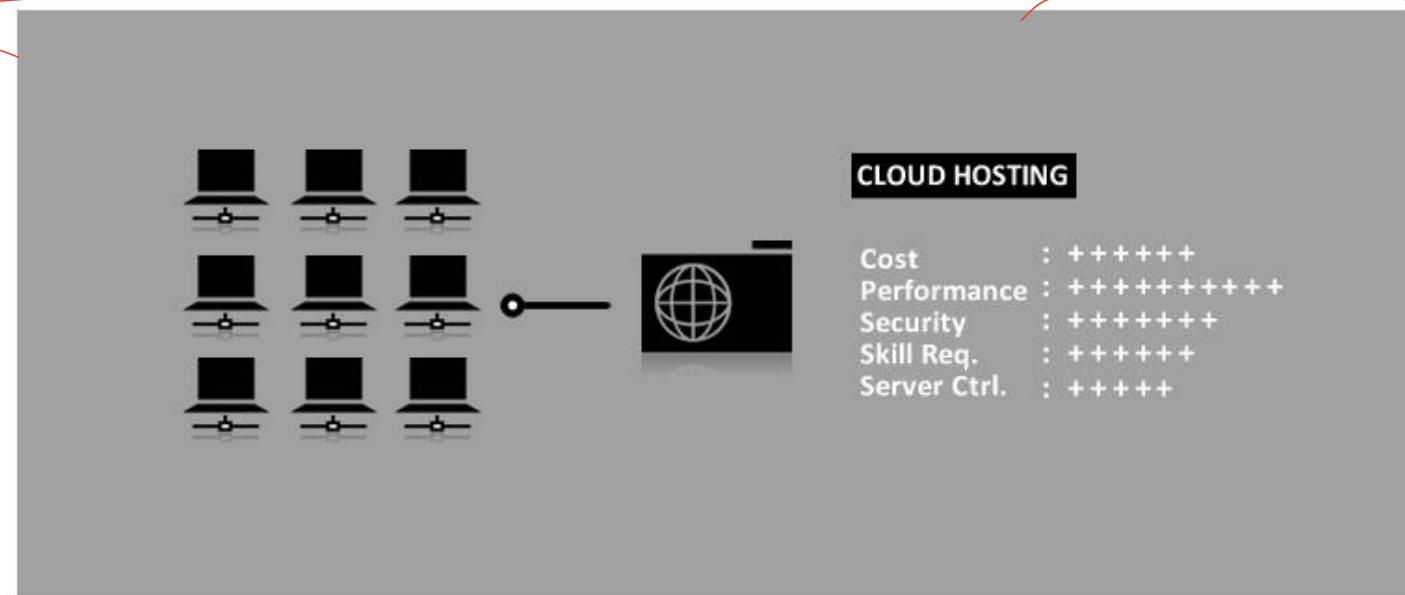
- **Disadvantages**

- Very expensive



# Cloud Hosting

- A team of servers (called a cloud) work together to host a group of websites.
- **Advantage**
  - Ability to handle high traffic or traffic spikes
- **Disadvantages**
  - Many cloud hosting setup do not offers root access (required to change server settings and install some software)
  - higher cost



# UCCD2323 Front-End Web Development



## Chapter 2 Cascading Style Sheet (CSS) Part 1.

# Topic Outline

- Introduction
- Inline Styles
- Embedded Style Sheets
- Conflicting Styles
- Linking External Style Sheets
- Positioning Elements: Absolute Positioning, z-index
- Positioning Elements: Relative Positioning, span
- Backgrounds
- Element Dimensions
- Box Model and Text Flow
- Media Types and Media Queries
- Drop-Down Menus
- (Optional) User Style Sheets
- Web Resources

CS

# Objectives

In this chapter you'll:

- Control a website's appearance with style sheets.
- Use a style sheet to give all the pages of a website the same look and feel.
- Use the class attribute to apply styles.
- Specify the precise font, size, color and other properties of displayed text.
- Specify element backgrounds and colors.
- Understand the box model and how to control margins, borders and padding.
- Use style sheets to separate presentation from content.

Template F ~~FAK~~

# Introduction

- **Cascading Style Sheets 3 (CSS3)**

Used to specify the presentation of elements separately from the structure of the document.

- **CSS validator**

➤ <https://jigsaw.w3.org/css-validator/>

➤ This tool can help you make sure that your code is correct and will work on CSS3-compliant browsers.

- **Types of Style Sheets**

➤ **Inline CSS**

~~By Tag~~

➤ **Embedded CSS**

?

➤ **External CSS**

File

# Style Specification Formats

- Inline CSS

```
<selector style =  
"property_1:value_1;property_2:value_2;...  
property_n:value_n">
```

```
<p style = "font-size: 20pt;">This text has the  
    <em>font-size</em> style applied to it, making it 20pt.  
</p>
```

```
<p style = "font-size: 20pt; color: deepskyblue;">  
    This text has the <em>font-size</em> and  
    <em>color</em> styles applied to it, making it  
    20pt and deep sky blue.</p>
```

- Embedded CSS

```
<style type="text/css">  
    rule_list  
</style>  
    selector  
    {property_1:value_1;property_2:value_2;...pr  
     operty_n:value_n}
```

```
<style type = "text/css">  
    em { font-weight: bold;  
          color: black; }  
    h1 { font-family: tahoma, helvetica, sans-serif; }  
    p { font-size: 12pt;  
        font-family: arial, sans-serif; }  
    .special { color: purple; }  
</style>
```

# Style Specification Formats

- External CSS

```
<link rel = "stylesheet" type = "text/css" href = "css/styles.css">
```

Selector {property\_1:value\_1;property\_n:value\_n;}

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Linking External Style Sheets</title>
    <link rel = "stylesheet" type = "text/css"
          href = "css/styles.css">
  </head>
  <body>
    <h1>Shopping list for <em>Monday</em></h1>

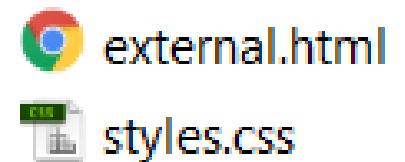
    <ul>
      <li>Milk</li>
      <li>Bread
        <ul>
          <li>White bread</li>
          <li>Rye bread</li>
          <li>Whole wheat bread</li>
        </ul>
      </li>
      <li>Carrots</li>
      <li>Yogurt</li>
      <li>Pizza <em>with mushrooms</em></li>
    </ul>

    <p><em>Go to the</em>
       <a class = "nodec" href = "http://www.deitel.com">
         Grocery store</a>
     </p>
  </body>
</html>
```

HTML document  
.html

CSS document  
.css

```
body { font-family: arial, helvetica, sans-serif; }
a.nodec { text-decoration: none; }
a:hover { text-decoration: underline; }
li em { font-weight: bold; }
h1, em { text-decoration: underline; }
ul { margin-left: 20px; }
ul ul { font-size: .8em; }
```



external.html  
styles.css

# Selector Forms

- Simple Selector Form

```
h1 {font-size: 24pt;}  
h2, h3 {font-size:20pt;}
```

- Class Selectors

```
p.normal {font-size:13pt;}  
p.warning {font-size:40pt;}
```

- Generic Selectors

```
.sale {property-value list}
```

- id Selectors

```
#section2 {font-size:20;}
```

- Universal Selectors

```
* {color: red;}
```

# CSS Selectors

SELECTOR	MEANING	EXAMPLE
UNIVERSAL SELECTOR	Applies to all elements in the document	* [] Targets all elements on the page
TYPE SELECTOR	Matches element names	h1, h2, h3 [] Targets the <h1>, <h2> and <h3> elements
CLASS SELECTOR	Matches an element whose class attribute has a value that matches the one specified after the period (or full stop) symbol	.note [] Targets any element whose class attribute has a value of note p.note {} Targets only <p> elements whose class attribute has a value of note
ID SELECTOR	Matches an element whose id attribute has a value that matches the one specified after the pound or hash symbol	#introduction [] Targets the element whose id attribute has a value of introduction
CHILD SELECTOR	Matches an element that is a direct child of another	li>a [] Targets any <a> elements that are children of an <li> element (but not other <a> elements in the page)
DESCENDANT SELECTOR	Matches an element that is a descendent of another specified element (not just a direct child of that element)	p a {} Targets any <a> elements that sit inside a <p> element, even if there are other elements nested between them

# How CSS Rules Cascade

chapter-10/cascade.html

HTML

```
<h1>Potatoes</h1>
<p id="intro">There are <i>dozens</i> of different
<b>potato</b> varieties.</p>
<p>They are usually described as early, second early
and maincrop potatoes.</p>
```

CSS

```
* {
  font-family: Arial, Verdana, sans-serif;}
h1 {
  font-family: "Courier New", monospace;}
i {
  color: green;}
i {
  color: red;}
b {
  color: pink;}
p b {
  color: blue !important;}
p b {
  color: violet;}
p#intro {
  font-size: 100%;}
p {
  font-size: 75%;}
```

RESULT

## Potatoes

There are *dozens* of different **potato** varieties.

They are usually described as early, second early and maincrop potatoes.

# Pseudo Classes

- A [CSS \*pseudo-class\*](#) is a keyword added to a selector that specifies a special state of the selected element(s).
- Pseudo-classes let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors.
- ```
<style type="text/css">
input:hover {background:pink; color:red;}
input:focus {background:lightblue; color:blue;}
</style>
```
- ```
<style type="text/css">
a:link {color:#FF0000;} /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
</style>
```

# Inline Styles

- **Inline style**
  - declare an individual element's format using the HTML5 attribute style.
- Figure 4.1 applies inline styles to p elements to alter their font size and color.
- Each CSS property is followed by a colon and the value of the attribute
  - Multiple property declarations are separated by a semicolon



## Software Engineering Observation 4.1

Inline styles do not truly separate presentation from content. To apply similar styles to multiple elements, use embedded style sheets or external style sheets, introduced later in this chapter.

# Inline Styles

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 4.1: inline.html -->
4 <!-- Using inline styles -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Inline Styles</title>
9   </head>
10  <body>
11    <p>This text does not have any style applied to it.</p>
12
13    <!-- The style attribute allows you to declare -->
14    <!-- inline styles. Separate multiple -->
15    <!-- style properties with a semicolon. -->
16    <p style = "font-size: 20pt;">This text has the
17      <em>font-size</em> style applied to it, making it 20pt.
18    </p>
19
20    <p style = "font-size: 20pt; color: deepskyblue;">
21      This text has the <em>font-size</em> and
22      <em>color</em> styles applied to it, making it
23      20pt and deep sky blue.</p>
24
25  </body>
26 </html>
```

Fig. 4.1 | Using inline styles. (Part 1 of 2.)



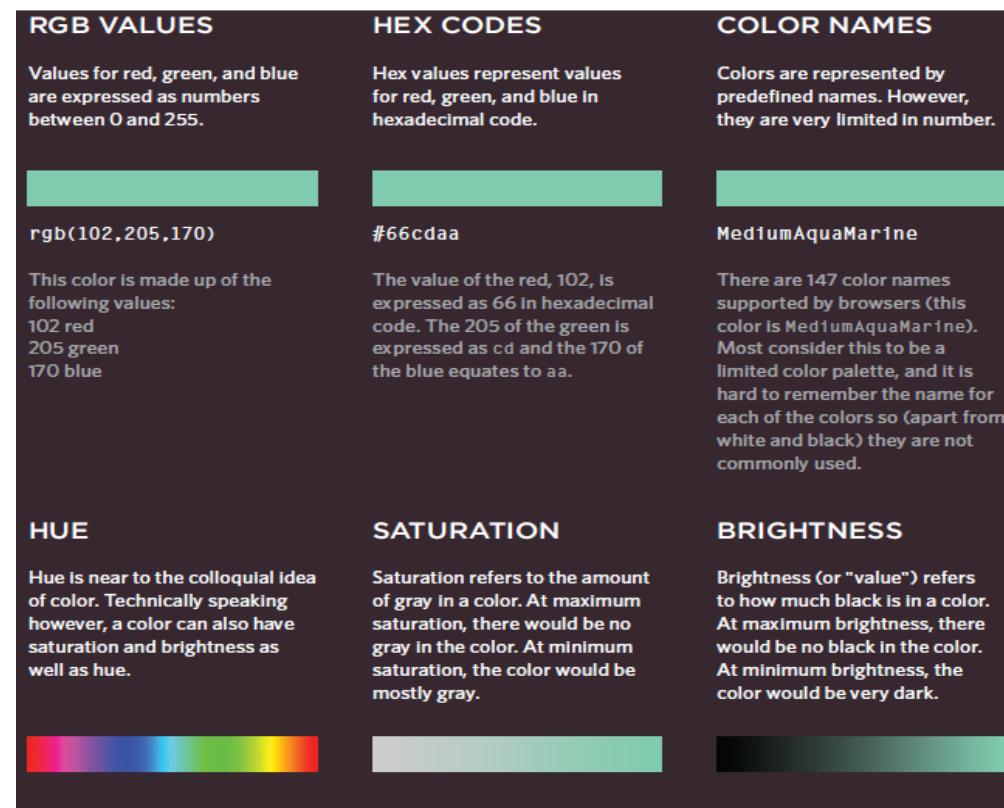
Fig. 4.1 | Using inline styles. (Part 2 of 2.)

# Inline Styles

- **color property sets text color**
  - Color names and hexadecimal codes may be used as the color property value.
  - Figure 4.2 contains the HTML standard color set.
  - A list of extended hexadecimal color codes and color names is provided in Appendix B.
  - You can also find a complete list of HTML standard and extended colors at <https://www.w3.org/TR/css-color-3/>

Color name	Value	Color name	Value
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	yellow	#FFFF00
maroon	#800000	white	#FFFFFF

**Fig. 4.2 |** HTML standard colors and hexadecimal RGB values.



# Embedded Style Sheets

- A second technique for using style sheets is **embedded style sheets**, which enable you to embed a CSS3 document in an HTML5 document's head section.
- Figure 4.3 creates an embedded style sheet containing four styles.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.3: embedded.html -->
4  <!-- Embedded style sheet. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Embedded Style Sheet</title>
9
10         <!-- this begins the style sheet section -->
11         <style type = "text/css">
12             em      { font-weight: bold;
13                         color: black; }
14             h1      { font-family: tahoma, helvetica, sans-serif; }
15             p       { font-size: 12pt;
16                         font-family: arial, sans-serif; }
17             .special { color: purple; }
18         </style>
19     </head>
```

Fig. 4.3 | Embedded style sheet. (Part 1 of 3.)

```
20    <body>
21        <!-- this attribute applies the .special style class -->
22        <h1 class = "special">Deitel & Associates, Inc.</h1>
23
24        <p>Deitel & Associates, Inc. is an authoring and
25            corporate training organization specializing in
26            programming languages, Internet and web technology,
27            iPhone and Android app development, and object
28            technology education.</p>
29
30        <h1>Clients</h1>
31        <p class = "special"> The company's clients include many
32            <em>Fortune 1000 companies</em>, government agencies,
33            branches of the military and business organizations.</p>
34    </body>
35  </html>
```

Fig. 4.3 | Embedded style sheet. (Part 2 of 3.)

# Embedded Style Sheets



Fig. 4.3 | Embedded style sheet. (Part 3 of 3.)

- ***The style Element and MIME Types***
  - ▶ Styles that are placed in a style element use selectors to apply style elements throughout the entire document
  - ▶ style element type attribute specifies the MIME type (the specific encoding format) of the style sheet. Style sheets use text/css.
  - ▶ Figure 4.4 lists common MIME types used in this book.

MIME type	Description
text/css	CSS documents
image/png	PNG images
text/javascript	JavaScript markup
text/plain	Plain text
image/jpeg	JPEG image
text/html	HTML markup

Fig. 4.4 | A few common MIME types.

# Embedded Style Sheets

- The style sheet's body declares the **CSS rules** for the style sheet.
- To achieve the separation between the CSS3 code and the HTML5 that it styles, we'll use a **CSS selector** to specify the elements that will be styled according to a rule.
- An **em element** indicates that its contents should be *emphasized*.
- Each rule body in a style sheet is enclosed in curly braces ({ and }).
- **font-weight property specifies the “boldness” of text. Possible values are:**
  - Bold
  - normal (the default)
  - bolder (bolder than bold text)
  - lighter (lighter than normal text)
  - Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., 100, 200, ..., 900). Text specified as normal is equivalent to 400, and bold text is equivalent to 700

# Embedded Style Sheets

- **Style Classes**
  - ▶ Style-class declarations are preceded by a period (.).
  - ▶ They define styles that can be applied to *any* element.
  - ▶ In this example, class special sets color to purple.
  - ▶ You can also declare id selectors.
  - ▶ If an element in your page has an id, you can declare a selector of the form `#elementId` to specify that element's style.
- ***font-family Property***
  - font-family property specifies the name of the font to use.
    - ❖ Generic font families allow authors to specify a type of font instead of a specific font, in case a browser does not support a specific font.

# Embedded Style Sheets

Generic font families	Examples
serif	times new roman, georgia
sans-serif	arial, verdana, futura
cursive	script
fantasy	critter
monospace	courier, fixedsys

Fig. 4.5 | Generic font families.



- ***font-size Property***
  - ▶ font-size property specifies the size used to render the font.
  - ▶ You can specify a point size or a relative value such as xx-small, x-small, small, smaller, medium, large, larger, x-large and xx-large.
  - ▶ Relative font-size values are preferred over points, because an author does not know the specific measurements of each client's display.
  - ▶ Relative values permit more flexible viewing of web pages.
    - For example, users can change font sizes the browser displays for readability.

# Embedded Style Sheets

- ***Applying a Style Class***
  - ▶ In many cases, the styles applied to an element (the **parent** or **ancestor element**) also apply to the element's *nested elements* (**child** or **descendant elements**).
  - ▶ Multiple values of one property can be set or inherited on the same element, so the browser must reduce them to one value for that property per element before they're rendered.
  - ▶ We discuss the rules for resolving these conflicts in the next section.

# Conflicting Styles

- Styles may be defined by a **user**, an **author** or a **user agent**.
  - Styles **cascade** (and hence the term “Cascading Style Sheets”), or **flow** together, such that the ultimate appearance of elements on a page results from combining styles defined in several ways.
  - Styles defined by the user take precedence over styles defined by the user agent.
  - Styles defined by authors take precedence over styles defined by the user.
- ▶ Most styles defined for parent elements are also **inherited** by child (nested) elements.
- **text-decoration** property applies decorations to text in an element
  - Underline 
  - Overline 
  - line-through 
  - blink 

# Conflicting Styles

- ▶ Figure 4.3 contains an example of inheritance in which a child em element inherits the font-size property from its parent p element.
- ▶ However, in Fig. 4.3, the child em element has a color property that conflicts with (i.e., has a different value than) the color property of its parent p element.
- ▶ Properties defined for child and descendant elements have a higher specificity than properties defined for parent and ancestor elements.
- ▶ Conflicts are resolved in favor of properties with a higher specificity, so the child's styles take precedence.
- ▶ Figure 4.6 illustrates examples of inheritance and specificity.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.6: advanced.html --&gt;
4  <!-- Inheritance in style sheets. --&gt;
5  &lt;html&gt;
6      &lt;head&gt;
7          &lt;meta charset = "utf-8"&gt;
8          &lt;title&gt;More Styles&lt;/title&gt;
9          &lt;style type = "text/css"&gt;
10             body { font-family: arial, helvetica, sans-serif; }
11             a.nodec { text-decoration: none; }
12             a:hover { text-decoration: underline; }
13             li em { font-weight: bold; }
14             h1, em { text-decoration: underline; }
15             ul { margin-left: 20px; }
16             ul ul { font-size: .8em; }
17
18         &lt;/style&gt;
19     &lt;/head&gt;</pre>
```

Fig. 4.6 | Inheritance in style sheets. (Part 1 of 4.)

```
19     <body>
20         <h1>Shopping list for Monday:</h1>
21
22         <ul>
23             <li>Milk</li>
24             <li>Bread
25                 <ul>
26                     <li>white bread</li>
27                     <li>Rye bread</li>
28                     <li>Whole wheat bread</li>
29                 </ul>
30             </li>
31             <li>Carrots</li>
32             <li>Yogurt</li>
33             <li>Pizza <em>with mushrooms</em></li>
34         </ul>
35
36         <p><em>Go to the</em>
37             <a class = "nodec" href = "http://www.deitel.com">
38                 Grocery store</a>
39             </p>
40         </body>
41     </html>
```

Fig. 4.6 | Inheritance in style sheets. (Part 2 of 4.)

# Conflicting Styles

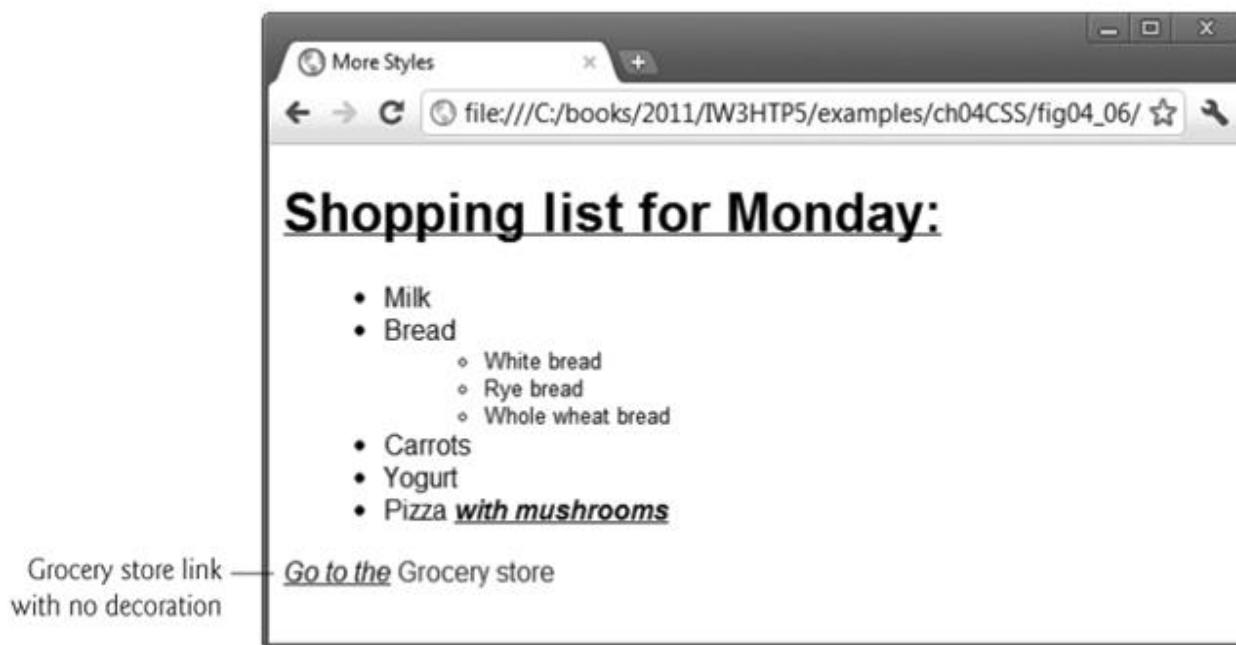


Fig. 4.6 | Inheritance in style sheets. (Part 3 of 4.)

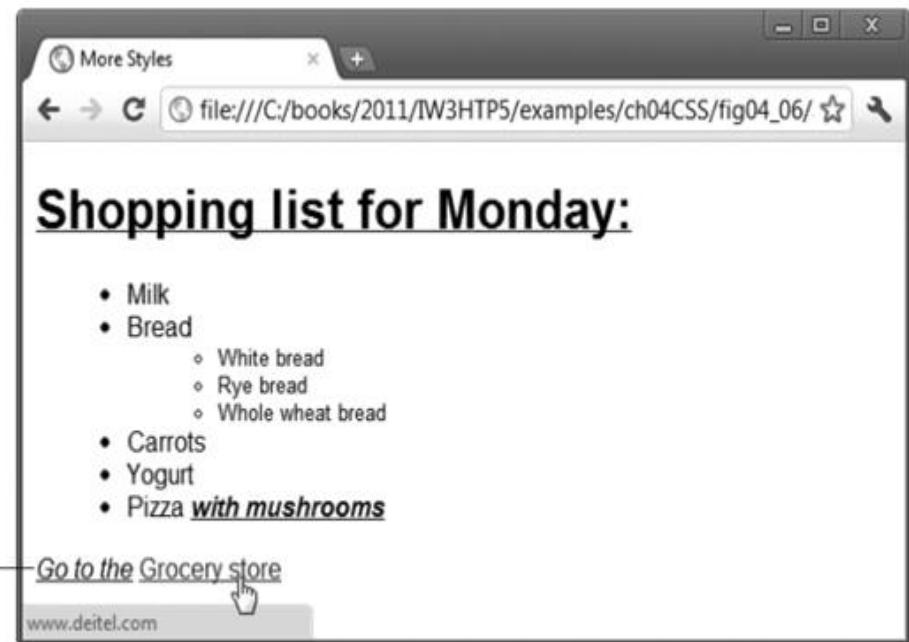


Fig. 4.6 | Inheritance in style sheets. (Part 4 of 4.)



## Portability Tip 4.1

To ensure that your style sheets work in various web browsers, test them on many client web browsers, and use the W3C CSS Validator.

# Conflicting Styles

- Pseudoclasses give you access to content that's not declared in the document.
- hover pseudoclass is activated when the user moves the mouse cursor over an element.
- **Relative length measurements:**
  - px (pixels – size varies depending on screen resolution)
  - em (usually the height of a font's uppercase M)
  - ex (usually the height of a font's lowercase x)
  - Percentages (of the font's default size)
- **Absolute-length measurements** (units that do not vary in size):
  - in (inches)
  - cm (centimeters)
  - mm (millimeters)
  - pt (points; 1 pt = 1/72 in)
  - pc (picas; 1 pc = 12 pt)



## Common Programming Error 4.1

Including a space before or after the colon separating a pseudo-class from the name of the element to which it's applied prevents the pseudo-class from being applied properly.

# The font-size Property

Text Values	Em Units	Px Units	Pt Units	Percentage
xx-small	.5 em	8 px	6 pt	50% <i>(12px)</i>
x-small	.60 em	11 px	8 pt	60%
small	.75 em	13 px	10 pt	75%
medium	1 em	16 px	12 pt	100%
large	1.15 em	18 px	13.5 pt	110%
x-large	1.5 em	24 px	18 pt	150%
xx-large	2 em	30 px	24 pt	200% <i>(36px)</i>

PIXELS	PERCENTAGES	EMS																										
<b>TWELVE PIXEL SCALE</b> <table border="1"> <tr><td>h1</td><td>24px</td></tr> <tr><td>h2</td><td>18px</td></tr> <tr><td>h3</td><td>14px</td></tr> <tr><td>body</td><td>12px</td></tr> </table>	h1	24px	h2	18px	h3	14px	body	12px	<b>SIXTEEN PIXEL SCALE</b> <table border="1"> <tr><td>h1</td><td>32px</td></tr> <tr><td>h2</td><td>24px</td></tr> <tr><td>h3</td><td>18px</td></tr> <tr><td>body</td><td>16px</td></tr> </table>	h1	32px	h2	24px	h3	18px	body	16px	<table border="1"> <tr><td>h1</td><td>1.5em</td></tr> <tr><td>h2</td><td>1.3em</td></tr> <tr><td>h3</td><td>1.17em</td></tr> <tr><td>body</td><td>100%</td></tr> <tr><td>p</td><td>0.75em</td></tr> </table>	h1	1.5em	h2	1.3em	h3	1.17em	body	100%	p	0.75em
h1	24px																											
h2	18px																											
h3	14px																											
body	12px																											
h1	32px																											
h2	24px																											
h3	18px																											
body	16px																											
h1	1.5em																											
h2	1.3em																											
h3	1.17em																											
body	100%																											
p	0.75em																											
		<table border="1"> <tr><td>h1</td><td>2em</td></tr> <tr><td>h2</td><td>1.5em</td></tr> <tr><td>h3</td><td>1.125em</td></tr> <tr><td>body</td><td>100%</td></tr> <tr><td>p</td><td>1em</td></tr> </table>	h1	2em	h2	1.5em	h3	1.125em	body	100%	p	1em																
h1	2em																											
h2	1.5em																											
h3	1.125em																											
body	100%																											
p	1em																											



## Good Programming Practice 4.1

Whenever possible, use relative-length measurements. If you use absolute-length measurements, your document may not scale well on some client browsers (e.g., smartphones).

# Linking External Style Sheets

- ▶ External style sheets are separate documents that contain only CSS rules.
- ▶ Help create a uniform look for a website
  - Separate pages can all use the same styles.
  - Modifying a single style-sheet file makes changes to styles across an entire website (or to a portion of one).
- ▶ When changes to the styles are required, you need to modify only a single CSS file to make style changes across *all* the pages that use those styles. This concept is sometimes known as **skinning**.
- ▶ Figure 4.7 presents an external style sheet.
- ▶ **CSS comments** may be placed in any type of CSS code (i.e., inline styles, embedded style sheets and external style sheets) and always start with /\* and end with \*/.

```
1  /* Fig. 4.7: styles.css */
2  /* External style sheet */
3  body      { font-family: arial, helvetica, sans-serif; }
4  a.nodec   { text-decoration: none; }
5  a:hover   { text-decoration: underline; }
6  li em    { font-weight: bold; }
7  h1, em   { text-decoration: underline; }
8  ul        { margin-left: 20px; }
9  ul ul    { font-size: .8em; }
```

**Fig. 4.7 | External style sheet.**

# Linking External Style Sheets

- ▶ Figure 4.8 contains an HTML5 document that references the external style sheet.
- ▶ link element
  - Uses rel attribute to specify a relationship between two documents
  - rel attribute declares the linked document to be a stylesheet for the document
- ▶ type attribute specifies the MIME type of the related document
- ▶ href attribute provides the URL for the document containing the style sheet

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.8: external.html -->
4  <!-- Linking an external style sheet. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Linking External Style Sheets</title>
9          <link rel = "stylesheet" type = "text/css"
10             href = "styles.css">
11      </head>
12      <body>
13          <h1>Shopping list for <em>Monday</em>:</h1>
14
15          <ul>
16              <li>Milk</li>
17              <li>Bread
18                  <ul>
19                      <li>white bread</li>
20                      <li>Rye bread</li>
21                      <li>Whole wheat bread</li>
22                  </ul>
23              </li>
24
25          <li>Carrots</li>
26          <li>Yogurt</li>
27          <li>Pizza <em>with mushrooms</em></li>
28      </ul>
29
30          <p><em>Go to the</em>
31              <a class = "nodec" href = "http://www.deitel.com">
32                  Grocery store</a>
33          </p>
34      </body>
35  </html>
```

Fig. 4.8 | Linking an external style sheet. (Part 1 of 4.)

Fig. 4.8 | Linking an external style sheet. (Part 2 of 4.)

# Linking External Style Sheets

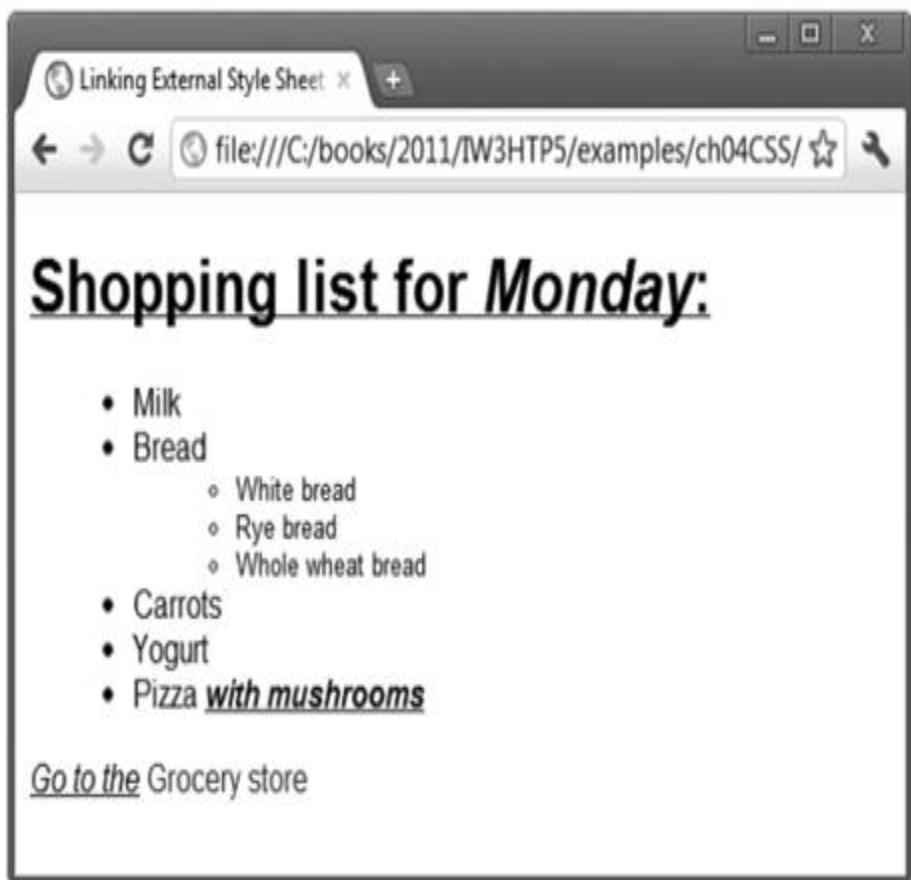


Fig. 4.8 | Linking an external style sheet. (Part 3 of 4.)

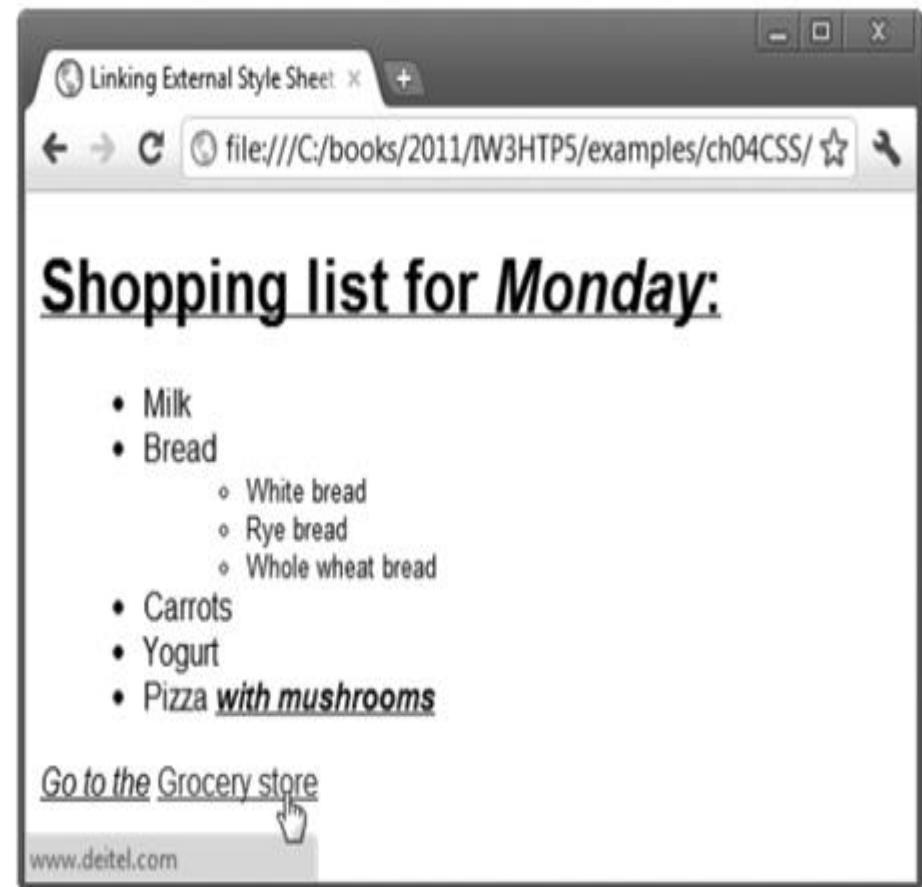


Fig. 4.8 | Linking an external style sheet. (Part 4 of 4.)

# Positioning Elements: Absolute Positioning, z-index

figue,  
text  
overlap

- **CSS position property**

- Allows absolute positioning, which provides greater control over where on a page elements reside.
- Normally, elements are positioned on the page in the order in which they appear in the HTML5 document.
- Specifying an element's position as **absolute** removes it from the normal flow of elements on the page and positions it according to distance from the top, left, right or bottom margin of its parent element.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 4.9: positioning.html -->
4 <!-- Absolute positioning of elements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Absolute Positioning</title>
9     <style type = "text/css">
10    .background_image { position: absolute;
11      top: 0px;
12      left: 0px;
13      z-index: 1; }
14    .foreground_image { position: absolute;
15      top: 25px;
16      left: 100px;
17      z-index: 2; }
18    .text
19      { position: absolute;
20      top: 25px;
21      left: 100px;
22      z-index: 3;
23      font-size: 20pt;
24      font-family: tahoma, geneva, sans-serif; }
25
26  </style>
27 </head>
```

z-index 高, 是上方

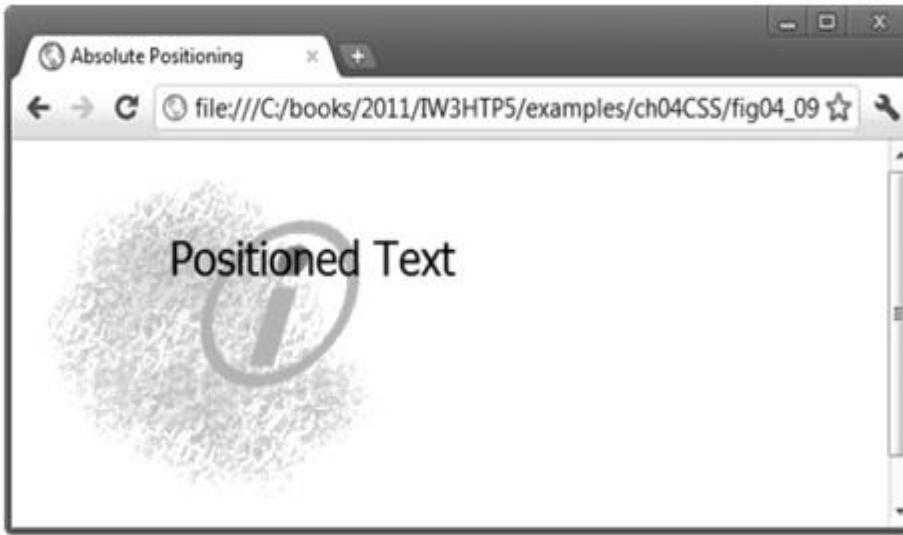
```
26 <body>
27   <p><img src = "background_image.png" class = "background_image"
28     alt = "First positioned image" /></p>
29
30   <p><img src = "foreground_image.png" class = "foreground_image"
31     alt = "Second positioned image" /></p>
32
33   <p class = "text">Positioned Text</p>
34 </body>
35 </html>
```



Fig. 4.9 | Absolute positioning of elements. (Part 1 of 3.)

Fig. 4.9 | Absolute positioning of elements. (Part 2 of 3.)

# Positioning Elements: Absolute Positioning, z-index



**Fig. 4.9 |** Absolute positioning of elements. (Part 3 of 3.)

- The z-index property allows a developer to layer overlapping elements.
- Elements that have higher z-index values are displayed in front of elements with lower z-index values.

# Positioning Elements: Relative Positioning, span

- Figure 4.10 demonstrates relative positioning, in which elements are positioned *relative to other elements*.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 4.10: positioning2.html -->
4 <!-- Relative positioning of elements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Relative Positioning</title>
9     <style type = "text/css">
10    p           { font-size: 1.3em;
11          font-family: verdana, arial, sans-serif; }
12    span         { color: red;
13          font-size: .6em;
14          height: 1em; }
15    .super       { position: relative;
16          top: -1ex; }
17    .sub         { position: relative;
18          bottom: -1ex; }
19    .shiftleft   { position: relative;
20          left: -1ex; }
21    .shiftright { position: relative;
22          right: -1ex; }
23
24   </style>
25   </head>
```

each Henry  
declare  
September  
Joy

```
25   <body>
26     <p>The text at the end of this sentence
27       <span class = "super">is in superscript</span>.</p>
28
29     <p>The text at the end of this sentence
30       <span class = "sub">is in subscript</span>.</p>
31
32     <p>The text at the end of this sentence
33       <span class = "shiftleft">is shifted left</span>.</p>
34
35     <p>The text at the end of this sentence
36       <span class = "shiftright">is shifted right</span>.</p>
37
38   </body>
39 </html>
```

Fig. 4.10 | Relative positioning of elements. (Part 2 of 3.)

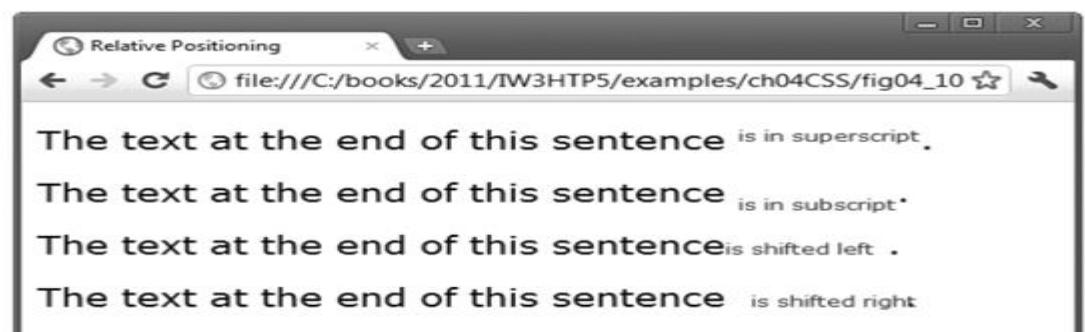


Fig. 4.10 | Relative positioning of elements. (Part 1 of 3.)

Fig. 4.10 | Relative positioning of elements. (Part 3 of 3.)

# Positioning Elements: Relative Positioning, span

- **Inline and Block-Level Elements**

- ▶ **Inline-level elements**

- Do not change the flow of the document

- Examples:

- img

- a

- em

- strong

- span

Grouping element

Does not apply any formatting to its contents

Creates a container for CSS rules or id attributes to be applied to a section

*format*

*(flow fn)*

Facebook

This text is in span

- **Block-level elements**

- Displayed on their own line

- Have virtual boxes around them

- Examples:

- P

- all headings (h1 through h6)

- div

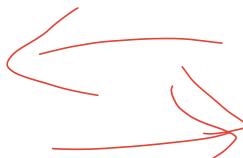
A grouping element like span



This text is in paragraph.

This text is in div

# Backgrounds

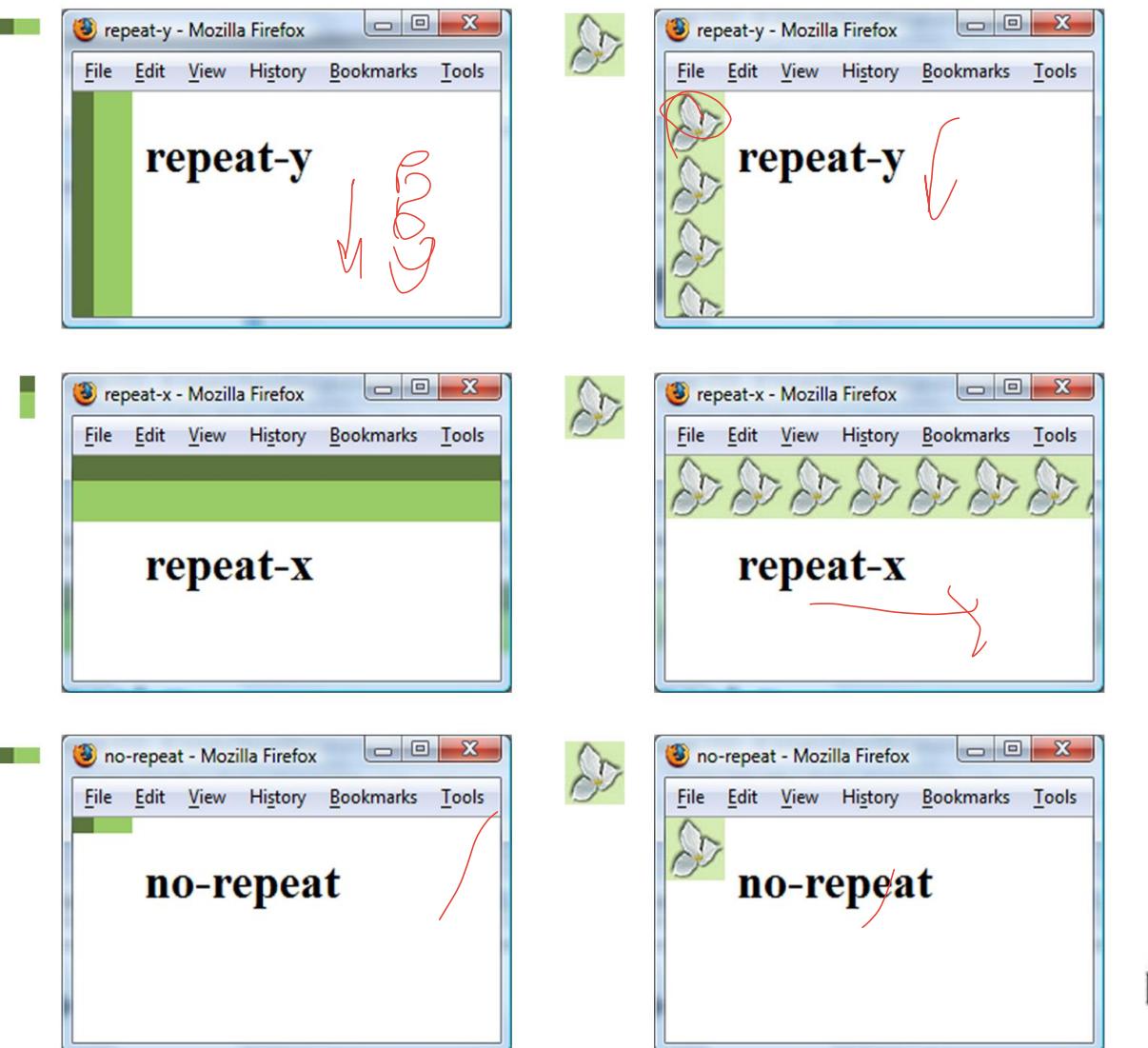


↳ *Brvgh*      ↳ *Omws Edy*

- CSS can control the backgrounds of block-level elements by adding:
  - Colors
  - Images
- Figure 4.11 adds a corporate logo to the bottom-right corner of the document. This logo stays fixed in the corner even when the user scrolls up or down the screen.

Image Type	File Extension	Com-pression	Trans-parency	Animation	Colors	Progressive Display
Graphic Interchange Format (GIF)	.gif	Lossless	Yes	Yes	256	Inter-lacing
Joint Photographic Experts Group (JPEG)	.jpg or .jpeg	Lossy	No	No	Millions	Progressive
Portable Network Graphic (PNG)	.png	Lossless	Yes (multiple levels)	No	Millions	Inter-lacing

# Backgrounds



```
1 <!DOCTYPE html>
2
3 <!-- Fig. 4.11: background.html -->
4 <!-- Adding background images and indentation -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Background Images</title>
9     <style type = "text/css">
10       body { background-image: url(logo.png);
11         background-position: bottom right;
12         background-repeat: no-repeat;
13         background-attachment: fixed;
14         background-color: lightgrey; }
15       p { font-size: 18pt;
16         color: Darkblue;
17         text-indent: 1em;
18         font-family: arial, sans-serif; }
19         .dark { font-weight: bold; }
20     </style>
21   </head>
```

A handwritten note in red ink is present on the right side of the code, pointing to the line "background-repeat: no-repeat;" with the text "Xh yes" written next to it.

Fig. 4.11 | Adding background images and indentation. (Part 1 of 3.)

# Backgrounds

```
22 <body>  
23   <p>  
24     This example uses the background-image,  
25     background-position and background-attachment  
26     styles to place the <span class = "dark">Deitel  
27     & Associates, Inc.</span> logo in the  
28     bottom-right corner of the page. Notice how the logo  
29     stays in the proper position when you resize the  
30     browser window. The background-color fills in where  
31     there is no image.  
32   </p>  
33 </body>  
34 </html>
```

Fig. 4.11 | Adding background images and indentation. (Part 2 of 3.)



Fig. 4.11 | Adding background images and indentation. (Part 3 of 3.)

# Backgrounds

- ***background-image Property***
  - ▶ Specifies the URL of the image, in the format url(fileLocation)
- ***background-position Property***
  - ▶ Places the image on the page using the values top, bottom, center, left and right individually or in combination for vertical and horizontal positioning. You can also position by using lengths. First value is the horizontal position and the second value is the vertical.
- ***background-repeat Property***
  - ▶ background-repeat property controls the **tiling** of the background image
    - Setting the tiling to no-repeat displays one copy of the background image on screen
    - Setting to repeat (the default) tiles the image vertically and horizontally
    - Setting to repeat-x tiles the image only horizontally
    - Setting to repeat-y tile the image only vertically

# Backgrounds

- **background-attachment: fixed Property**
  - Fixes the image in the position specified by background-position.
  - Scrolling the browser window will not move the image from its set position.
  - The default value, scroll, moves the image as the user scrolls the window
- **text-indent Property**
  - Indents the first line of text in the element by the specified amount
- **font-style Property**
  - Allows you to set text to none, italic or oblique

# Element Dimensions

- Figure 4.12 demonstrates how to set the dimensions of elements.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 4.12: width.html -->
4 <!-- Element dimensions and text alignment. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Box Dimensions</title>
9     <style type = "text/css">
10    p { background-color: lightskyblue;
11        margin-bottom: .5em;
12        font-family: arial, helvetica, sans-serif; }
13   </style>
14 </head>
15 <body>
16   <p style = "width: 20%">Here is some
17     text that goes in a box which is
18     set to stretch across twenty percent
19     of the width of the screen.</p>
20
21   <p style = "width: 80%; text-align: center">
22     Here is some CENTERED text that goes in a box
23     which is set to stretch across eighty percent of
24     the width of the screen.</p>
```

Fig. 4.12 | Element dimensions and text alignment. (Part 1 of 3.)

```
25
26   <p style = "width: 20%; height: 150px; overflow: scroll">
27     This box is only twenty percent of
28     the width and has a fixed height.
29     What do we do if it overflows? Set the
30     overflow property to scroll!</p>
31   </body>
32 </html>
```

Fig. 4.12 | Element dimensions and text alignment. (Part 2 of 3.)

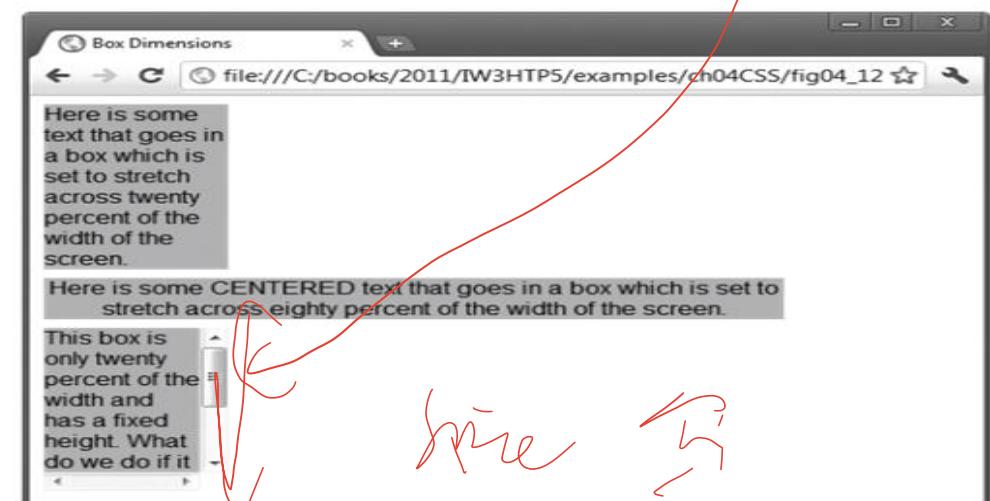


Fig. 4.12 | Element dimensions and text alignment. (Part 3 of 3.)

# Element Dimensions

- ***Specifying the width and height of an Element***
  - ▶ Dimensions of elements on a page can be set with CSS by using properties height and width
    - Their values can be relative or absolute
- ***text-align Property***
  - ▶ Text in an element can be centered using text-align: center; other values for the text-align property are left and right
- ***overflow Property and Scroll Bars***
  - ▶ Problem with setting both vertical and horizontal dimensions of an element
    - Content might sometimes exceed the set boundaries, in which case the element must be made large enough for all the content to fit
    - Can set the overflow property to scroll, which adds scroll bars if the text overflows the boundaries set for it

# Box Model and Text Flow

- ▶ Block-level HTML5 elements have a virtual box drawn around them based on the box model
- ▶ When the browser renders an element using the box model, the content is surrounded by padding, a margin and a border.
- ▶ Padding
  - The padding property determines the distance between the content inside an element and the edge of the element
  - Padding can be set for each side of the box by using padding-top, padding-right, padding-left and padding-bottom
- ▶ Margin
  - Determines the distance between the element's edge and any outside text
  - Margins for individual sides of an element can be specified by using margin-top, margin-right, margin-left and margin-bottom
- ▶ Border
  - The border is controlled using the properties:
    - **border-width**
      - May be set to any of the CSS lengths or to the predefined value of thin, medium or thick
    - **border-color**
      - Sets the color used for the border
    - **border-style**
      - Options are: none, hidden, dotted, dashed, solid, double, groove, ridge, inset and outset

# Box Model and Text Flow

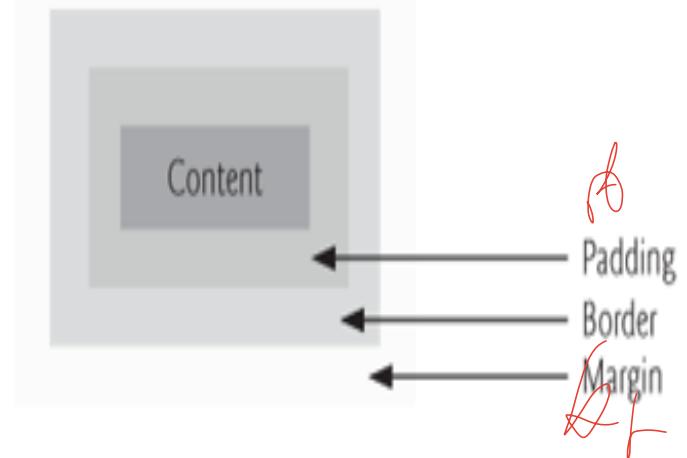


Fig. 4.13 | Box model for block-level elements.



- CSS controls the border using three properties: border-width, border-color and border-style.
- We illustrate these properties in Fig. 4.14.

# Box Model and Text Flow

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 4.14: borders.html -->
4 <!-- Borders of block-level elements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Borders</title>
9     <style type = "text/css">
10    div { text-align: center;
11      width: 50%;
12      position: relative;
13      left: 25%;
14      border-width: 6px; }
15    .thick { border-width: thick; }
16    .medium { border-width: medium; }
17    .thin { border-width: thin; }
18    .solid { border-style: solid; }
19    .double { border-style: double; }
20    .groove { border-style: groove; }
21    .ridge { border-style: ridge; }
22    .dotted { border-style: dotted; }
23    .inset { border-style: inset; }
24    .outset { border-style: outset; }
```

Fig. 4.14 | Borders of block-level elements. (Part 1 of 3.)

```
25   .dashed { border-style: dashed; }
26   .red { border-color: red; }
27   .blue { border-color: blue; }
28 </style>
29 </head>
30 <body>
31   <div class = "solid">Solid border</div><hr>
32   <div class = "double">Double border</div><hr>
33   <div class = "groove">Groove border</div><hr>
34   <div class = "ridge">Ridge border</div><hr>
35   <div class = "dotted">Dotted border</div><hr>
36   <div class = "inset">Inset border</div><hr>
37   <div class = "thick dashed">Thick dashed border</div><hr>
38   <div class = "thin red solid">Thin red solid border</div><hr>
39   <div class = "medium blue outset">Medium blue outset border</div>
40 </body>
41 </html>
```

Fig. 4.14 | Borders of block-level elements. (Part 2 of 3.)

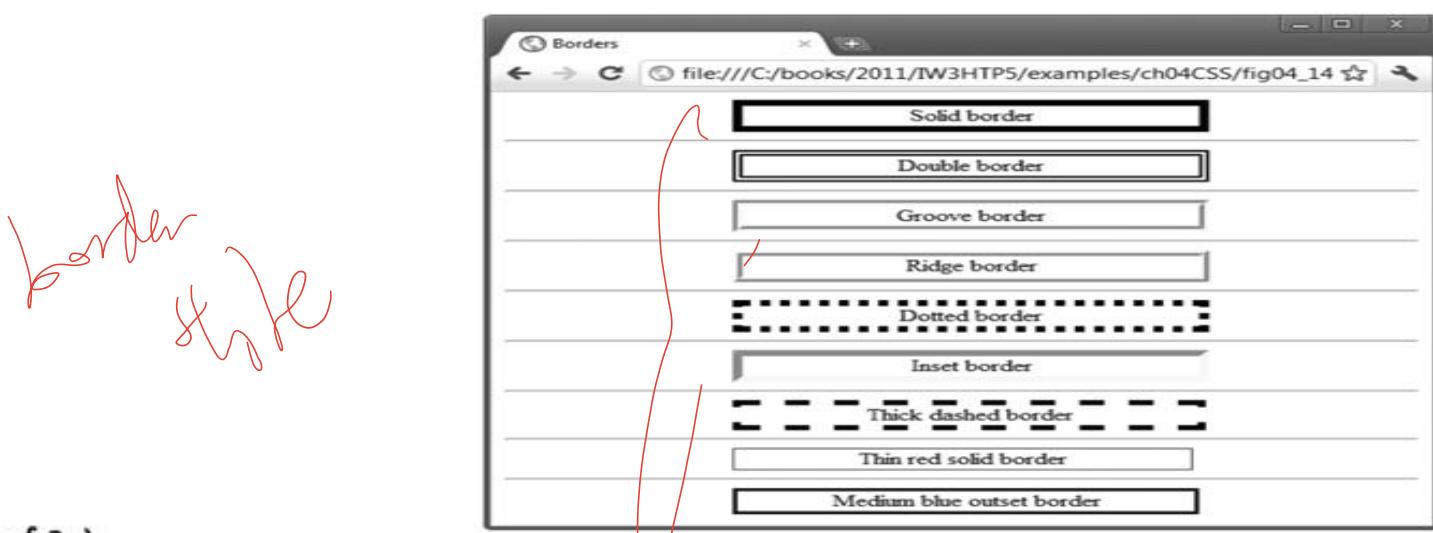


Fig. 4.14 | Borders of block-level elements. (Part 3 of 3.)

# Box Model and Text Flow

- **Floating Elements**
  - ▶ Floating allows you to move an element to one side of the screen; other content in the document then *flows around* the floated element.
  - ▶ Figure 4.15 demonstrates how floating elements and the box model can be used to control the layout of an entire page.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.15: floating.html -->
4  <!-- Floating elements. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Flowing Text Around Floating Elements</title>
9          <style type = "text/css">
10             header { background-color: skyblue;
11                 text-align: center;
12                 font-family: arial, helvetica, sans-serif;
13                 padding: .2em; }
14             p { text-align: justify;
15                 font-family: verdana, geneva, sans-serif;
16                 margin: .5em; }
17             h1 { margin-top: 0px; }
```

Fig. 4.15 | Floating elements. (Part 1 of 4.)

```
18         .floated { background-color: lightgrey;
19             font-size: 1.5em;
20             font-family: arial, helvetica, sans-serif;
21             padding: .2em;
22             margin-left: .5em;
23             margin-bottom: .5em;
24             float: right;
25             text-align: right;
26             width: 50%; }
27
28         section { border: 1px solid skyblue; }
29     </style>
30 </head>
31 <body>
32     <header><img src = "deitel.png" alt = "Deitel" /></header>
33     <section>
34         <h1 class = "floated">Corporate Training and Authoring</h1>
35         <p>Deitel & Associates, Inc. is an internationally
36             recognized corporate training and authoring organization
37             specializing in programming languages, Internet/web
38             technology, iPhone and Android app development and
39             object technology education. The company provides courses
40             on Java, C++, C#, Visual Basic, C, Internet and web
41             programming, Object Technology and iPhone and Android
42             app development.</p>
43     </section>
```

Fig. 4.15 | Floating elements. (Part 2 of 4.)

# Box Model and Text Flow

```
43 <section>
44   <h1 class = "floated">Programming Books and Videos</h1>
45   <p>Through its publishing
46     partnership with Pearson, Deitel & Associates,
47     Inc. publishes leading-edge programming textbooks,
48     professional books and interactive web-based and DVD
49     LiveLessons video courses.</p>
50   </section>
51 </body>
52 </html>
```

Fig. 4.15 | Floating elements. (Part 3 of 4.)

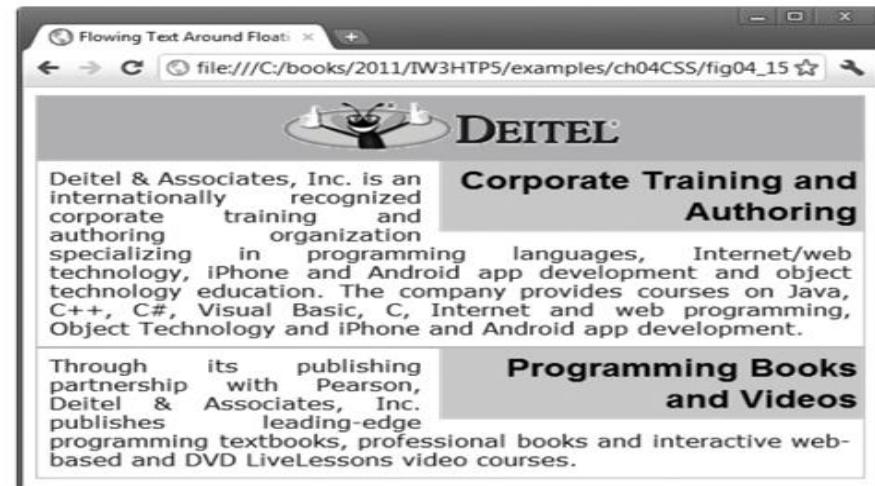


Fig. 4.15 | Floating elements. (Part 4 of 4.)

- ***margin and padding Properties***
  - ▶ The **margin** property sets the space between the outside of an element's border and all other content on the page.
  - ▶ The **padding** property determines the distance between the content inside an element and the inside of the element's border.
  - ▶ Margins for individual sides of an element can be specified by using the properties **margin-top**, **margin-right**, **margin-left** and **margin-bottom**.
  - ▶ Padding can be specified in the same way, using **padding-top**, **padding-right**, **padding-left** and **padding-bottom**.

# Margin and padding Properties

- All the margin properties can have the following values:
  - **auto** - the browser calculates the margin
  - **length** - specifies a margin in px, pt, cm, etc.
  - **%** - specifies a margin in % of the width of the containing element
  - **inherit** - specifies that the margin should be inherited from the parent element

```
margin: 25px 50px 75px 100px; top  
margin is 25px  
right margin is 50px  
bottom margin is 75px  
left margin is 100px
```

# Media Types and Media Queries

- **CSS media types**
  - allow you to decide what a page should look like depending on the kind of media being used to display the page
  - Most common media type for a web page is the screen media type, which is a standard computer screen
- A block of styles that applies to all media types is declared by `@media all` and enclosed in curly braces
- To create a block of styles that apply to a single media type such as print, use `@media print` and enclose the style rules in curly braces
- Other media types in CSS include:
  - **handheld**
    - Designed for mobile Internet devices
  - **braille**
    - For machines that can read or print web pages in braille
  - **speech**
    - Allow the programmer to give a speech-synthesizing web browser more information about the content of the web page
  - **print**
    - Affects a web page's appearance when it is printed

# Media Types and Media Queries

- Figure 4.16 gives a simple classic example that applies one set of styles when the document is viewed on all media (including screens) other than a printer, and another when the document is printed.
- To see the difference, look at the screen captures below the paragraph or use the Print Preview feature in your browser if it has one.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.16: mediatypes.html -->
4  <!-- CSS media types. -->
5  <html>
6    <head>
7      <meta charset = "utf-8">
8      <title>Media Types</title>
9      <style type = "text/css">
10        @media all
11        {
12          body { background-color: steelblue; }
13          h1 { font-family: verdana, helvetica, sans-serif;
14              color: palegreen; }
15          p { font-size: 12pt;
16              color: white;
17              font-family: arial, sans-serif; }
18        } /* End @media all declaration. */
```

Fig. 4.16 | CSS media types. (Part 1 of 4.)

```
19
20  @media print
21  {
22    body { background-color: white; }
23    h1 { color: seagreen; }
24    p { font-size: 14pt;
25        color: steelblue;
26        font-family: "times new roman", times, serif; }
27    } /* End @media print declaration. */
28  </style>
29  </head>
30  <body>
31    <h1>CSS Media Types Example</h1>
32
33    <p>
34      This example uses CSS media types to vary how the page
35      appears in print and how it appears on any other media.
36      This text will appear in one font on the screen and a
37      different font on paper or in a print preview. To see
38      the difference in Internet Explorer, go to the Print
39      menu and select Print Preview. In Firefox, select Print
40      Preview from the File menu.
41    </p>
42  </body>
43  </html>
```

Fig. 4.16 | CSS media types. (Part 2 of 4.)

# Media Types and Media Queries

a) Background color appears on the screen.



Fig. 4.16 | CSS media types. (Part 3 of 4.)

## Look-and-Feel Observation 4.1

Pages with dark background colors and light text use a lot of ink and may be difficult to read when printed, especially on a black-and-white printer. Use the **print** media type to avoid this.



b) Background color is set to white for the **print** media type.

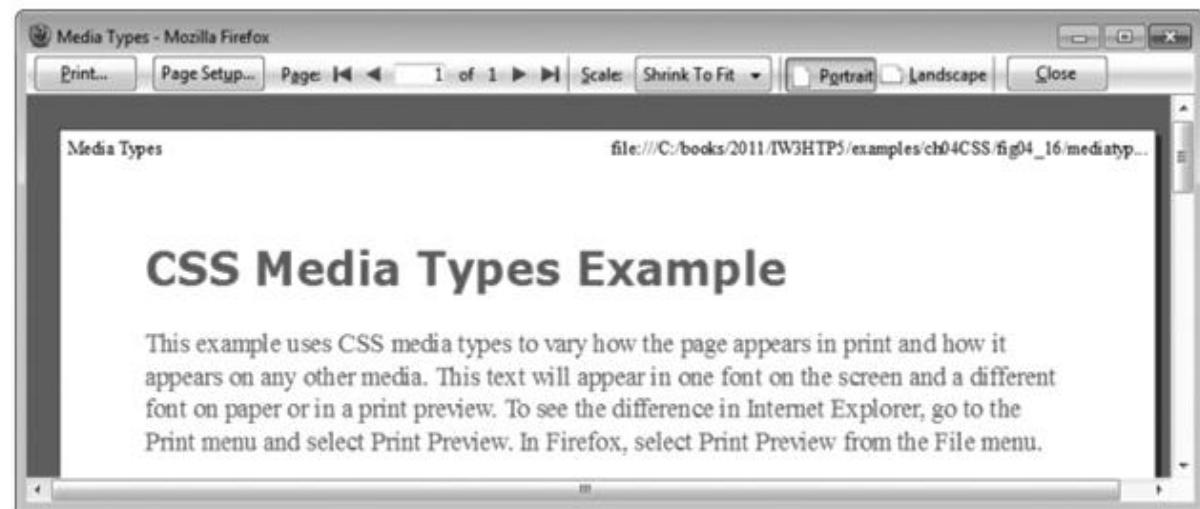


Fig. 4.16 | CSS media types. (Part 4 of 4.)

## Look-and-Feel Observation 4.2

In general, sans-serif fonts look better on a screen, while serif fonts look better on paper. The **print** media type allows your web page to display a sans-serif font on a screen and change to a serif font when it's printed.



# Media Types and Media Queries

- **Media Queries**
  - ▶ Allow you to format your content to specific output devices.
  - ▶ Include a media type and expressions that check the media features of the output device.
  - ▶ Common media features include:
    - **width**—the width of the part of the screen on which the document is rendered, including any scrollbars  
`@media screen and (min-width: 480px) {}`
    - **height**—the height of the part of the screen on which the document is rendered, including any scrollbars
    - **device-width**—the width of the screen of the output device  
`@media screen and (max-device-width: 480px)`
    - **device-height**—the height of the screen of the output device
    - **orientation**—if the height is greater than the width, orientation is portrait, and if the width is greater than the height, orientation is landscape
    - **aspect-ratio**—the ratio of width to height `@media (aspect ratio: 1/1)`
    - **device-aspect-ratio**—the ratio of device-width to device-height

# Drop-Down Menus

- **:hover pseudoclass**
  - used to apply styles to an element when the mouse cursor is over it
- **display property**
  - allows a programmer to decide if an element is displayed as a **block element**, **inline element**, or is **not rendered at all (none)**

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 4.17: dropdown.html --&gt;
4  &lt;!-- CSS drop-down menu. --&gt;
5  &lt;html&gt;
6    &lt;head&gt;
7      &lt;meta charset = "utf-8"&gt;
8      &lt;title&gt;
9        Drop-Down Menu
10     &lt;/title&gt;
11     &lt;style type = "text/css"&gt;
12       body
13         { font-family: arial, sans-serif }
14       nav
15         { font-weight: bold;
16           color: white;
17           border: 2px solid royalblue;
18           text-align: center;
19           width: 10em;
20           background-color: royalblue; }
21       nav ul
22         { display: none;
23           list-style: none;
24           margin: 0;
25           padding: 0; }
26       nav:hover ul
27         { display: block }</pre>
```

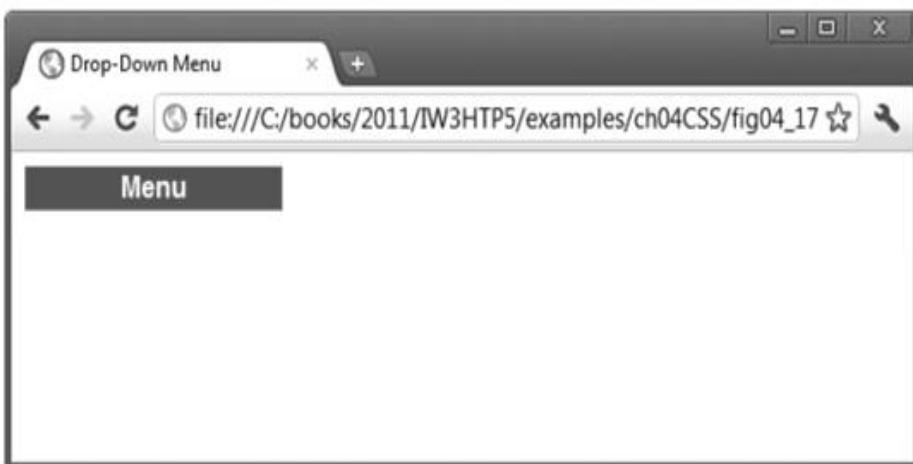
Fig. 4.17 | CSS drop-down menu. (Part 1 of 5.)

```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
nav ul li
  { border-top: 2px solid royalblue;
    background-color: white;
    width: 10em;
    color: black; }
nav ul li:hover
  { background-color: powderblue; }
  { text-decoration: none; }
a
</style>
</head>
<body>
<nav>Menu
  <ul>
    <li><a href = "#">Home</a></li>
    <li><a href = "#">News</a></li>
    <li><a href = "#">Articles</a></li>
    <li><a href = "#">Blog</a></li>
    <li><a href = "#">Contact</a></li>
  </ul>
</nav>
</body>
</html>
```

Fig. 4.17 | CSS drop-down menu. (Part 2 of 5.)

# Drop-Down Menus

a) A collapsed menu



b) A drop-down menu is displayed when the mouse cursor is hovered over **Menu**

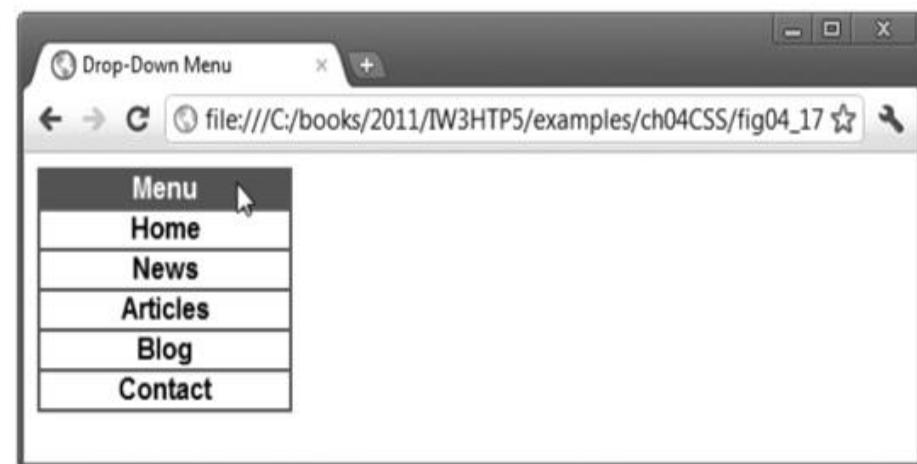


Fig. 4.17 | CSS drop-down menu. (Part 3 of 5.)

c) Hovering the mouse cursor over a menu link highlights the link



Fig. 4.17 | CSS drop-down menu. (Part 5 of 5.)

Fig. 4.17 | CSS drop-down menu. (Part 4 of 5.)

# UCCD2323 Front-End Web Development



## Chapter 2 Cascading Style Sheet (CSS) Part 2.

# Topic Outline

Introduction  
Text Shadows  
Rounded Corners  
Color  
Box Shadows  
Linear Gradients; Introducing Vendor Prefixes  
Radial Gradients  
(Optional: WebKit Only) Text Stroke  
Multiple Background Images  
(Optional: WebKit Only) Reflections  
Image Borders  
Animation; Selectors  
Transitions and Transformations  
5.13.1 transition and transform Properties  
5.12.2 Skew  
5.12.3 Transitioning Between Images  
Downloading Web Fonts and the @font-face Rule  
Flexible Box Layout Module and :nth-child Selectors  
Multicolumn Layout  
Media Queries  
Web Resources

# Objectives

In this chapter you'll:

- Add text shadows and text-stroke effects.
- Create rounded corners.
- Add shadows to elements.
- Create linear and radial gradients, and reflections.
- Create animations, transitions and transformations.
- Use multiple background images and image borders.
- Create a multicolumn layout.
- Use flexible box model layout and :nth-child selectors.
- Use the @font-face rule to specify fonts for a web page.
- Use RGBA and HSLA colors.
- Use vendor prefixes.
- Use media queries to customize content to fit various screen sizes.

## Text Shadows

- ▶ The CSS3 **text-shadow property** makes it easy to add a **text shadow effect** to *any* text (Fig. 5.1).
- ▶ The **text-shadow property** has four values which represent:
  - **Horizontal offset of the shadow**—the number of pixels that the text-shadow will appear to the *left* or the *right* of the text. A *negative value moves the text-shadow to the left*; a *positive value moves it to the right*.
  - **Vertical offset of the shadow**—the number of pixels that the text-shadow will be shifted *up* or *down* from the text. A *negative value moves the shadow up*, whereas a *positive value moves it down*.
  - **blur radius**—the blur (in pixels) of the shadow. A blur-radius of 0px would result in a shadow with a sharp edge (no blur). The greater the value, the greater the blurring of the edges.
  - **color**—determines the color of the text-shadow.

```
h1 { text-shadow: -4px 4px 6px dimgrey; }
```

# Text Shadows

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.1: textshadow.html -->
4 <!-- Text shadow in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Text Shadow</title>
9     <style type = "text/css">
10    h1
11    {
12      text-shadow: -4px 4px 6px dimgray; /* add shadow */
13      font-size: 400%; /* increasing the font size */
14    }
15  </style>
16 </head>
17 <body>
18   <h1>Text Shadow</h1>
19 </body>
20 </html>
```

Fig. 5.1 | Text shadow in CSS3. (Part 1 of 2.)



Fig. 5.1 | Text shadow in CSS3. (Part 2 of 2.)

## Rounded Corners

- ▶ The **border-radius** property allows you to add rounded corners to an element (Fig. 5.2).
- ▶ For the first rectangle, we set the border-radius to 15px. This adds slightly rounded corners to the rectangle.
- ▶ For the second rectangle, we increase the border-radius to 50px, making the left and right sides completely round.
- ▶ Any border-radius value greater than half of the shortest side length produces a completely round end.
- ▶ You can also specify the radius for each corner with **border-top-left-radius**, **border-top-right-radius**, **border-bottom-left-radius** and **border-bottom-right-radius**.

## Rounded Corners

- ▶ **Four values – border-radius: 15px 50px 30px 5px;** (first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner):
- ▶ **Three values – border-radius: 15px 50px 30px;** (first value applies to top-left corner, second value applies to top-right and bottom-left corners, and third value applies to bottom-right corner):
- ▶ **Two values – border-radius: 15px 50px;** (first value applies to top-left and bottom-right corners, and the second value applies to top-right and bottom-left corners):
- ▶ **One value – border-radius: 15px;** (the value applies to all four corners, which are rounded equally)

# Rounded Corners

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.2: roundedcorners.html -->
4 <!-- Using border-radius to add rounded corners to two elements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Rounded Corners</title>
9     <style type = "text/css">
10    div
11    {
12      border: 3px solid navy;
13      padding: 5px 20px;
14      background: lightcyan;
15      width: 200px;
16      text-align: center;
17      border-radius: 15px; /* adding rounded corners */
18      margin-bottom: 20px;
19    }
```

Fig. 5.2 | Using border-radius to add rounded corners to two elements. (Part 1 of 3.)

*Styling under*

```
20 #round2
21 {
22   border: 3px solid navy;
23   padding: 5px 20px;
24   background: lightcyan;
25   width: 200px;
26   text-align: center;
27   border-radius: 50px; /* increasing border-radius */
28 }
29 </style>
30 </head>
31 <body>
32   <div>The border-radius property adds rounded corners
33   to an element.</div>
34   <div id = "round2">Increasing the border-radius rounds the corners
35   of the element more.</div>
36 </body>
37 </html>
```

Fig. 5.2 | Using border-radius to add rounded corners to two elements. (Part 2 of 3.)

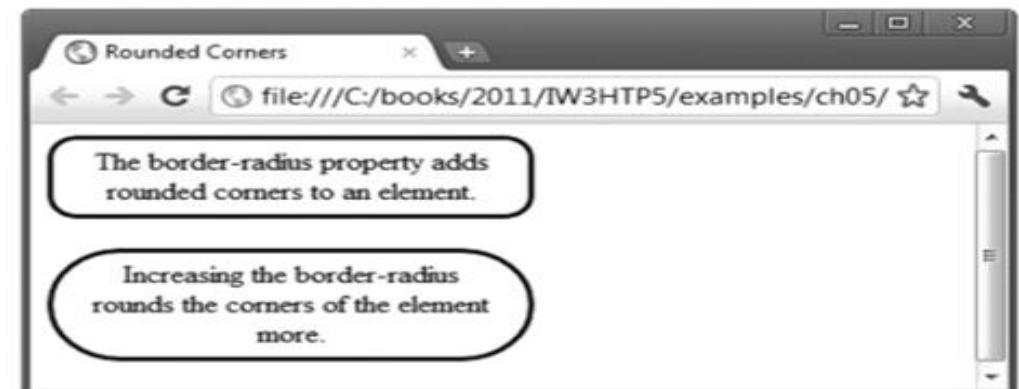


Fig. 5.2 | Using border-radius to add rounded corners to two elements. (Part 3 of 3.)

# Color

- ▶ CSS3 allows you to express color in several ways in addition to standard color names (such as Aqua) or hexadecimal RGB values (such as #00FFFF for Aqua).
- ▶ **RGB (Red, Green, Blue)** or **RGBA (Red, Green, Blue, Alpha)** gives you greater control over the exact colors in your web pages.
- ▶ The value for each color—red, green and blue—can range from 0 to 255.
- ▶ The *alpha* value—which represents *opacity*—can be any value in the range 0.0 (fully transparent) through 1.0 (fully opaque).
- ▶ If you were to set the background color as follows:
  - `background: rgba(255, 0, 0, 0.5);`the resulting color would be a half-opaque red.
- ▶ There are over 140 HTML color names, whereas there are 16,777,216 different RGB colors ( $256 \times 256 \times 256$ ) and varying opacities of each.

```
body {background-color: rgba(201,  
76, 76, 0.3);} //RGBA
```

```
body {background-color: rgb(201,  
76, 76);} //RGB
```

# Color

- ▶ CSS3 also allows you to express color using HSL (hue, saturation, lightness) or HSLA (hue, saturation, lightness, alpha) values.
- ▶ The *hue* is a color or shade expressed as a value from 0 to 359 representing the degrees on a color wheel (a wheel is 360 degrees).
- ▶ The colors on the wheel progress in the order of the colors of the rainbow—red, orange, yellow, green, blue, indigo and violet.
- ▶ The value for red, which is at the beginning of the wheel, is 0.
- ▶ Green hues have values around 120 and blue hues have values around 240.
- ▶ A hue value of 359, which is just left of 0 on the wheel, would result in a red hue.
- ▶ The *saturation*—the intensity of the hue—is expressed as a percentage, where 100% is fully saturated (the full color) and 0% is gray.

```
body {background-color: hsl(89,  
43%, 51%);}
```

```
body {background-color: hsl(89,  
43%, 51%, 1);}
```

Hue color picker:  
<http://hslpicker.com/>

# Convert from Decimal to Hexadecimal

- To convert from decimal to hexadecimal, you need to divide the decimal number by 16 repeatedly and write down the remainders in reverse order. You can use the table below to find the hexadecimal equivalent of the remainders from 0 to 15.

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6

Decimal	Hexadecimal
7	7
8	8
9	9
10	A
11	B
12	C
13	D

Decimal	Hexadecimal
14	E
15	F

- Convert 495 to Hexadecimal**
- Divide 495 by 16 and get a quotient of 30 and a remainder of 15. Write down F for the remainder.
- Divide 30 by 16 and get a quotient of 1 and a remainder of 14. Write down E for the remainder.
- Divide 1 by 16 and get a quotient of 0 and a remainder of 1. Write down 1 for the remainder.
- Since the quotient is 0, stop the process and write the remainders in reverse order. The hexadecimal number is 1EF.

## Color

- ▶ **Lightness**—the intensity of light or luminance of the hue—is also expressed as a percentage.
- ▶ A lightness of 50% is the actual hue.
- ▶ If you *decrease* the amount of light to 0%, the color appears completely dark (black).
- ▶ If you *increase* the amount of light to 100%, the color appears completely light (white).
- ▶ For example, if you wanted to use an `hsla` value to get the same color red as in our example of an `rgba` value, you would set the `background` property as follows:
  - `background: hsla(0, 100%, 50%, 0.5);`

## Box Shadows

- ▶ You can shadow *any* block-level element in CSS3.
- ▶ Figure 5.3 shows you how to create a **box shadow**.
  - **Horizontal offset of the shadow**—the number of pixels that the box-shadow will appear to the left or the right of the box. A *positive* value moves the box-shadow to the *right*.
  - **Vertical offset of the shadow**—the number of pixels the box-shadow will be shifted up or down from the box. A *positive* value moves the box-shadow *down*.
  - **Blur radius**—A blur-radius of 0px would result in a shadow with a sharp edge (no blur). The greater the value, the more the edges of the shadow are blurred.
  - **Color**—the box-shadow's color.

box-shadow: 25px 25px 50px dimgrey;

# Box Shadows

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.3: boxshadow.html -->
4 <!-- Creating box-shadow effects. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Box Shadow</title>
9     <style type = "text/css">
10    div
11    {
12      width: 200px;
13      height: 200px;
14      background-color: plum;
15      box-shadow: 25px 25px 50px dimgrey;
16      float: left;
17      margin-right: 120px;
18      margin-top: 40px;
19    }
20
```

Fig. 5.3 | Creating box-shadow effects. (Part 1 of 3.)

```
21 #box2
22 {
23   width: 200px;
24   height: 200px;
25   background-color: plum;
26   box-shadow: -25px -25px 50px dimgrey;
27 }
28 h2
29 {
30   text-align: center;
31 }
32 </style>
33 </head>
34 <body>
35 <div><h2>Box Shadow Bottom and Right</h2></div>
36 <div id = "box2"><h2>Box Shadow Top and Left</h2></div>
37 </body>
38 </html>
```

Fig. 5.3 | Creating box-shadow effects. (Part 2 of 3.)

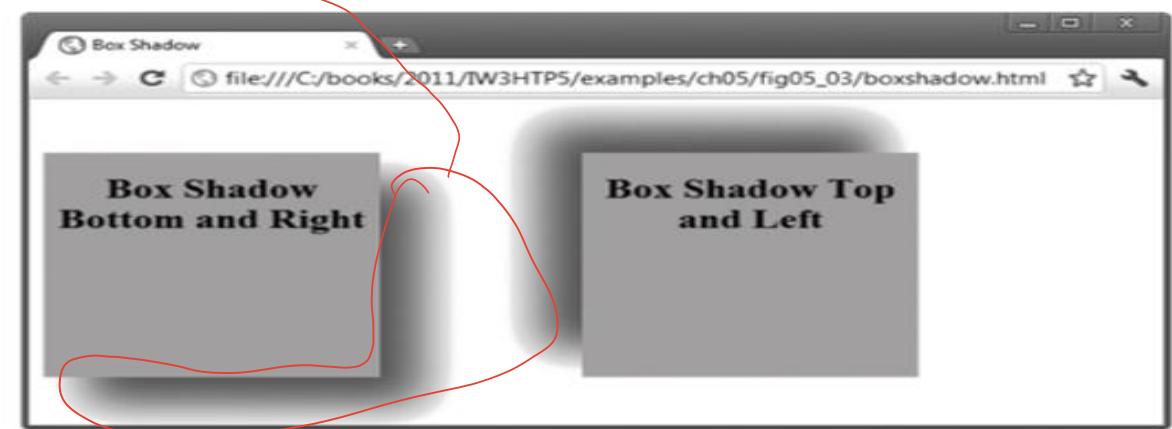


Fig. 5.3 | Creating box-shadow effects. (Part 3 of 3.)

## Linear Gradients; Introducing Vendor Prefixes

- ▶ **Linear gradients** are a type of image that gradually transitions from one color to the next horizontally, vertically or diagonally.
- ▶ You can transition between as many colors as you like and specify the points at which to change colors, called **color-stops**, represented in pixels or percentages along the *gradient line*—the angle at which the gradient extends.
- ▶ *You can use gradients in any property that accepts an image.*

# Linear Gradients; Introducing Vendor Prefixes

## *Creating Linear Gradients*

- ▶ In Fig. 5.4, we create three linear gradients—*vertical*, *horizontal* and *diagonal*—in separate rectangles.
- ▶ The background property for each of the three linear gradient styles (vertical, horizontal and diagonal) is defined multiple times in each style—once for WebKit-based browsers, once for Mozilla Firefox and once using the standard CSS3 syntax for linear gradients.
- ▶ This occurs frequently when working with CSS3, because many of its features are not yet finalized.
- ▶ Many of the browsers have gone ahead and begun implementing these features so you can use them now.

# Linear Gradients; Introducing Vendor Prefixes

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.4: lineargradient.html -->
4 <!-- Linear gradients in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Linear Gradient</title>
9     <style type = "text/css">
10    div
11    {
12      width: 200px;
13      height: 200px;
14      border: 3px solid navy;
15      padding: 5px 20px;
16      text-align: center;
17      background: -webkit-gradient(
18        linear, center top, center bottom,
19        color-stop(15%, white), color-stop(50%, lightsteelblue),
20        color-stop(75%, navy) );
21      background: -moz-linear-gradient(
22        top center, white 15%, lightsteelblue 50%, navy 75% );
```

copying it

```
23
24   background: linear-gradient(
25     to bottom, white 15%, lightsteelblue 50%, navy 75% );
26   float: left;
27   margin-right: 15px;
28 }
29 #horizontal
30 {
31   width: 200px;
32   height: 200px;
33   border: 3px solid orange;
34   padding: 5px 20px;
35   text-align: center;
36   background: -webkit-gradient(
37     linear, left top, right top,
38     color-stop(15%, white), color-stop(50%, yellow),
39     color-stop(75%, orange) );
40   background: -moz-linear-gradient(
41     left, white 15%, yellow 50%, orange 75% );
42   background: linear-gradient(
43     90deg, white 15%, yellow 50%, orange 75% );
44 }
```

Fig. 5.4 | Linear gradients in CSS3. (Part 1 of 4.)

Fig. 5.4 | Linear gradients in CSS3. (Part 2 of 4.)

# Linear Gradients; Introducing Vendor Prefixes

```
45      #angle
46      {
47          width: 200px;
48          height: 200px;
49          border: 3px solid Purple;
50          padding: 5px 20px;
51          text-align: center;
52          background: -webkit-gradient(
53              linear, left top, right bottom,
54              color-stop(15%, white), color-stop(50%, plum),
55              color-stop(75%, purple) );
56          background: -moz-linear-gradient(
57              top left, white 15%, plum 50%, purple 75% );
58          background: linear-gradient(
59              45deg, white 15%, plum 50%, purple 75% );
60      }
61      </style>
62  </head>
63  <body>
64      <div><h2>Vertical Linear Gradient</h2></div>
65      <div id = "horizontal"><h2>Horizontal Linear Gradient</h2></div>
66      <div id = "angle"><h2>Diagonal Linear Gradient</h2></div>
67  </body>
68  </html>
```

Fig. 5.4 | Linear gradients in CSS3. (Part 3 of 4.)

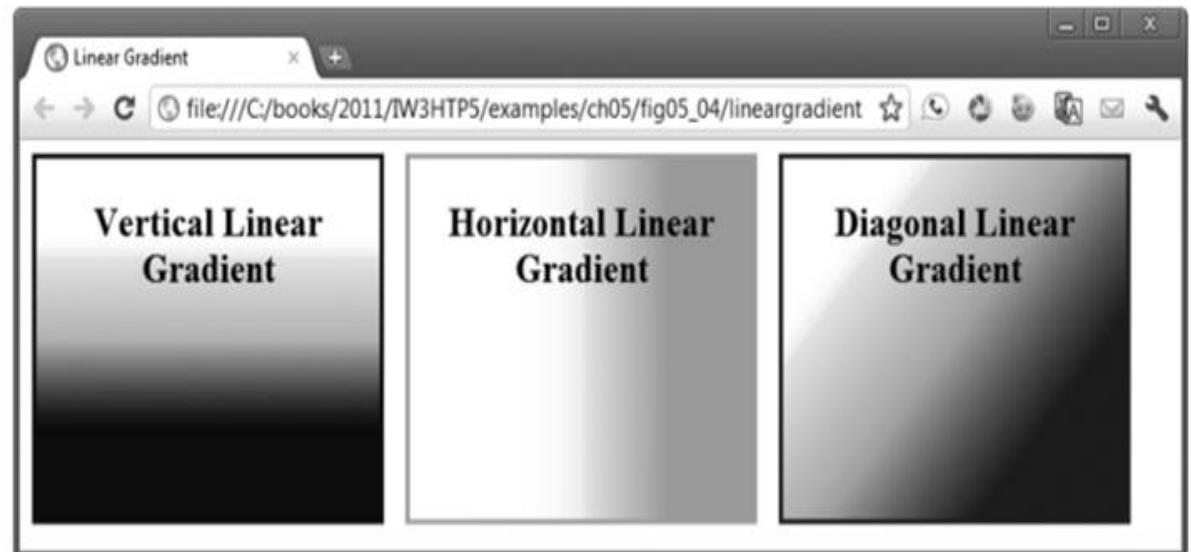


Fig. 5.4 | Linear gradients in CSS3. (Part 4 of 4.)

# Linear Gradients; Introducing Vendor Prefixes

## ***WebKit Vertical Linear Gradient***

- ▶ Begin with the background property.
- ▶ The linear gradient syntax for WebKit differs slightly from that for Firefox.
- ▶ For WebKit browsers, we use `-webkit-gradient`.
- ▶ We then specify the **type of gradient (linear)** and the **direction of the linear gradient**, from center top to center bottom.
- ▶ Next, we specify the **color-stops** for the linear gradient.
- ▶ Within each color-stop are two values—the first is the **location of the stop** and the **second is the color**.
- ▶ You can use as many color-stops as you like.

# **Linear Gradients; Introducing Vendor Prefixes**

## ***Mozilla Vertical Linear Gradient***

- ▶ For Mozilla browsers, we use -moz-linear-gradient.
- ▶ We specify the gradient-line (top center), which is the direction of the gradient.
- ▶ After the gradient-line we specify each color and color-stop.

## ***Standard Vertical Linear Gradient***

- ▶ The standard CSS3 syntax for linear gradients is also slightly different.
- ▶ First, we specify the linear-gradient.
- ▶ We include the values for the gradient—the direction of the gradient (top), followed by each color and color-stop.

# Linear Gradients; Introducing Vendor Prefixes

## ***Horizontal Linear Gradient***

- ▶ Next we create a rectangle with a *horizontal* (left-to-right) gradient.
- ▶ For WebKit, the direction of the gradient is left top to right top, followed by the colors and color-stops.
- ▶ For Mozilla, we specify the gradient-line (left), followed by the colors and color-stops.
- ▶ The standard CSS3 syntax begins with the direction (left), indicating that the gradient changes from left to right, followed by the colors and color-stops.
- ▶ The direction can also be specified in degrees, with 0 degrees straight up and positive degrees progressing clockwise.



# Linear Gradients; Introducing Vendor Prefixes

## *Diagonal Linear Gradient*

- ▶ In the third rectangle we create a *diagonal* linear gradient.
- ▶ For WebKit, the direction of the gradient is left top to right bottom, followed by the colors and color-stops.
- ▶ For Mozilla, we specify the gradient-line (top left), followed by the colors and color-stops.
- ▶ The standard CSS3 syntax begins with the direction (135deg), indicating that the gradient changes at a 45-degree angle, followed by the colors and color-stops.

# Linear Gradients; Introducing Vendor Prefixes

## *Vendor Prefixes*

- ▶ **Vendor prefixes** (Fig. 5.5) and are used for properties that are still being finalized in the CSS specification but have already been implemented in various browsers.

Vendor prefix	Browsers
-ms-	Internet Explorer
-moz-	Mozilla-based browsers, including Firefox
-o-	Opera and Opera Mobile
-webkit-	WebKit-based browsers, including Google Chrome, Safari (and Safari on the iPhone) and Android

Fig. 5.5 | Vendor prefixes.

# Linear Gradients; Introducing Vendor Prefixes

- ▶ Prefixes are *not* available for every browser or for every property.
- ▶ If we remove the prefixed versions of the linear gradient styles in this example, the gradients will *not* appear when the page is rendered in a WebKit-based browser or Firefox.
- ▶ If you run this program in browsers that don't support gradients yet, the gradients will *not* appear.
- ▶ It's good practice to include the multiple prefixes when they're available so that your pages render properly in the various browsers.
- ▶ As the CSS3 features are finalized and incorporated fully into the browsers, the prefixes will become unnecessary.
- ▶ Many of the new CSS3 features have not yet been implemented in Internet Explorer.
- ▶ When using vendor prefixes in styles, always place them *before* the nonprefixed version.
- ▶ The last version of the style that a given browser supports takes precedence and the browser will use it. By listing the standard non-prefixed version last, the browser will use the standard version over the prefixed version when the standard version is supported.

## Radial Gradients

- ▶ Radial gradients are similar to linear gradients, but the color changes gradually from an inner point (the *start*) to an outer circle (the *end*) (Fig. 5.6).
- ▶ In this example, the **radial-gradient property** has three values.
- ▶ The first is the position of the start of the radial gradient—in this case, the center of the rectangle. Other possible values for the position include top, bottom, left and right.
- ▶ The second value is the *start color* (yellow), and the third is the *end color* (red).
- ▶ The resulting effect is a box with a yellow center that gradually changes to red in a circle around the starting position.
- ▶ In this case, notice that other than the vendor prefixes, the syntax of the gradient is identical for WebKit browsers, Mozilla and the standard CSS3 radial-gradient.

# Radial Gradients

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.6: radialgradient.html -->
4 <!-- Radial gradients in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Radial Gradient</title>
9     <style type = "text/css">
10    div
11    {
12      width: 200px;
13      height: 200px;
14      padding: 5px;
15      text-align: center;
16      background: -webkit-radial-gradient(center, yellow, red);
17      background: -moz-radial-gradient(center, yellow, red);
18      background: radial-gradient(center, yellow, red);
19    }
20  </style>
21 </head>
```

Fig. 5.6 | Radial gradients in CSS3. (Part 1 of 2.)

```
22   <body>
23     <div><h2>Radial Gradient</h2></div>
24   </body>
25 </html>
```

Radial gradient begins with yellow in the center, then changes to red in a circle as it moves toward the edges of the box

F  
M }  
} style  
for browser SO

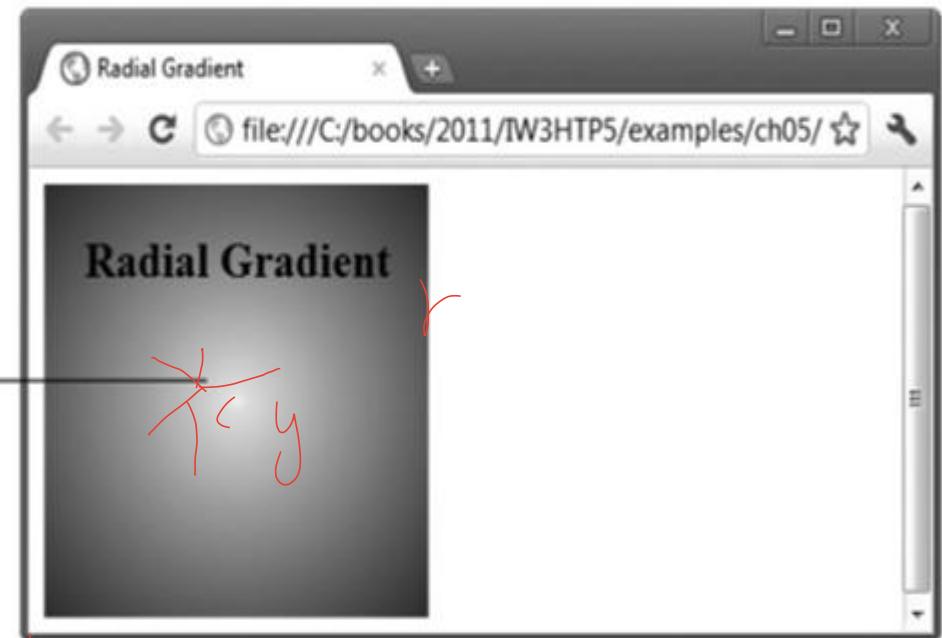


Fig. 5.6 | Radial gradients in CSS3. (Part 2 of 2.)

## (Optional: WebKit Only) Text Stroke

- ▶ The `-webkit-text-stroke` property is a nonstandard property for WebKit-based browsers that allows you to add an outline (text stroke) around text.
- ▶ Four of the seven browsers we use in this book are WebKit based—Safari and Chrome on the desktop and the mobile browsers in iOS and Android.
- ▶ In Fig. 5.7, we set the color of the h1 text to LightCyan.
- ▶ We add a `-webkit-text-stroke` with two values—the outline *thickness* (2px) and the *color* of the text stroke (black).
- ▶ We used the `font-size` 500% here so you could see the outline better.

## (Optional: WebKit Only) Text Stroke

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.7: textstroke.html -->
4 <!-- Text stroke in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Text Stroke</title>
9     <style type = "text/css">
10    h1
11    {
12      color: lightcyan;
13      -webkit-text-stroke: 2px black; /* vendor prefix */
14      font-size: 500%; /* increasing the font size */
15    }
16  </style>
17  </head>
18  <body>
19    <h1>Text Stroke</h1>
20  </body>
21 </html>
```

Fig. 5.7 | A text-stroke rendered in Chrome. (Part 1 of 2.)

color is lightcyan  
text-stroke is 2px black

Inside color font

2px



Fig. 5.7 | A text-stroke rendered in Chrome. (Part 2 of 2.)

# Multiple Background Images

- ▶ CSS3 allows you to add **multiple background images** to an element (Fig. 5.8).
- ▶ The style begins by adding two **background-images**—`logo.png` and `ocean.png`.
- ▶ Next, we specify each **image's placement** using property **background-position**.
- ▶ The **comma-separated list of values** matches the order of the comma-separated list of images in the **background-image** property.
- ▶ The first value—`bottom right`—places the first image, `logo.png`, in the bottom-right corner of the **background** in the **border-box**.
- ▶ The last value—`100% center`—centers the entire second image, `ocean.png`, in the **content-box** so that it appears behind the content and stretches to fill the **content-box**.
- ▶ The **background-origin** determines where each image is placed using the box model we discussed in Fig. 4.13.
- ▶ The first image (`logo.png`) is in the outermost **border-box**, and the second image (`ocean.png`) is in the innermost **content-box**.

# Multiple Background Images

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.8: multiplebackgrounds.html -->
4 <!-- Multiple background images in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Multiple Backgrounds</title>
9     <style type = "text/css">
10    div.background
11    {
12      background-image: url(logo.png), url(ocean.png);
13      background-position: bottom right, 100% center;
14      background-origin: border-box, content-box;
15      background-repeat: no-repeat, repeat;
16    }
17    div.content
18    {
19      padding: 10px 15px;
20      color: white;
21      font-size: 150%;
22    }
23  </style>
24 </head>
```

2 nelye

```
25 <body>
26   <div class = "background">
27     <div class = "content">
28       <p>Deitel & Associates, Inc., is an internationally recognized
29         authoring and corporate training organization. The company
30         offers instructor-led courses delivered at client sites
31         worldwide on programming languages and other software topics
32         such as C++, Visual C++®, C, Java™, C#®, Visual
33         Basic®, Objective-C®, XML®, Python®, JavaScript,
34         Internet and web programming, and Android and iPhone app
35         development.</p>
36     </div></div>
37   </body>
38 </html>
```

Fig. 5.8 | Multiple background images in CSS3. (Part 2 of 3.)



The second image  
(ocean.png) is  
centered behind the  
content and  
stretched as needed  
to fill the

The first image  
(logo.png) is  
placed at the  
bottom right with

Fig. 5.8 | Multiple background images in CSS3. (Part 3 of 3.)

Fig. 5.8 | Multiple background images in CSS3. (Part 1 of 3.)

## Animation; Selectors

- ▶ In Fig. 5.11, we create a simple *animation* of an image that moves in a diamond pattern as it changes opacity.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.11: animation.html -->
4 <!-- Animation in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Animation</title>
9     <style type = "text/css">
10    img
11    {
12      position: relative;
13      -webkit-animation: movingImage linear 10s 1s 2 alternate;
14      -moz-animation: movingImage linear 10s 1s 2 alternate;
15      animation: movingImage linear 10s 2 1s alternate;
16    }
```

Fig. 5.11 | Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 1 of 4.)

```
17   @-webkit-keyframes movingImage
18   {
19     0%   {opacity: 0; left: 50px; top: 0px;}
20     25%  {opacity: 1; left: 0px; top: 50px;}
21     50%  {opacity: 0; left: 50px; top: 100px;}
22     75%  {opacity: 1; left: 100px; top: 50px;}
23     100% {opacity: 0; left: 50px; top: 0px;}
24   }
25   @-moz-keyframes movingImage
26   {
27     0%   {opacity: 0; left: 50px; top: 0px;}
28     25%  {opacity: 1; left: 0px; top: 50px;}
29     50%  {opacity: 0; left: 50px; top: 100px;}
30     75%  {opacity: 1; left: 100px; top: 50px;}
31     100% {opacity: 0; left: 50px; top: 0px;}
32   }
```

Fig. 5.11 | Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 2 of 4.)

# Animation; Selectors

```
33 @keyframes movingImage
34 {
35     0% {opacity: 0; left: 50px; top: 0px;}
36     25% {opacity: 1; left: 0px; top: 50px;}
37     50% {opacity: 0; left: 50px; top: 100px;}
38     75% {opacity: 1; left: 100px; top: 50px;}
39     100% {opacity: 0; left: 50px; top: 0px;}
40 }
41 </style>
42 </head>
43 <body>
44     <img src = "jhtp.png" width = "138" height = "180"
45         alt = "Java How to Program book cover">
46     <div></div>
47 </body>
48 </html>
```

**Fig. 5.11 |** Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 3 of 4.)

The animation starts and ends at the top of the diamond, moving the image in the counterclockwise direction initially. When the animation reaches the top of the diamond, the animation reverses, continuing in the clockwise direction. The animation terminates when the image reaches the top of the diamond for a second time.



**Fig. 5.11 |** Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 4 of 4.)

### *animation Property*

- ▶ The **animation property** allows you to represent several animation properties in a *shorthand* notation, rather than specifying each separately, as in:
  - `animation-name: movingImage;`
  - `animation-timing-function: linear;`
  - `animation-duration: 10s;`
  - `animation-delay: 1s;`
  - `animation-iteration-count: 2;`
  - `animation-direction: alternate;`

## Animation; Selectors

- ▶ In the shorthand notation, the values are listed in the following order:
  - **animation-name**—represents the name of the animation (`movingImage`).
  - **animation-timing-function**—determines how the animation progresses in one cycle of its duration. Possible values include `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`, `cubic-bezier`.
    - `linear` specifies that the animation will move at the same speed from start to finish.
    - The default value, `ease`, starts slowly, increases speed, then ends slowly.
    - `ease-in` starts slowly, then speeds up, whereas the `ease-out` value starts faster, then slows down.
    - `ease-in-out` starts and ends slowly.
    - `cubic-bezier` allows you to customize the timing function with four values between 0 and 1, such as `cubic-bezier(1,0,0,1)`

## Animation; Selectors

- **animation-duration**—specifies the time in seconds (s) or milliseconds (ms) that the animation takes to complete one iteration (10s in this case).
- **animation-delay**—specifies the number of seconds (1s in this case) or milliseconds after the page loads before the animation begins.
- **animation-iteration-count**—specifies the number of times the animation will run. You may use the value infinite to repeat the animation continuously.
- **animation-direction**—specifies the direction in which the animation will run. The value alternate used here specifies that the animation will run in alternating directions—in this case, counterclockwise (as we define with our keyframes), then clockwise. The default value, normal, would run the animation in the same direction for each cycle.
  - ▶ The shorthand animation property cannot be used with the **animation-play-state property**—it must be specified separately.
  - ▶ If you do not include the **animation-play-state**, which specifies whether the animation is paused or running, it defaults to running.

# Animation; Selectors

## *@keyframes Rule and Selectors*

- ▶ The **@keyframes rule** defines the element's properties that will change during the animation, the values to which those properties will change, and when they'll change.
- ▶ The **@keyframes rule** is followed by the name of the animation (`movingImage`) to which the keyframes are applied.
- ▶ CSS **rules** consist of one or more **selectors** followed by a **declaration block** in curly braces ({}).
- ▶ Selectors enable you to apply styles to elements of a particular type or attribute.
- ▶ A declaration block consists of one or more declarations, each of which includes the property name followed by a colon (:), a value and a semicolon (;).
- ▶ In this example, the **@keyframes rule** includes five selectors to represent the points-in-time for our animation.
- ▶ You can break down the animation into as many points as you like.

## Transitions and Transformations

- ▶ With CSS3 **transitions**, you can change an element's style over a specified duration.
- ▶ CSS3 **transformations** allow you to *move*, *rotate*, *scale* and *skew* elements.
- ▶ Transitions are similar in concept to the animations, but transitions allow you to specify only the starting and ending values of the CSS properties being changed.
- ▶ An animation's keyframes enable you to control intermediate states throughout the animation's duration.

## Transitions and Transformations Properties

- ▶ Figure 5.12 uses the **transition** and **transform properties** to scale and rotate an image 360 degrees when the cursor *hovers* over it.
- ▶ We begin by defining the transition. For each property that will change, the transition property specifies the duration of that change.
- ▶ We could specify a comma-separated list of property names that will change and the individual durations over which each property will change. For example:
  - `transition: transform 4s, opacity 2s;`
- ▶ indicates that a transform takes four seconds to apply and the opacity changes over two seconds—thus, the transform will continue for another two seconds after the opacity change completes.
- ▶ In this example, we define the transform only when the user *hovers* the mouse over the image.

# Transitions and Transformations Properties

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.12: transitions.html -->
4 <!-- Transitions in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Transitions</title>
9     <style type = "text/css">
10    img
11    {
12      margin: 80px;
13      -webkit-transition: -webkit-transform 4s;
14      -moz-transition: -moz-transform 4s;
15      -o-transition: -o-transform 4s;
16      transition: transform 4s;
17    }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

Fig. 5.12 | Transitioning an image over a four-second duration and applying rotate and scale transforms. (Part 1 of 3.)

18         
19  
20  
21  
22  
23  
24  
25         
26  
27  
28         
29  
30  
31

```
img:hover
{
  -webkit-transform: rotate(360deg) scale(2, 2);
  -moz-transform: rotate(360deg) scale(2, 2);
  -o-transform: rotate(360deg) scale(2, 2);
  transform: rotate(360deg) scale(2, 2);
}
```

Fig. 5.12 | Transitioning an image over a four-second duration and applying rotate and scale transforms. (Part 2 of 3.)

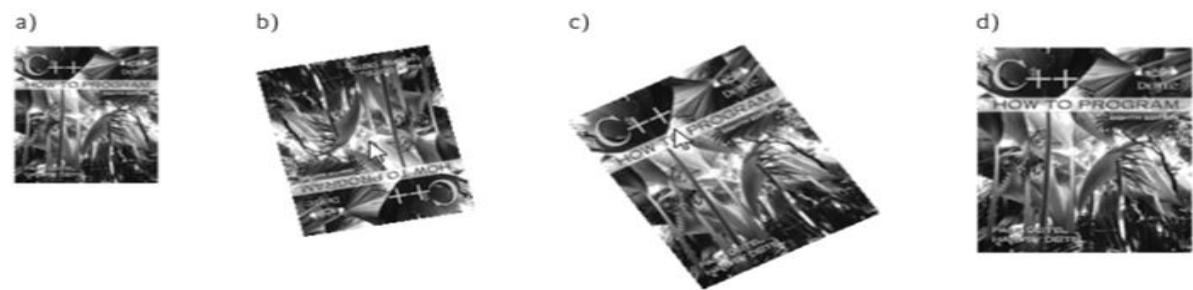


Fig. 5.12 | Transitioning an image over a four-second duration and applying rotate and scale transforms. (Part 3 of 3.)

# Transitions and Transformations Properties

- ▶ The `:hover` pseudo-class formerly worked only for anchor elements but now works with *any* element.
- ▶ We use `:hover` to begin the rotation and scaling of the image.
- ▶ The `transform` property specifies that the image will rotate 360deg and will scale to twice its original width and height when the mouse hovers over the image.
- ▶ The `transform` property uses **transformation functions**, such as `rotate` and `scale`, to perform the transformations.
- ▶ The `rotate` transformation function receives the number of degrees. Negative values cause the element to rotate left.
- ▶ The `scale` transformation function specifies how to scale the width and height. The value 1 represents the original width or original height, so values greater than 1 increase the size and values less than 1 decrease the size.

## Skew

- ▶ CSS3 transformations also allow you to **skew** block-level elements, slanting them at an angle either horizontally (**skewX**) or vertically (**skewY**).
- ▶ We use the **animation** and **transform** properties to skew a rectangle and text horizontally by 45 degrees (Fig. 5.13).
- ▶ First we create a rectangle with a **LightGreen** background, a solid **DarkGreen** border and rounded corners.
- ▶ The **animation** property specifies that the element will skew in a three-second (3s) interval for an infinite duration.
- ▶ The fourth value, **linear**, is the **animation-timing-function**.
- ▶ Next, we use the **@keyframes** rule and selectors to specify the angle of the **skew** transformation at different intervals.

# Skew

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.13: skew.html -->
4 <!-- Skewing and transforming elements in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Skew</title>
9     <style type = "text/css">
10    .skew .textbox
11    {
12      margin-left: 75px;
13      background: lightgreen;
14      height: 100px;
15      width: 200px;
16      padding: 25px 0;
17      text-align: center;
18      font-size: 250%;
19      border: 3px solid DarkGreen;
20      border-radius: 15px;
21      -webkit-animation: skew 3s infinite linear;
22      -moz-animation: skew 3s infinite linear;
23      animation: skew 3s infinite linear;
24    }
```

```
25      @-webkit-keyframes skew
26      {
27        from { -webkit-transform: skewX(0deg); }
28        25% { -webkit-transform: skewX(45deg); }
29        50% { -webkit-transform: skewX(0); }
30        75% { -webkit-transform: skewX(-45deg); }
31        to { -webkit-transform: skewX(0); }
32      }
33      @-moz-keyframes skew
34      {
35        from { -webkit-transform: skewX(0deg); }
36        25% { -webkit-transform: skewX(45deg); }
37        50% { -webkit-transform: skewX(0); }
38        75% { -webkit-transform: skewX(-45deg); }
39        to { -webkit-transform: skewX(0); }
40      }
41      @keyframes skew
42      {
43        from { -webkit-transform: skewX(0deg); }
44        25% { -webkit-transform: skewX(45deg); }
45        50% { -webkit-transform: skewX(0); }
46        75% { -webkit-transform: skewX(-45deg); }
47        to { -webkit-transform: skewX(0); }
48      }
49    </style>
```

Phases ~

Fix

Fig. 5.13 | Skewing and transforming elements in CSS3. (Part 1 of 3.)

Fig. 5.13 | Skewing and transforming elements in CSS3. (Part 2 of 3.)

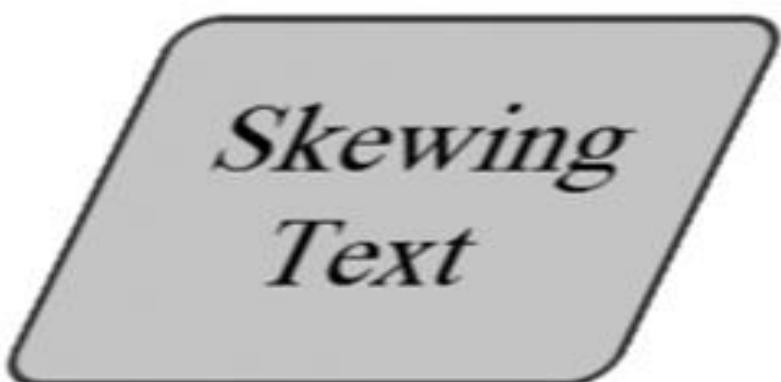
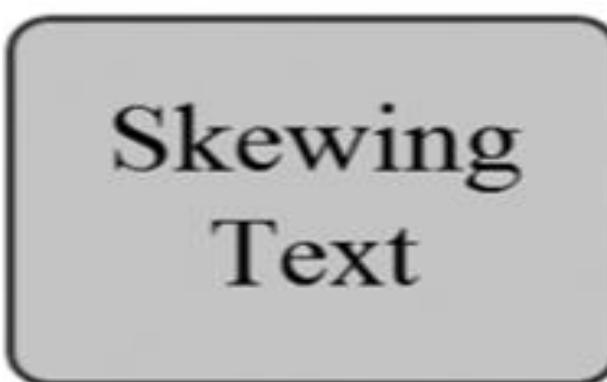
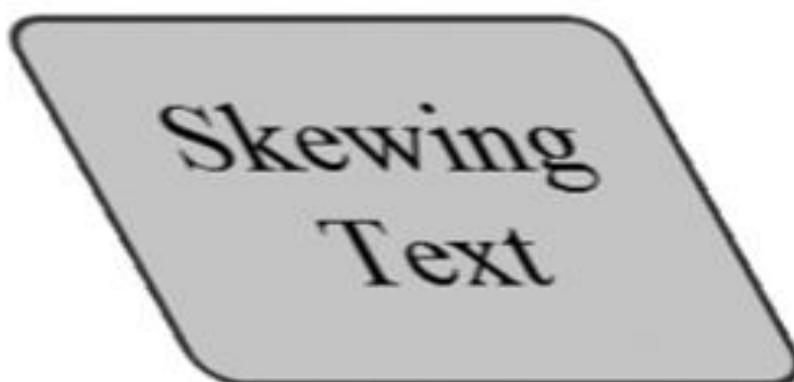
# Skew

```
50    </head>
51    <body>
52      <div class = "box skew">
53        <div class = "textbox">Skewing Text</div>
54      </div>
55    </body>
56  </html>
```

a) Bordered div at skewed left position

b) Bordered div at centered position

c) Bordered div at skewed right position



**Fig. 5.13 | Skewing and transforming elements in CSS3. (Part 3 of 3.)**

## Transitioning Between Images

- ▶ We can use the transition property to create the visually beautiful effect of *melting* one image into another (Fig. 5.14).
- ▶ The transition property includes three values. First, we specify that the transition will occur on the opacity of the image.
- ▶ The second value, 4s, is the transition-duration.
- ▶ The third value, ease-in-out, is the transition-timing-function.
- ▶ Next, we define :hover with an opacity of 0, so when the cursor hovers over the top image, its opacity becomes fully transparent, revealing the bottom image directly behind it. We then add the bottom and top images, placing one directly behind the other.

# Transitioning Between Images

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.14: meltingimages.html -->
4 <!-- Melting one image into another using CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Melting Images</title>
9     <style type = "text/css">
10    #cover
11      {
12        position: relative;
13        margin: 0 auto;
14      }
15      #cover img
16      {
17        position: absolute;
18        left: 0;
19        -webkit-transition: opacity 4s ease-in-out;
20        transition: opacity 4s ease-in-out;
21      }
22      #cover img.top:hover
23      {
24        opacity:0;
25      }
26    </style>
```

```
25      </head>
26      <body>
27        <div id = "cover">
28          <img class = "bottom" src = "jhtp.png" alt = "Java 9e cover">
29          <img class = "top" src = "jhtp8.png" alt = "Java 8e cover">
30        </div>
31      </body>
32    </html>
```

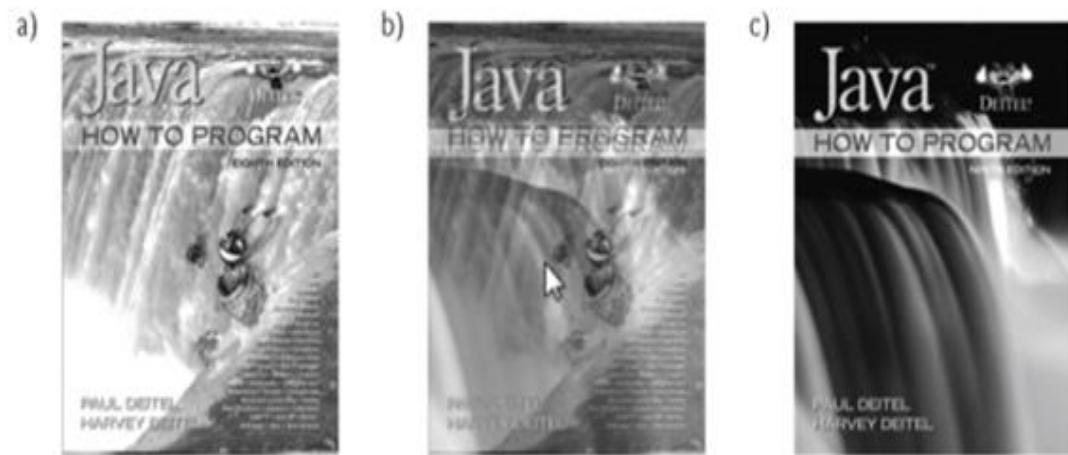


Fig. 5.14 | Melting one image into another using CSS3. (Part 2 of 2.)

Fig. 5.14 | Melting one image into another using CSS3. (Part 1 of 2.)

## Downloading Web Fonts and the @font-face Rule

- ▶ Using the **@font-face rule**, you can specify fonts for a web page, even if they're not installed on the user's system.
- ▶ You can use *downloadable fonts* to help ensure a uniform look across client sites.
- ▶ In Fig. 5.15, we use the Google web font named "Calligraffiti."
- ▶ You can find numerous free, open-source web fonts at <http://www.google.com/webfonts>.
- ▶ *Make sure the fonts you get from other sources have no legal encumbrances.*

# Downloading Web Fonts and the @font-face Rule

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.15: embeddedfonts.html -->
4 <!-- Embedding fonts for use in your web page. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Embedded Fonts</title>
9     <link href = 'http://fonts.googleapis.com/css?family=Calligraffiti'
10        rel = 'stylesheet' type = 'text/css'>
11     <style type = "text/css">
12       body
13       {
14         font-family: "Calligraffiti";
15         font-size: 48px;
16         text-shadow: 3px 3px 3px DimGrey;
17       }
18     </style>
19   </head>
```

*font +  
from Google  
) instead*

```
20   <body>
21     <div>
22       <b>Embedding the Google web font "Calligraffiti"</b>
23     </div>
24   </body>
25 </html>
```



Fig. 5.15 | Embedding fonts for use in your web page. (Part 1 of 2.)

Fig. 5.15 | Embedding fonts for use in your web page. (Part 2 of 2.)

## Downloading Web Fonts and the @font-face Rule

- To get Google's Calligraffiti font, go to <http://www.google.com/webfonts> and use the search box on the site to find the font "Calligraffiti."
- Next, click Quick-use to get the link to the style sheet that contains the @font-face rule.
- Paste that link element into the head section of your document. The referenced CSS style sheet contains the following CSS rules:

```
@media screen {  
    @font-face {  
        font-family: 'calligraffiti';  
        font-style: normal;  
        font-weight: normal;  
        src: local('calligraffiti'),  
            url('http://themes.googleusercontent.com/static/fonts/  
                calligraffiti/v1/vLVN2Y-z65rVu1R71WdvyKIZAuDcNtpCwuPSaIR0Ie8  
                .woff') format('woff');  
    }  
}
```

## Downloading Web Fonts and the @font-face Rule

- ▶ The @media screen rule specifies that the font will be used when the document is rendered on a computer screen.
- ▶ The @font-face rule includes the font-family (Calligraffiti), font-style (normal) and font-weight (normal).
- ▶ The @font-face rule also includes the *location* of the font.

## Flexible Box Layout Module and :nth-child Selectors

- ▶ Flexible Box Layout Module (FBLM) makes it easy to align the contents of boxes, change their size, change their order dynamically, and lay out the contents in any direction.
- ▶ In Fig. 5.16, we create flexible divs for four programming tips. When the mouse hovers over one of the divs, the div expands, the text changes from black to white, the background color changes and the layout of the text changes.

# Flexible Box Layout Module and :nth-child Selectors

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.16: fblm.html -->
4 <!-- Flexible Box Layout Module. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Flexible Box Layout Model</title>
9     <link href = 'http://fonts.googleapis.com/css?family=Rosario'
10       rel = 'stylesheet' type = 'text/css'>
11   <style type = "text/css">
12     .flexbox
13   {
14     width: 600px;
15     height: 420px;
16     display: -webkit-box;
17     display: box;
18     -webkit-box-orient: horizontal;
19     box-orient: horizontal;
20 }
```

```
.flexbox {
  width: 800px;
  height: 420px;
  display: flex;
}
```

```
21   .flexbox > div
22   {
23     -webkit-transition: 1s ease-out;
24     transition: 1s ease-out;
25     -webkit-border-radius: 10px;
26     border-radius: 10px;
27     border: 2px solid black;
28     width: 120px;
29     margin: 10px -10px 10px 0px;
30     padding: 20px 20px 20px 20px;
31     box-shadow: 10px 10px 10px dimgray;
32 }
33 .flexbox > div:nth-child(1){ background-color: lightgrey; }
34 .flexbox > div:nth-child(2){ background-color: lightgrey; }
35 .flexbox > div:nth-child(3){ background-color: lightgrey; }
36 .flexbox > div:nth-child(4){ background-color: lightgrey; }
37 .flexbox > div:hover { 44 BYF
38   width: 200px; color: white; font-weight: bold; }
39 .flexbox > div:nth-child(1):hover { background-color: royalblue; }
40 .flexbox > div:nth-child(2):hover { background-color: crimson; }
41 .flexbox > div:nth-child(3):hover { background-color: crimson; }
42 .flexbox > div:nth-child(4):hover { background-color: darkgreen; }
43 p { height: 250px; overflow: hidden; font-family: "Rosario" }
44
45 </style>
```

```
.flexbox>div {
  transition: 1s ease-out;
  border-radius: 10px;
  border: 2px solid black;
  width: 120px;
  margin: 10px -10px 10px 0px;
  padding: 20px 20px 20px 20px;
  box-shadow: 10px 10px 10px dimgray;
}
```

Fig. 5.16 | Flexible Box Layout Module. (Part 1 of 5.)

Fig. 5.16 | Flexible Box Layout Module. (Part 2 of 5.)

# Flexible Box Layout Module and :nth-child Selectors

```
46 </head>
47 <body>
48 <div class = "flexbox">
49 <div><img src = "GPP.png" alt = "Good programming practice icon">
50 <p>Good Programming Practices call attention to techniques that
51 will help you produce programs that are clearer, more
52 understandable and more maintainable.</p></div>
53 <div><img src = "EPT.png" alt = "Error prevention tip icon">
54 <p>Error-Prevention Tips contain suggestions for exposing bugs
55 and removing them from your programs; many describe aspects of
56 programming that prevent bugs from getting into programs in
57 the first place.</p></div>
58 <div><img src = "CPE.png" alt = "Common programming error icon">
59 <p>Common Programming Errors point out the errors that students
60 tend to make frequently. These Common Programming Errors reduce
61 the likelihood that you'll make the same mistakes.</p></div>
62 <div><img src = "SEO.png"><p>Software Engineering Observations
63 highlight architectural and design issues that affect the
64 construction of software systems, especially large-scale
65 systems.</p></div>
66 </div>
67 </body>
68 </html>
```

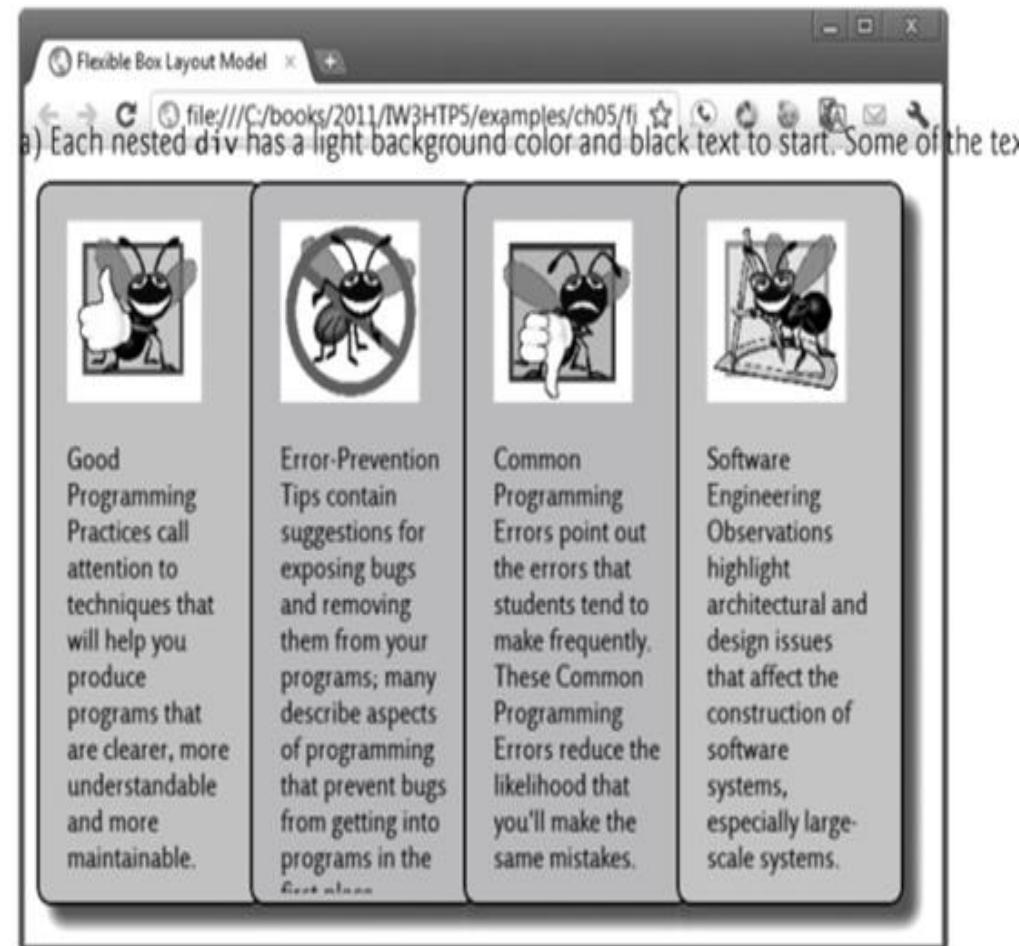
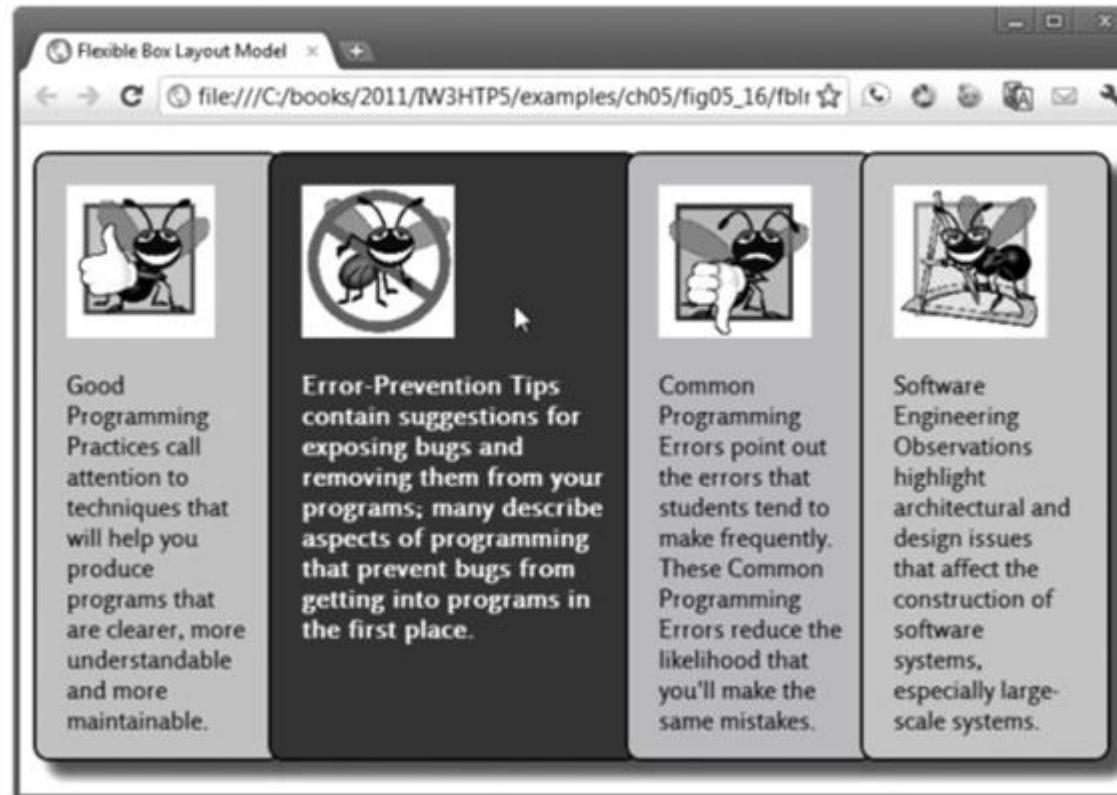


Fig. 5.16 | Flexible Box Layout Module. (Part 3 of 5.)

Fig. 5.16 | Flexible Box Layout Module. (Part 4 of 5.)

# Flexible Box Layout Module and :nth-child Selectors

b) When the mouse hovers over :nth-child(2), the flexbox expands, the background-color changes to Crimson, the overflow text is revealed and the text changes to a bold



**Fig. 5.16 | Flexible Box Layout Module. (Part 5 of 5.)**

## Flexible Box Layout Module and :nth-child Selectors

- ▶ We define a div to which we apply the flexbox CSS class. That div contains four other divs.
- ▶ The flexbox class's display property is set to the new CSS3 value box.
- ▶ The **box-orient property** specifies the orientation of the box layout. The default value is horizontal (which we specified anyway). You can also use vertical.
- ▶ For the nested divs, we specify a one-second ease-out transition. This will take effect when the :hover pseudo-class style is applied to one of these divs to expand it.

# Flexible Box Layout Module and :nth-child Selectors

## :nth-child Selectors

- ▶ In CSS3, you can use selectors to easily select elements to style based on their *attributes*.
- ▶ We use **:nth-child selectors** to select each of the four div elements in the flexbox div to style.
- ▶ div:nth-child(1) selects the div element that's the *first* child of its parent and applies the background-color LightBlue.
- ▶ Similarly, div:nth-child(2) selects the div element that's the *second* child of its parent, div:nth-child(3) selects the *third* child of its parent, and div:nth-child(4) selects the *fourth* child of its parent—each applies a specified background-color.

## Flexible Box Layout Module and :nth-child Selectors

- ▶ Next, we define styles that are applied to the nested `div` elements when the mouse hovers over them—the width (200px), color (white) and font-weight (bold).
- ▶ We use `:nth-child` selectors to specify a new background color for each nested `div`.
- ▶ Finally, we style the `p` element—the text within each `div`.
- ▶ In the output, notice that the text in the *second* child element (the Error-Prevention Tips), the overflow text is hidden. When the mouse hovers over the element, all of the text is revealed.

## Multicolumn Layout

- ▶ CSS3 allows you to easily create **multicolumn layouts**.
- ▶ In Figure 5.17, we create a three-column layout by setting the **column-count property** to 3 and the **column-gap property** (the spacing between columns) to 30px.
- ▶ We then add a thin black line between each column using the **column-rule property**.
- ▶ When you run this example, try resizing your browser window. The width of the columns changes to fit the three-column layout in the browser.

# Multicolumn Layout

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.17: multicolumns.html -->
4 <!-- Multicolumn text in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Multicolumns</title>
9     <style type = "text/css">
10    p
11      { margin:0.9em 0em; }
12      .multicolumns
13      {
14        /* setting the number of columns to 3 */
15        -webkit-column-count: 3;
16        -moz-column-count: 3;
17        -o-column-count: 3;
18        column-count: 3;
19        /* setting the space between columns to 30px */
20        -webkit-column-gap: 30px;
21        -moz-column-gap: 30px;
22        -o-column-gap: 30px;
23        column-gap: 30px;
```

b3r S

```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
/* adding a 1px black line between each column */
-webkit-column-rule: 1px outset black;
-moz-column-rule: 1px outset black;
-o-column-rule: 1px outset black;
column-rule: 1px outset black;
}
</style>
</head>
<body>
<header>
  <h1>Computers, Hardware and Software</h1>
</header>
<div class = "multicolumns">
  <p>A computer is a device that can perform computations and make
  logical decisions phenomenally faster than human beings can.
  Many of today's personal computers can perform billions of
  calculations in one second—more than a human can perform
  in a lifetime. Supercomputers are already performing thousands
  of trillions (quadrillions) of instructions per second! To put
  that in perspective, a quadrillion-instruction-per-second
  computer can perform in one second more than 100,000
  calculations for every person on the planet! And—these
  "upper limits" are growing quickly!</p>
  <p>Computers process data under the control of sets of
  instructions called computer programs. These programs guide
```

Fig. 5.17 | Multicolumn text in CSS3. (Part 1 of 5.)

Fig. 5.17 | Multicolumn text in CSS3. (Part 2 of 5.)

# Multicolumn Layout

the computer through orderly sets of actions specified by people called computer programmers. The programs that run on a computer are referred to as software. In this book, you'll learn today's key programming methodology that's enhancing programmer productivity, thereby reducing software-development costs&mdash;object-oriented programming.</p><p>A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disks, memory, DVDs and processing units). Computing costs are dropping dramatically, owing to rapid developments in hardware and software technologies. Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each. Ironically, silicon is one of the most abundant materials&mdash;it's an ingredient in common sand. Silicon-chip technology has made computing so economical that more than a billion general-purpose computers are in use worldwide, and this is expected to double in the next few years.</p><p>Computer chips (microprocessors) control countless devices. These embedded systems include anti-lock brakes in cars, navigation systems, smart home appliances, home security systems, cell phones and smartphones, robots, intelligent traffic intersections, collision avoidance systems, video game controllers and more. The vast majority of the microprocessors

produced each year are embedded in devices other than general-purpose computers.</p><pre>74<br>75<br>76<br>77<br>78<br>79<br>80<br>81<br>82</pre>

Fig. 5.17 | Multicolumn text in CSS3. (Part 4 of 5.)



Fig. 5.17 | Multicolumn text in CSS3. (Part 3 of 5.)

Fig. 5.17 | Multicolumn text in CSS3. (Part 5 of 5.)

## Media Queries

- ▶ With CSS3 *media queries* you can determine the finer attributes of the media on which the user is viewing the page, such as the *length* and *width* of the viewing area on the screen, to better customize your presentation.
- ▶ In Fig. 5.18, we modify the multicolumn example to alter the numbers of columns and the rules between columns based on the screen size of the device on which the page is viewed.

# Media Queries

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.18: mediaqueries.html -->
4 <!-- Using media queries to reformat a page based on the device width. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Media Queries</title>
9     <style type = "text/css">
10    p
11      { margin: 0.9em 0em; }
12      /* styles for smartphones with screen widths 480px or smaller */
13      @media handheld and (max-width: 480px),
14        screen and (max-device-width: 480px),
15        screen and (max-width: 480px)
16      {
17        div {
18          -webkit-column-count: 1;
19          column-count: 1; }
20      }
21
22      /* styles for devices with screen widths of 481px to 1024px */
23      @media only screen and (min-width: 481px) and
24        (max-width: 1024px)
25      {
26        div {
27          -webkit-column-count: 2;
28          column-count: 2;
29          -webkit-column-gap: 30px;
30          column-gap: 30px;
31          -webkit-column-rule: 1px outset black;
32          column-rule: 1px outset black; }
33
34      /* styles for devices with screen widths of 1025px or greater */
35      @media only screen and (min-width: 1025px)
36      {
37        div {
38          -webkit-column-count: 3;
39          column-count: 3;
40          -webkit-column-gap: 30px;
41          column-gap: 30px;
42          -webkit-column-rule: 1px outset black;
43          column-rule: 1px outset black; }
```

**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 1 of 8.)

**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 2 of 8.)

# Media Queries

```
44      </style>
45  </head>
46  <body>
47    <header>
48      <h1>Computers, Hardware and Software</h1>
49    </header>
50    <div>
51      <p>A computer is a device that can perform computations and make
52        logical decisions phenomenally faster than human beings can.
53        Many of today's personal computers can perform billions of
54        calculations in one second—more than a human can perform
55        in a lifetime. Supercomputers are already performing thousands
56        of trillions (quadrillions) of instructions per second! To put
57        that in perspective, a quadrillion-instruction-per-second
58        computer can perform in one second more than 100,000
59        calculations for every person on the planet! And—these
60        "upper limits" are growing quickly!</p>
61      <p>Computers process data under the control of sets of
62        instructions called computer programs. These programs guide
63        the computer through orderly sets of actions specified by
64        people called computer programmers. The programs that run on a
65        computer are referred to as software. In this book, you'll
66        learn today's key programming methodology that's enhancing
```

```
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
```

programmer productivity, thereby reducing software-development costs—object-oriented programming.</p>

<p>A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disks, memory, DVDs and processing units). Computing costs are dropping dramatically, owing to rapid developments in hardware and software technologies. Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each. Ironically, silicon is one of the most abundant materials—it's an ingredient in common sand. Silicon-chip technology has made computing so economical that more than a billion general-purpose computers are in use worldwide, and this is expected to double in the next few years.</p>

<p>Computer chips (microprocessors) control countless devices. These embedded systems include anti-lock brakes in cars, navigation systems, smart home appliances, home security systems, cell phones and smartphones, robots, intelligent traffic intersections, collision avoidance systems, video game controllers and more. The vast majority of the microprocessors produced each year are embedded in devices other than general-purpose computers.</p>

**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 3 of 8.)

**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 4 of 8.)

# Media Queries

```
90 <footer>
91     <em>© 2012 by Pearson Education, Inc.
92     All Rights Reserved.</em>
93 </footer>
94 </div>
95 </body>
96 </html>
```

**Fig. 5.18 | Using media queries to reformat a page based on the device width. (Part 5 of 8.)**

a) Styles for smartphones with screen widths 480px or smaller



**Fig. 5.18 | Using media queries to reformat a page based on the device width. (Part 6 of 8.)**

b) Styles for devices with screen widths of 481px to 1024px



**Fig. 5.18 | Using media queries to reformat a page based on the device width. (Part 7 of 8.)**

c) Styles for devices with screen widths of 1024px or greater



**Fig. 5.18 | Using media queries to reformat a page based on the device width. (Part 8 of 8.)**

# Media Queries

## *@media-Rule*

- ▶ The **@media rule** is used to determine the type *and size* of device on which the page is rendered.
- ▶ When the browser looks at the rule, the result is either *true* or *false*. The rule's styles are applied only if the result is true.
- ▶ First, we use the @media rule to determine whether the page is being rendered on a handheld device (e.g., a smartphone) with a **max-width** of 480px, or a device with a screen that has a **max-device-width** of 480px, or on a screen having **max-width** of 480px.
- ▶ If this is *true*, we set the **column-count** to 1—the page will be rendered in a single column on handheld devices such as an iPhone or in browser windows that have been resized to 480px or less.

# Media Queries

- ▶ If the condition of the first @media rule is *false*, a second @media rule determines whether the page is being rendered on devices with a min-width of 481px and a max-width of 1024px.
- ▶ If this condition is *true*, we set the column-count to 2, the column-gap (the space between columns) to 30px and the column-rule (the vertical line between the columns) to 1px outset black.
- ▶ If the conditions in the first two @media rules are *false*, we use a third @media rule to determine whether the page is being rendered on devices with a min-width of 1025px.
- ▶ If the condition of this rule is *true*, we set the column-count to 3, the column-gap to 30px (lines 39–40) and the column-rule to 1px outset black.

# UCCD2323 Front-End Web Development



## Chapter 2 Formatting and Presenting Information Part 1.

# Topic Outline

- INTRODUCTION
- NEW HTML5 FORM INPUT TYPES (color, date, datetime, datime-local, email, month, number, range, search, tel, time, url, week)
- INPUT AND DATALIST ELEMENTS AND AUTOCOMPLETE ATTRIBUTE. (input Element autocomplete Attribute, Datalist Element)
- PAGE STRUCTURE ELEMENTS ( header, nav, figure, article, summary and details, section, aside, meter, footer and text-level semantics.

## Objectives

- At the end of this lecture, students will be able to:
  1. Build a form using the new HTML5 input types.
  2. Specify input element in a form as the one should receive the focus by default
  3. Use self-validating input elements
  4. Specify temporary placeholder text in various input elements
  5. Use autocomplete input elements that helps users re-enter text that they've previously entered in a form
  6. Use a datalist to specify a list of values that can be entered in an input element and to autocomplete entries as the user types.
  7. Use HTML5's new page-structure elements to delineate parts of a page including headers, sections, figures, footers and more.

# New HTML5 Form input Types

- Figure 3.1 demonstrates HTML5's new form input types.
- These are not yet universally supported by all browsers.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 3.1: newforminputtypes.html --&gt;
4  &lt;!-- New HTML5 form input types and attributes. --&gt;
5  &lt;html&gt;
6      &lt;head&gt;
7          &lt;meta charset="utf-8"&gt;
8          &lt;title&gt;New HTML5 Input Types&lt;/title&gt;
9      &lt;/head&gt;
10
11     &lt;body&gt;
12         &lt;h1&gt;New HTML5 Input Types Demo&lt;/h1&gt;
13         &lt;p&gt;This form demonstrates the new HTML5 input types
14             and the placeholder, required and autofocus attributes.
15         &lt;/p&gt;
16
17         &lt;form method = "post" action = "http://www.deitel.com"&gt;
18             &lt;p&gt;
19                 &lt;label&gt;Color:
20                     &lt;input type = "color" autofocus /&gt;
21                         (Hexadecimal code such as #ADD8E6)
22                 &lt;/label&gt;
23             &lt;/p&gt;
24
25             &lt;p&gt;
26                 &lt;label&gt;Date:
27                     &lt;input type = "date" /&gt;
28                         (yyyy-mm-dd)
29                 &lt;/label&gt;
30             &lt;/p&gt;
31
32             &lt;p&gt;
33                 &lt;label&gt;Datetime:
34                     &lt;input type = "datetime" /&gt;
35                         (yyyy-mm-ddThh:mm:ss, such as 2012-01-27T03:15)
36                 &lt;/label&gt;
37             &lt;/p&gt;
38
39             &lt;p&gt;
40                 &lt;label&gt;Datetime-local:
41                     &lt;input type = "datetime-local" /&gt;
42                         (yyyy-mm-ddThh:mm, such as 2012-01-27T03:15)
43                 &lt;/label&gt;
44             &lt;/p&gt;
45
46             &lt;p&gt;
47                 &lt;label&gt;Email:
48                     &lt;input type = "email" placeholder = "name@domain.com"
49                         required /&gt; (name@domain.com)
50                 &lt;/label&gt;
51             &lt;/p&gt;</pre>
```

Fig. 3.1 | New HTML5 form input types and attributes. (Part 1 of 5.)

Fig. 3.1 | New HTML5 form input types and attributes. (Part 2 of 5.)

# New HTML5 Form input Types

```
48 <p>
49   <label>Month:
50     <input type = "month" /> (yyyy-mm)
51   </label>
52 </p>
53 <p>
54   <label>Number:
55     <input type = "number"
56       min = "0"
57       max = "7"
58       step = "1"
59       value = "4" />
60   </label> (Enter a number between 0 and 7)
61 </p>
62 <p>
63   <label>Range:
64     0 <input type = "range"
65       min = "0"
66       max = "20"
67       value = "10" /> 20
68   </label>
69 </p>
```

Fig. 3.1 | New HTML5 form input types and attributes. (Part 3 of 5.)

```
70 <p>
71   <label>Search:
72     <input type = "search" placeholder = "search query" />
73   </label> (Enter your search query here.)
74 </p>
75 <p>
76   <label>Tel:
77     <input type = "tel" placeholder = "(###) ###-####"
78       pattern = "\(\d{3}\) +\d{3}-\d{4}" required />
79       (###) ###-####
80   </label>
81 </p>
82 <p>
83   <label>Time:
84     <input type = "time" /> (hh:mm:ss.ff)
85   </label>
86 </p>
87 <p>
88   <label>URL:
89     <input type = "url"
90       placeholder = "http://www.domainname.com" />
91       (http://www.domainname.com)
92   </label>
93 </p>
```

Fig. 3.1 | New HTML5 form input types and attributes. (Part 4 of 5.)

# New HTML5 Form input Types

```
94      <p>
95          <label>Week:
96              <input type = "week" />
97                  (yyyy-Wnn, such as 2012-W01)
98          </label>
99      </p>
100     <p>
101         <input type = "submit" value = "Submit" />
102         <input type = "reset" value = "Clear" />
103     </p>
104     </form>
105   </body>
106 </html>
```

**Fig. 3.1 |** New HTML5 form input types and attributes. (Part 5 of 5.)

# Input Type Color

- The **color input type** enables the user to enter a color.
- At the time of this writing, most browsers render the color input type as a text field in which the user can enter a hexadecamal code or a color name.
- In the future, when you click a color input, browsers will likely display a *color picker* similar to the Microsoft Windows color dialog shown in Fig. 3.2.

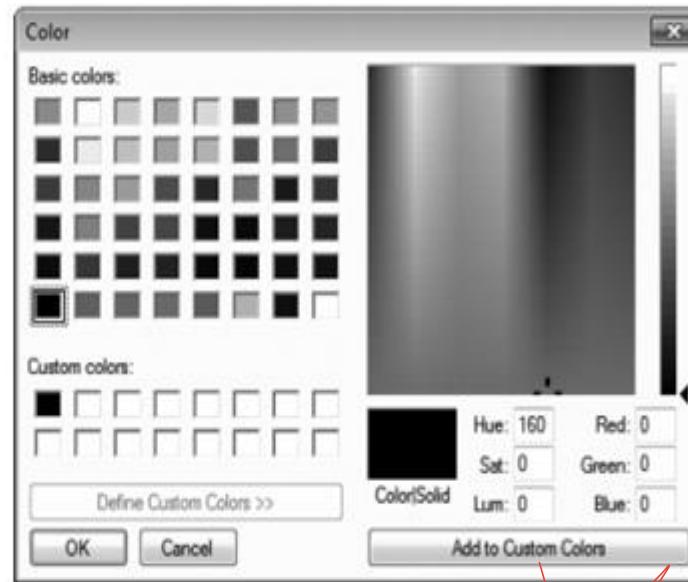


Fig. 3.2 | A dialog for choosing colors.

time  
hex

Color  
code

# Input Type Color

- ***autofocus Attribute***
- The **autofocus attribute**—an optional attribute that can be used in only one input element on a form—automatically gives the focus to the input element, allowing the user to begin typing in that element immediately.
- Figure 3.3 shows autofocus on the color element—the first input element in our form—as rendered in Chrome. You do not need to include autofocus in your forms.

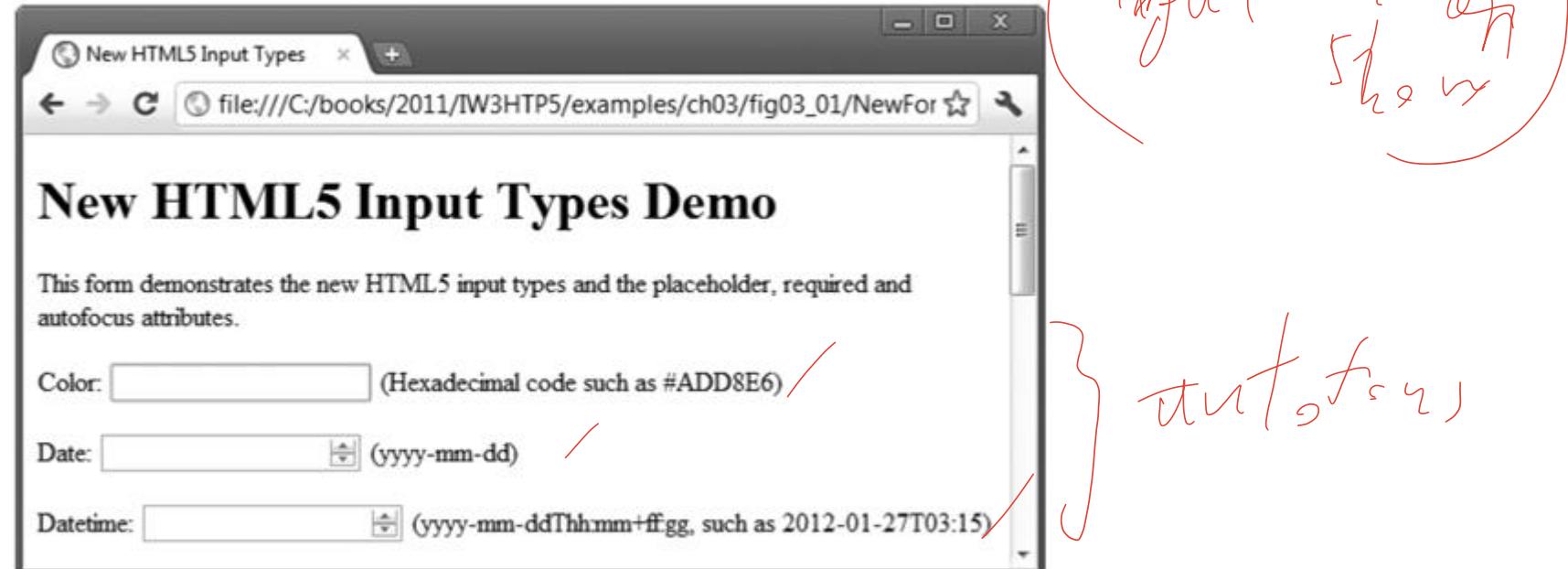


Fig. 3.3 | Autofocus in the color input element using Chrome.

# Input Type Color

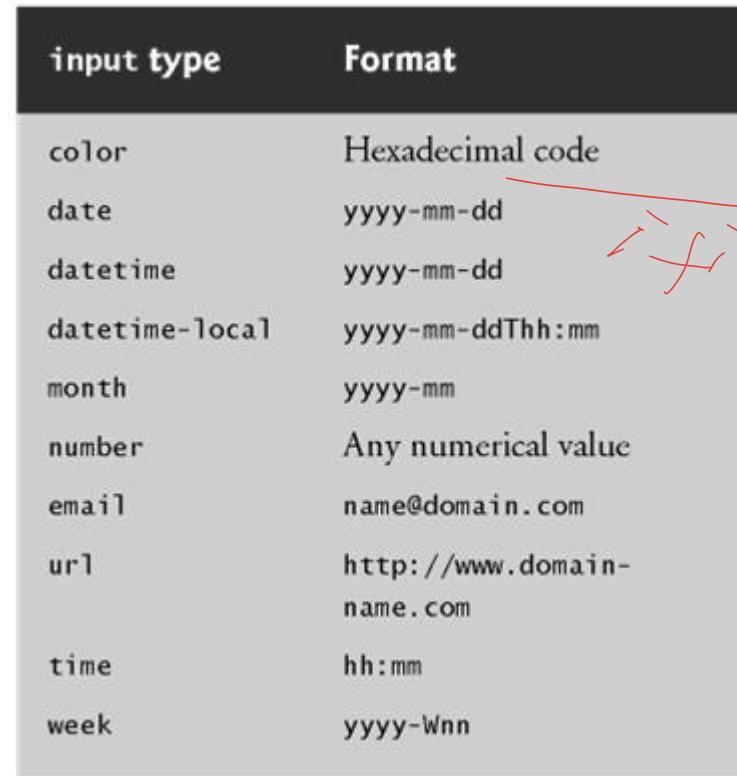
- **Validation**
- The new HTML 5 input types are *self validating* on the client side, eliminating the need to add complicated JavaScript code to your web pages to validate user input, reducing the amount of invalid data submitted and consequently reducing Internet traffic between the server and the client to correct invalid input.
- *The server should still validate all user input.*
- When a user enters data into a form then submits the form the browser immediately checks the self-validating elements to ensure that the data is correct (Fig. 3.4).



Fig. 3.4 | Validating a color input in Chrome.

# Input Type Color

- Figure 3.5 lists each of the new HTML5 input types and provides examples of the proper formats required for each type of data to be valid.



input type	Format
color	Hexadecimal code
date	yyyy-mm-dd
datetime	yyyy-mm-dd
datetime-local	yyyy-mm-ddThh:mm
month	yyyy-mm
number	Any numerical value
email	name@domain.com
url	http://www.domain-name.com
time	hh:mm
week	yyyy-Wnn

Fig. 3.5 | Self-validating input types.

- If you want to bypass validation, you can add the `formnovalidate` attribute to input type submit in line 101:  
`<input type = "submit" value = "Submit" formnovalidate />`

# Input Type Date

- The date input type enables the user to enter a date in the form yyyy-mm-dd.
- Firefox and Internet Explorer display a text field in which a user can enter a date such as 2012-01-27.
- Chrome and Safari display a spinner control—a text field with an up-down arrow () on the right side—allowing the user to select a date by clicking the up or down arrow.
- The start date is the *current date*.
- Opera displays a calendar from which you can choose a date.
- In the future, when the user clicks a date input, browsers are likely to display a date control similar to the Microsoft Windows one shown in Fig. 3.6.



Fig. 3.6 | A date chooser control.

## **Input Type Datetime**

- The **datetime input type** enables the user to enter a date (year, month, day), time (hour, minute, second, fraction of a second) and the time zone set to UTC (Coordinated Universal Time or Universal Time, Coordinated).
- Currently, most of the browsers render datetime as a text field; Chrome renders an up-down control and Opera renders a date and time control.

## **Input Type Datetime-Local**

- The **datetime-local input type** enables the user to enter the date and time in a *single* control.
- The data is entered as year, month, day, hour, minute, second and fraction of a second.
- Internet Explorer, Firefox and Safari all display a text field.
- Opera displays a date and time control.

# Input Type Email

(Get standard)

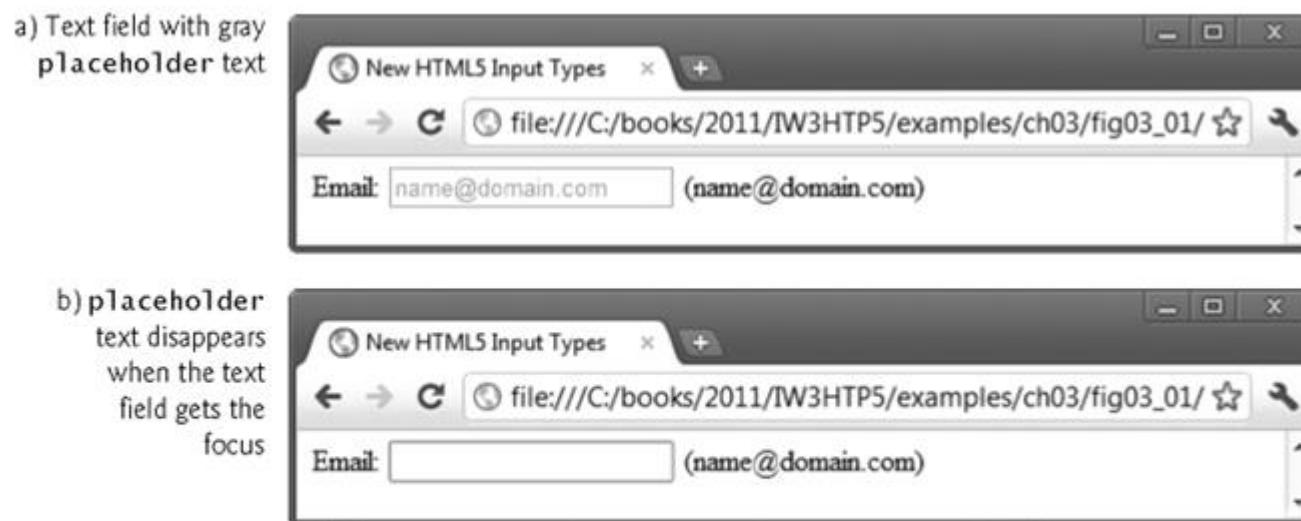
- The **email** input type enables the user to enter an e-mail address or a list of e-mail addresses separated by commas (if the multiple attribute is specified).
- Currently, all of the browsers display a text field.
- If the user enters an *invalid* e-mail address (i.e., the text entered is *not* in the proper format) and clicks the Submit button, a callout asking the user to enter an e-mail address is rendered pointing to the input element (Fig. 3.7).
- HTML5 does not check whether an e-mail address entered by the user actually exists—rather it just validates that the e-mail address is in the *proper format*.



Fig. 3.7 | Validating an e-mail address in Chrome.

# Input Type Email

- **placeholder Attribute**
- The **placeholder** attribute allows you to place temporary text in a text field.
- Generally, placeholder text is *light gray* and provides an example of the text and/or text format the user should enter (Fig. 3.8).
- When the *focus* is placed in the text field (i.e., the cursor is in the text field), the placeholder text disappears—it's not “submitted” when the user clicks the Submit button (unless the user types the same text).



**Fig. 3.8 |** placeholder text disappears when the `input` element gets the focus.

# Input Type Email

- HTML5 supports placeholder text for only six input types—text, search, url, tel, email and password.
- ***required Attribute***
- The **required attribute** forces the user to enter a value before submitting the form.
- You can add required to any of the input types.
- In this example, the user *must* enter an e-mail address and a telephone number to submit the form (Fig. 3.9).

New HTML5 Input Types Demo

This form demonstrates the new HTML5 input types and the placeholder, required and autofocus attributes.

Color:  (Hexadecimal code such as #ADD8E6)

Date:  (yyyy-mm-dd)

Datetime:  (yyyy-mm-ddThh:mm+ff:gg, such as 2012-01-27T03:15)

Datetime-local:  (yyyy-mm-ddThh:mm, such as 2012-01-27T03:15)

Email:  (name@domain.com)

Month:  Please fill out this field. (y-mm)

Fig. 3.9 | Demonstrating the **required** attribute in Chrome.

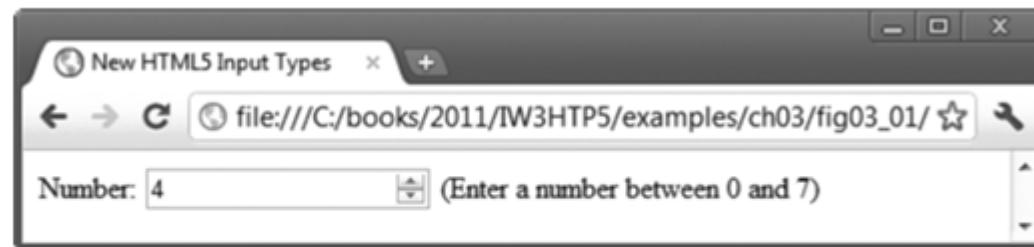
# Input Type Month

- The **month** input type enables the user to enter a year and month in the format yyyy-mm, such as 2012-01.
- If the user enters the data in an improper format (e.g., January 2012) and submits the form, a callout stating that an invalid value was entered appears.

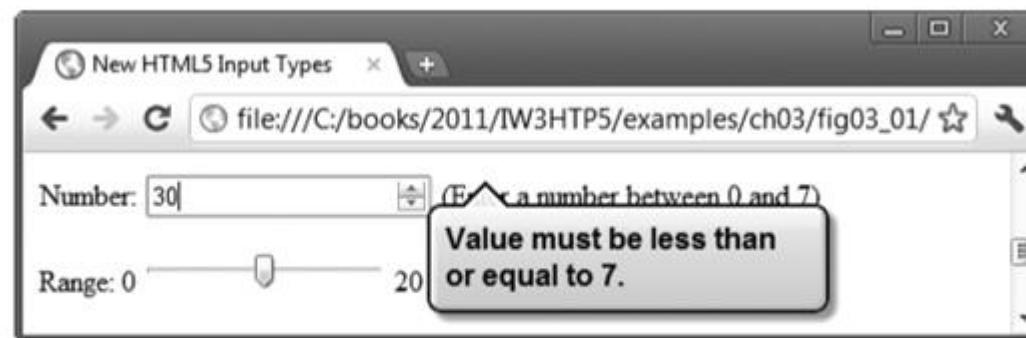
## Input Type Number

- The **number** input type enables the user to enter a numerical value—mobile browsers typically display a numeric keypad for this input type.
- Internet Explorer, Firefox and Safari display a text field in which the user can enter a number. Chrome and ~~Opera~~ render a spinner control for adjusting the number.
- The **min** attribute sets the minimum valid number.
- The **max** attribute sets the maximum valid number.
- The **step** attribute determines the increment in which the numbers increase.
- The **value** attribute sets the initial value displayed in the form (Fig. 3.10).
- The spinner control includes only the valid numbers.
- If the user attempts to enter an invalid value by typing in the text field, a callout pointing to the number input element will instruct the user to enter a valid value.

# Input Type Number



**Fig. 3.10** | `input type number` with a `value` attribute of 4 as rendered in Chrome.



**Fig. 3.11** | Chrome checking for a valid number.

# Input Type Range

- The range input type appears as a *slider control* in Chrome, Safari and Opera (Fig. 3.12).
- You can set the minimum and maximum and specify a value.
- The range input type is *inherently self-validating* when it is rendered by the browser as a slider control, because *the user is unable to move the slider outside the bounds of the minimum or maximum value*.



**Fig. 3.12** | range slider with a value attribute of 10 as rendered in Chrome.

# Input Type Search

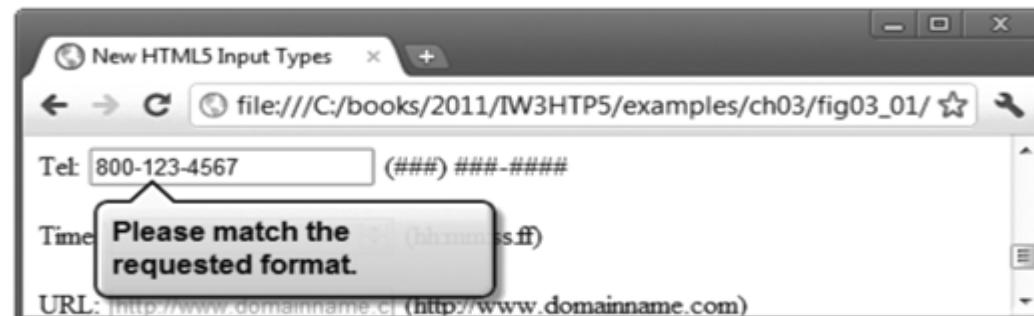
- The `search` input type provides a search field for entering a query.
- This input element is functionally equivalent to an input of type `text`.
- When the user begins to type in the search field, Chrome and Safari display an X that can be clicked to clear the field (Fig. 3.13).



**Fig. 3.13 |** Entering a search query in Chrome.

# Input Type Tel

- The `tel` input type enables the user to enter a telephone number—mobile browsers typically display a keypad specific to entering phone numbers for this input type.
- At the time of this writing, the `tel` input type is rendered as a text field in all of the browsers.
- HTML5 does *not* self validate the `tel` input type.
- To ensure that the user enters a phone number in a proper format, we've added a `pattern` attribute that uses a *regular expression* to determine whether the number is in the format:  
`(555) 555-5555`
  - When the user enters a phone number in the wrong format, a callout appears requesting the proper format, pointing to the `tel` input element (Fig. 3.14).



**Fig. 3.14 |** Validating a phone number using the `pattern` attribute in the `tel` input type.

# Input Type Time

- The **time** input type enables the user to enter an hour, minute, seconds and fraction of second (Fig. 3.15).
- The HTML5 specification indicates that a time must have two digits representing the **hour**, followed by a **colon (:) and two digits representing the minute**.
- Optionally, you can also include a colon followed by two digits representing the seconds and a period followed by one or more digits representing a fraction of a second (shown as ff in our sample text to the right of the time input element in Fig. 3.15).



Fig. 3.15 | time input as rendered in Chrome.

# Input Type URL

- The `url` input type enables the user to enter a URL.
- The element is rendered as a text field, and the proper format is `http://www.deitel.com`.
- If the user enters an improperly formatted URL (e.g., `www.deitel.com` or `www.deitelcom`), the URL will *not* validate (Fig. 3.16).
- HTML5 does not check whether the URL entered is valid; rather it validates that the URL entered is in the proper format.



Fig. 3.16 | Validating a URL in Chrome.

## Input Type Week

- The **week input type** enables the user to select a year and week number in the format yyyy-Wnn, where nn is 01–53—for example, 2012-W01 represents the first week of 2012. Internet Explorer, Firefox and Safari render a text field.
- Chrome renders an up-down control.
- Opera renders *week control* with a down arrow that, when clicked, brings up a calendar for the current month with the corresponding week numbers listed down the left side.

# Input and Data list Elements and Autocomplete Attribute

- Figure 3.17 shows how to use the new autocomplete attribute and datalist element.

```
1 <!DOCTYPE html>  
2  
3 <!-- Fig. 3.17: autocomplete.html -->  
4 <!-- New HTML5 form autocomplete attribute and datalist element. -->  
5 <html>  
6   <head>  
7     <meta charset="utf-8">  
8     <title>New HTML5 autocomplete Attribute and datalist Element</title>  
9   </head>  
10  
11 <body>  
12   <h1>Autocomplete and Datalist Demo</h1>  
13   <p>This form demonstrates the new HTML5 autocomplete attribute  
14     and the datalist element.  
15   </p>  
16  
17   <!-- turn autocomplete on -->  
18   <form method = "post" autocomplete = "on">  
19     <p><label>First Name:  
20       <input type = "text" id = "firstName"  
21         placeholder = "First name" /> (First name)  
22     </label></p>
```

Fig. 3.17 | New HTML5 form autocomplete attribute and datalist element. (Part I of 6.)

# Input and Data list Elements and Autocomplete Attribute

```
23 <p><label>Last Name:</label>
24   <input type = "text" id = "lastName"
25     placeholder = "Last name" /> (Last name)
26   </label></p>
27 <p><label>Email:</label>
28   <input type = "email" id = "email"
29     placeholder = "name@domain.com" /> (name@domain.com)
30   </label></p>
31 <p><label for = "txtList">Birth Month:</label>
32   <input type = "text" id = "txtList"
33     placeholder = "Select a month" list = "months" />
34   <datalist id = "months">
35     <option value = "January">
36     <option value = "February">
37     <option value = "March">
38     <option value = "April">
39     <option value = "May">
40     <option value = "June">
41     <option value = "July">
42     <option value = "August">
43     <option value = "September">
44     <option value = "October">
45     <option value = "November">
```

Fig. 3.17 | New HTML5 form autocomplete attribute and datalist element. (Part 2 of 6.)

```
46   <option value = "December">
47   </datalist>
48 </label></p>
49 <p><input type = "submit" value = "Submit" />
50   <input type = "reset" value = "Clear" /></p>
51 </form>
52 </body>
53 </html>
```

Fig. 3.17 | New HTML5 form autocomplete attribute and datalist element. (Part 3 of 6.)

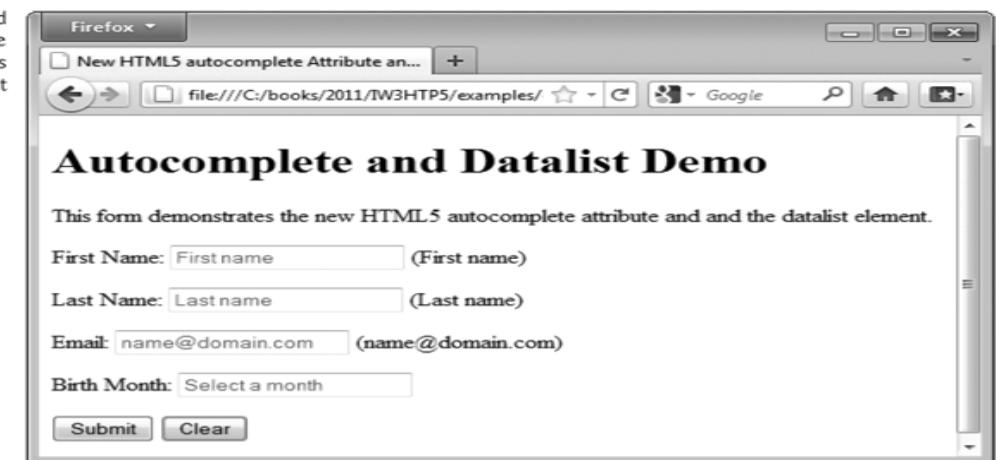


Fig. 3.17 | New HTML5 form autocomplete attribute and datalist element. (Part 4 of 6.)

# Input and Data list Elements and Autocomplete Attribute

b) autocomplete automatically fills in the data when the user returns to a form submitted previously and begins typing in the First Name input element; clicking Jane inserts that value in the input

Firefox

New HTML5 autocomplete Attribute an...

file:///C:/books/2011/IW3HTP5/examples/ Google

## Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: J (First name)

Last Name: Last name (Last name)

Email: name@domain.com (name@domain.com)

Birth Month: Select a month

Submit Clear

c) autocomplete with a datalist showing the previously entered value (June) followed by all items that match what the user has typed so far; clicking an item in the autocomplete list inserts that value in the input

Firefox

New HTML5 autocomplete Attribute an...

file:///C:/books/2011/IW3HTP5/examples/ Google

## Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: Jane (First name)

Last Name: Blue (Last name)

Email: jane@domain.com (name@domain.com)

Birth Month: j

June

January

June

July

Submit

Fig. 3.17 | New HTML5 form autocomplete attribute and datalist element. (Part 5 of 6.)

Fig. 3.17 | New HTML5 form autocomplete attribute and datalist element. (Part 6 of 6.)

# Input Element Autocomplete Attribute

- The **autocomplete** attribute can be used on input types to automatically fill in the user's information based on previous input—such as name, address or e-mail.
- You can enable autocomplete for an entire form or just for specific elements.
- For example, an online order form might set `autocomplete = "on"` for the name and address inputs and set `autocomplete = "off"` for the credit card and password inputs for security purposes.



## Error-Prevention Tip 3.1

The `autocomplete` attribute works only if you specify a `name` or `id` attribute for the `input` element.

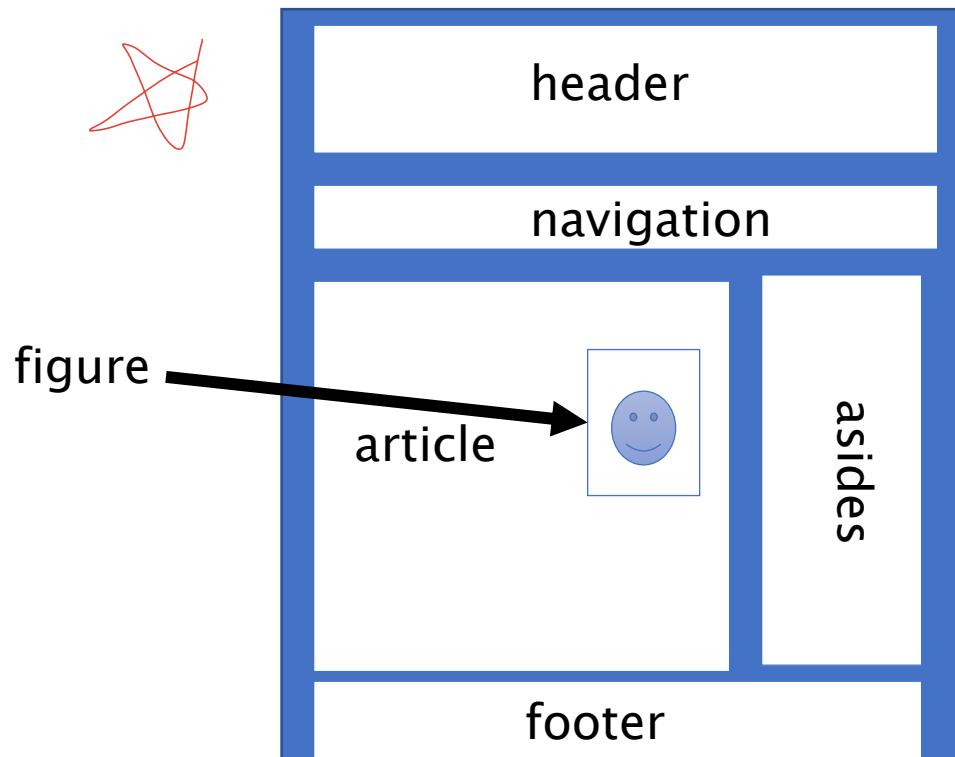
# Datalist Element

- The [datalist element](#) provides input options for a text input element.
- At the time of this writing, datalist support varies by browser.
- In this example, we use a datalist element to obtain the user's birth month.
- Using Opera, when the user clicks in the text field, a drop-down list of the months of the year appears. If the user types "M" in the text field, the list on months is narrowed to March and May.
- When using Firefox, the drop-down list of months appears only after the user begins typing in the text field. If the user types "M", all months containing the letter "M" or "m" appear in the drop-down list—March, May, September, November and December.

# Page-Structure Elements

pr of 2  
6/9/14

- HTML5 introduces several new page-structure elements (Fig. 3.18) that meaningfully identify areas of the page as headers, footers, articles, navigation areas, asides, figures and more.



+ template

# Page-Structure Elements

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 3.18: sectionelements.html -->
4  <!-- New HTML5 section elements. -->
5  <html>
6      <head>
7          <meta charset="utf-8">
8          <title>New HTML5 Section Elements</title>
9      </head>
10
11     <body> APL
12         <header> <!-- header element creates a header for the page -->
13             <img src = "deitellogo.png" alt = "Deitel logo" />
14             <h1>Welcome to the Deitel Buzz Online</h1>
15
16             <!-- time element inserts a date and/or time -->
17             <time>2012-01-17</time>
18
19     </header>
20
21     <section id = "1"> <!-- Begin section 1 -->
22         <nav> <!-- nav element groups navigation links -->
23             <h2> Recent Publications</h2>
```

Fig. 3.18 | New HTML5 section elements. (Part 1 of 13.)

```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

        <ul>
            <li><a href = "http://www.deitel.com/books/iw3htp5">
                Internet & World Wide Web How to Program, 5/e</a></li>
            <li><a href = "http://www.deitel.com/books/androidfp/">
                Android for Programmers: An App-Driven Approach</a>
            </li>
            <li><a href = "http://www.deitel.com/books/iphonefp">
                iPhone for Programmers: An App-Driven Approach</a></li>
            <li><a href = "http://www.deitel.com/books/jhtp9/">
                Java How to Program, 9/e</a></li>
            <li><a href = "http://www.deitel.com/books/cpphtp8/">
                C++ How to Program, 8/e</a></li>
            <li>
                <a href = "http://www.deitel.com/books/vcsharp2010htp">
                    Visual C# 2010 How to Program, 4/e</a></li>
            <li><a href = "http://www.deitel.com/books/vb2010htp">
                Visual Basic 2010 How to Program</a></li>
            </ul>
        </nav>
    </section>

    <section id = "2"> <!-- Begin section 2 -->
        <h2>How to Program Series Books</h2>
        <h3><em>Java How to Program, 9/e</em></h3>
```

Fig. 3.18 | New HTML5 section elements. (Part 2 of 13.)

# Page-Structure Elements

```
48
49      <figure> <!-- figure element describes the image -->
50          <img src = "jhtp.jpg" alt = "Java How to Program, 9/e" />
51
52          <!-- figurecaption element inserts a figure caption -->
53          <figcaption><em>Java How to Program, 9/e</em>
54              cover.</figcaption>
55      </figure>
56
57      <!--article element represents content from another source -->
58      <article>
59          <header>
60              <h5>From
61                  <em>
62                      <a href = "http://www.deitel.com/books/jhtp9/">
63                          Java How to program, 9/e: </a>
64                  </em>
65              </h5>
66          </header>
67
```

Fig. 3.18 | New HTML5 section elements. (Part 3 of 13.)

```
68
69      <p>Features include:
70          <ul>
71              <li>Rich coverage of fundamentals, including
72                  <!-- mark element highlights text -->
73                  <mark>two chapters on control statements.</mark></li>
74              <li>Focus on <mark>real-world examples.</mark></li>
75              <li><mark>Making a Difference exercises set.</mark></li>
76              <li>Early introduction to classes, objects,
77                  methods and strings.</li>
78              <li>Integrated exception handling.</li>
79              <li>Files, streams and object serialization.</li>
80              <li>Optional modular sections on language and
81                  library features of the new Java SE 7.</li>
82              <li>Other topics include: Recursion, searching,
83                  sorting, generic collections, generics, data
84                  structures, applets, multimedia,
85                  multithreading, databases/JDBC&trade;, web-app
86                  development, web services and an optional
87                  ATM Object-Oriented Design case study.</li>
88          </ul>
```

Fig. 3.18 | New HTML5 section elements. (Part 4 of 13.)

# Page-Structure Elements

```
89    <!-- summary element represents a summary for the -->
90    <!-- content of the details element -->
91    <details>
92        <summary>Recent Edition Testimonials</summary>
93        <ul>
94            <li>"Updated to reflect the state of the
95                art in Java technologies; its deep and
96                crystal clear explanations make it
97                indispensable. The social-consciousness
98                [Making a Difference] exercises are
99                something really new and refreshing."
100               <strong>&mdash;José Antonio
101                  González Seco, Parliament of
102                  Andalusia</strong></li>
103               <li>"Gives new programmers the benefit of the
104                  wisdom derived from many years of software
105                  development experience."<strong>
106                      &mdash;Edward F. Gehringer, North Carolina
107                      State University</strong></li>
108               <li>"Introduces good design practices and
109                  methodologies right from the beginning.
110                  An excellent starting point for developing
111                  high-quality robust Java applications."
112                 <strong>&mdash;Simon Ritter,
113                     Oracle Corporation</strong></li>
114
115
116
117
118
119
120
121
122
123
124           </ul>
125       </details>
126   </p>
127 </article>
128
129   <!-- aside element represents content in a sidebar that's -->
130   <!-- related to the content around the element -->
131   <aside>
132       The aside element is not formatted by the browsers.
133   </aside>
134
```

Fig. 3.18 | New HTML5 section elements. (Part 5 of 13.)

Fig. 3.18 | New HTML5 section elements. (Part 6 of 13.)

# Page-Structure Elements

```
135 <h2>Deitel Developer Series Books</h2>
136 <h3><em>Android for Programmers: An App-Driven Approach
137   </em></h3>
138   Click <a href = "http://www.deitel.com/books/androidfp/">
139     here</a> for more information or to order this book.
140
141 <h2>LiveLessons Videos</h2>
142 <h3><em>C# 2010 Fundamentals LiveLessons</em></h3>
143   Click <a href = "http://www.deitel.com/Books/LiveLessons/">
144     here</a> for more information about our LiveLessons videos.
145 </section>
146
147 <section id = "3"> <!-- Begin section 3 -->
148   <h2>Results from our Facebook Survey</h2>
149   <p>If you were a nonprogrammer about to learn Java for the first
150     time, would you prefer a course that taught Java in the
151     context of Android app development? Here are the results from
152     our survey:</p>
153
154   <!-- meter element represents a scale within a range -->
155   0 <meter min = "0"
156     max = "54"
157     value = "14"></meter> 54
```

Fig. 3.18 | New HTML5 section elements. (Part 7 of 13.)

```
158   <p>Of the 54 responders, 14 (green) would prefer to
159     learn Java in the context of Android app development.</p>
160 </section>
161
162   <!-- footer element represents a footer to a section or page, -->
163   <!-- usually containing information such as author name, -->
164   <!-- copyright, etc. -->
165 <footer>
166   <!-- wbr element indicates the appropriate place to break a -->
167   <!-- word when the text wraps -->
168   <h6>&copy; 1992-2012 by Deitel & Associates, Inc.
169   All Rights Reserved.<h6>
170   <!-- address element represents contact information for a -->
171   <!-- document or the nearest body element or article -->
172 <address>
173   Contact us at <a href = "mailto:deitel@deitel.com">
174     deitel@deitel.com</a>
175 </address>
176   </footer>
177 </body>
178 </html>
```

Fig. 3.18 | New HTML5 section elements. (Part 8 of 13.)

# Page-Structure Elements

a) Chrome browser showing the header element and a nav element that contains an unordered list of links



b) Chrome browser showing the beginning of a section containing a figure and a figurecaption



Fig. 3.18 | New HTML5 section elements. (Part 9 of 13.)

Fig. 3.18 | New HTML5 section elements. (Part 10 of 13.)

# Page-Structure Elements

c) Chrome browser showing an **article** containing a **header**, some content and a collapsed **details** element, followed by an **aside** element

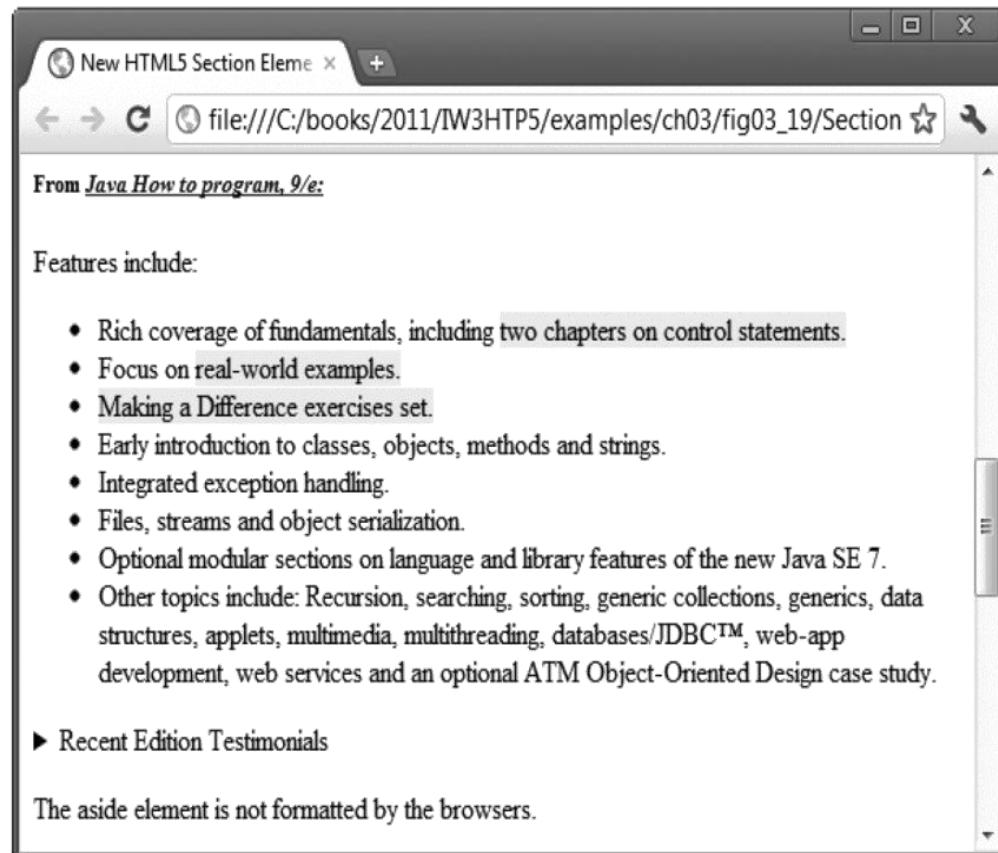


Fig. 3.18 | New HTML5 section elements. (Part 11 of 13.)

d) Chrome browser showing the end of the **section** that started in part (b)



Fig. 3.18 | New HTML5 section elements. (Part 12 of 13.)

# Page-Structure Elements

e) Chrome browser showing the last **section** containing a **meter** element, followed by a **footer** element



**Fig. 3.18 | New HTML5 section elements. (Part 13 of 13.)**

# Header Element

- The **header element** creates a header for this page that contains both text and graphics.  
*any*
- The header element can be used multiple times on a page and can include HTML headings (`<h1>` through `<h6>`), navigation, images and logos and more.
- ***time Element***
- The **time element**, which does not need to be enclosed in a header, enables you to identify a date (as we do here), a time or both.

## Nav Element

- The **nav element** groups **navigation links**. *Pages ↪*
- In this example, we used the heading Recent Publications and created a **ul** element with seven **li** elements that link to the corresponding web pages for each book.

## **Figure Element and Figcaption Element**

- The **figure element** describes a figure (such as an image, chart or table) in the document so that it could be moved to the side of the page or to another page.
- The **figcaption element** provides a caption for the image in the figure element.

## *Content* Article Element

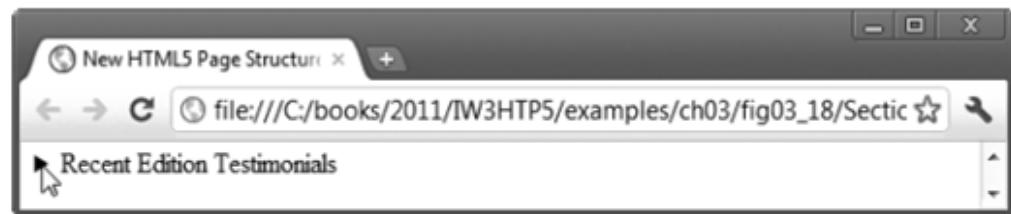
- The **article** element describes standalone content that could potentially be used or distributed elsewhere, such as a news article, forum post or blog entry.
- You can nest article elements. For example, you might have reader comments about a magazine nested as an article within the magazine article.

## Summary Element and Details Element

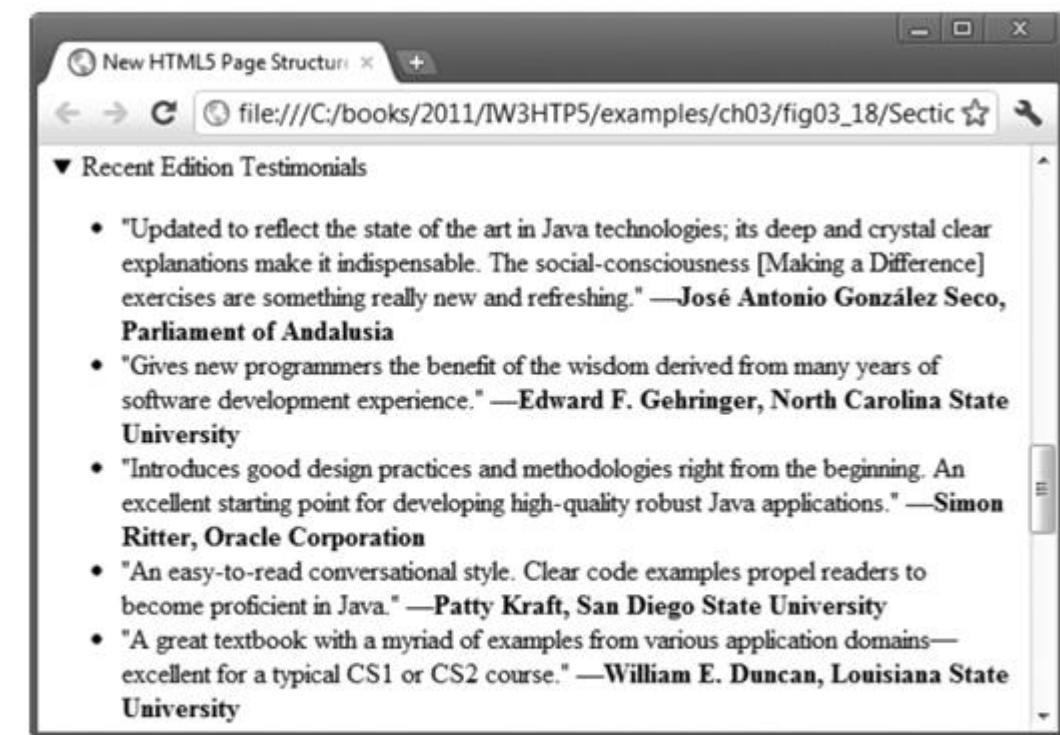
- The **summary element** displays a right-pointing arrow next to a summary or caption when the document is rendered in a browser (Fig. 3.19).
- When clicked, the arrow points downward and reveals the content in the **details element**.

# Summary Element and Details Element

✓ by



**Fig. 3.19 | Demonstrating the summary and detail elements.  
(Part 1 of 2.)**



**Fig. 3.19 | Demonstrating the summary and detail elements.  
(Part 2 of 2.)**

## Section Element

- The **section element** describes a section of a document, usually with a heading for each section—these elements can be nested.
- In this example, we broke the document into three sections—the first is Recent Publications.  
*+ eos}*
- The section element may also be nested in an article.

## Aside Element

- The **aside element** describes content that's related to the surrounding content (such as an article) but is somewhat ~~separate from the flow of the text.~~  
*Mention*
- For example, an aside in a news story might include some background history.

## Meter Element

- The **meter element** renders a visual representation of a measure within a range (Fig. 3.20).
- In this example, we show the results of a recent web survey we did.
- The min attribute is "0" and a max attribute is "54" —indicating the total number of responses to our survey.
- The value attribute is "14", representing the total number of people who responded “yes” to our survey question.

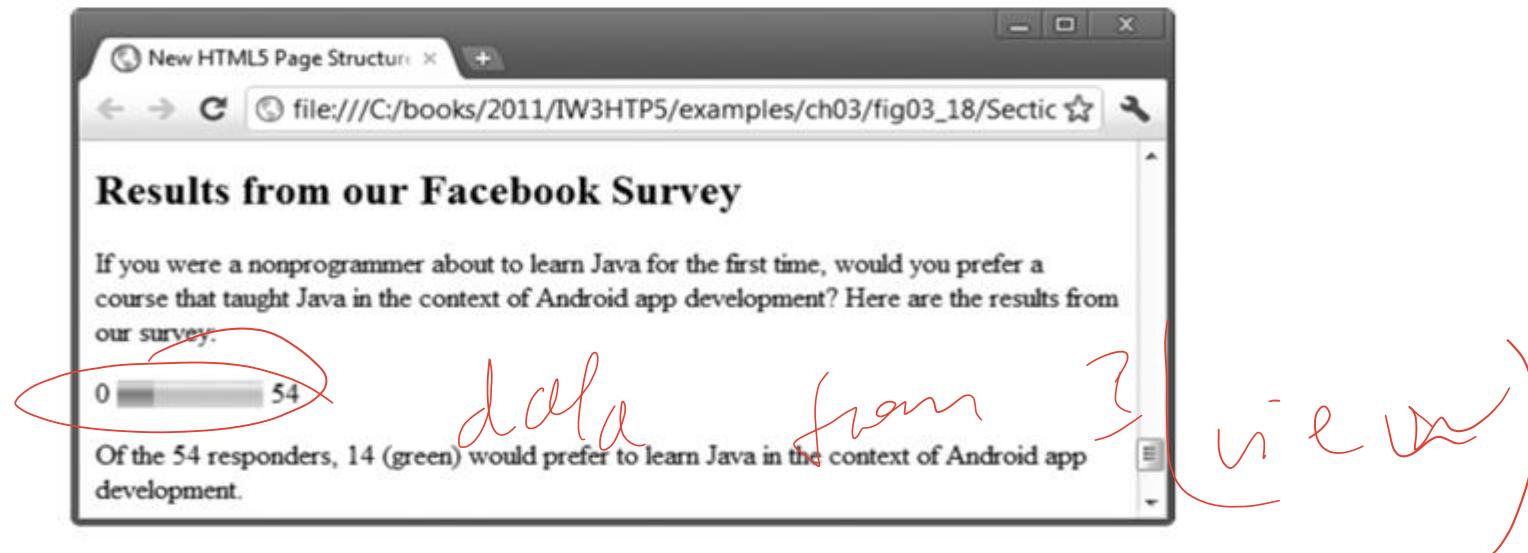


Fig. 3.20 | Chrome rendering the meter element.

## Footer Element

(c) 2016 Pearson Education, Inc.

- The **footer element** describes a *footer*—content that usually appears at the bottom of the content or section element.
- In this example, we use the footer to describe the copyright notice and contact information.

Refrence | go

## **Text-Level Semantics: Mark Element and wbr Element**

- The **mark element** highlights the text that's enclosed in the element.
- The **wbr element** indicates the appropriate place to break a word when the text wraps to multiple lines.
- You might use wbr to prevent a word from breaking in an awkward place.

# Traditional HTML Layouts

```
<body>
```

```
  <div id="page">
```

```
    <div id="header">
```

```
      <div id="nav">
```

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

```
      <div class="article">
```

```
        <div class="article">
```

```
      <div id="sidebar">
```

```
    <div id="footer">
```

# New HTML5 Layout Elements

```
<body>
```

```
  <div id="page">
```

```
    <header>
```

```
    <nav>
```

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

```
    <article>
```

```
    <article>
```

```
  <aside>
```

```
  <footer>
```

# UCCD2323 Front-End Web Development



## Chapter 2 Formatting and Presenting Information Part 2.

# Topic Outline

- **DEFINITION MULTIMEDIA**
- **HTML VIDEO**
- **HTML AUDIO**
- **HTML PLUG-INS**
- **HTML OBJECT TAG**
- **HTML EMBED TAG**
- **MULTIMEDIA ACCESSIBILITY**

## Objectives

- **At the end of this lecture, students will be able to:**
- 1. Enhancing the web pages with sound and video in Web Pages.**

# What is Multimedia?

- Multimedia comes in many different formats. It can be almost anything you can hear or see.
- Examples: Images, music, sound, videos, records, films, animations, and more.
- Web pages often contain multimedia elements of different types and formats.

# Common Video Formats

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5 (See MP4).
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers. (See MP4)
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
WebM	.webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube.

# HTML Video – Browser Support

Browser	MP4	WebM	Ogg
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

The numbers in the table specify the first browser version that fully supports the `<video>` element.

Element					
<code>&lt;video&gt;</code>	4.0	9.0	3.5	4.0	10.5

# HTML Video – Media Types

File Format	Media Type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

## HTML5 ~~Video~~

- The controls attribute adds video controls, like play, pause, and volume.
- It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

```
<video controls poster="sparky.jpg"
       width="160" height="150">
    <source src="sparky.mp4" type="video/mp4">
    <source src="sparky.ogv" type="video/ogg">
    <a href="sparky.mov">Sparky the Dog</a> (.mov)
</video>
```

# HTML5 Video

- The `<source>` element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
- The text between the `<video>` and `</video>` tags will only be displayed in browsers that do not support the `<video>` element.

## HTML5 Video Tags

Tag	Description
<code>&lt;video&gt;</code>	Defines a <u>video or movie</u>
<code>&lt;source&gt;</code>	Defines multiple media resources for <u>media elements</u> , such as <code>&lt;video&gt;</code> and <code>&lt;audio&gt;</code>

# HTML5 Video Autoplay

- To start a video automatically use the **autoplay** attribute
- **loop** attribute specifies that the video will start over again, every time it is finished.
- M4V (.m4v) is developed by Apple Inc. and it is specifically designed for Apple products such as the iPhone, iTunes store, iPod, etc.
- MP4 (.mp4) is developed by Moving Picture Experts Group (MPEG) and it can be played by most media players and devices. It is one of most popular video format streaming on the Internet thanks to its great compression and low-bandwidth requirement.

```
<video controls poster="sparky.jpg"  
       width="160" height="150" autoplay loop>  
  <source src="sparky.m4v" type="video/mp4">  
  <source src="sparky.ogv" type="video/ogg">  
  <a href="sparky.mov">Sparky the Dog</a> (.mov)  
</video>
```

# HTML5 Video Optional Attributes

Attribute	Value	Description
<u>autoplay</u>	autoplay	Specifies that the video will start playing as soon as it is ready
<u>controls</u>	controls	Specifies that video controls should be displayed (such as a play/pause button etc).
<u>height</u>	<i>pixels</i>	Sets the height of the video player
<u>loop</u>	loop	Specifies that the video will start over again, every time it is finished
<u>muted</u>	muted	Specifies that the audio output of the video should be muted
<u>poster</u>	<i>URL</i>	Specifies an image to be shown while the video is downloading, or until the user hits the play button
<u>preload</u>	auto metadata none	Specifies if and how the author thinks the video should be loaded when the page loads
<u>src</u>	<i>URL</i>	Specifies the URL of the video file
<u>width</u>	<i>pixels</i>	Sets the width of the video player

# Common Audio Formats

Format	File	Description
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Commonly used in music players. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML5.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. MP4 video is the upcoming video format on the internet. This leads to automatic support for MP4 audio by all browsers.

# HTML Audio – Browser Support

There are three supported audio formats: MP3, WAV, and OGG. The browser support for the different formats is:

Browser	MP3	WAV	OGG
Edge/IE	YES	YES*	YES*
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

\*From Edge 79

The numbers in the table specify the first browser version that fully supports the `<audio>` element.

Element					
<code>&lt;audio&gt;</code>	4.0	9.0	3.5	4.0	10.5

# HTML Audio – Media Types

## File Format

## Media Type

MP3

audio/mpeg

Ogg

audio/ogg

Wav

audio/wav

## HTML5 Audio

```
<audio controls>
    <source src="soundloop.mp3" type="audio/mpeg">
    <source src="soundloop.ogg" type="audio/ogg">
    <a href="soundloop.mp3">Download the Audio File</a> (MP3)
</audio>
```

# HTML5 Audio

- The `controls` attribute adds audio controls, like play, pause, and volume.
- The `<source>` element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.
- The text between the `<audio>` and `</audio>` tags will only be displayed in browsers that do not support the `<audio>` element.

## HTML5 Audio Tags

Tag	Description
<code>&lt;audio&gt;</code>	Defines sound content
<code>&lt;source&gt;</code>	Defines multiple media resources for media elements, such as <code>&lt;video&gt;</code> and <code>&lt;audio&gt;</code>

# HTML5 Audio Attributes

Attribute	Value	Description
<u>autoplay</u>	autoplay	Specifies that the audio will start playing as soon as it is ready
<u>controls</u>	controls	Specifies that audio controls should be displayed (such as a play/pause button etc)
<u>loop</u>	loop	Specifies that the audio will start over again, every time it is finished
<u>muted</u>	muted	Specifies that the audio output should be muted
<u>preload</u>	auto metadata none	Specifies if and how the author thinks the audio should be loaded when the page loads
<u>src</u>	URL	Specifies the URL of the audio file

# HTML Plug - ins

- Adobe Flash Player
- Adobe Reader
- Windows Media Player
- Apple Quicktime
- Java Applets
  
- Plug-ins can be added to web pages with the <object> tag or the <embed> tag.

## Warning !

Most browsers no longer support Java Applets and Plug-ins.

ActiveX controls are no longer supported in any browser.

The support for Shockwave Flash has also been turned off in modern browsers.

# HTML <object> Tag

- The <object> element is supported by all browsers.
- The <object> element defines an embedded object within an HTML document.
- It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages.

```
<object data="fall15.swf" width="640" height="100"></object>
```

## HTML <embed> Tag

- The <embed> element is supported in all major browsers.
- The <embed> element also defines an embedded object within an HTML document.
- Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

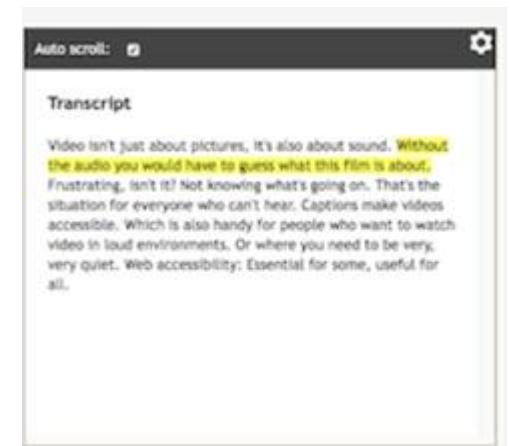
```
<embed type="application/x-shockwave-flash" src="fall15.swf"
       width="640" height="100"
       quality="high" title="Fall Nature Hikes">
```

# HTML <embed> Attributes

Attribute	Value	Description
<u>height</u>	<i>pixels</i>	Specifies the height of the embedded content
<u>src</u>	<i>URL</i>	Specifies the address of the external file to embed
<u>type</u>	<i>media_type</i>	Specifies the media type of the embedded content
<u>width</u>	<i>pixels</i>	Specifies the width of the embedded content

# Multimedia Accessibility

- Multimedia accessibility ensure that website work is accessible to all, including people with disabilities who cannot hear or see audio.
- **Transcript** - required. For most W3C media, such as recordings of teleconferences, you only need to get/make and post a transcript to provide basic accessibility.
- **Captions** - nice to have for most W3C media, required for some. (Captions are essentially the transcript synchronized with the video or audio.) Captions are important when people need to see what's happening in the video and get the audio information in text at the same time



# UCCD2323 Front-End Web Development



## Chapter 2 Responsive Layout Basics.

# Learning Outcomes

A Syah

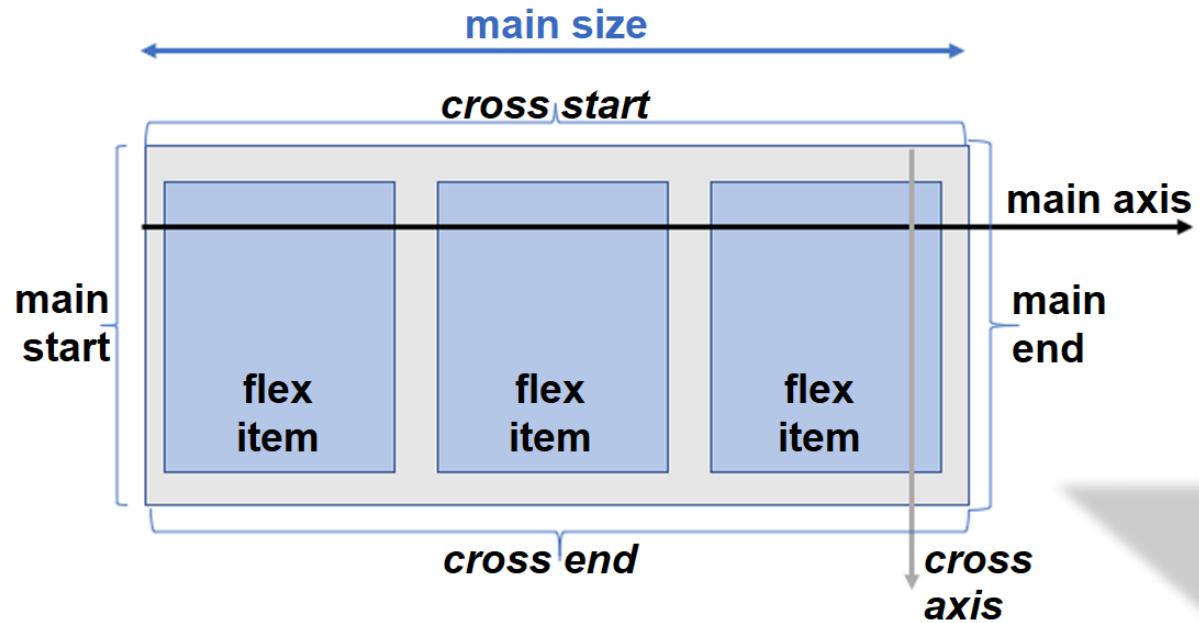
- Describe the purpose of CSS Flexible Box Layout
- Configure a web page that applies CSS Flexible Box Layout
- Describe the purpose of CSS Grid Layout
- Configure a grid container
- Configure grid rows, grid columns, grid gaps, and grid areas
- Create responsive page layouts with CSS Grid Layout
- Configure web pages for mobile display using the viewport meta tag
- Apply responsive web design techniques with CSS media queries
- Apply responsive image techniques using the picture element
- Apply responsive image techniques with the img element's loading attribute

# CSS Flexible Box Layout aka Flexbox

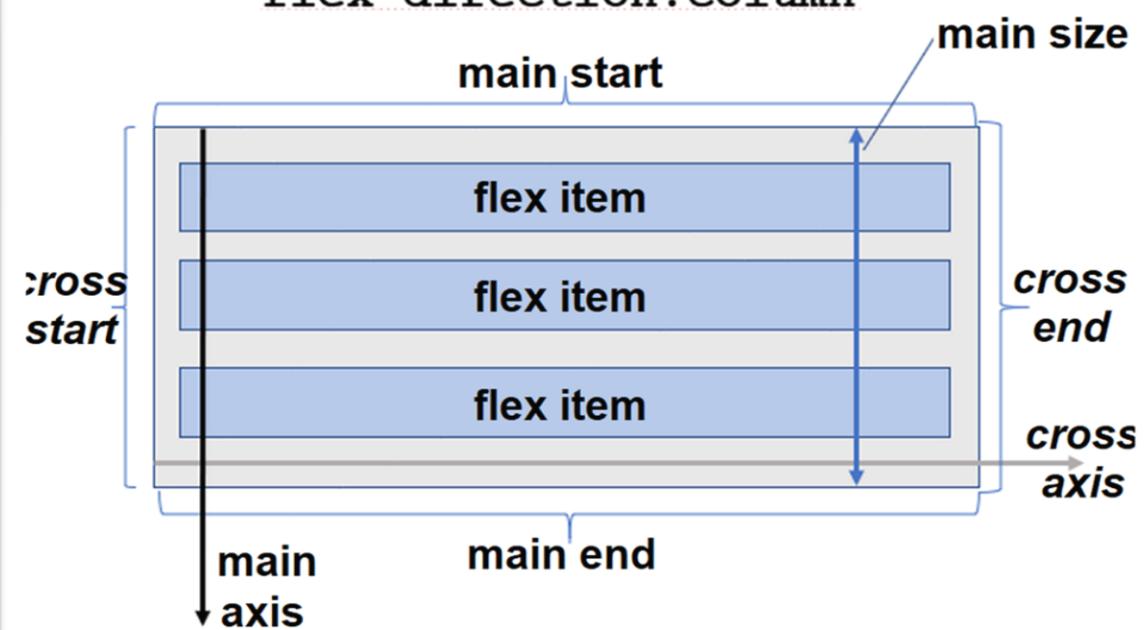
- **Purpose:**
- Provide for a flexible, responsive layout  
Best used for one dimension – a row or a column
- **The display property**
  - Configures a flexbox container display: flex;
- **Flex Item -- a child element of the flex container**
- **The flex-wrap property**
  - Determines whether flex items are displayed on multiple lines
    - Values: nowrap (default), wrap, wrap-reverse
- **The flex-direction property**
  - Configures the flow direction
    - Values: row,(default), column, row-reverse, and column-reverse

# Diagram of a Flex Container

flex container with flex-direction:row

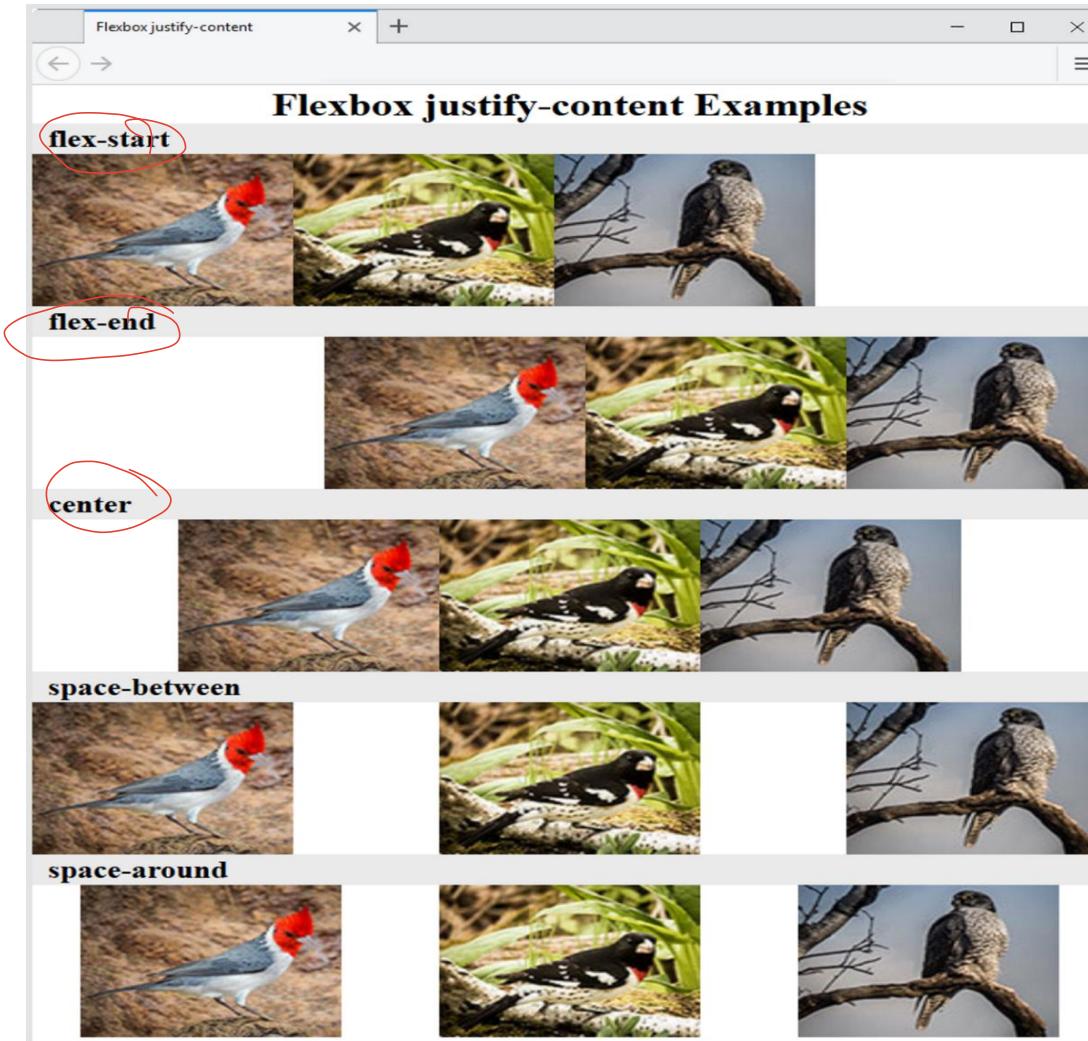


flex container with flex-direction:column



# The justify - content Property

- Configures how the extra space along the main axis should be displayed



# More flexbox Properties

- **The align-items Property**

Configures how the browser displays extra space along the cross-axis

- **The flex-flow Property**

Shorthand to configure flex-direction and flex-wrap properties

- **The order Property**

Causes the browser to display flex items in different order than they are coded

Warning – this could be an accessibility issue

- **The gap Property**

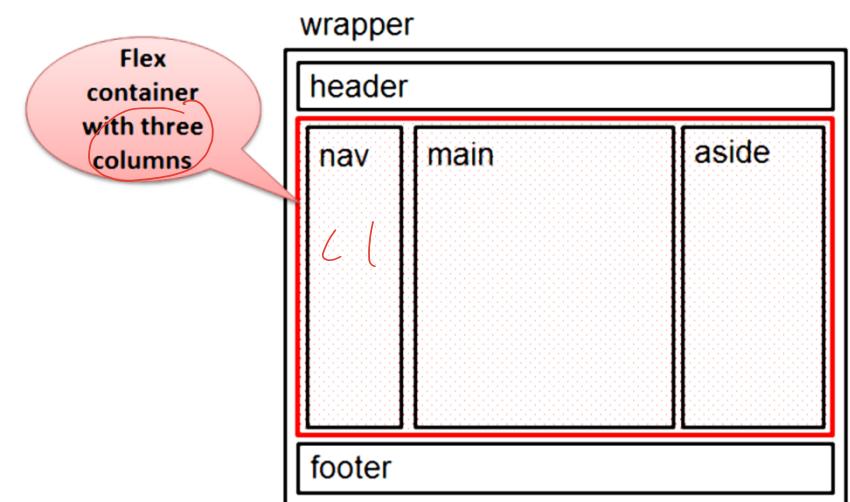
Configures space between adjacent items

Shorthand to configure column-gap and row-gap

# Configure Flex Items

- By default flex items are flexible in size and allocated the same amount of space in the flex container
- The flex property
  - Customizes the size of each flex item
  - Indicated the flex grow factor
  - Indicates the flex shrink factor
  - Can be used to indicate a proportional flexible item

```
nav { flex: 1; }  
main { flex: 7; } ratio  
aside { flex: 2; }
```



# Configure Flex Items

```
<html>  
  
<head>  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <style>  
    /* Common styles for all displays */  
    .wrapper {  
      max-width: 1200px;  
      margin: 30px;  
    }  
  
    header,  
    nav,  
    main,  
    aside,  
    footer {  
      padding: 10px;  
      border: 1px solid black;  
    }  
  
    header {  
      background-color: lightblue;  
    }  
  
    nav {  
      background-color: lightgreen;  
    }  
  
    main {  
      background-color: lightyellow;  
    }  
  
    aside {  
      background-color: lightpink;  
    }  
  
    footer {  
      background-color: lightgray;  
    }  
  </style>  
</head>
```

```
/* Large Display */  
@media (min-width: 901px) {  
  .wrapper {  
    display: flex;  
    flex-direction: column;  
    margin: 30px;  
  }  
  
  div.wrapper2 {  
    display: flex;  
    flex-direction: row;  
  }  
  
  nav {  
    flex: 1;  
  }  
  
  main {  
    flex: 7;  
  }  
  
  aside {  
    flex: 1;  
  }  
}  
</style>  
</head>
```

# Configure Flex Items

```
<body>
  <div class="wrapper">
    <header>Header</header>
    <div class="wrapper2">
      <nav>Navigation</nav>
      <main>
        <p>Main Content</p>
        <p> Bacon ipsum dolor amet proident ea adipisicing incididunt t-bone, eiusmod landjaeger anim bresaola  

          beef fatback labore. Commodo consectetur exercitation beef incididunt. Salami velit fugiat rump  

          ground round aliqua jerky flank turkey chicken. Leberkas corned beef alcatra tail ut ham hock  

          officia sint chuck tenderloin. Sint frankfurter corned beef, laborum chislic ribeye bacon ullamco  

          meatball. Qui adipisicing burgdoggen pancetta do chislic. Duis shankle ex, chuck est adipisicing  

          commodo ad kevin brisket meatloaf eu shank ribeye.
        </p>

        <p> Aute capicola chuck, est irure porchetta prosciutto reprehenderit chislic proident anim ham jerky  

          ipsum. In quis deserunt salami buffalo boudin rump nulla pancetta shankle strip steak. Quis cow  

          shankle, pariatur ipsum sausage porchetta leberkas jowl commodo ad pork chop. Ex cupim reprehenderit  

          consequat ut enim t-bone tenderloin jowl pork belly.
        </p>
      </main>
      <aside>Sidebar</aside>
    </div>
    <footer>Footer</footer>
  </div>
</body>

</html>
```

# CSS Grid Layout

- Purpose: Configure a two-dimensional grid-based layout

The grid can be fixed in dimension  
or flexible and responsive.

- The display property

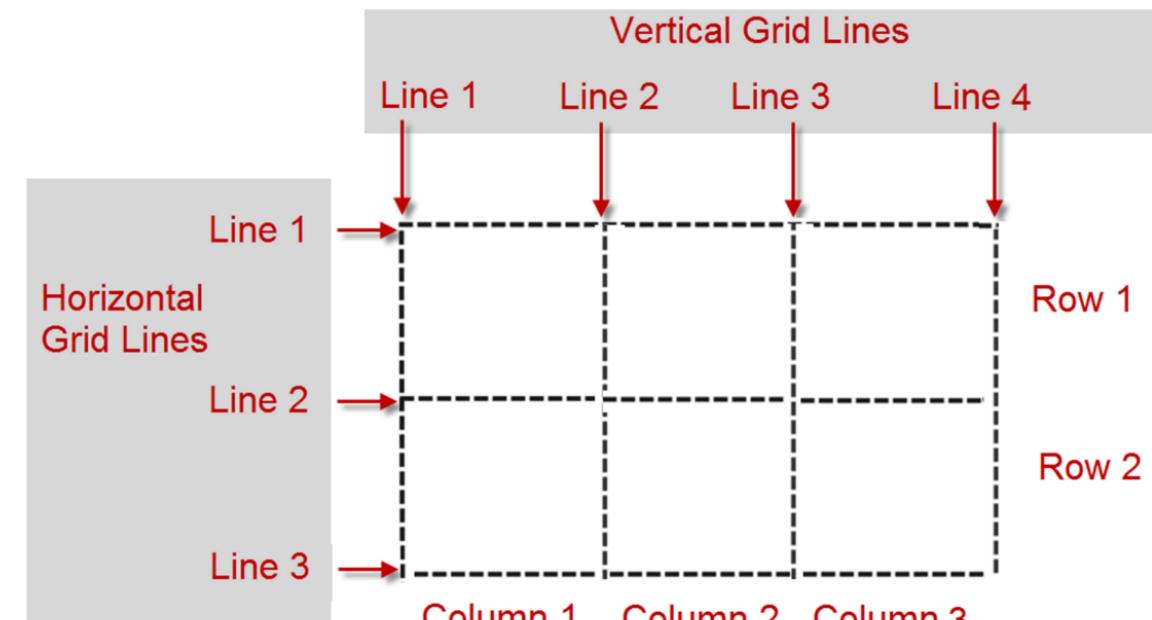
- Configures a grid container  
`display: grid;`

- Grid Item -- a child element of the grid container

- Grid Terms

- Grid line
- Grid row
- Grid column
- Grid track
- Grid gap
- Grid area

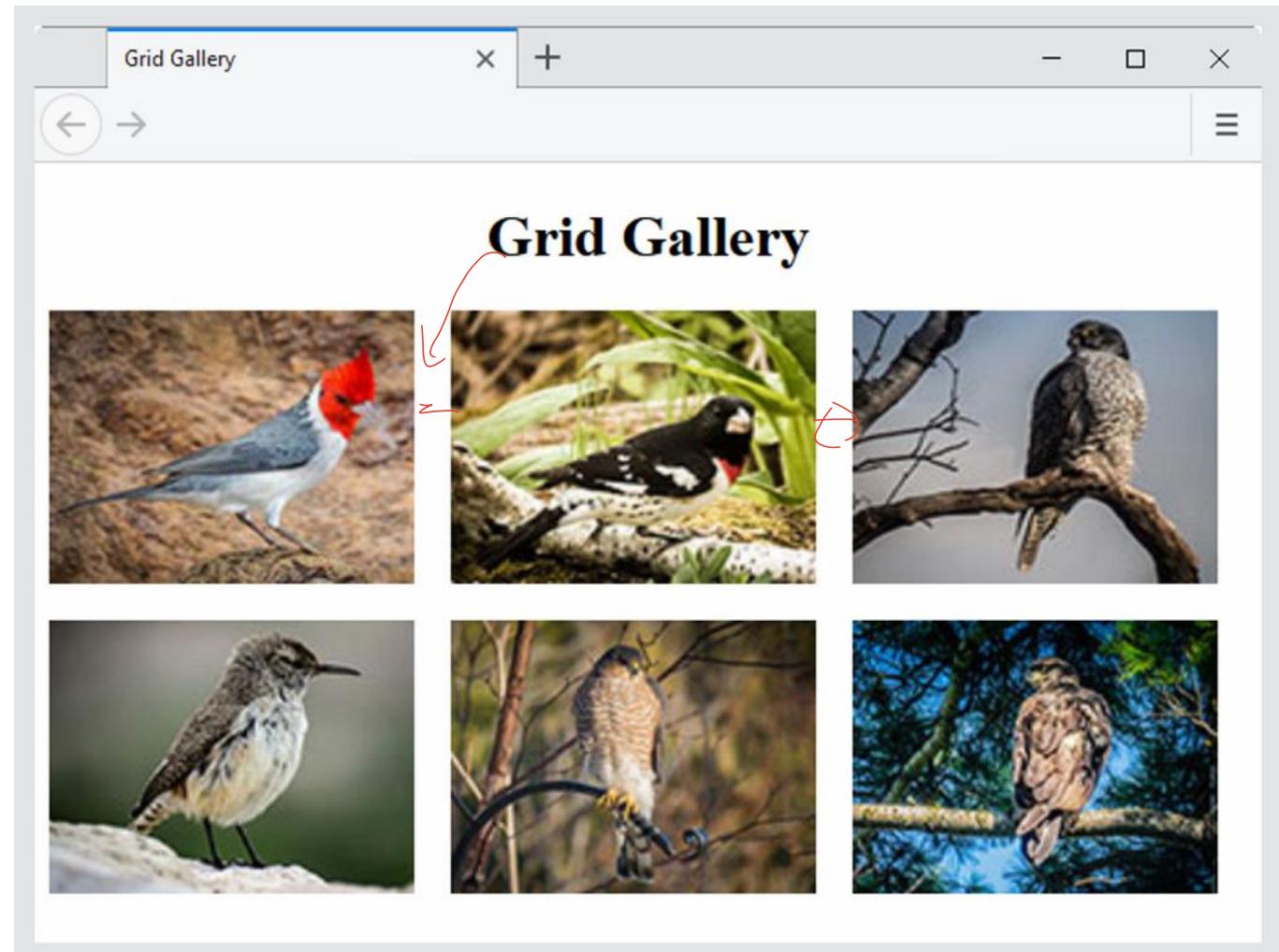
- The order property



# Configure Grid Columns and Grid Rows

- The grid-template-columns property
- The grid-template-rows property
- Example:

```
#gallery { display: grid;  
grid-template-columns: 220px 220px 220px;  
grid-template-rows: 170px 170px; }
```



# Grid Columns & Grid Rows

- Commonly used values for grid-template-columns and grid-template-rows
- <https://www.w3.org/TR/css-grid-1/#propdef-grid-template-columns>

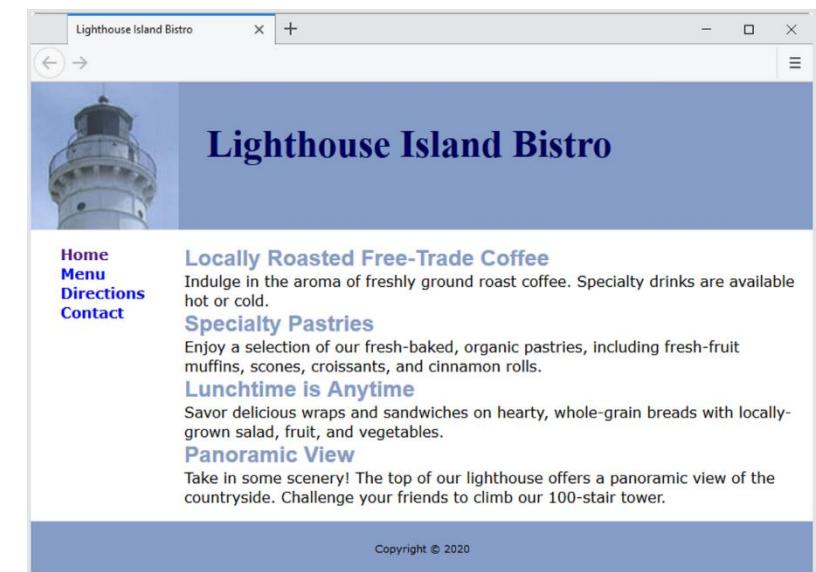
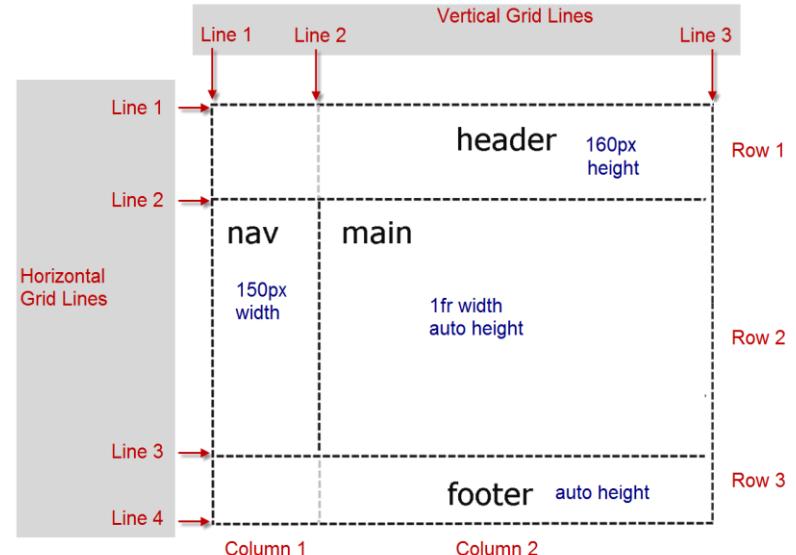
Value	Description
<b>numeric length unit</b>	Configures a fixed size with a length unit such as px or em Example: 220px
<b>numeric percentage</b>	Configures a percentage size; Example: 20%
<b>numeric fr unit</b>	Configures a flex factor unit (denoted by fr) that directs the browser to allocate a fractional part of the remaining space
<b>auto</b>	Configures a size to hold the maximum content
<b>minmax (min, max)</b>	Configures a size range greater or equal to min value and less than or equal to max value. The max can be set to a flex factor.
<b>repeat(repetition amount, format value)</b>	Repeats the column or row the number of times specified by the repetition amount numeric value or keyword and uses the format value to configure the column or row. The auto-fill keyword indicates to repeat but stop before an overflow. Example: repeat (autofill, 250px)

# Configure Grid Items

- The **grid-row** property
  - configures the area in rows that is reserved for the item in the grid
- The **grid-column** property
  - configures the area in columns that is reserved for the item in the grid
- Grid Line Numbers
  - Identify the starting and ending line number separated by a / character

put in standard

```
header { grid-row: 1 / 2; grid-column: 1 / 3; }
nav { grid-row: 2 / 3; grid-column: 1 / 2; }
main { grid-row: 2 / 3; grid-column: 2 / 3; }
footer { grid-row: 3 / 4; grid-column: 1 / 3; }
```



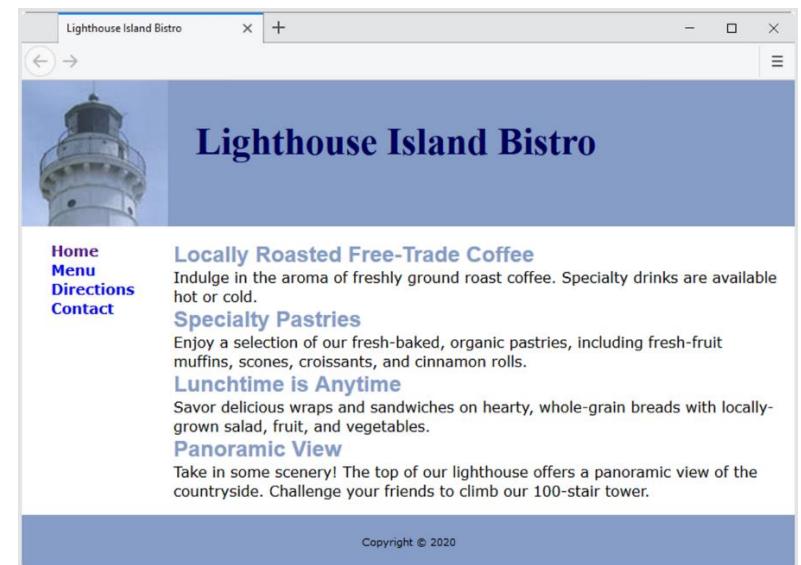
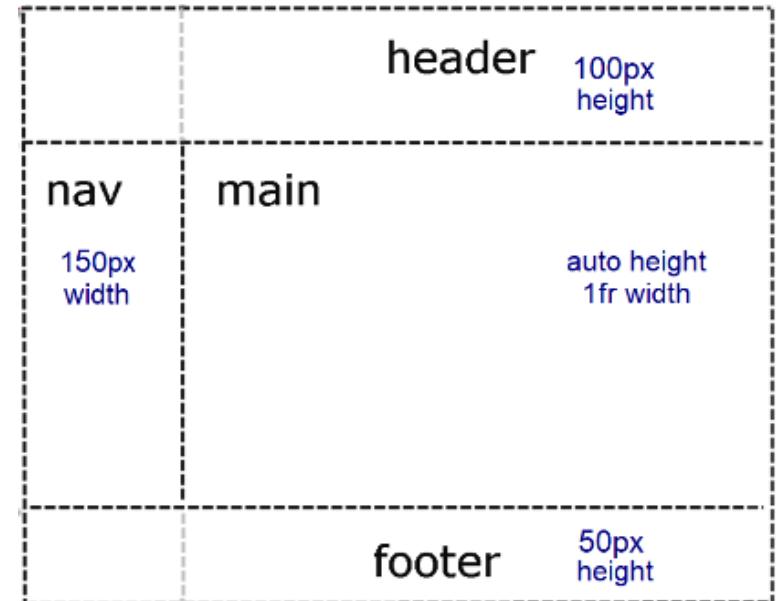
# Configure Grid Areas (1)

- grid-area Property

```
header { grid-area: header; }
nav { grid-area: nav; }
main { grid-area: main; }
footer { grid-area: footer; }
```

- grid-template-areas  
Property

```
#wrapper { display: grid;
  grid-template-columns: 150px 1fr;
  grid-template-rows: 100px auto 50px;
  grid-template-areas:
    "header header"
    "nav main"
    "footer footer"; }
```



# Configure Grid Areas (1)

```
<html>  
  
<head>  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  
  <style>  
    /* Common styles for all displays */  
    .wrapper {  
      max-width: 1200px;  
      margin: 0 auto;  
    }  
  
    header,  
    nav,  
    main,  
    aside,  
    footer {  
      padding: 10px;  
      border: 1px solid black;  
    }  
  
    header {  
      background-color: lightblue;  
    }  
  
    nav {  
      background-color: lightgreen;  
    }  
  
    main {  
      background-color: lightyellow;  
    }  
  
    footer {  
      background-color: lightgray;  
    }  
  </style>
```

```
/* Large Display */  
@media (min-width: 901px) {  
  .wrapper {  
    display: grid;  
  
    header {  
      grid-area: header;  
    }  
    nav {  
      grid-area: nav;  
    }  
    main {  
      grid-area: main;  
    }  
    footer {  
      grid-area: footer;  
    }  
    header {  
      grid-row: 1 / 2;  
      grid-column: 1 / 3;  
    }  
    nav {  
      grid-row: 2 / 3;  
      grid-column: 1 / 2;  
    }  
    main {  
      grid-row: 2 / 3;  
      grid-column: 2 / 3;  
    }  
    footer {  
      grid-row: 3 / 4;  
      grid-column: 1 / 3;  
    }  
  }  
  grid-template-columns: 150px 1fr 150px;  
  grid-template-rows: 100px auto 50px;  
  grid-template-areas: "header header header"  
                      "nav main"  
                      "footer footer footer";  
}</style>
```

# Configure Grid Areas (1)

```
</head>

<body>
  <div class="wrapper">
    <header>Header</header>
    <nav>Navigation</nav>
    <main>
      <p>Main Content</p>
      <p> Bacon ipsum dolor amet proident ea adipisicing incididunt t-bone, eiusmod landjaeger anim bresaola
          beef fatback labore. Commodo consectetur exercitation beef incididunt. Salami velit fugiat rump
          ground round aliqua jerky flank turkey chicken. Leberkas corned beef alcatra tail ut ham hock
          officia sint chuck tenderloin. Sint frankfurter corned beef, laborum chislic ribeye bacon ullamco
          meatball. Qui adipisicing burgdoggen pancetta do chislic. Duis shankle ex, chuck est adipisicing
          commodo ad kevin brisket meatloaf eu shank ribeye.
      </p>

      <p> Aute capicola chuck, est irure porchetta prosciutto reprehenderit chislic proident anim ham jerky
          ipsum. In quis deserunt salami buffalo boudin rump nulla pancetta shankle strip steak. Quis cow
          shankle, pariatur ipsum sausage porchetta leberkas jowl commodo ad pork chop. Ex cupid reprehenderit
          consequat ut enim t-bone tenderloin jowl pork belly.
      </p>
    </main>
    <footer>Footer</footer>
  </div>
</body>

</html>
```

# Configure Grid Areas (2)

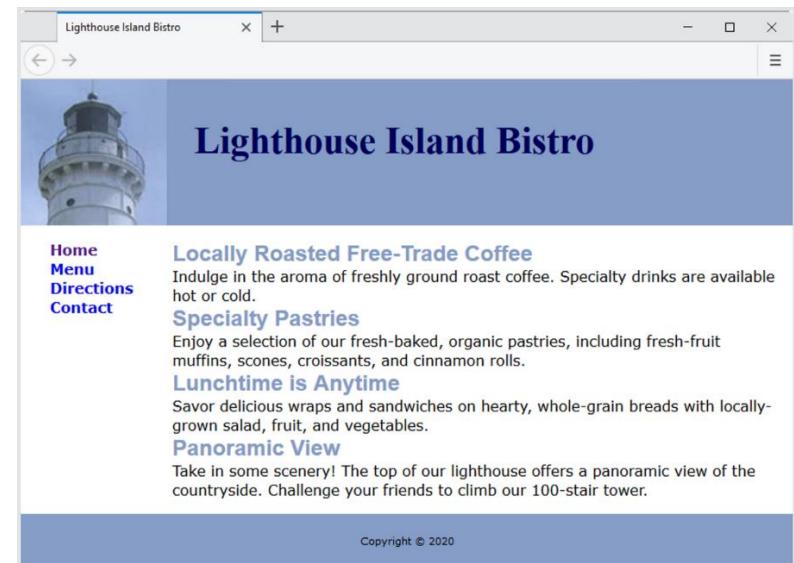
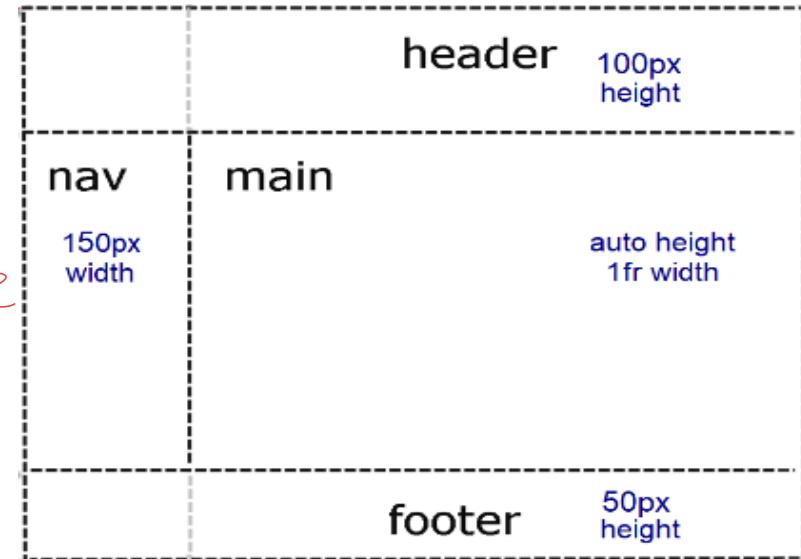
- grid-area Property

```
header { grid-area: header; }  
nav { grid-area: nav; }  
main { grid-area: main; }  
footer { grid-area: footer; }
```

for add free else  
Layout width

- grid-template Property

```
#wrapper { display: grid;  
grid-template:  
"header header" 100px  
"nav main"      auto  
"footer footer" 50px  
/ 150px 1fr; }
```



# Progressive Enhancement with Grid

- **CSS Feature Query**
  - A **feature query** is a conditional that can be used to test for support of a CSS property, and if support is found, apply the specified style rules.

```
@supports ( display: grid ) {  
.... grid styles go here ...  
}
```
- **Progressive Enhancement Strategy**
  - Configure web page layout with normal flow or float for browsers and devices that do not support grid
  - Configure a feature query with grid layout for modern browsers

# Viewport Meta Tag

- Default action for most mobile devices is to zoom out and scale the web page
- **Viewport Meta Tag**
  - Created as an Apple extension to configure display on mobile devices
  - Configures width and initial scale of browser viewport

<meta name="viewport" content="width=device-width, initial-scale=1.0">  
density ~



# Telephone & Text Message Hyperlinks

- **Telephone Scheme**

<a href="tel:888-555-5555">Call 888-555-5555</a>

- Many mobile browsers will initiate a phone call when the hyperlink is clicked.

- **SMS Scheme**

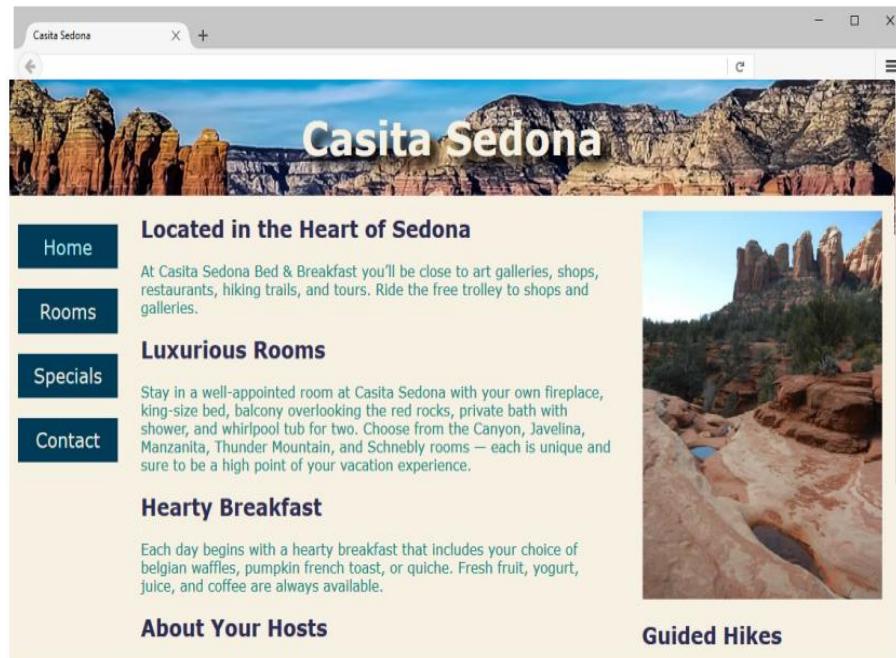
<a href="sms:888-555-5555">Text 888-555-5555</a>

- Many mobile browsers will initiate a text message to the phone number when the hyperlink is clicked.

# Responsive Web Design

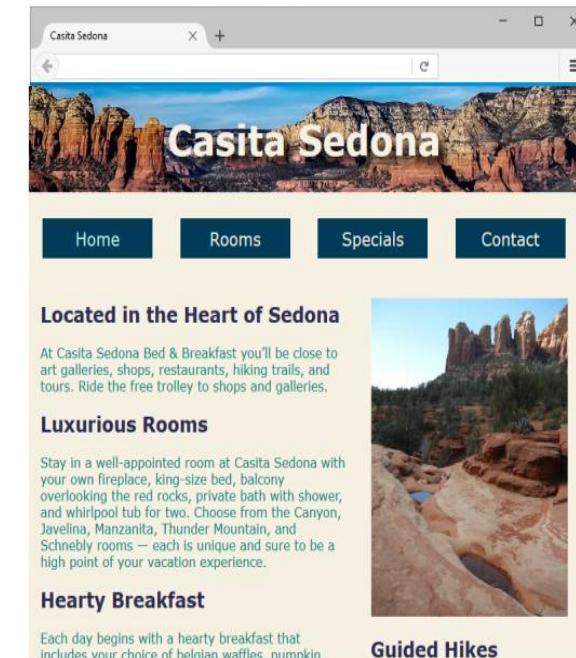
- Term coined by Ethan Marcotte
- Progressively enhancing a web page for different viewing contexts
- **Techniques:**
  - Fluid Layout
  - CSS Media Queries
  - Flexible Images

Desktop Browser



~~Interact~~

Tablet  
Display Width



Smartphone  
Display Width



# Mobile First Approach

- Responsive design layout strategy
- Term coined by Luke Wroblewski
- The Mobile First Process:
  1. Configure a single-column page layout for narrow screens – smartphones!  
Test with a small browser window if needed.
  2. Resize the browser viewport to be larger until the design “breaks” and needs to be reworked for a pleasing display—this is the point where you may need to code a media query.
  3. Continue resizing the browser viewport to be larger until the design breaks and code additional media queries.

# CSS Media Queries

- **Media Query**
  - Determines the capability of the mobile device, such as screen resolution
  - Directs the browser to styles configured specifically for those capabilities

- **Link Element Example:**

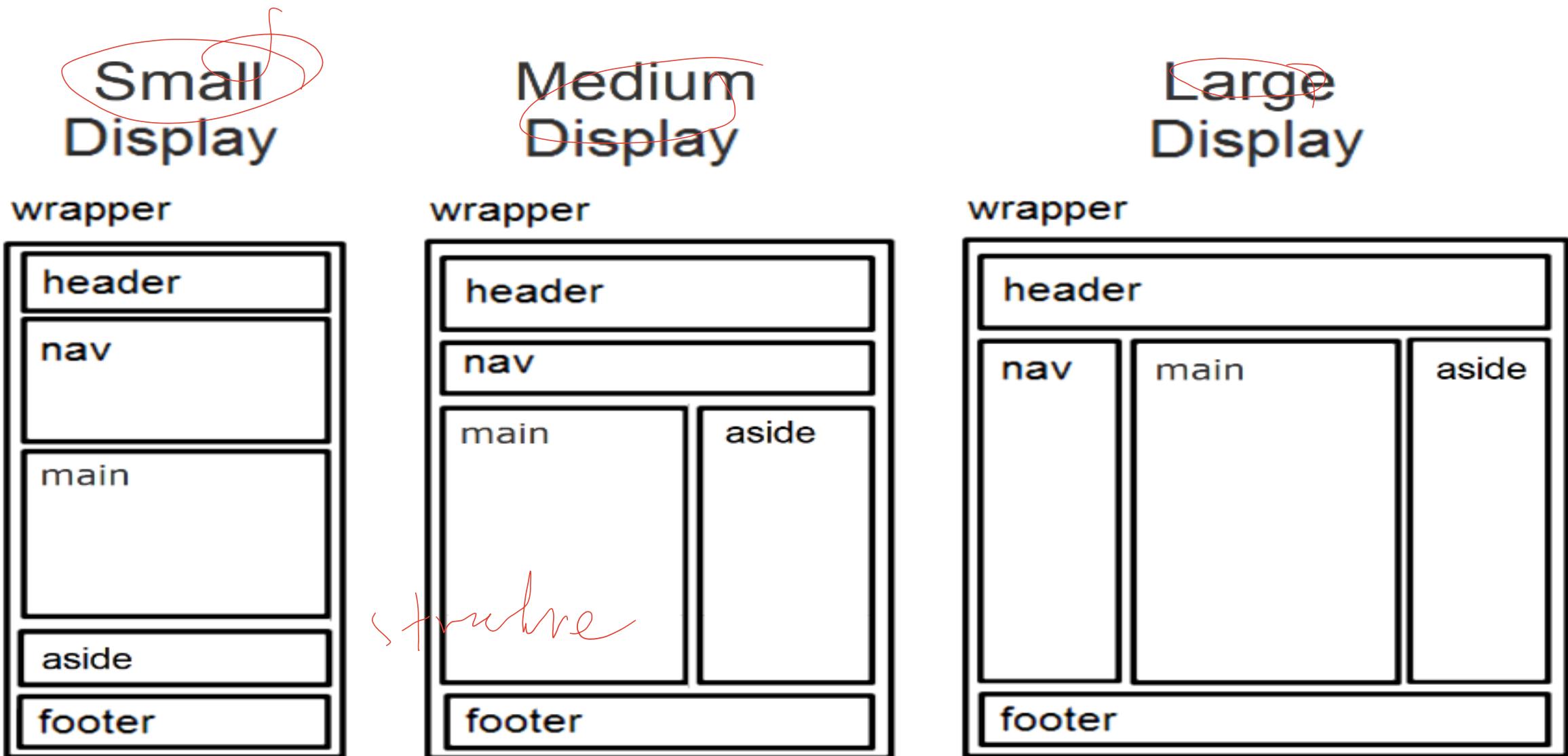
```
<link href="lighthousemobile.css" media="only screen and (max-device-width: 480px)">
```

- **CSS Example:**

```
@media only screen and (max-width: 480px) {  
    header { background-image: url(mobile.gif);  
    }  
}
```



# Responsive Grid Layout with Media Queries



# Flexible Images

- **Edit HTML:**  
remove height and width attributes

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

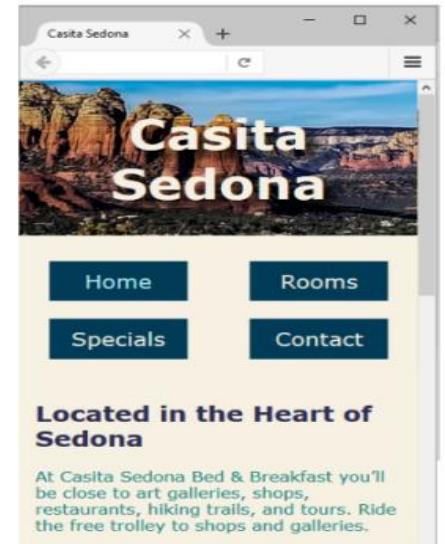
Desktop Browser



Tablet Display Width



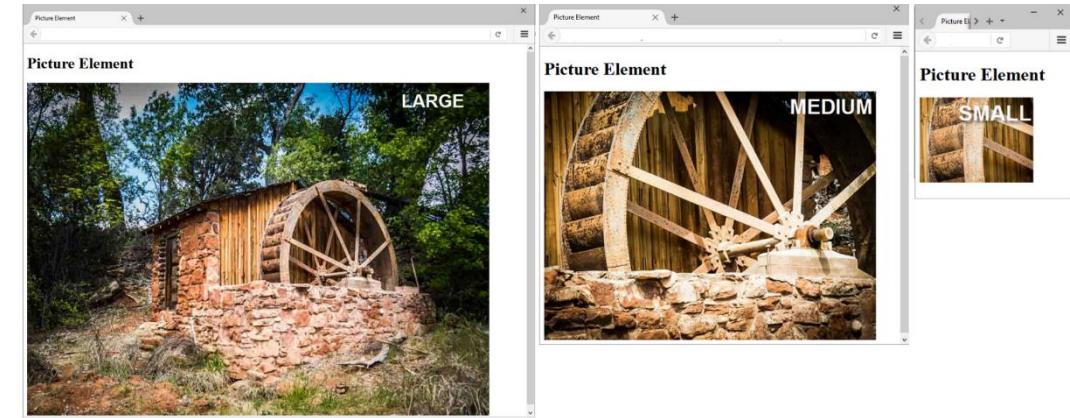
Smartphone Display Width



# Responsive Images: HTML Picture Element

```
<picture>
<source type="image/webp" media="(max-width: 480px)" srcset="small.webp">
<source type="image/webp" media="(max-width: 800px)" srcset="medium.webp">
<source type="image/webp" srcset="large.webp">
<source media="(max-width: 480px)" srcset="small.jpg">
<source media="(max-width: 800px)" srcset="medium.jpg">
<source srcset="large.jpg">

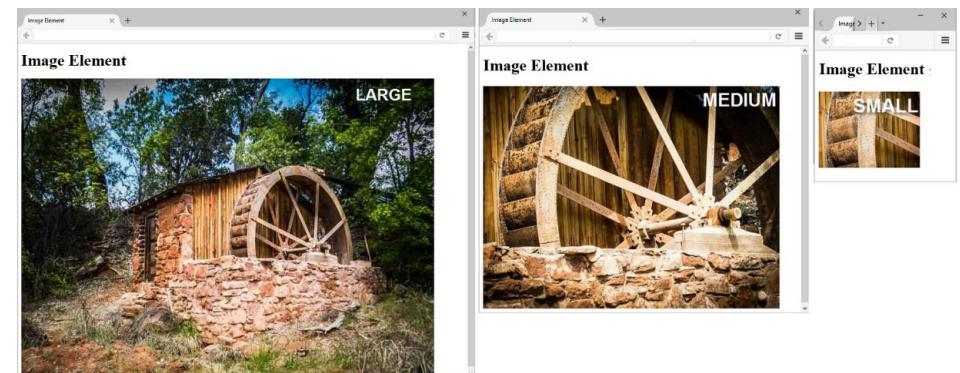
</picture>
```



## Responsive Images: img Element Sizes & srcset Attributes

```

```

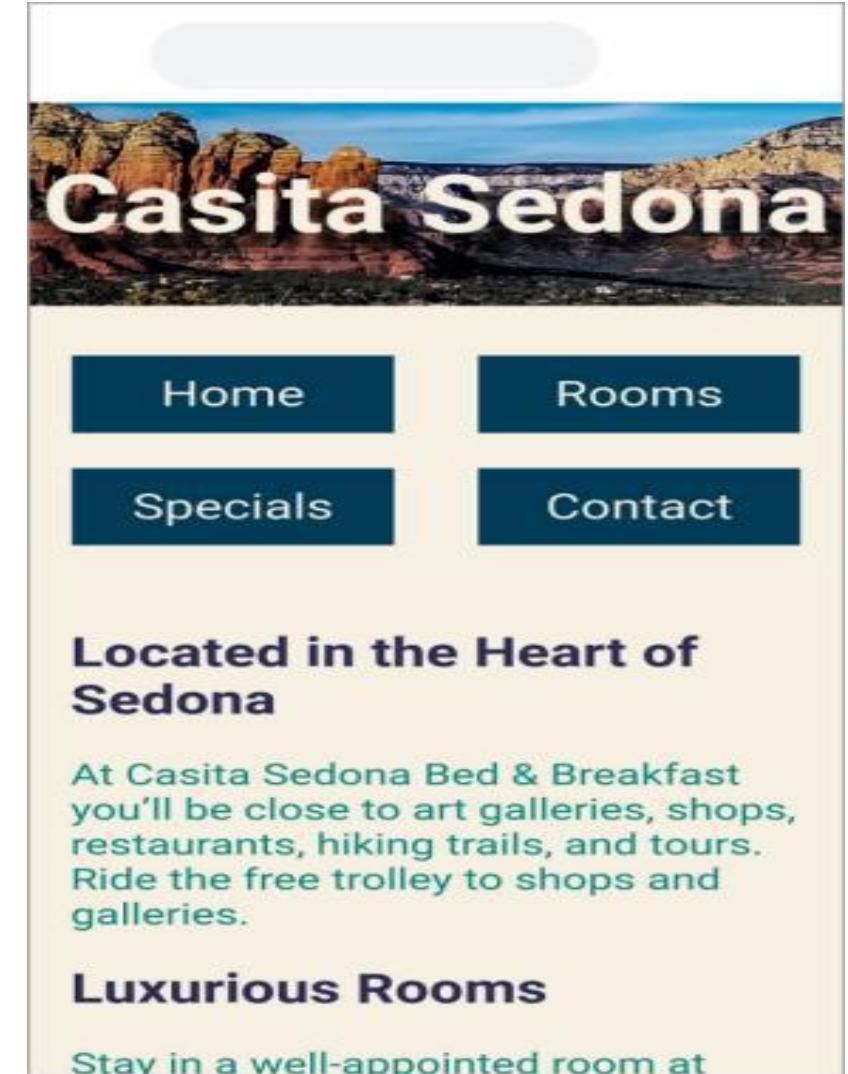


# Responsive Images The loading Attributes

- Useful to decrease perceived load time for pages with multiple large images
- Prevents requesting images that are not in or near the viewport for download
- img element Attribute
- Values:
  - **eager (default)**  
Image is requested immediately for download
  - **lazy**  
Image is requested when it is near or in the viewport

# Testing Mobile Display Options

- **Test with a mobile device**
- **Test with a desktop browser**
- **Responsive Testing Tools**
  - Web Developer Extension
  - Viewport Dimensions Extension
  - Responsive Design Checker
- **Built-in Browser Tools**
  - Google Chrome Dev Tools
  - Firefox Responsive Design Mode
  - Microsoft Edge (Chromium) Developer Tools
- **More Mobile Testing Options**
  - Browserstack.com
  - CrossBrowserTesting.com
  - IOS and Andriod SDKs



# CSS Debugging Tips

- Manually check syntax errors
- Use W3C CSS Validator to check syntax errors
  - <https://jigsaw.w3.org/css-validator/>
- Configure temporary background colors
- Configure temporary borders
- Use CSS comments to find the unexpected
  - /\* the browser ignores this code \*/
- Don't expect your pages to look exactly the same in all browsers!
- Be patient!

# Summary

- This chapter expanded your CSS and HTML skillset.
- You configured web pages with CSS Flexible Box Layout (flexbox).
- You configured web pages with CSS Grid Layout
- You configured web pages for mobile display using the viewport meta tag.
- You applied responsive web design techniques with CSS media queries and feature queries.
- You applied responsive image techniques including the picture element.
- You explored techniques for testing the mobile display of a web page.

# UCCD2323 Front-End Web Development



## Chapter 3 Client-side Programming on Event Handling Part 1.

# Topic Outline

Introduction

Your First Script: Displaying a Line of Text with JavaScript in a Web Page

Modifying Your First Script

Obtaining User Input with prompt Dialogs

6.4.1 Dynamic Welcome Page

6.4.2 Adding Integers

Memory Concepts

Arithmetic

Decision Making: Equality and Relational Operators

Web Resources

# Objectives

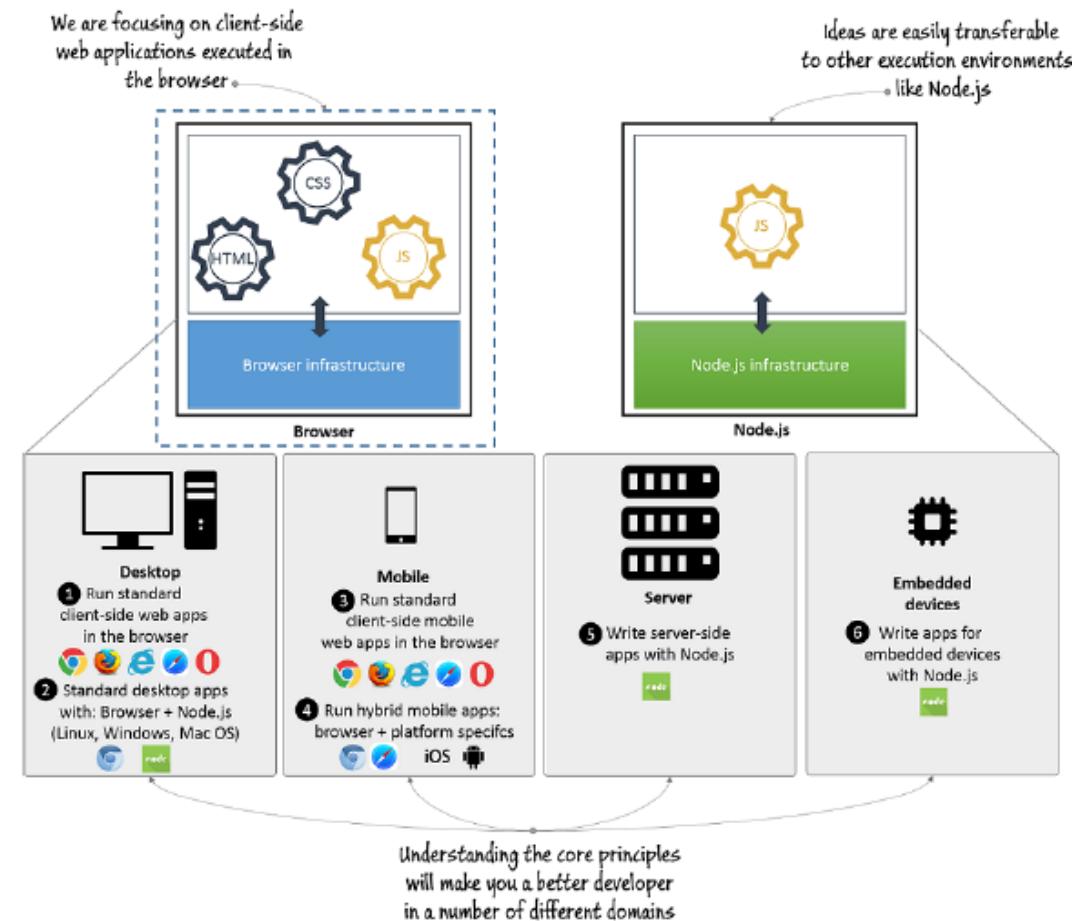
In this chapter you will:

- Write simple JavaScript programs.
- Use input and output statements.
- Learn basic memory concepts.
- Use arithmetic operators.
- Learn the precedence of arithmetic operators.
- Write decision-making statements to choose among alternative courses of action.
- Use relational and equality operators to compare data items.

# The JavaScript eco-system

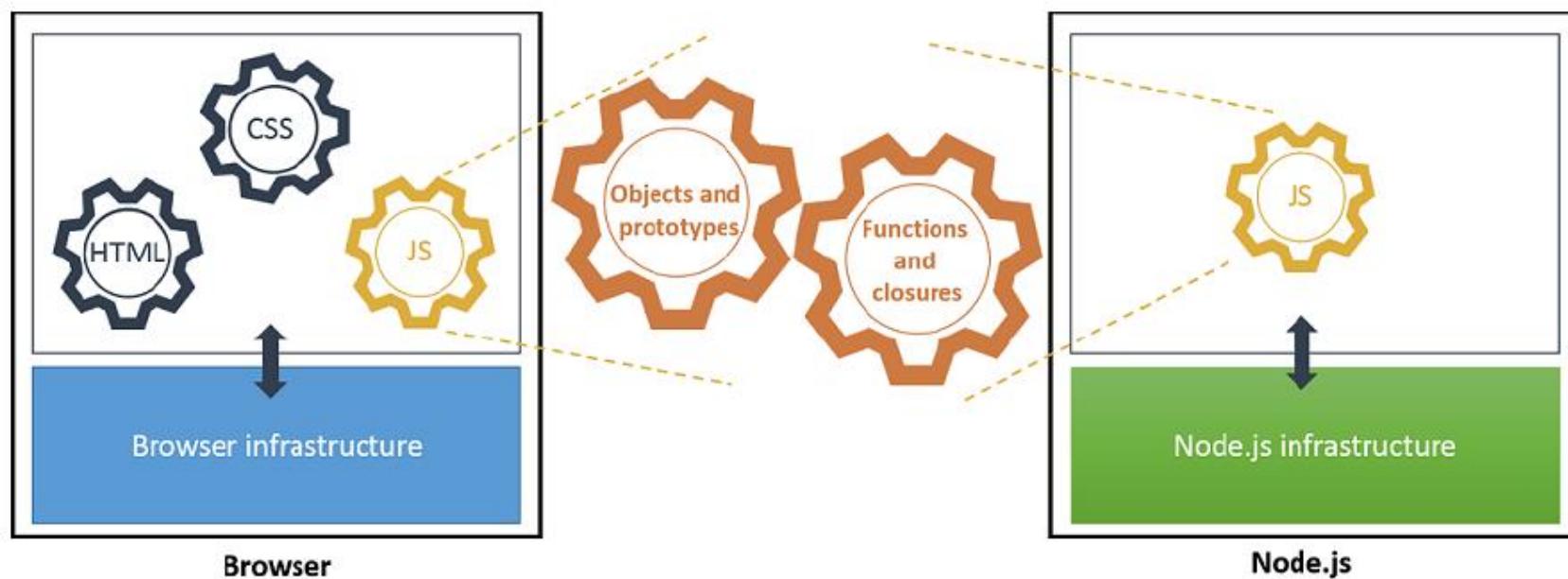
- The JavaScript eco-system: JavaScript applications can nowadays be executed on different devices (desktop computers, mobile devices, servers, and even on embedded devices).

*dynamic web*



# JavaScript

- JavaScript consists of a close relationship between objects and prototypes, and functions and closures; concepts applicable across multiple domains.



# Introduction

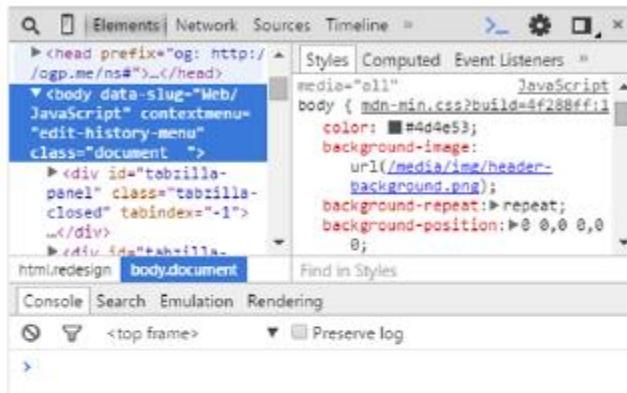
- JavaScript
  - Scripting language which is used to enhance the functionality and appearance of web pages.
- Before you can run code examples with JavaScript on your computer, you may need to change your browser's security settings.
  - IE9 *prevents* scripts on the local computer from running by default.
  - Firefox, Chrome, Opera, Safari (including on the iPhone) and the Android browser have JavaScript enabled by default.

# JavaScript Debugging

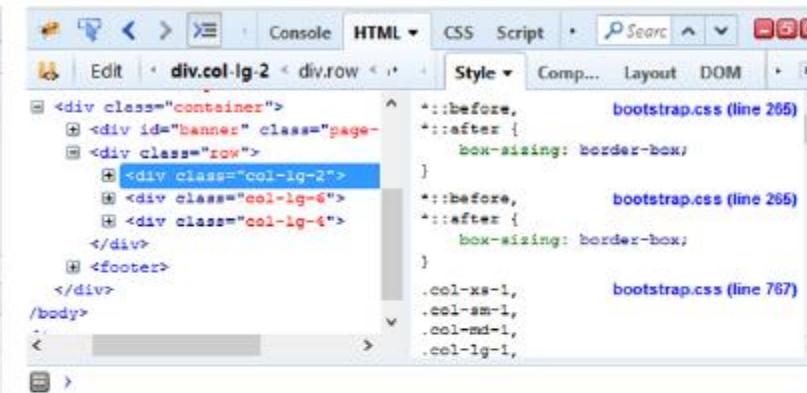
- Debugging JavaScript used to mean using alert to verify the value of variables. Fortunately, the ability to debug JavaScript code has dramatically improved, in no small part due to the popularity of the Firebug developer extension for Firefox. Similar tools have been developed for all major browsers:
- Firebug – The popular developer extension for Firefox that got the ball rolling (<http://getfirebug.com/>).
- Chrome Dev Tools – developed by the Chrome team, used in Chrome and Opera.
- Firefox DevTools – a tool included into Firefox, built by the Firefox team.
- F12 developer tools – Included in Internet Explorer and Microsoft Edge.
- WebKit Inspector – used by Safari.

# JavaScript Debugging

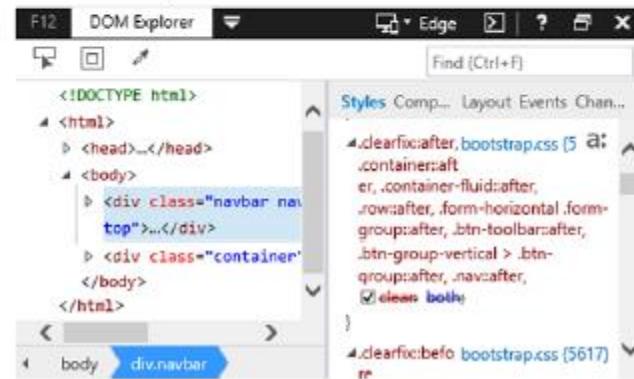
Chrome DevTools used from Chrome



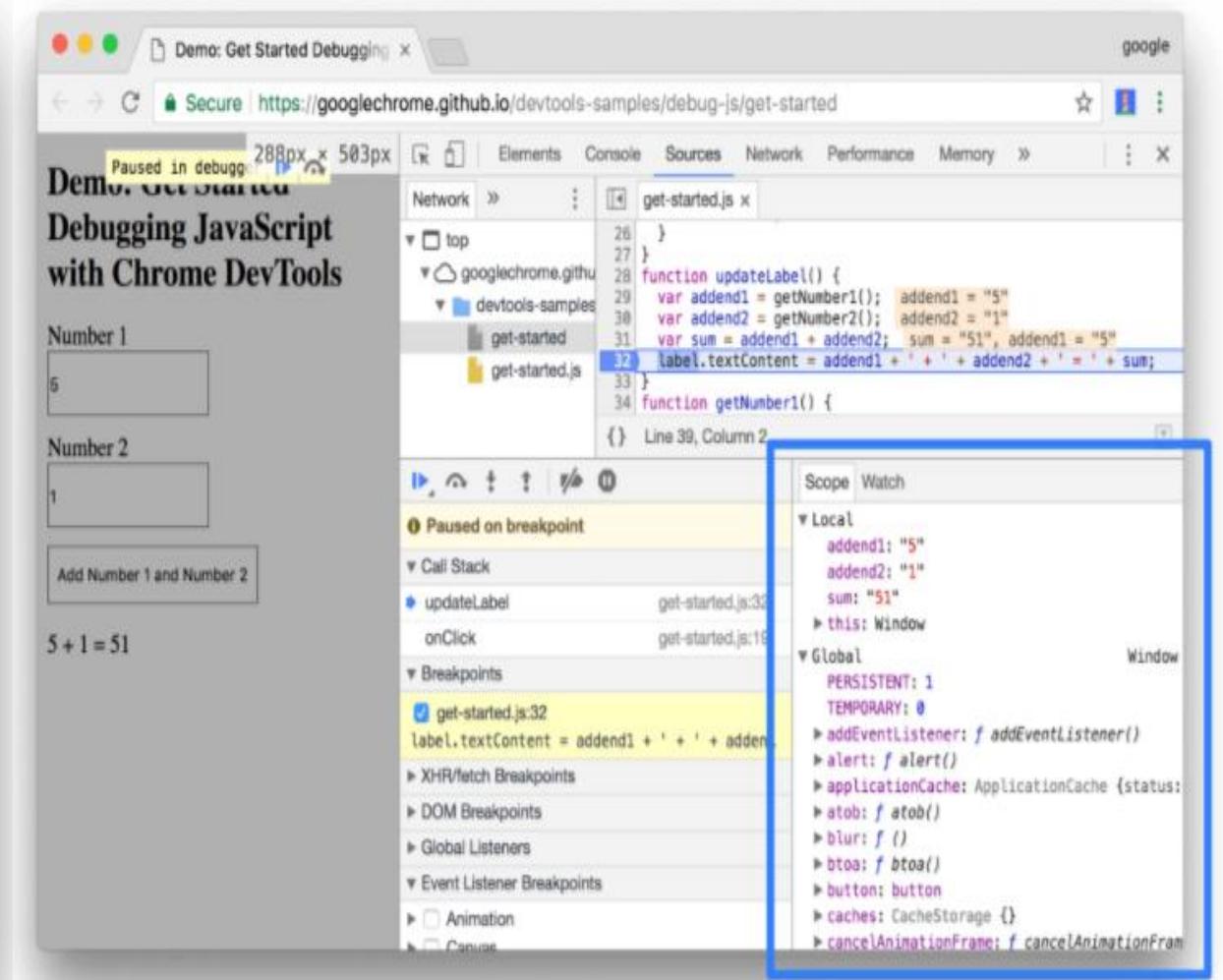
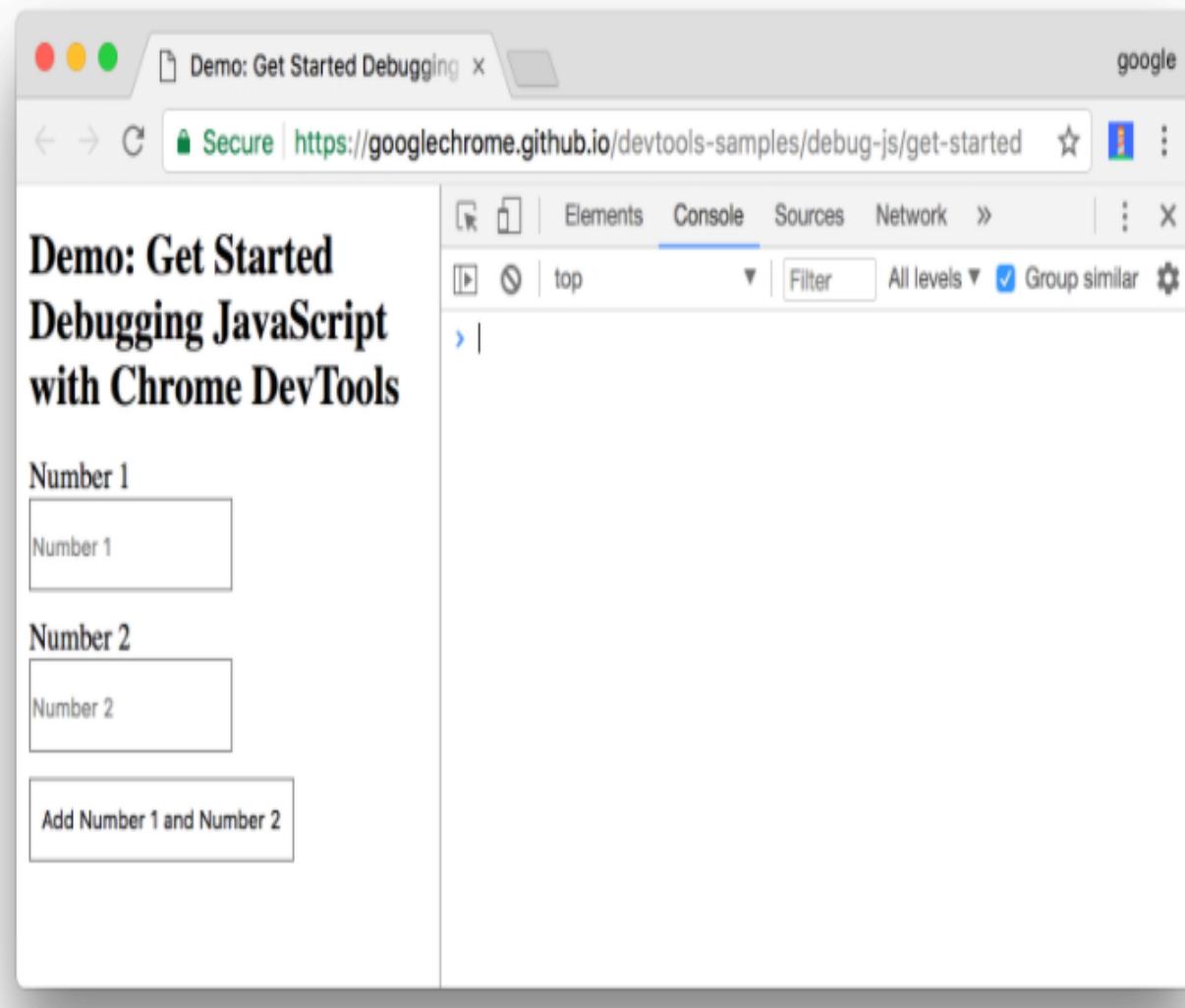
Firebug used from Firefox



F12 Developer tools from IE



# JavaScript Debugging



# Declaring Variables in JavaScript

- Using 'var' keyword
  - Variables declared with var are function-scoped, meaning they are only visible within the function where they are declared.
  - They can also be globally scoped if declared outside of any function.
  - Example,
    - `var x = 10;`
- Using 'let' keyword
  - Variables declared with let are block-scoped, meaning they are only visible within the block (statement or compound statement) where they are declared.
  - let allows you to reassign values.
  - Example,
    - `let y = 20;`

# Declaring Variables in JavaScript

- Using 'const' keyword
  - Variables declared with const are also block-scoped.
  - The value assigned to a const variable cannot be reassigned after declaration.
  - It is commonly used for values that should not be changed.
  - Example,
    - `const z = 30;`
- Variables can also be declared without an initial value, and you can assign a value to them later:
  - Example,
    - `let name; // Variable declaration without initialization`
    - `name = "John"; // Variable assignment`
  - Use const by default and only use let when you know the variable's value will change.

# Naming of Variables

- Valid variable names must adhere to certain rules and conventions.
- Identifier Rules
  - Variable names in JavaScript are called "identifiers."
  - Identifiers must begin with a letter (a-z, A-Z), underscore (\_), or dollar sign (\$). 
  - After the first character, identifiers can also contain digits (0-9).
- Reserved Words
  - You cannot use JavaScript reserved words as variable names. These include keywords like if, else, while, function, etc.
- Case Sensitivity
  - JavaScript is case-sensitive, so myVariable and MyVariable are considered different variables.
- Camel Case Convention *Conventions*
  - It's a common convention in JavaScript to use camel case for variable names (e.g., myVariable, totalAmount, calculateSum), especially for variables and functions.
- Avoid Single Letter Names 
  - While technically valid, it's good practice to use meaningful names that convey the purpose of the variable.

# Naming of Variables

- Examples of invalid variable names

// Invalid: Starts with a number

let 1stPlace = "First";

  63

// Invalid: Contains a special character other than underscore or dollar sign

let my-variable = "Invalid";

// Invalid: Reserved word

let if = "Condition";

// Invalid: Space in the variable name

let my variable = "Invalid";

# Create Array in JavaScript

- Array Literal
  - You can create an array using square brackets and separating elements with commas.
  - Example,
    - `let fruits = ['apple', 'orange', 'banana'];`
- Array Constructor
  - You can use the Array constructor to create an array.
  - Example,
    - `let numbers = new Array(1, 2, 3, 4, 5);`
- Empty Array
  - You can create an empty array and later add elements to it.
  - Example,
    - `let emptyArray = [];`
    - `emptyArray.push('first', 'second');`
- Array.from()
  - You can use Array.from() to create an array from an iterable or array-like object.
  - Example,
    - `let newArray = Array.from('hello'); // Result: ['h', 'e', 'l', 'l', 'o']`

# Create Array in JavaScript

- Array of a Specific Length
  - You can create an array of a specific length and fill it with default values.
  - Example,
    - `let newArray = new Array(3); // Creates an array with length 3  
// Result: [undefined, undefined, undefined]`
    - `let filledArray = new Array(3).fill(0); // Creates an array with length 3, filled with 0  
// Result: [0, 0, 0]`
- Spread Operator *(calc app ok)*
  - You can use the spread operator (...) to create a new array by spreading elements from an existing array or iterable.
  - Example,
    - `let originalArray = [1, 2, 3];`
    - `let newArray = [...originalArray];`

# Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- We begin with a simple script that displays the text "Welcome to JavaScript Programming!" in the HTML5 document.
- All major web browsers contain JavaScript **interpreters**, which process the commands written in JavaScript.
- The JavaScript code and its result are shown in Fig. 6.1.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.1: welcome.html -->
4  <!-- Displaying a line of text. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>A First Program in JavaScript</title>
9          <script type = "text/javascript">
10             document.writeln(
11                 "<h1>Welcome to JavaScript Programming!</h1>" );
12
13
14     </script>
15     </head><body></body>
16 </html>
```

declare under  
head / body

Script result



Fig. 6.1 | Displaying a line of text. (Part 2 of 2.)

Fig. 6.1 | Displaying a line of text. (Part 1 of 2.)

# Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- Spacing displayed by a browser in a web page is determined by the HTML5 elements used to format the page
- Often, JavaScripts appear in the <head> section of the HTML5 document
- The browser interprets the contents of the <head> section first
- The <script> tag indicates to the browser that the text that follows is part of a script. Attribute type specifies the scripting language used in the script—such as **text/javascript**

CJ

- ***The script Element and Commenting Your Scripts***
  - ▶ The <script> tag indicates to the browser that the text which follows is part of a script.
  - ▶ The type attribute specifies the MIME type of the script as well as the **scripting language** used in the script—in this case, a text file written in javascript.
  - ▶ In HTML5, the default MIME type for a <script> is "text/html", so you can omit the type attribute from your <script> tags.
  - ▶ You'll see it in legacy HTML documents with embedded JavaScripts.

# Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- A string of characters can be contained between double quotation ("") marks (also called a **string literal**).
- Browser's document object represents the HTML5 document currently being displayed in the browser
  - Allows you to specify HTML5 text to be displayed in the HTML5 document
- Browser contains a complete set of objects that allow script programmers to access and manipulate every element of an HTML5 document
- Object
  - Resides in the computer's memory and contains information used by the script
  - The term object normally implies that attributes (data) and behaviors (methods) are associated with the object
  - An object's methods use the attributes' data to perform useful actions for the client of the object—the script that calls the methods



## Software Engineering Observation 6.1

Strings in JavaScript can be enclosed in either double quotation marks ("") or single quotation marks ('').

# Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- The parentheses following the name of a method contain the arguments that the method requires to perform its task (or its action)
- Every statement should end with a semicolon (also known as the **statement terminator**), although none is required by JavaScript
- JavaScript is case sensitive
  - Not using the proper uppercase and lowercase letters is a syntax error



## Good Programming Practice 6.1

Terminate every statement with a semicolon. This notation clarifies where one statement ends and the next statement begins.



## Common Programming Error 6.1

Forgetting the ending `</script>` tag for a script may prevent the browser from interpreting the script properly and may prevent the HTML5 document from loading properly.

# Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- The document object's writeln method
  - Writes a line of HTML5 text in the HTML5 document
  - Does not guarantee that a corresponding line of text will appear in the HTML5 document.
  - Text displayed is dependent on the contents of the string written, which is subsequently rendered by the browser.
  - Browser will interpret the HTML5 elements as it normally does to render the final text in the document



## Common Programming Error 6.2

JavaScript is case sensitive. Not using the proper uppercase and lowercase letters is a syntax error. A syntax error occurs when the script interpreter cannot recognize a statement. The interpreter normally issues an error message to help you locate and fix the incorrect statement. Syntax errors are violations of the rules of the programming language. The interpreter notifies you of a syntax error when it attempts to execute the statement containing the error. Each browser has its own way to display JavaScript Errors. For example, Firefox has the Error Console (in its Web Developer menu) and Chrome has the JavaScript console (in its Tools menu). To view script errors in IE9, select *Internet Options...* from the *Tools* menu. In the dialog that appears, select the *Advanced* tab and click the checkbox labeled *Display a notification about every script error* under the *Browsing* category.

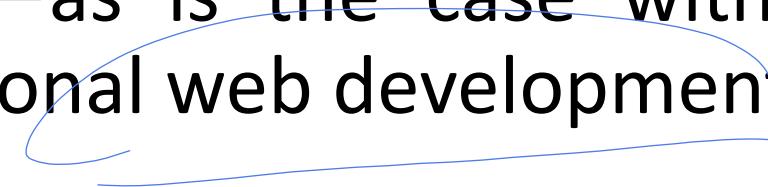


## Error-Prevention Tip 6.1

When the interpreter reports a syntax error, sometimes the error is not in the line indicated by the error message. First, check the line for which the error was reported. If that line does not contain errors, check the preceding several lines in the script.

# Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- ***A Note About Embedding JavaScript Code into HTML5 Documents***
  - ▶ JavaScript code is typically placed in a separate file, then included in the HTML5 document that uses the script.
  - ▶ This makes the code more reusable, because it can be included into any HTML5 document—as is the case with the many JavaScript libraries used in professional web development today.



# Modifying Your First Script

- A script can display Welcome to JavaScript Programming! in many ways.
- Figure 6.2 displays the text in magenta, using the CSS color property.
- Method write displays a string like writeln, but does not position the output cursor in the HTML5 document at the beginning of the next line after writing its argument

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.2: welcome2.html -->
4 <!-- Printing one line with multiple statements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Printing a Line with Multiple Statements</title>
9     <script type = "text/javascript">
10       <!--
11         document.write( "h1 style = 'color: magenta' " );
12         document.write( "Welcome to JavaScript " +
13           "Programming!"</h1> );
14       // --
15     </script>
16   </head><body></body>
17 </html>
```



Fig. 6.2 | Printing one line with separate statements. (Part 2 of 2.)

Fig. 6.2 | Printing one line with separate statements. (Part 1 of 2.)

# Modifying Your First Script

- The + operator (called the “concatenation operator” when used in this manner) joins two strings together



## Common Programming Error 6.3

Splitting a JavaScript statement in the middle of a string is a syntax error.



## Common Programming Error 6.4

Many people confuse the writing of HTML5 text with the rendering of HTML5 text. Writing HTML5 text creates the HTML5 that will be rendered by the browser for presentation to the user.

# Modifying Your First Script

- ***Displaying Text in an Alert Dialog***

- ▶ **Dialogs**

- Useful to display information in windows that “pop up” on the screen to grab the user’s attention
- Typically used to display important messages to the user browsing the web page
- Browser’s window object uses method alert to display an alert dialog
- Method alert requires as its argument the string to be displayed

```
I <!DOCTYPE html>
2
3 <!-- Fig. 6.3: welcome3.html -->
4 <!-- Alert dialog displaying multiple lines. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Printing Multiple Lines in a Dialog Box</title>
9     <script type = "text/javascript">
10       <!--
11         window.alert( "Welcome to\nJavaScript\nProgramming!" );
12       // -->
13     </script>
14   </head>
15   <body>
16     <p>Click Refresh (or Reload) to run this script again.</p>
17   </body>
18 </html>
```



**Fig. 6.3 | Alert dialog displaying multiple lines. (Part 2 of 2.)**

**Fig. 6.3 | Alert dialog displaying multiple lines. (Part 1 of 2.)**

# Modifying Your First Script

- **Escape Sequences**
  - ▶ When a backslash is encountered in a string of characters, the next character is combined with the backslash to form an **escape sequence**. The escape sequence `\n` is the **newline character**. It causes the cursor in the HTML5 document to move to the beginning of the next line.



Escape sequence	Description
<code>\n</code>	<i>New line</i> —position the screen cursor at the beginning of the next line.
<code>\t</code>	<i>Horizontal tab</i> —move the screen cursor to the next tab stop.
<code>\\</code>	<i>Backslash</i> —used to represent a backslash character in a string.
<code>\"</code>	<i>Double quote</i> —used to represent a double-quote character in a string contained in double quotes. For example,  <code>window.alert( \"in double quotes\" );</code>  displays "in double quotes" in an alert dialog.
<code>'</code>	<i>Single quote</i> —used to represent a single-quote character in a string. For example,  <code>window.alert( '\'in single quotes\' );</code>  displays 'in single quotes' in an alert dialog.

Fig. 6.4 | Some common escape sequences.

# Dynamic Welcome Page- Obtaining User Input with prompt Dialogs

- Scripting
  - Gives you the ability to generate part or all of a web page's content at the time it is shown to the user
  - Such web pages are said to be dynamic, as opposed to static, since their content has the ability to change
- The next script creates a dynamic welcome page that obtains the user's name, then displays it on the page.
- The script uses another *predefined* dialog box from the window object—a **prompt** dialog—which allows the user to enter a value that the script can use.
- Figure 6.5 presents the script and sample output.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.5: welcome4.html -->
4  <!-- Prompt box used on a welcome screen -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Using Prompt and Alert Boxes</title>
9          <script type = "text/javascript">
10             <!--
11             var name; // string entered by the user
12
13             // read the name from the prompt box as a string
14             name = window.prompt( "Please enter your name" );
15
16             document.writeln( "<h1>Hello " + name +
17                 ", welcome to JavaScript programming!</h1>" );
18             // -->
19         </script>
20     </head><body></body>
21 </html>
```

Fig. 6.5 | Prompt box used on a welcome screen. (Part 1 of 2.)



Fig. 6.5 | Prompt box used on a welcome screen. (Part 2 of 2.)

# Dynamic Welcome Page

- **Keywords** are words with special meaning in JavaScript
- Keyword var
  - Used to declare the names of variables
  - A variable is a location in the computer's memory where a value can be stored for use by a script
  - All variables have a name, type and value, and should be declared with a var statement before they are used in a script
- A variable name can be any valid **identifier** consisting of letters, digits, underscores ( \_ ) and dollar signs (\$) that does not begin with a digit and is not a reserved JavaScript keyword.



## Good Programming Practice 6.2

Choosing meaningful variable names helps a script to be “self-documenting” (i.e., easy to understand by simply reading the script).



## Good Programming Practice 6.3

By convention, variable-name identifiers begin with a lowercase first letter. Each subsequent word should begin with a capital first letter. For example, identifier `itemPrice` has a capital P in its second word, `Price`.



## Common Programming Error 6.5

Splitting a statement in the middle of an identifier is a syntax error.

# Dynamic Welcome Page

- ▶ Declarations end with a semicolon (;) and can be split over several lines, with each variable in the declaration separated by a comma (forming a comma-separated list of variable names)
  - Several variables may be declared in one declaration or in multiple declarations.
- ▶ Comments
  - A single-line **comment** begins with the characters // and terminates at the end of the line
  - Comments do not cause the browser to perform any action when the script is interpreted; rather, comments are ignored by the JavaScript interpreter
  - Multiline **comments** begin with delimiter /\* and end with delimiter \*/
    - All text between the delimiters of the comment is ignored by the interpreter.



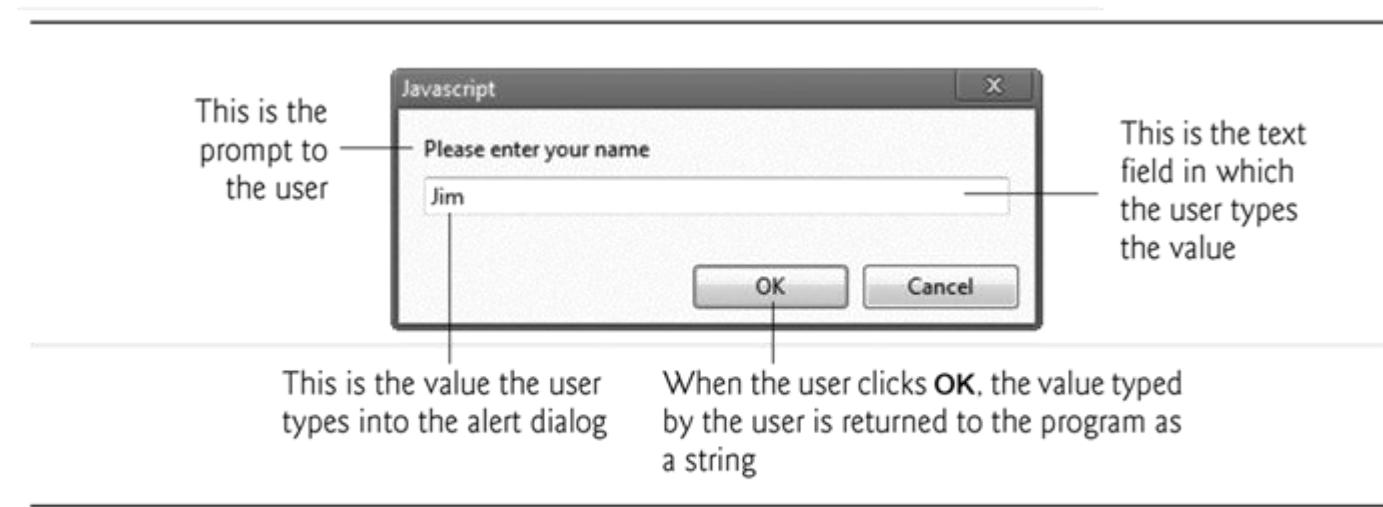
## Good Programming Practice 6.4

Although it's not required, declare each variable on a separate line. This allows for easy insertion of a comment next to each declaration. This is a widely followed professional coding standard.

ek+

# Dynamic Welcome Page

- The window object's prompt method displays a dialog into which the user can type a value.
  - The first argument is a message (called a prompt) that directs the user to take a specific action.
  - The optional second argument is the default string to display in the text field.
- Script can then use the value that the user inputs.



**Fig. 6.6 |** Prompt dialog displayed by the window object's prompt method.

# Dynamic Welcome Page

- A variable is assigned a value with an assignment statement, using the assignment operator, =.
- The = operator is called a binary operator, because it has two operands.



## Good Programming Practice 6.5

Place a space on each side of a binary operator. This format makes the operator stand out and makes the script more readable.

# Dynamic Welcome Page

- null keyword
  - Signifies that a variable has no value
  - null is not a string literal, but rather a predefined term indicating the absence of value
  - Writing a null value to the document, however, displays the word “null”
- Function parseInt
  - converts its string argument to an integer
  - JavaScript has a version of the + operator for string concatenation that enables a string and a value of another data type (including another string) to be concatenated

# Adding Integers

- Our next script illustrates another use of prompt dialogs to obtain input from the user.
- Figure 6.7 inputs two *integers* (whole numbers, such as 7, -11, 0 and 31914) typed by a user at the keyboard, computes the sum of the values and displays the result.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.7: addition.html -->
4 <!-- Addition script. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>An Addition Program</title>
9     <script type = "text/javascript">
10    <!--
11    var firstNumber; // first string entered by user
12    var secondNumber; // second string entered by user
13    var number1; // first number to add
14    var number2; // second number to add
15    var sum; // sum of number1 and number2
16
17    // read in first number from user as a string
18    firstNumber = window.prompt( "Enter first integer" );
19
20    // read in second number from user as a string
21    secondNumber = window.prompt( "Enter second integer" );
22
```

*(stry get) +*

```
23   // convert numbers from strings to integers
24   number1 = parseInt( firstNumber );
25   number2 = parseInt( secondNumber );
26   sum = number1 + number2; // add the numbers
27
28
29   // display the results on screen
30   document.writeln( "<h1>The sum is " + sum + "</h1>" );
31   // -->
32 </script>
33 </head><body></body>
34 </html>
```

*str → int*

Fig. 6.7 | Addition script. (Part 1 of 3.)

Fig. 6.7 | Addition script. (Part 2 of 3.)

# Adding Integers

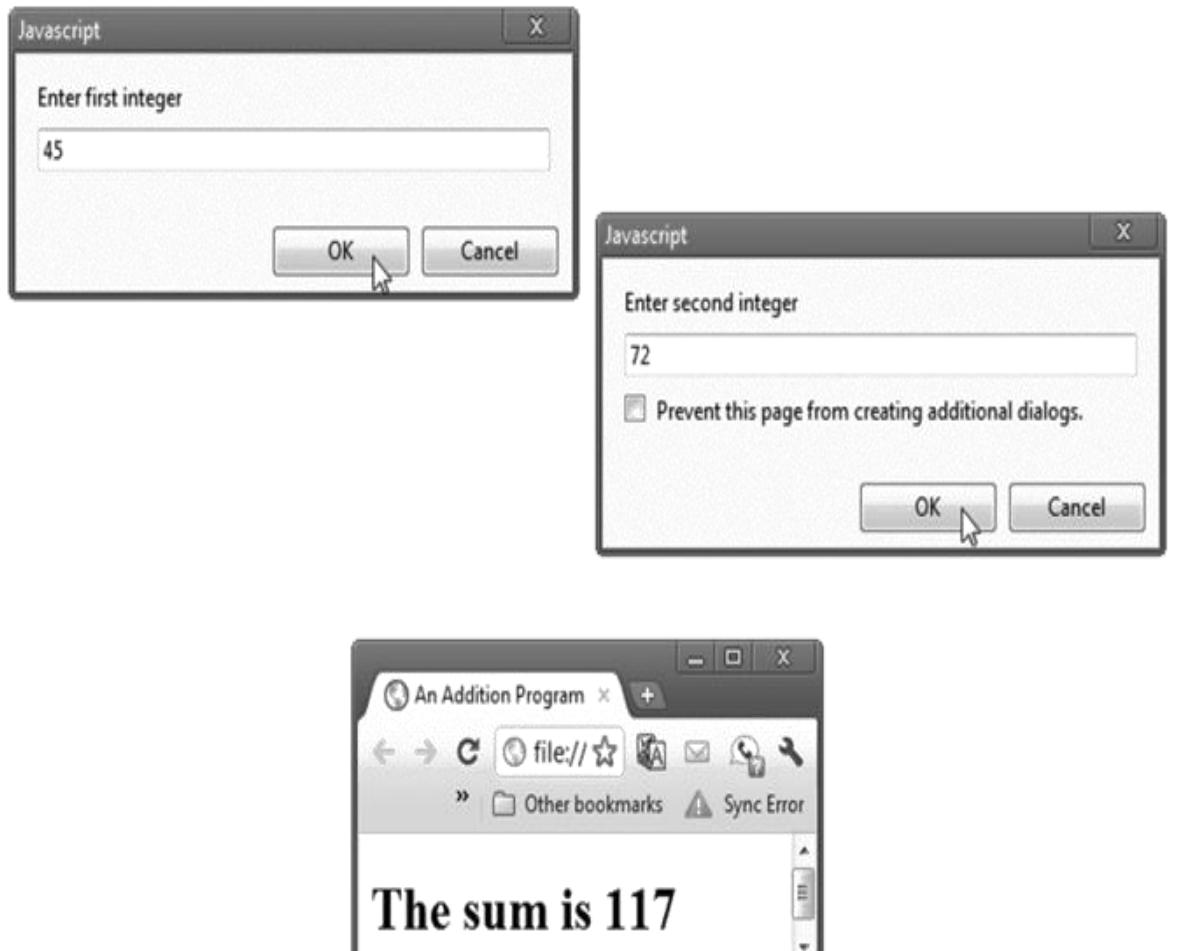


Fig. 6.7 | Addition script. (Part 3 of 3.)

## Common Programming Error 6.6

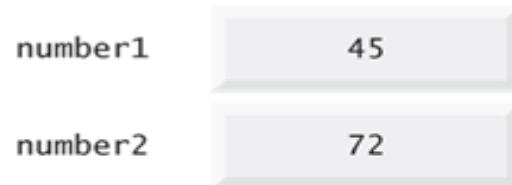
Confusing the `+` operator used for string concatenation with the `+` operator used for addition often leads to undesired results. For example, if integer variable `y` has the value 5, the expression `"y + 2 = " + y + 2` results in `"y + 2 = 52"`, not `"y + 2 = 7"`, because first the value of `y` (i.e., 5) is concatenated with the string `"y + 2 = "`, then the value 2 is concatenated with the new, larger string `"y + 2 = 5"`. The expression `"y + 2 = " + (y + 2)` produces the string `"y + 2 = 7"` because the parentheses ensure that `y + 2` is calculated.

# Memory Concepts

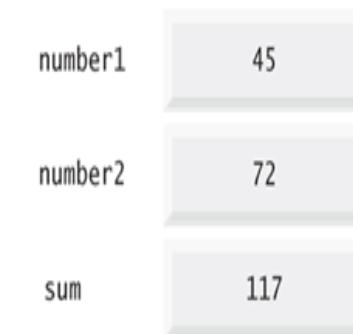
- Variable names correspond to locations in the computer's memory.
- Every variable has a name, a type and a value.
- When a value is placed in a memory location, the value replaces the previous value in that location.
- When a value is read out of a memory location, the process is nondestructive.



**Fig. 6.8** | Memory location showing the name and value of variable number1.



**Fig. 6.9** | Memory locations after inputting values for variables number1 and number2.



**Fig. 6.10** | Memory locations after calculating the sum of number1 and number2.

# Memory Concepts

- ▶ JavaScript does not require variables to have a type before they can be used in a script
- ▶ A variable in JavaScript can contain a value of any data type, and in many situations, JavaScript automatically converts between values of different types for you
- ▶ JavaScript is referred to as a loosely typed language
- ▶ When a variable is declared in JavaScript, but is not given a value, it has an undefined value.
  - Attempting to use the value of such a variable is normally a logic error.
- ▶ When variables are declared, they are not assigned default values, unless specified otherwise by the programmer.
  - To indicate that a variable does not contain a value, you can assign the value null to it.

# Arithmetic

- The basic arithmetic operators (+, -, \*, /, and %) are binary operators, because they each operate on two operands
- JavaScript provides the remainder operator, %, which yields the remainder after division
- Arithmetic expressions in JavaScript must be written in straight-line form to facilitate entering programs into the computer

JavaScript operation	Arithmeti c operator	Algebraic expression	JavaScript expressio n
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	$bm$	<code>b * m</code>
Division	/	$x/y$ or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Remainder	%	$r \bmod s$	<code>r % s</code>

**Fig. 6.11 |** Arithmetic operators.

# Arithmetic

- ▶ Parentheses can be used to group expressions as in algebra.
- ▶ Operators in arithmetic expressions are applied in a precise sequence determined by the rules of operator precedence:
  - Multiplication, division and remainder operations are applied first. ✓
  - If an expression contains several of these operations, operators are applied from left to right.
  - Multiplication, division and remainder operations are said to have the same level of precedence.
  - Addition and subtraction operations are applied next.
  - If an expression contains several of these operations, operators are applied from left to right.
  - Addition and subtraction operations have the same level of precedence.
- ▶ When we say that operators are applied from left to right, we are referring to the associativity of the operators. Some operators associate from right to left.

Operator(s)	Operation(s)	Order of evaluation (precedence)
* , / or %	Multiplication Division Remainder	Evaluated first. If there are several such operations, they're evaluated from left to right.
+ or -	Addition Subtraction	Evaluated last. If there are several such operations, they're evaluated from left to right.

Fig. 6.12 | Precedence of arithmetic operators.

# Decision Making: Equality and Relational Operators

- ▶ if statement allows a script to make a decision based on the truth or falsity of a condition
  - If the condition is met (i.e., the condition is true), the statement in the body of the if statement is executed
  - If the condition is not met (i.e., the condition is false), the statement in the body of the if statement is not executed
- ▶ Conditions in if statements can be formed by using the equality operators and relational operators
- ▶ Equality operators both have the same level of precedence, which is lower than the precedence of the relational operators.
- ▶ The equality operators associate from left to right.



## Common Programming Error 6.7

Confusing the equality operator, `==`, with the assignment operator, `=`, is a logic error. The equality operator should be read as “is equal to,” and the assignment operator should be read as “gets” or “gets the value of.” Some people prefer to read the equality operator as “double equals” or “equals equals.”

Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
<i>Equality operators</i>			
<code>=</code>	<code>==</code>	<code>x == y</code>	<code>x</code> is equal to <code>y</code>
<code>≠</code>	<code>!=</code>	<code>x != y</code>	<code>x</code> is not equal to <code>y</code>
<i>Relational operators</i>			
<code>&gt;</code>	<code>&gt;</code>	<code>x &gt; y</code>	<code>x</code> is greater than <code>y</code>
<code>&lt;</code>	<code>&lt;</code>	<code>x &lt; y</code>	<code>x</code> is less than <code>y</code>
<code>≥</code>	<code>≥</code>	<code>x ≥ y</code>	<code>x</code> is greater than or equal to <code>y</code>
<code>≤</code>	<code>≤</code>	<code>x ≤ y</code>	<code>x</code> is less than or equal to <code>y</code>

Fig. 6.13 | Equality and relational operators.

# Decision Making: Equality and Relational Operators

- The script in Fig. 6.14 uses four if statements to display a time-sensitive greeting on a welcome page.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.14: welcome5.html -->
4 <!-- Using equality and relational operators. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Using Relational Operators</title>
9     <script type = "text/javascript">
10    <!--
11      var name; // string entered by the user
12      var now = new Date();      // current date and time
13      var hour = now.getHours(); // current hour (0-23)
14
15      // read the name from the prompt box as a string
16      name = window.prompt( "Please enter your name" );
17
18      // determine whether it's morning
19      if ( hour < 12 )
20        document.write( "<h1>Good Morning, " );
21
```

Fig. 6.14 | Using equality and relational operators. (Part 1 of 3.)

```
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
// determine whether the time is PM
if ( hour >= 12 )
{
  // convert to a 12-hour clock
  hour = hour - 12;

  // determine whether it is before 6 PM
  if ( hour < 6 )
    document.write( "<h1>Good Afternoon, " );

  // determine whether it is after 6 PM
  if ( hour >= 6 )
    document.write( "<h1>Good Evening, " );
} // end if

document.writeln( name +
  ", welcome to JavaScript programming!</h1>" );
// -->
</script>
</head><body></body>
</html>
```

Fig. 6.14 | Using equality and relational operators. (Part 2 of 3.)

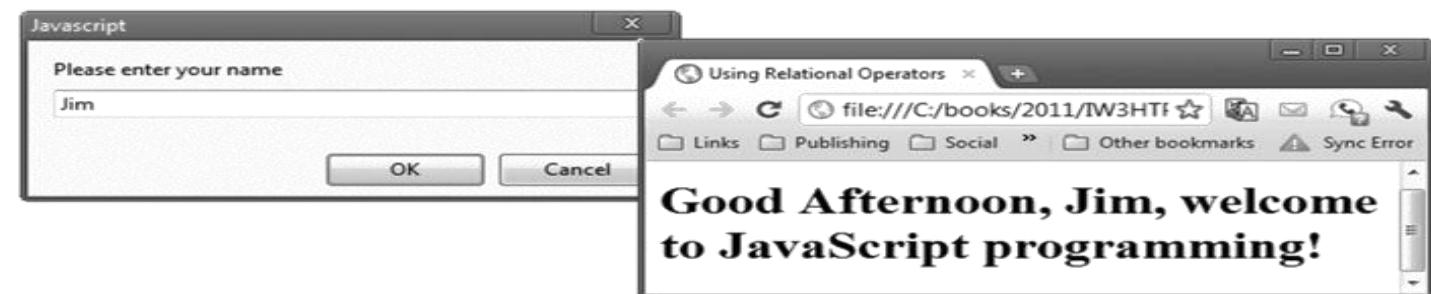


Fig. 6.14 | Using equality and relational operators. (Part 3 of 3.)

# Decision Making: Equality and Relational Operators

- Date object
  - Used acquire the current local time
  - Create a new instance of an object by using the new operator followed by the type of the object, Date, and a pair of parentheses



## Good Programming Practice 6.6

Include comments after the closing curly brace of control statements (such as `if` statements) to indicate where the statements end, as in line 35 of Fig. 6.14.



## Good Programming Practice 6.7

Indent the statement in the body of an `if` statement to make the body of the statement stand out and to enhance script readability.



## Error-Prevention Tip 6.2

A lengthy statement may be spread over several lines. If a single statement must be split across lines, choose breaking points that make sense, such as after a comma in a comma-separated list or after an operator in a lengthy expression. If a statement is split across two or more lines, indent all subsequent lines.

# Decision Making: Equality and Relational Operators



## Good Programming Practice 6.8

Refer to the operator precedence chart when writing expressions containing many operators. Confirm that the operations are performed in the order in which you expect them to be performed. If you're uncertain about the order of evaluation, use parentheses to force the order, exactly as you would do in algebraic expressions. Be sure to observe that some operators, such as assignment ( $=$ ), associate from right to left rather than from left to right.

Operators	Associativity	Type
$*$ $/$ $\%$	left to right	multiplicative
$+$ $-$	left to right	additive
$<$ $\leq$ $>$ $\geq$	left to right	relational
$==$ $!=$ $====$ $!====$	left to right	equality
$=$	right to left	assignment

**Fig. 6.15** | Precedence and associativity of the operators discussed so far.

# JavaScript Data Types

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

Note: JavaScript BigInt is a new datatype (ES2020) that can be used to store integer values that are too big to be represented by a normal JavaScript Number.

Reference:

[https://www.w3schools.com/js/js\\_datatypes.asp](https://www.w3schools.com/js/js_datatypes.asp)

# The Object Datatype

1. An object
2. An array
3. A date

## Examples

```
// Numbers:  
let length = 16;  
let weight = 7.5;  
  
// Strings:  
let color = "Yellow";  
let lastName = "Johnson";  
  
// Booleans  
let x = true;  
let y = false;  
  
// Object:  
const person = {firstName:"John", lastName:"Doe"};  
  
// Array object:  
const cars = ["Saab", "Volvo", "BMW"];  
  
// Date object:  
const date = new Date("2022-03-25");
```

# UCCD2323 Front-End Web Development



## Chapter 3 Client-side Programming on Event Handling Part 2.

# Topic Outline

Introduction  
Program Modules in JavaScript  
Function Definitions  
9.3.1 Programmer-Defined Function square  
9.3.2 Programmer-Defined Function maximum  
Notes on Programmer-Defined Functions  
Random Number Generation  
9.5.1 Scaling and Shifting Random Numbers  
9.5.2 Displaying Random Images  
9.5.3 Rolling Dice Repeatedly and Displaying Statistics  
Example: Game of Chance; Introducing the HTML5 audio and video Elements  
Scope Rules  
JavaScript Global Functions

# Objectives

In this chapter you will:

- Construct programs modularly from small pieces called functions.
- Define new functions.
- Pass information between functions.
- Use simulation techniques based on random number generation.
- Use the new HTML5 audio and video elements
- Use additional global methods.
- See how the visibility of identifiers is limited to specific regions of programs.

# Introduction

- Experience has shown that the best way to develop and **maintain a large program** is to construct it from **small, simple pieces, or modules**. This technique is called **divide and conquer**. This chapter describes many key features of JavaScript that facilitate the design, implementation, operation and maintenance of large scripts.

*=> page 1, 2/3*

## Program Modules in JavaScript

- ▶ You'll combine new functions that you write with **prepackaged functions** and objects available in JavaScript
- ▶ The prepackaged functions that belong to JavaScript objects (such as `Math.pow`, introduced previously) are called **methods**.
- ▶ JavaScript provides several objects that have a rich collection of methods for performing common mathematical calculations, string manipulations, date and time manipulations, and manipulations of collections of data called arrays.

# Program Modules in JavaScript

- You can define **programmer-defined functions** that perform specific tasks and use them at many points in a script
  - The actual statements defining the function are written only once and are hidden from other functions
- Functions are invoked by writing the name of the function, followed by a left parenthesis, followed by a comma-separated list of zero or more arguments, followed by a right parenthesis  
*maximum(value1,value2,value3)*      *reflect*
- Methods are called in the same way as functions, but require the name of the object to which the method belongs and a dot preceding the method name
- Function (and method) arguments may be constants, variables or expressions

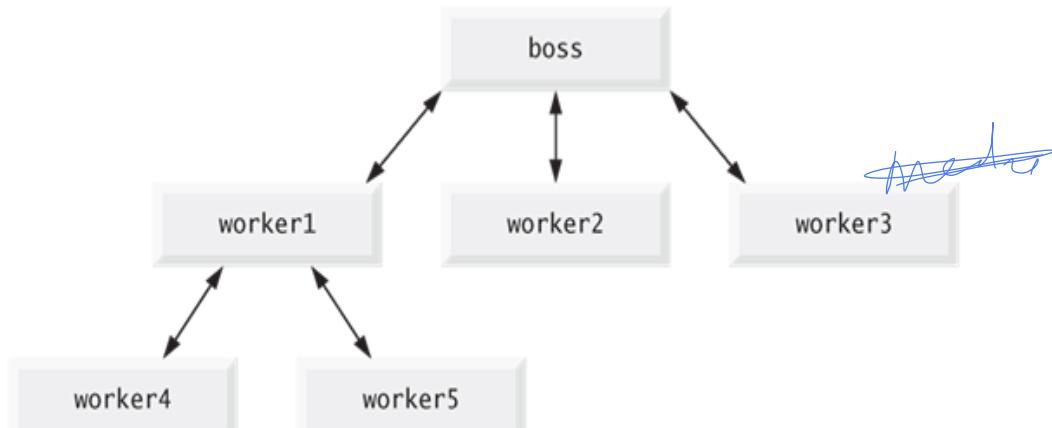


Fig. 9.1 | Hierarchical boss-function/worker-function relationship.

Common analogy for this structure is the hierarchical form of management:  
A boss (the **calling function**, or **caller**) asks a worker (the **called function**) to perform a task and **return** (i.e., report back) the results when the task is done. *The boss function does not know how the worker function performs its designated tasks.* The worker may call other worker functions—the boss will be unaware of this. Figure 9.1 shows the boss function communicating with several worker functions in a hierarchical manner. Note that worker1 also acts as a “boss” function to worker4 and worker5, and worker4 and worker5 report back to worker1.

# Programmer-Defined Function square

- return statement
  - passes information from inside a function back to the point in the program where it was called
- A function must be called explicitly for the code in its body to execute
- The format of a function definition is

```
function function-name( parameter-list )  
{  
    declarations and statements  
}
```

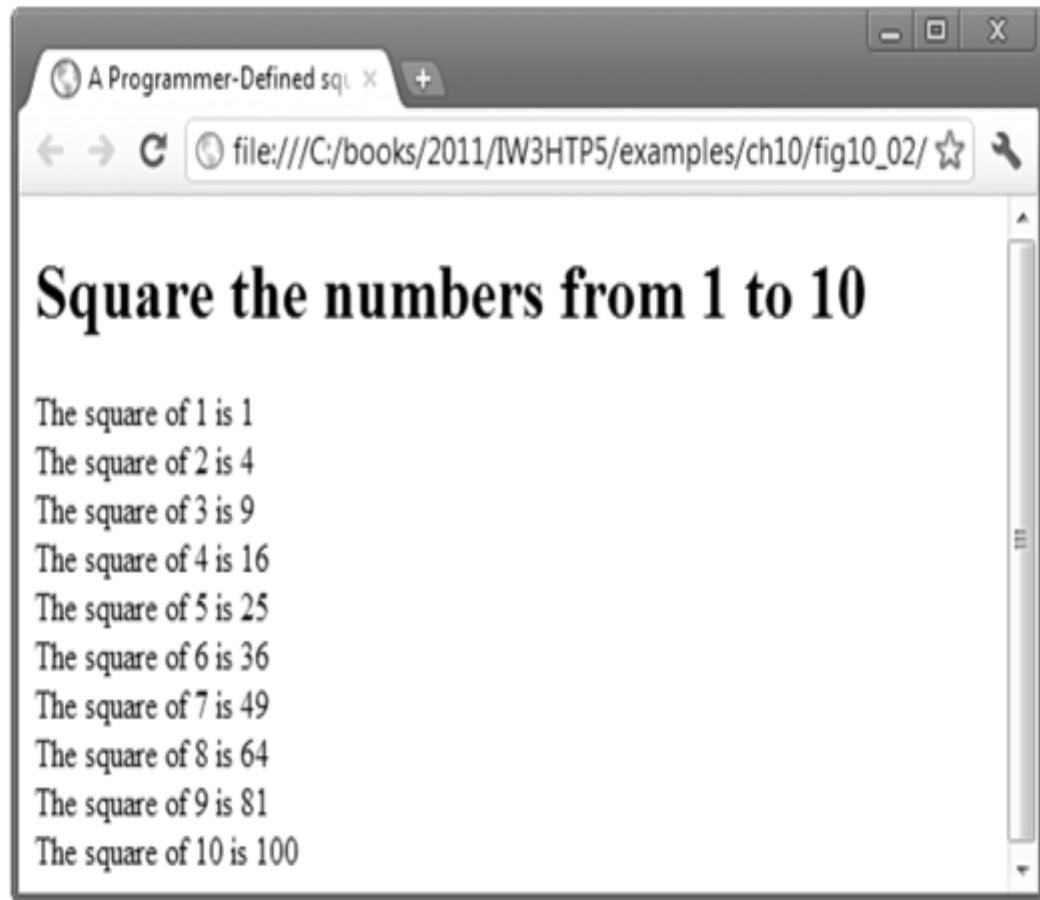
```
1 <!DOCTYPE html>  
2  
3 <!-- Fig. 9.2: SquareInt.html -->  
4 <!-- Programmer-defined function square. -->  
5 <html>  
6     <head>  
7         <meta charset = "utf-8">  
8         <title>A Programmer-Defined square Function</title>  
9         <style type = "text/css">  
10            p { margin: 0; }  
11        </style>  
12        <script>  
13  
14             document.writeln( "<h1>Square the numbers from 1 to 10</h1>" );  
15  
16             // square the numbers from 1 to 10  
17             for ( var x = 1; x <= 10; ++x )  
18                 document.writeln( "<p>The square of " + x + " is " +  
19                     square( x ) + "</p>" );  
20
```

Fig. 9.2 | Programmer-defined function square. (Part 1 of 3.)

```
21     // The following square function definition's body is executed  
22     // only when the function is called explicitly as in line 19  
23     function square( y )  
24     {  
25         return y * y;  
26     } // end function square  
27  
28     </script>  
29     </head><body></body> <!-- empty body element -->  
30 </html>
```

Fig. 9.2 | Programmer-defined function square. (Part 2 of 3.)

# Programmer-Defined Function square



The screenshot shows a Java application window titled "A Programmer-Defined sq". The URL in the address bar is "file:///C:/books/2011/IW3HTP5/examples/ch10/fig10\_02/". The main content area displays the following text:

```
Square the numbers from 1 to 10
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100
```



## Common Programming Error 9.1

Forgetting to return a value from a function that's supposed to return a value is a logic error.

Fig. 9.2 | Programmer-defined function square. (Part 3 of 3.)

# Programmer-Defined Function square

- Three ways to return control to the point at which a function was invoked
  - Reaching the function-ending right brace
  - Executing the statement `return;`
  - Executing the statement “`return expression;`” to return the value of *expression* to the caller
- When a return statement executes, control returns immediately to the point at which the function was invoked

# Programmer-Defined Function maximum

- The script in our next example (Fig. 9.3) uses a programmer-defined function called maximum to determine and return the largest of three floating-point values.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 9.3: maximum.html -->
4 <!-- Programmer-Defined maximum function. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Maximum of Three Values</title>
9     <style type = "text/css">
10       p { margin: 0; }
11     </style>
12     <script>
13
14     var input1 = window.prompt( "Enter first number", "0" );
15     var input2 = window.prompt( "Enter second number", "0" );
16     var input3 = window.prompt( "Enter third number", "0" );
17
18     var value1 = parseFloat( input1 );
19     var value2 = parseFloat( input2 );
20     var value3 = parseFloat( input3 );
21
22     var maxValue = maximum( value1, value2, value3 );
23
24
25   document.writeln( "<p>First number: " + value1 + "</p>" +
26                     "<p>Second number: " + value2 + "</p>" +
27                     "<p>Third number: " + value3 + "</p>" +
28                     "<p>Maximum is: " + maxValue + "</p>" );
29
30   // maximum function definition (called from line 22)
31   function maximum( x, y, z )
32   {
33     return Math.max( x, Math.max( y, z ) );
34   } // end function maximum
35
36   </script>
37 </head><body></body>
38 </html>
```

Fig. 9.3 | Programmer-defined maximum function. (Part 1 of 4.)

Fig. 9.3 | Programmer-defined maximum function. (Part 2 of 4.)

# Programmer-Defined Function maximum

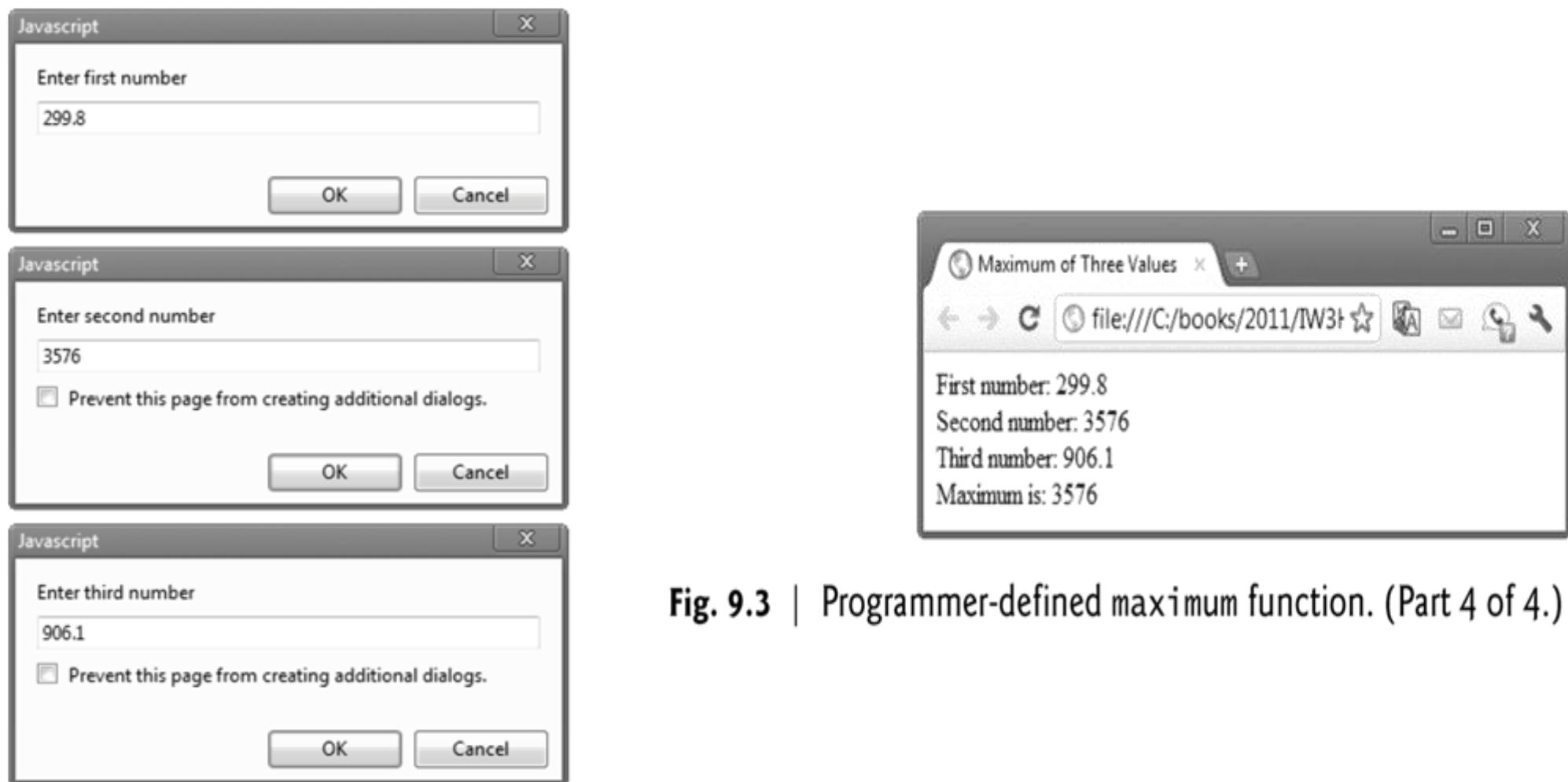


Fig. 9.3 | Programmer-defined maximum function. (Part 4 of 4.)

Fig. 9.3 | Programmer-defined maximum function. (Part 3 of 4.)

# Notes on Programmer-Defined Functions

- All variables declared with the keyword var in function definitions are local variables—this means that they can be accessed only in the function in which they're defined.
- A function's parameters are also considered to be local variables.
- There are several reasons for modularizing a program with functions.
  - Divide-and-conquer approach makes program development more manageable.
  - Software reusability.
  - Avoid repeating code in a program.



## Software Engineering Observation 9.1

If a function's task cannot be expressed concisely, perhaps the function is performing too many different tasks. It's usually best to break such a function into several smaller functions.



## Common Programming Error 9.2

Redefining a function parameter as a local variable in the function is a logic error.



## Good Programming Practice 9.1

Do not use the same name for an argument passed to a function and the corresponding parameter in the function definition. Using different names avoids ambiguity.



## Software Engineering Observation 9.2

To promote software reusability, every function should be limited to performing a single, well-defined task, and the name of the function should describe that task effectively. Such functions make programs easier to write, debug, maintain and modify.

# Random Number Generation

- random method generates a floating-point value from 0.0 up to, but *not* including, 1.0
- Random integers in a certain range can be generated by scaling and shifting the values returned by random, then using Math.floor to convert them to integers
  - The scaling factor determines the size of the range (i.e. a scaling factor of 4 means four possible integers)
  - The shift number is added to the result to determine where the range begins (i.e. shifting the numbers by 3 would give numbers between 3 and 7)
- Method Math.floor rounds its argument down to the closest integer

```
I  <!DOCTYPE html>
2
3  <!-- Fig. 9.4: RandomInt.html -->
4  <!-- Random integers, shifting and scaling. -->
5  <html>
6    <head>
7      <meta charset = "utf-8">
8      <title>Shifted and Scaled Random Integers</title>
9      <style type = "text/css">
10        p, ol { margin: 0; }
11        li { display: inline; margin-right: 10px; }
12      </style>
13      <script>
14
15        var value;
16
17        document.writeln( "<p>Random Numbers</p><ol>" );
18
19        for ( var i = 1; i <= 30; ++i ) {
20          value = Math.floor( 1 + Math.random() * 6 );
21          document.writeln( "<li>" + value + "</li>" );
22        } // end for
23
24
25        document.writeln( "</ol>" );
26
27      </script>
28    </head><body></body>
29  </html>
```

Fig. 9.4 | Random integers, shifting and scaling. (Part 1 of 2.)

```
25      document.writeln( "</ol>" );
26
27    </script>
28  </head><body></body>
29 </html>
```

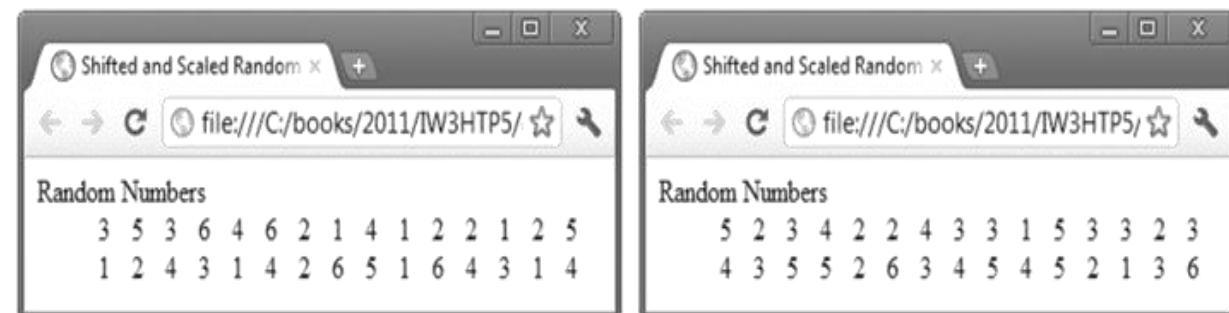


Fig. 9.4 | Random integers, shifting and scaling. (Part 2 of 2.)

## Scope Rules

- Each identifier in a program has a scope
- The **scope** of an identifier for a variable or function is the portion of the program in which the identifier can be referenced
- **Global variables** or **script-level variables** are accessible in any part of a script and are said to have **global scope**
  - Thus every function in the script can potentially use the variables
- Identifiers declared inside a function have function (or local) scope and can be used only in that function  
*St. Objd and*
- Function scope begins with the opening left brace ({}) of the function in which the identifier is declared and ends at the terminating right brace (})
- Local variables of a function and function parameters have function scope
- If a local variable in a function has the same name as a global variable, the global variable is “hidden” from the body of the function.

# Scope Rules



## Good Programming Practice 9.2

Avoid local-variable names that hide global-variable names. This can be accomplished by simply avoiding the use of duplicate identifiers in a script.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 9.9: scoping.html -->
4 <!-- Scoping example. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Scoping Example</title>
9     <style type = "text/css">
10       p      { margin: 0px; }
11       p.space { margin-top: 10px; }
12     </style>
```

```
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

```
<script>
  var output; // stores the string to display
  var x = 1; // global variable

  function start()
  {
    var x = 5; // variable local to function start

    output = "<p>local x in start is " + x + "</p>";

    functionA(); // functionA has local x
    functionB(); // functionB uses global variable x
    functionA(); // functionA reinitializes local x
    functionB(); // global variable x retains its value

    output += "<p class='space'>local x in start is " + x +
              "</p>";
    document.getElementById( "results" ).innerHTML = output;
  } // end function start
```

Fig. 9.9 | Scoping example. (Part 1 of 4.)

Fig. 9.9 | Scoping example. (Part 2 of 4.)

# Scope Rules

```
33 function functionA()
34 {
35     var x = 25; // initialized each time functionA is called
36
37     output += "<p class='space'>local x in functionA is " + x +
38         " after entering functionA</p>";
39     ++x; add after increment
40     output += "<p>local x in functionA is " + x +
41         " before exiting functionA</p>";
42 } // end functionA
43
44 function functionB()
45 {
46     output += "<p class='space'>global variable x is " + x +
47         " on entering functionB";
48     x *= 10;
49     output += "<p>global variable x is " + x +
50         " on exiting functionB</p>";
51 } // end functionB
52
53 window.addEventListener( "load", start, false );
54 </script>
55 </head>
```

```
56 <body>
57     <div id = "results"></div>
58 </body>
59 </html>
```



Fig. 9.9 | Scoping example. (Part 3 of 4.)

Fig. 9.9 | Scoping example. (Part 4 of 4.)

# JavaScript Global Functions

- JavaScript provides nine global functions as part of a Global object
- This object contains
  - all the global variables in the script
  - all the user-defined functions in the script
  - all the built-in global functions listed in the following slide
- You do not need to use the Global object directly; JavaScript uses it for you

Global function	Description
isFinite	Takes a numeric argument and returns <code>true</code> if the value of the argument is not <code>Nan</code> , <code>Number.POSITIVE_INFINITY</code> or <code>Number.NEGATIVE_INFINITY</code> (values that are not numbers or numbers outside the range that JavaScript supports)—otherwise, the function returns <code>false</code> .
isNaN	Takes a numeric argument and returns <code>true</code> if the value of the argument is not a number; otherwise, it returns <code>false</code> . The function is commonly used with the return value of <code>parseInt</code> or <code>parseFloat</code> to determine whether the result is a proper numeric value.
parseFloat	Takes a string argument and attempts to convert the <i>beginning</i> of the string into a floating-point value. If the conversion is unsuccessful, the function returns <code>Nan</code> ; otherwise, it returns the converted value (e.g., <code>parseFloat("abc123.45")</code> returns <code>Nan</code> , and <code>parseFloat("123.45abc")</code> returns the value <code>123.45</code> ).
Global function	Description
parseInt	Takes a string argument and attempts to convert the beginning of the string into an integer value. If the conversion is unsuccessful, the function returns <code>Nan</code> ; otherwise, it returns the converted value (for example, <code>parseInt("abc123")</code> returns <code>Nan</code> , and <code>parseInt("123abc")</code> returns the integer value <code>123</code> ). This function takes an optional second argument, from 2 to 36, specifying the radix (or base) of the number. Base 2 indicates that the first argument string is in binary format, base 8 that it's in octal format and base 16 that it's in hexadecimal format. See Appendix E, for more information on binary, octal and hexadecimal numbers.

Fig. 9.10 | JavaScript global functions. (Part 1 of 2.)

Fig. 9.10 | JavaScript global functions. (Part 2 of 2.)

# UCCD2323 Front-End Web Development



## Chapter 3 Client-side Programming on Event Handling Part 3.

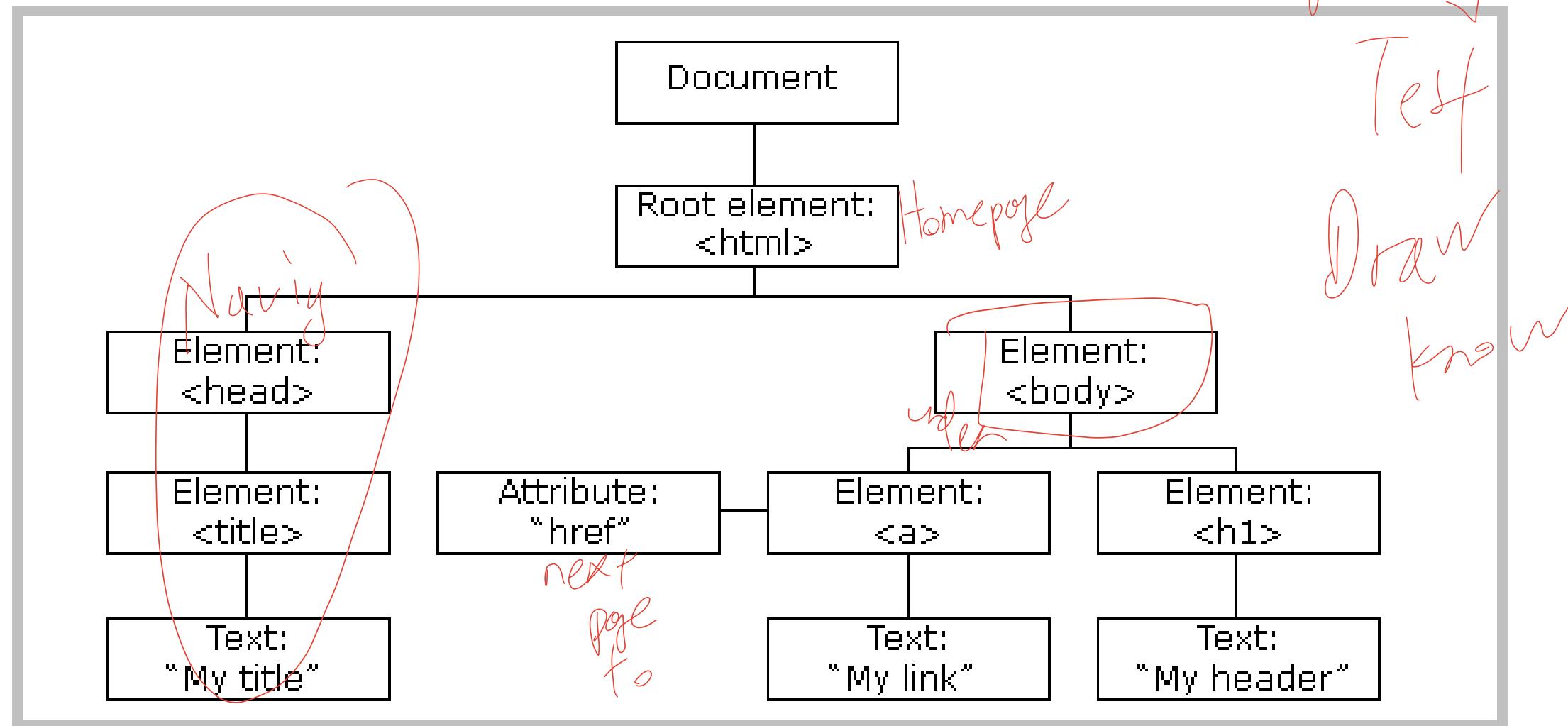
# Objectives

- In this chapter you will:
  - Learn about the Document Object Model
  - Learn about events
  - Writing event handlers and handling the events

Draw event  
occurs

# The HTML DOM Tree of Objects

Comprehension



## What is the HTML DOM?

Sketch the & page Subpage X pages

- The HTML DOM is a standard object model and programming interface for HTML. It defines:
  - The HTML elements as objects
  - The properties of all HTML elements
  - The methods to access all HTML elements
  - The events for all HTML elements
- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

# JavaScript – HTML DOM Methods

- HTML DOM methods are actions you can perform (on HTML Elements).
- HTML DOM properties are values (of HTML Elements) that you can set or change.

## The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript.
- In the DOM, all HTML elements are defined as objects.
- The programming interface is the properties and methods of each object.
- A property is a value that you can get or set (like changing the content of an HTML element).
- A method is an action you can do (like add or deleting an HTML element).

# The DOM Programming Interface

- Change the content of the `<p>` element with `id="demo"` using the `innerHTML` property.

```
<html>
<body>
    <p id="demo"></p>

    <script> $ 
        document.getElementById("demo").innerHTML = "Hello World!";
    </script>

    </body>
</html>
```

HTML tag →

Handwritten annotations: A red circle highlights the `id="demo"` attribute in the `<p>` tag. Another red circle highlights the `document.getElementById("demo")` part of the JavaScript code. A handwritten note `HTML tag` with an arrow points to the `<p>` tag. A handwritten note `HTML tag` with an arrow points to the `innerHTML` property in the code.

- Output:

My First Page

Hello World!

# The DOM Programming Interface

- getElementById is a method, while innerHTML is a property.
- The getElementById method used id="demo" to find the element
- The innerHTML property is used to get or replace the content of HTML elements  
*Value for find bulk that element*
- Note: The innerHTML property can be used to get or change any HTML element, including <html> and <body>

# Finding HTML Elements

## Method

document.getElementById(*id*)

## Description

Find an element by element id

document.getElementsByTagName(*name*)

Find elements by tag name

document.getElementsByClassName(*name*)

Find elements by class name

# Changing HTML Elements

## Property

`element.innerHTML = new html content`

## Description

Change the inner HTML of an element

`element.attribute = new value`

Change the attribute value of an HTML element

`element.style.property = new style CSS`

*or*  
*HTML*

Change the style of an HTML element

## Method

`element.setAttribute(attribute, value)`

*declare*

## Description

Change the attribute value of an HTML element

# Adding and Deleting Elements

## Method

## Description

`document.createElement(element)`

Create an HTML element

`document.removeChild(element)`

Remove an HTML element

`document.appendChild(element)`

Add an HTML element

`document.replaceChild(new, old)`

Replace an HTML element

`document.write(text)`

Write into the HTML output stream

# Introduction to events

- What are events?
  - Events are actions or occurrences that happen in the system you are programming
- Example of events:
  - user clicks a button on a webpage
  - The user clicking the mouse over a certain element or hovering the cursor over a certain element.
  - The user pressing a key on the keyboard.
  - The user resizing or closing the browser window.
  - A web page finishing loading.
  - A form being submitted.
  - A video being played, or paused, or finishing play.
  - An error occurring.

# Introduction to events

- Need to respond to the events through the use of event handler
  - Eg. Displaying an information box
  - Need to register the event handler
- What is event handler?
  - Event handler is a block of code (usually a JavaScript function that you as a programmer create) that will be run when the event fires
- What are event listeners?
  - Event listeners listen for events
  - The listener listens out for the event happening, and the handler is the code that is run in response to it happening.

# Introduction to events

- (html)

```
<button>Change color</button>
```

me c(,7k → event bind(e)

- (JavaScript)

```
const btn = document.querySelector('button');

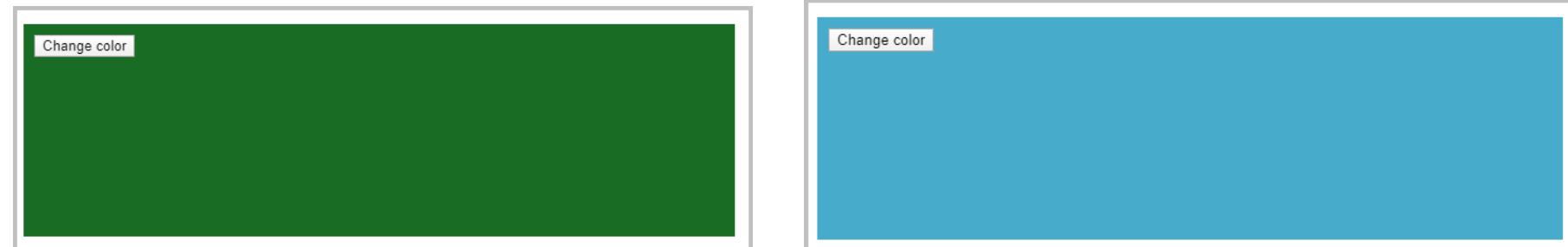
function random(255number) {
    return Math.floor(Math.random() * (number+1));
}

do btn.onclick = function() {
    const rndCol = 'rgb(' + random(255) + ', ' + random(255) + ', ' + random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}
```

for each

# Introduction to events

- A reference to the button is stored inside a constant called btn using the Document.querySelector() function
- The “random” function returns a random number
- The btn constant points to a <button> element
- Listening for the click event firing by setting the onclick event handler property to equal an anonymous function
- The code runs whenever the click event fires on the <button> element, that is, whenever a user clicks on it.



# Ways of using web events

- (1) Event handler properties
- (2) Inline event handlers — **don't use these**
- (3) addEventListener() and removeEventListener()

## (1) Event handler **properties**

- In Eg1.html, the onclick property is the event handler property
- Setting the handler property to be equal to an anonymous function as in Eg1.html or to a named function name as in the example below:

```
const btn = document.querySelector('button');

function bgChange() {
    const rndCol = 'rgb(' + random(255) + ', ' + random(255) + ', ' + random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}

btn.onclick = bgChange;
```

## (2) ~~Inline~~ event handlers – don't use these

- (html)

```
<button onclick="bgChange()">Press me</button>
```

- (JavaScript)

```
function bgChange() {  
    const rndCol = 'rgb(' + random(255) + ', ' + random(255) + ', ' + random(255) + ')';  
    document.body.style.backgroundColor = rndCol;  
}
```

- Explanation:
  - The attribute value is literally the JavaScript code you want to run when the event occurs.
  - This method is unmanageable and inefficient.
  - With JavaScript, you could easily add an event handler function to all the buttons on the page no matter how many there were, using something like this:

## (2) Inline event handlers – use the methods shown below

- Method (1)

```
const buttons = document.querySelectorAll('button');
Fixed (buttons)
for (let i = 0; i < buttons.length; i++) {
    buttons[i].onclick = bgChange;
}
```

- Method (2)

```
buttons.forEach(function(button) {
    button.onclick = bgChange;
});
```

### (3) addEventListener() and removeEventListener()

- This is defined in the Document Object Model (DOM) Level 2 Events Specification
- Example,
  - (JavaScript)

The code:

```
const btn = document.querySelector('button');

function bgChange() {
    const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}

btn.addEventListener('click', bgChange);
```

Annotations:

- A red circle surrounds the `bgChange()` function.
- A red arrow points from the word `click` in the first parameter of `addEventListener` to a callout box containing the text: "The name of the event we want to register this handler for".
- A red arrow points from the `bgChange()` parameter in the same line to another callout box containing the text: "The code that comprises the handler function".
- Handwritten notes:
  - "bgChange" is written next to the `bgChange()` function.
  - "change" is written below the `click` in the `addEventListener` parameter.
  - "is ?" is written below the `bgChange()` parameter.

### (3) addEventListener() and removeEventListener()

- Can be used to register multiple handlers for the same listener.
- Note that it is perfectly appropriate to put all the code inside the addEventListener() function, in an anonymous function, like this:
- Example,
  - (JavaScript)

```
btn.addEventListener('click', function() {  
    var rndCol = 'rgb(' + random(255) + ', ' + random(255) + ', ' + random(255) + ')';  
    document.body.style.backgroundColor = rndCol;  
});
```

### (3) addEventListener() and removeEventListener()

- To remove a previously added listener, use removeEventListener()
- This allows you to have the same button performing different actions in different circumstances — all you have to do is add or remove event handlers as appropriate.
- Example,
  - (JavaScript)

```
btn.removeEventListener('click', bgChange);
```

### (3) addEventListener() and removeEventListener()

- To register multiple handlers for the same listener,
  - Example,
  - (JavaScript)

```
myElement.addEventListener('click', functionA);
myElement.addEventListener('click', functionB);
```

multiple  
handlers  
declare sepeate

- The example below will not register both the handlers. The second line overwrites the value of onclick set by the first line

```
myElement.onclick = functionA;
myElement.onclick = functionB;
```

for  
function A  
function B

## What mechanism should I use?

- Event handler properties have less power and options, but better cross-browser compatibility (being supported as far back as Internet Explorer 8). You should probably start with these as you are learning.
- DOM Level 2 Events (`addEventListener()`, etc.) are more powerful, but can also become more complex and are less well supported (supported as far back as Internet Explorer 9). You should also experiment with these, and aim to use them where possible.

# Other event concepts

- Event objects
- Preventing default behaviour
- Event bubbling and capture

## Event objects

- Event object is parameter specified with a name such as event, evt, or simply e in the event handler function.
- Their values are automatically passed to event handlers to provide extra features and information.
- Example
  - (JavaScript)

```
function bgChange(e) {  
    const rndCol = 'rgb(' + random(255) + ', ' + random(255) + ', ' + random(255) + ')';  
    e.target.style.backgroundColor = rndCol;  
    console.log(e);  
}  
  
btn.addEventListener('click', bgChange);
```

# Event objects

- Explanation
  - `e.target` refers to the button itself.
  - The `target` property of the event object is always a reference to the element that the event has just occurred upon.
  - In this example, a random background color is set on the button.
- Example 2
  - Select all `<div>` elements using `document.querySelectorAll()`, then loop through each one, adding an `onclick` handler to each that makes it so that a random color is applied to each one when clicked.
  - (JavaScript)

```
const divs = document.querySelectorAll('div');
for (let i = 0; i < divs.length; i++) {
    divs[i].onclick = function(e) {
        e.target.style.backgroundColor = bgChange();
    }
}
```

The code above is annotated with handwritten notes:

- A red circle highlights the word "target" in the line `e.target.style.backgroundColor`.
- Handwritten text next to the circle says "1 div" and "2 divs".
- A red bracket is drawn around the entire assignment expression `= bgChange();`. Handwritten text next to the bracket says "color" and "background color".

# Preventing default behavior

Server will change  
D 不要走 js

- Usage:
  - prevent an event from doing what it does by default
- Example,
  - Default behavior of a submit button is for the data to be submitted to a specified page on the server for processing, and the browser to be redirected to the page of some kind (or the same page, if another is not specified.)

- (HTML):

```
<form>
  <div>
    <label for="fname">First name: </label>
    <input id="fname" type="text">
  </div>
  <div>
    <label for="lname">Last name: </label>
    <input id="lname" type="text">
  </div>
  <div>
    <input id="submit" type="submit">
  </div>
</form>
<p></p>
```

Azure  
14 day  
temp  
use

↓ (on H form, hmc ~ (put))

# Preventing default behavior

- (JavaScript):

```
const form = document.querySelector('form');
const fname = document.getElementById('fname');
const lname = document.getElementById('lname');
const para = document.querySelector('p');

form.onsubmit = function(e) {
    if (fname.value === '' || lname.value === '') {
        e.preventDefault();
        para.textContent = 'You need to fill in both names!';
    }
}
```

JS / -W not store DB

- Explanation

- The code inside onsubmit event handler (the submit event is fired on a form when it is submitted) tests whether the text fields are empty. If they are, the preventDefault() function on the event object stops the form submission and then display an error message in the paragraph below our form to tell the user what's wrong.

- Output:

First name:

Last name:

Submit

You need to fill in both names!

# Event bubbling and capture

- **What happens in the capturing phase.**

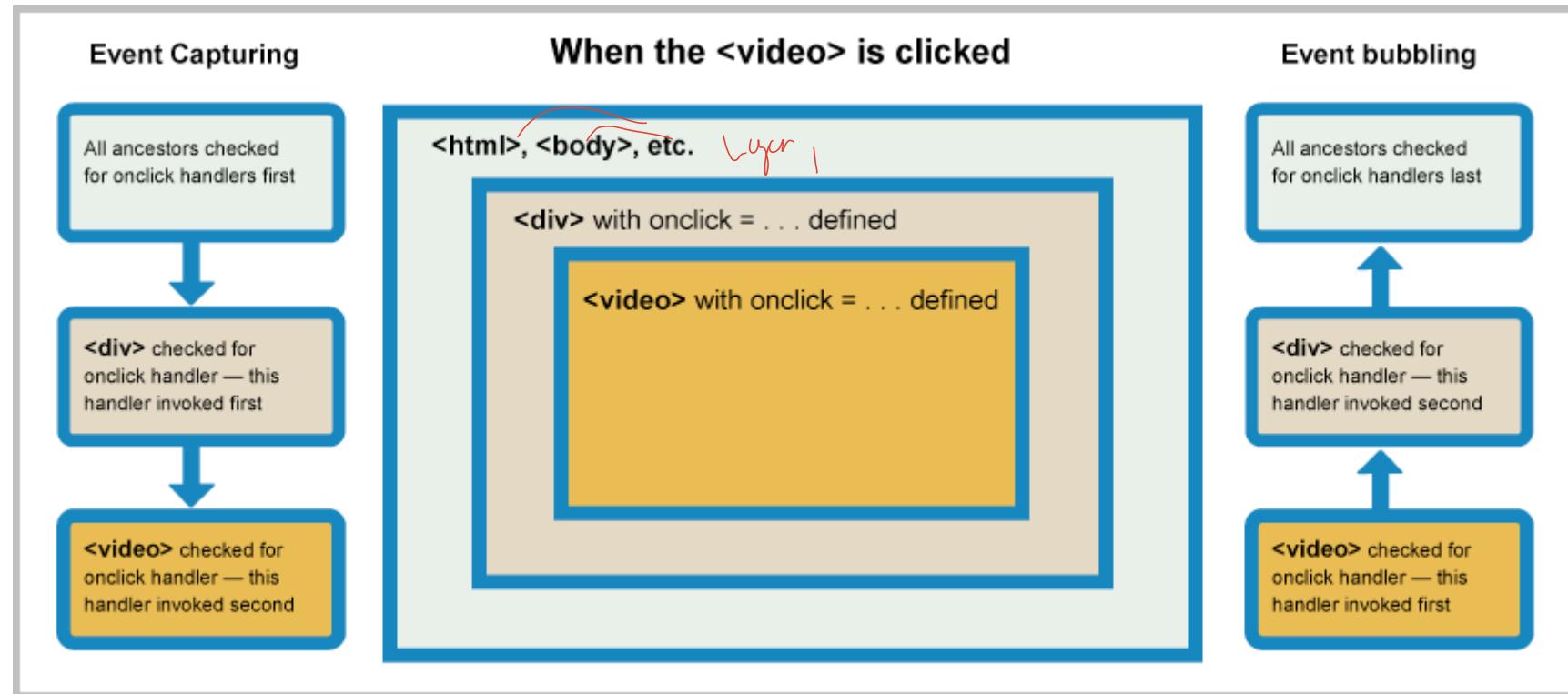
- The browser checks to see if the element's outer-most ancestor (<html>) has an onclick event handler registered on it in the capturing phase, and runs it if so.
- Then it moves on to the next element inside <html> and does the same thing, then the next one, and so on until it reaches the element that was actually clicked on.

↑ error = buffer we expect

- **What happens in the bubbling phase.**

- The browser checks to see if the element that was actually clicked on has an onclick event handler registered on it in the bubbling phase, and runs it if so.
- Then it moves on to the next immediate ancestor element and does the same thing, then the next one, and so on until it reaches the <html> element.

# Event bubbling and capture



- In modern browsers, by default, all event handlers are registered in the bubbling phase.
- We can use the `addEventListener(type, listener, useCapture)` to register event handlers in the bubbling phase (default) or in the capturing phase.

# Event bubbling and capture

- Example, Eg2.html

- (HTML):

```
<button>Display video</button>

<div class="hidden">
  <video>
    <source src="rabbit320.mp4" type="video/mp4">
    <source src="rabbit320.webm" type="video/webm">
    <p>Your browser doesn't support HTML5 video. Here is a <a href="rabbit320.mp4">link to the video</a> instead.</p>
  </video>
</div>
```

```
<style>
  div {
    position: absolute;
    top: 50%;
    transform: translate(-50%, -50%);
    width: 480px;
    height: 380px;
    border-radius: 10px;
    background-color: #eee;
    background-image: linear-gradient (to bottom, rgba(0, 0, 0, 0),
      rgba(0, 0, 0, 0.1));
  }
  .hidden {
    left: -50%;
  }
  .showing {
    left: 50%;
  }
  div video {
    display: block;
    width: 400px;
    margin: 40px auto;
  }
</style>
```

- (CSS):

# Event bubbling and capture

- (JavaScript):

```
<script> fixd pointer
const btn = document.querySelector('button');
const videoBox = document.querySelector('div');
const video = document.querySelector('video');

btn.onclick = function() {
  videoBox.setAttribute('class', 'showing');
}

videoBox.onclick = function() {
  videoBox.setAttribute('class', 'hidden');
};

video.onclick = function() {
  video.play();
};

</script>
```

Video  
in  
a  
box  
fixd  
pointer



- Explanation

- When the <button> is clicked, the video is displayed, by changing the class attribute on the <div> from hidden to showing

```
btn.onclick = function() {
  videoBox.setAttribute('class', 'showing');
}
```

# Event bubbling and capture

- When the video is clicked, it starts to play, but it causes the <div> to also be hidden at the same time. This is because the video is inside the <div>. Clicking on the video actually runs both the above event handlers.

```
videoBox.onclick = function() {  
    videoBox.setAttribute('class', 'hidden');  
};  
  
video.onclick = function() {  
    video.play();  
};
```

## Fixing the problem with stopPropagation()

- The standard Event object has a function available on it called `stopPropagation()` which, when invoked on a handler's event object, makes it so that first handler is run but the event doesn't bubble any further up the chain, so no more handlers will be run.
- (JavaScript)

dihel

```
video.onclick = function(e) {  
    e.stopPropagation(); →  
    video.play();  
};
```

# UCCD2323 Front-End Web Development



## Chapter 3 JavaScript Object Part 1.

# Topic Outline

Introduction

Math Object

String Object

11.3.1 Fundamentals of Characters and Strings

11.3.2 Methods of the String Object

11.3.3 Character-Processing Methods

11.3.4 Searching Methods

11.3.5 Splitting Strings and Obtaining Substrings

Date Object

Boolean and Number Objects

document Object

Favorite Twitter Searches: HTML5 Web Storage

Using JSON to Represent Objects

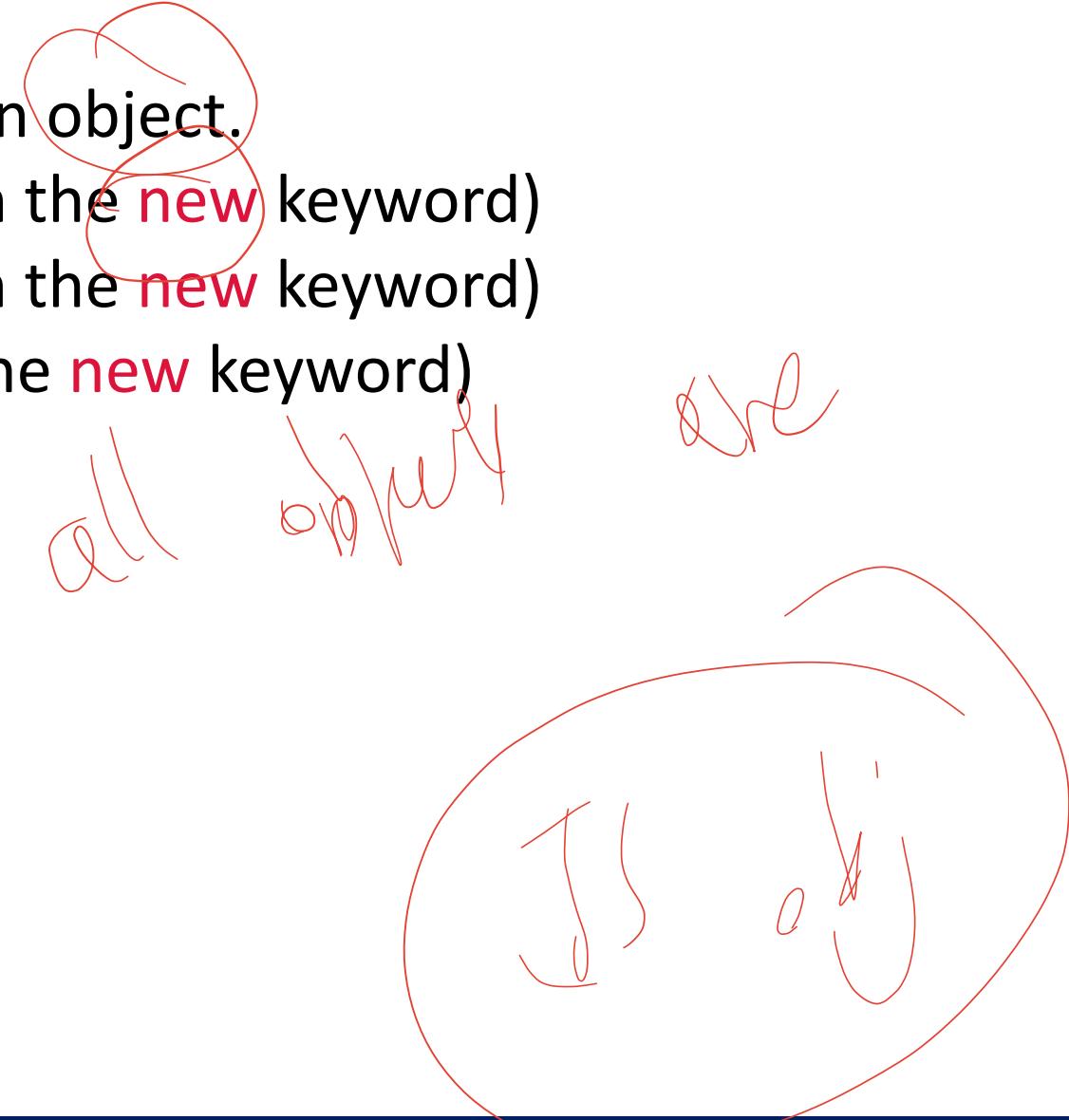
# Objectives

In this chapter you'll:

- Learn object-based programming terminology and concepts.
- Learn the concepts of encapsulation and data hiding.
- Learn the value of object orientation.
- Use the methods of the JavaScript objects Math, String, Date, Boolean and Number.
- Use HTML5 web storage to create a web application that stores user data locally.
- Represent objects simply using JSON.

# Introduction

- In JavaScript, almost "everything" is an object.
- Booleans can be objects (if defined with the **new** keyword)
- Numbers can be objects (if defined with the **new** keyword)
- Strings can be objects (if defined with the **new** keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects



# Objects in JavaScript

- JavaScript supports two kinds of objects
  - 1) **Intrinsic (or "built-in") objects**, includes **Array, Boolean, Date, Math** and **String**.
  - 2) **User-define objects**

## Built-in Object- Math Object

- JMath object methods enable you to conveniently perform many common mathematical calculations.
- An object's methods are called by writing the **name of the object** followed by a dot operator (.) and the **name of the method**  
e.g: **Math.pow (2,3)**
- In parentheses following the method name is **arguments to the method**



### Software Engineering Observation 11.1

The difference between invoking a stand-alone function and invoking a method of an object is that an object name and a dot are not required to call a stand-alone function.

# Built-in Object- Math Object

Method	Description	Examples
<code>abs( x )</code>	Absolute value of x.	<code>abs( 7.2 )</code> is 7.2 <code>abs( 0 )</code> is 0 <code>abs( -5.6 )</code> is 5.6
<code>ceil( x )</code>	Rounds x to the smallest integer not less than x.	<code>ceil( 9.2 )</code> is 10 <code>ceil( -9.8 )</code> is -9.0
<code>cos( x )</code>	Trigonometric cosine of x (x in radians).	<code>cos( 0 )</code> is 1
<code>exp( x )</code>	Exponential method $e^x$ .	<code>exp( 1 )</code> is 2.71828 <code>exp( 2 )</code> is 7.38906
<code>floor( x )</code>	Rounds x to the largest integer not greater than x.	<code>floor( 9.2 )</code> is 9 <code>floor( -9.8 )</code> is -10.0
<code>log( x )</code>	Natural logarithm of x (base e).	<code>log( 2.718282 )</code> is 1 <code>log( 7.389056 )</code> is 2
<code>max( x, y )</code>	Larger value of x and y.	<code>max( 2.3, 12.7 )</code> is 12.7 <code>max( -2.3, -12.7 )</code> is -2.3

Fig. 11.1 | Math object methods. (Part 1 of 2.)

Method	Description	Examples
<code>min( x, y )</code>	Smaller value of x and y.	<code>min( 2.3, 12.7 )</code> is 2.3 <code>min( -2.3, -12.7 )</code> is -12.7
<code>pow( x, y )</code>	x raised to power y ( $x^y$ ).	<code>pow( 2, 7 )</code> is 128 <code>pow( 9, .5 )</code> is 3.0
<code>round( x )</code>	Rounds x to the closest integer.	<code>round( 9.75 )</code> is 10 <code>round( 9.25 )</code> is 9
<code>sin( x )</code>	Trigonometric sine of x (x in radians).	<code>sin( 0 )</code> is 0
<code>sqrt( x )</code>	Square root of x.	<code>sqrt( 900 )</code> is 30 <code>sqrt( 9 )</code> is 3
<code>tan( x )</code>	Trigonometric tangent of x (x in radians).	<code>tan( 0 )</code> is 0

Fig. 11.1 | Math object methods. (Part 2 of 2.)

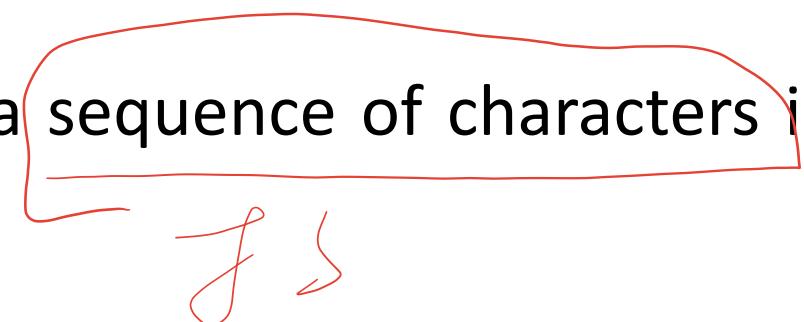
# Built-in Object- Math Object

Constant	Description	Value
Math.E	Base of a natural logarithm ( $e$ ).	Approximately 2.718
Math.LN2	Natural logarithm of 2.	Approximately 0.693
Math.LN10	Natural logarithm of 10.	Approximately 2.302
Math.LOG2E	Base 2 logarithm of $e$ .	Approximately 1.442
Math.LOG10E	Base 10 logarithm of $e$ .	Approximately 0.434
Math.PI	$\pi$ —the ratio of a circle's circumference to its diameter.	Approximately 3.141592653589793
Math.SQRT1_2	Square root of 0.5.	Approximately 0.707
Math.SQRT2	Square root of 2.0.	Approximately 1.414

**Fig. 11.2 | Properties of the Math object.**

# Built-in Object- String Object

- Characters are the building blocks of JavaScript programs
- Every program is composed of a sequence of characters grouped together meaningfully that is interpreted by the computer as a series of instructions used to accomplish a task
- A string is a series of characters treated as a single unit
- A string may include letters, digits and various **special characters**, such as +, -, \*, /, and \$
- JavaScript supports Unicode, which represents a large portion of the world's languages
- **String literals or string constants** are written as a sequence of characters in double or single quotation marks



# Methods of the String Object

- Combining strings is called **concatenation**

Method	Description	Method	Description
<code>charAt( index )</code>	Returns a string containing the character at the specified <i>index</i> . If there's no character at the <i>index</i> , <code>charAt</code> returns an empty string. The first character is located at <i>index</i> 0.	<code>lastIndexOf( substring, index )</code>	Searches for the <i>last</i> occurrence of <i>substring</i> starting from position <i>index</i> and searching toward the beginning of the string that invokes the method. The method returns the starting index of <i>substring</i> in the source string or -1 if <i>substring</i> is not found. If the <i>index</i> argument is not provided, the method begins searching from the <i>end</i> of the source string.
<code>charCodeAt( index )</code>	Returns the Unicode value of the character at the specified <i>index</i> , or <code>NaN</code> (not a number) if there's no character at that <i>index</i> .	<code>replace( searchString, replaceString )</code>	Searches for the substring <i>searchString</i> , replaces the first occurrence with <i>replaceString</i> and returns the modified string, or returns the original string if no replacement was made.
<code>concat( string )</code>	Concatenates its argument to the end of the string on which the method is invoked. The original string is not modified; instead a new <code>String</code> is returned. This method is the same as adding two strings with the string-concatenation operator + (e.g., <code>s1.concat(s2)</code> is the same as <code>s1 + s2</code> ).	<code>slice( start, end )</code>	Returns a string containing the portion of the string from index <i>start</i> through index <i>end</i> . If the <i>end</i> index is not specified, the method returns a string from the <i>start</i> index to the end of the source string. A negative <i>end</i> index specifies an offset from the end of the string, starting from a position one past the end of the last character (so -1 indicates the last character position in the string).
<code>fromCharCode( value1, value2, ... )</code>	Converts a list of Unicode values into a string containing the corresponding characters.		
<code>indexOf( substring, index )</code>	Searches for the <i>first</i> occurrence of <i>substring</i> starting from position <i>index</i> in the string that invokes the method. The method returns the starting index of <i>substring</i> in the source string or -1 if <i>substring</i> is not found. If the <i>index</i> argument is not provided, the method begins searching from index 0 in the source string.		

Fig. 11.3 | Some String-object methods. (Part 1 of 3.)

Fig. 11.3 | Some String-object methods. (Part 2 of 3.)

# Methods of the String Object

Method	Description
<code>split( string )</code>	Splits the source string into an array of strings (tokens), where its <i>string</i> argument specifies the delimiter (i.e., the characters that indicate the end of each token in the source string).
<code>substr( start, length )</code>	Returns a string containing <i>length</i> characters starting from index <i>start</i> in the source string. If <i>length</i> is not specified, a string containing characters from <i>start</i> to the end of the source string is returned.
<code>substring( start, end )</code>	Returns a string containing the characters from index <i>start</i> up to but not including index <i>end</i> in the source string.
<code>toLowerCase()</code>	Returns a string in which all uppercase letters are converted to lowercase letters. Non-letter characters are not changed.
<code>toUpperCase()</code>	Returns a string in which all lowercase letters are converted to uppercase letters. Non-letter characters are not changed.

texting  
put pose

**Fig. 11.3 | Some String-object methods. (Part 3 of 3.)**

# Character Processing Methods

- String method charAt
  - Returns the character at a specific position
  - Indices for the characters in a string start at 0 (the first character) and go up to (but do not include) the string's length
  - If the index is outside the bounds of the string, the method returns an empty string
- String method charCodeAt
  - Returns the Unicode value of the character at a specific position
  - If the index is outside the bounds of the string, the method returns NaN.
- String method fromCharCode
  - Returns a string created from a series of Unicode values
- String method toLowerCase
  - Returns the lowercase version of a string
- String method toUpperCase
  - Returns the uppercase version of a string

String  
method

# Character Processing Methods

```
<html>
  <head>
    <meta charset = "utf-8">
    <title>Character Processing</title>
  </head>
  <body>

    <script>
      var s = "ZEBRA"; how many words
      var s2 = "AbCdEfG";
      result = "";

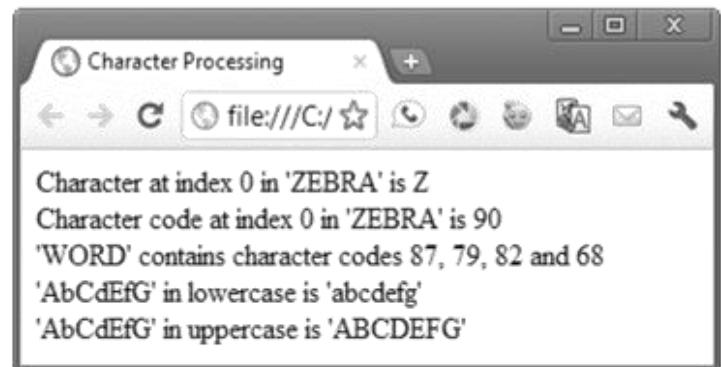
      result = "<p>Character at index 0 in '" + s + "' is " +
      s.charAt(0) + "</p>"; //Z
      result += "<p>Character code at index 0 in '" + s + "' is " +
      s.charCodeAt(0) + "</p>"; //90 try this ↪

      result += "<p>" + String.fromCharCode( 87, 79, 82, 68 ) +
      " contains character codes 87, 79, 82 and 68</p>"; //WORD

      result += "<p>" + s2 + "' in lowercase is '" +
      s2.toLowerCase() + "'</p>"; //abcdefg
      result += "<p>" + s2 + "' in uppercase is '" +
      s2.toUpperCase() + "'</p>"; //ABCDEFG

      document.write (result)

    </script>
  </body>
</html>
```



**Fig. 11.4** | HTML5 document to demonstrate methods `charAt`, `charCodeAt`, `fromCharCode`, `toLowerCase` and `toUpperCase`. (Part 2 of 2.)

## Searching Methods

- String method indexOf
  - Determines the location of the first occurrence of its argument in the string used to call the method
  - If the substring <sup>PwP</sup> is found, the index at which the first occurrence of the substring begins is returned; otherwise, -1 is returned
  - Receives an optional second argument specifying the index from which to begin the search
- String method lastIndexOf
  - Determines the location of the last occurrence of its argument in the string used to call the method
  - If the substring is found, the index at which the last occurrence of the substring begins is returned; otherwise, -1 is returned
  - Receives an optional second argument specifying the index from which to begin the search

# Searching Methods

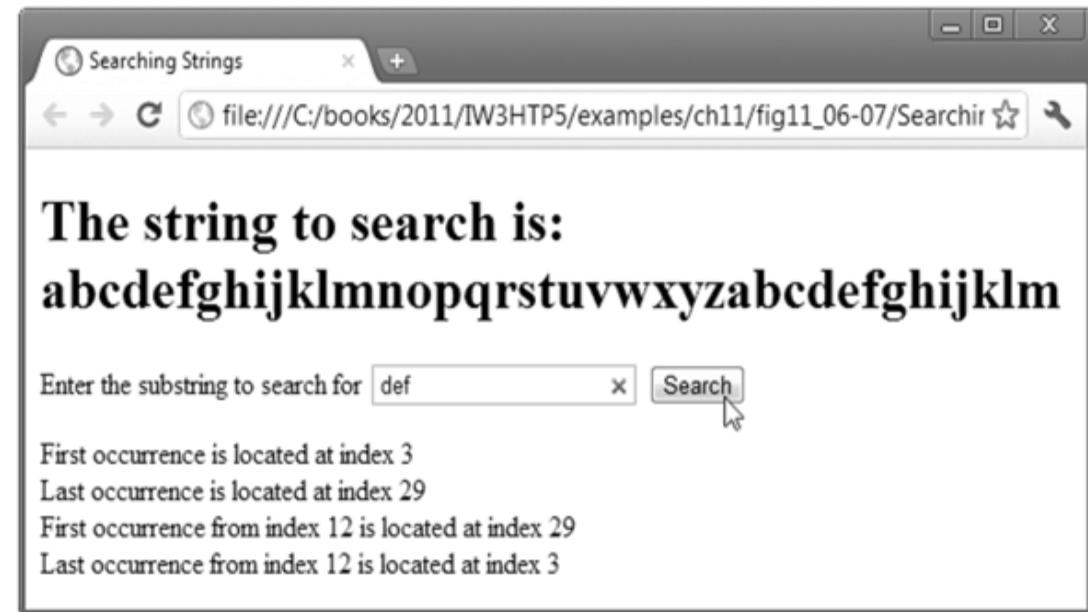
```
<body>
  <form id = "searchForm" action = "#">
    <h1>The string to search is:
      abcdefghijklmnopqrstuvwxyzabcdefghijklm</h1>
    <p>Enter the substring to search for
    <input id = "inputField" type = "text">
    <input id = "searchButton" type = "submit" value = "Search"></p>
    <div id = "display"></div>
  </form>
  <script>

$(function () {

  $("#searchForm").submit(function(event) {
    event.preventDefault();
    var letter = "abcdefghijklmnopqrstuvwxyzabcdefghijklm";
    var str = $("#inputField").val();

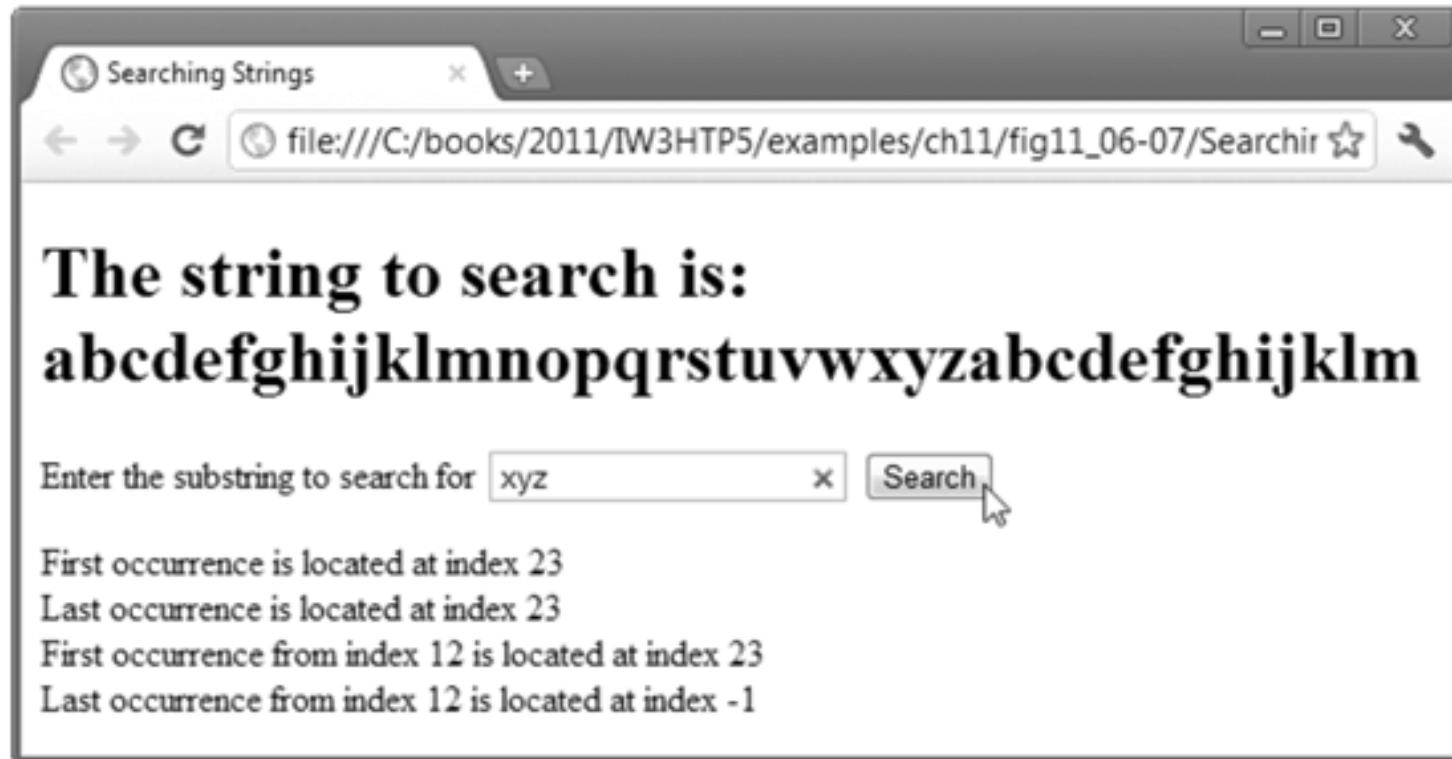
    $("#display").html( "<p>First occurrence is located at index " +
      letter.indexOf(str) + "</p>" +
      "<p>Last occurrence is located at index " +
      letter.lastIndexOf(str) + "</p>" +
      "<p>First occurrence from index 12 is located at index " +
      letter.indexOf(str, 12) + "</p>" +
      "<p>Last occurrence from index 12 is located at index " +
      letter.lastIndexOf(str, 12) + "</p>" );
  });
});

</script>
</body>
```



**Fig. 11.6 |** HTML document to demonstrate methods `indexOf` and `lastIndexOf`. (Part 2 of 3.)

# Searching Methods



**Fig. 11.6** | HTML document to demonstrate methods `indexOf` and `lastIndexOf`. (Part 3 of 3.)

# Splitting Strings and Obtaining Substrings

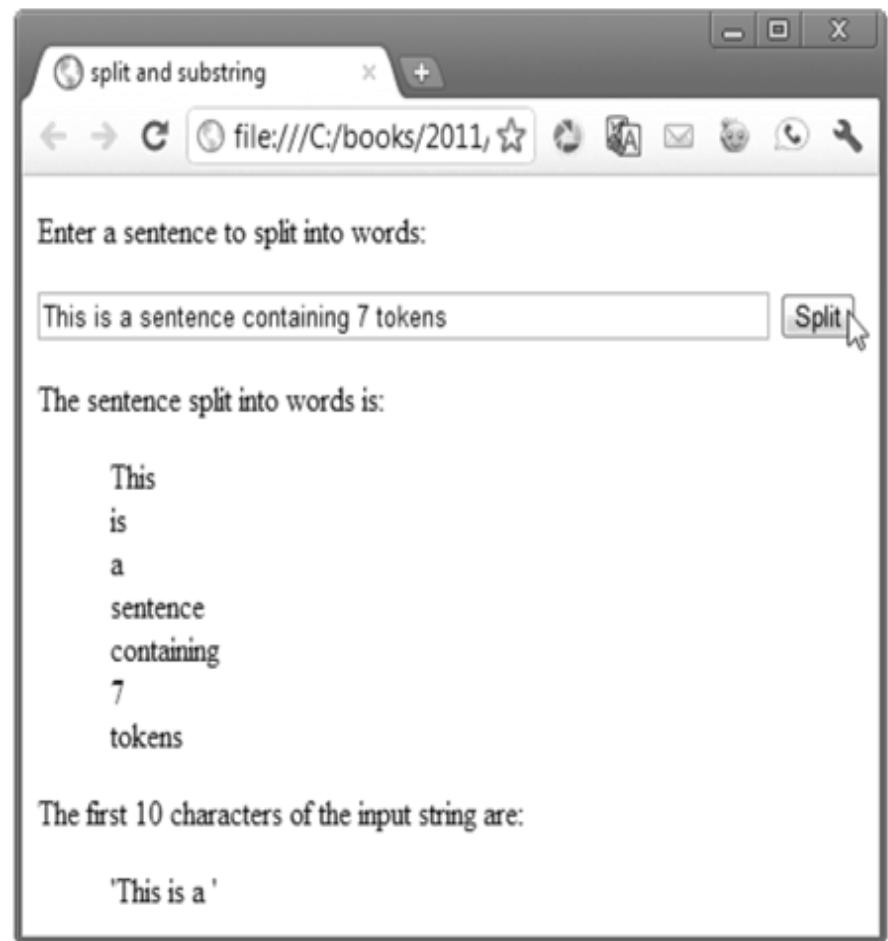
- Breaking a string into tokens is called **tokenization**
- Tokens are separated from one another by delimiters, typically white-space characters such as blank, tab, newline and carriage return
  - Other characters may also be used as delimiters to separate tokens
- String method `split`
  - Breaks a string into its component tokens
  - Argument is the delimiter string
  - Returns an array of strings containing the tokens
- String method `substring`
  - Returns the substring from the starting index (its first argument) up to but not including the ending index (its second argument)
  - If the ending index is greater than the length of the string, the substring returned includes the characters from the starting index to the end of the original string

# Splitting Strings and Obtaining Substrings

```
<body>
  <form id = "splitstring">
    <p>Enter a sentence to split into words:</p>
    <p><input id = "inputField" type = "text">
      <input id = "splitButton" type = "submit" value = "Split"></p>
    <div id = "results"></p>
  </form>
  <script>
    $(function () {
      $("#splitstring").submit(function(event) {
        event.preventDefault();

        var inputString = $("#inputField").val();
        var tokens = inputString.split(" ");
        (tokens taken 7)

        $("#results").html("The sentence split into words is: </p>" +
          "<p class = 'indent'>" +
          (tokens.join(" <br> ") + "</p>" +
          "<p>The first 10 characters of the input string are: </p>" +
          "<p class = 'indent'>" + inputString.substring(0, 10) + "</p>");
      });
    });
  </script>
</body>
```



**Fig. 11.8 |** HTML document demonstrating String methods `split` and `substring`. (Part 2 of 2.)

# Splitting Strings and Obtaining Substrings

- delimiter string  
the string that determines the end of each token in the original string.
- The method returns the substring from the **starting index** (0 in this example) up to but not including the **ending index** (10 in this example).
- If the ending index is greater than the length of the string, the substring returned includes the characters from the starting index to the end of the original string.

Alvin

# Built-in Object-~~Date~~ Object

- Date object provides methods for date and time manipulations
  - Based either on the ~~computer's local time~~ zone or on World Time Standard's Coordinated Universal Time (abbreviated UTC)
- Most methods have a local time zone and a UTC version
- Empty parentheses after an object name indicate a call to the object's constructor with no arguments
  - A constructor is an initializer method for an object
  - Called automatically when an object is allocated with new
  - The Date constructor with no arguments initializes the Date object with the local computer's current date and time
  - A new Date object can be initialized by passing the number of milliseconds since midnight, January 1, 1970, to the Date constructor
  - Can also create a new Date object by supplying arguments to the Date constructor for year, month, date, hours, minutes, seconds and milliseconds.
    - Hours, minutes, seconds and milliseconds arguments are all optional
    - If any one of these arguments is not specified, a zero is supplied in its place
    - If an argument is specified, all arguments to its left must be specified

```
var d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

*Standard University*

```
var d = new Date();
```

# Date Object

*needed for use*

Method	Description	Method	Description
getDate()	Returns a number from 1 to 31 representing the day of the month in local time or UTC.	getTime()	Returns the number of milliseconds between January 1, 1970, and the time in the Date object.
getUTCDate()		getTimezoneOffset()	Returns the difference in minutes between the current time on the local computer and UTC (Coordinated Universal Time).
getDay()	Returns a number from 0 (Sunday) to 6 (Saturday) representing the day of the week in local time or UTC.	setDate( val )	Sets the day of the month (1 to 31) in local time or UTC.
getUTCDay()		setUTCDate( val )	
getFullYear()	Returns the year as a four-digit number in local time or UTC.	setFullYear( y, m, d )	Sets the year in local time or UTC. The second and third arguments representing the month and the date are optional. If an optional argument is not specified, the current value in the Date object is used.
getUTCFullYear()		setUTCFullYear( y, m, d )	
getHours()	Returns a number from 0 to 23 representing hours since midnight in local time or UTC.	setHours( h, m, s, ms )	Sets the hour in local time or UTC. The second, third and fourth arguments, representing the minutes, seconds and milliseconds, are optional. If an optional argument is not specified, the current value in the Date object is used.
getUTCHours()		setUTCHours( h, m, s, ms )	
getMilliseconds()	Returns a number from 0 to 999 representing the number of milliseconds in local time or UTC, respectively. The time is stored in hours, minutes, seconds and milliseconds.	setMilliSeconds( ms )	Sets the number of milliseconds in local time or UTC.
getUTCMilliSeconds()		setUTCMilliseconds( ms )	
getMinutes()	Returns a number from 0 to 59 representing the minutes for the time in local time or UTC.		
getUTCMinutes()			
getMonth()	Returns a number from 0 (January) to 11 (December) representing the month in local time or UTC.		
getUTCMonth()			
getSeconds()	Returns a number from 0 to 59 representing the seconds for the time in local time or UTC.		
getUTCSecs()			

Fig. 11.10 | Date-object methods. (Part 1 of 4.)

Fig. 11.10 | Date-object methods. (Part 2 of 4.)

# Built-in Object-Date Object

Method	Description
<code>setMinutes( m, s, ms )</code>	Sets the minute in local time or UTC. The second and third arguments, representing the seconds and milliseconds, are optional. If an optional argument is not specified, the current value in the Date object is used.
<code>setUTCMinutes( m, s, ms )</code>	
<code>setMonth( m, d )</code>	Sets the month in local time or UTC. The second argument, representing the date, is optional. If the optional argument is not specified, the current date value in the Date object is used.
<code>setUTCMonth( m, d )</code>	
<code>setSeconds( s, ms )</code>	Sets the seconds in local time or UTC. The second argument, representing the milliseconds, is optional. If this argument is not specified, the current milliseconds value in the Date object is used.
<code>setUTCSeconds( s, ms )</code>	
<code>setTime( ms )</code>	Sets the time based on its argument—the number of elapsed milliseconds since January 1, 1970.
<code>toLocaleString()</code>	Returns a string representation of the date and time in a form specific to the computer's locale. For example, September 13, 2007, at 3:42:22 PM is represented as <i>09/13/07 15:47:22</i> in the United States and <i>13/09/07 15:47:22</i> in Europe.

Method	Description
<code>toUTCString()</code>	Returns a string representation of the date and time in the form: <i>15 Sep 2007 15:47:22 UTC</i> .
<code>toString()</code>	Returns a string representation of the date and time in a form specific to the locale of the computer ( <i>Mon Sep 17 15:47:22 EDT 2007</i> in the United States).
<code>valueOf()</code>	The time in number of milliseconds since midnight, January 1, 1970. (Same as <code>getTime()</code> .)

**Fig. 11.10 | Date-object methods. (Part 4 of 4.)**

**Fig. 11.10 | Date-object methods. (Part 3 of 4.)**

# Built-in Object-Date Object

- Date method **parse**
  - Receives as its argument a string representing a date and time and returns the number of milliseconds between midnight, January 1, 1970, and the specified date and time

```
var d = Date.parse("March 21, 2012");
```

- Output : 1332259200000
- Date method UTC
  - Returns the number of milliseconds between midnight, January 1, 1970, and the date and time specified as its arguments
  - Arguments include the required year, month and date, and the optional hours, minutes, seconds and milliseconds
  - If an argument is not specified, a 0 is supplied in its place
  - For hours, minutes and seconds, if the argument to the right of any of these arguments is specified, that argument must also be specified

```
var d = Date.UTC(2012, 02, 30);
```

declare

- Output : 1333065600000

↓ return in Universal Time Code

# Built-in Object-Date Object

```
<!DOCTYPE html>

<!-- Fig. 11.11: DateTime.html -->
<!-- HTML document to demonstrate Date object methods. -->
<html>
  <head>
    <meta charset = "utf-8">
    <title>Date and Time Methods</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jq

  </head>
  <body>
    <h1>String representations and valueOf</h1>
    <section id = "strings"></section>
    <h1>Get methods for local time zone</h1>
    <section id = "getMethods"></section>
    <h1>Specifying arguments for a new Date</h1>
    <section id = "newArguments"></section>
    <h1>Set methods for local time zone</h1>
    <section id = "setMethods"></section>
```

```
<script>
$(function () {
    JS

    var current = new Date();

    // string formatting methods and valueof

    $("#strings").html( "<p>toString: " + current.toString() + "</p>" +
        "<p>toLocaleString: " + current.toLocaleString() + "</p>" +
        "<p>toUTCString: " + current.toUTCString() + "</p>" +
        "<p>valueOf: " + current.valueOf() + "</p>" );

    //getmethod
    $("#getMethods").html("<p>getDate: " + current.getDate() + "</p>" +
        "<p>getDay: " + current.getDay() + "</p>" +
        "<p>getMonth: " + current.getMonth() + "</p>" +
        "<p>getFullYear: " + current.getFullYear() + "</p>" +
        "<p>getTime: " + current.getTime() + "</p>" +
        "<p>getHours: " + current.getHours() + "</p>" +
        "<p>getMinutes: " + current.getMinutes() + "</p>" +
        "<p>getSeconds: " + current.getSeconds() + "</p>" +
        "<p>getMilliseconds: " + current.getMilliseconds() + "</p>" +
        "<p>getTimezoneOffset: " + current.getTimezoneOffset() + "</p>" );

    // creating a Date not by retreat back
    var anotherDate = new Date( 2011, 2, 18, 1, 5, 0, 0 );
    $ ("#newArguments" ).html("<p>Date: " + anotherDate + "</p>");

    anotherDate.setDate( 31 );
    anotherDate.setMonth( 11 );
    anotherDate.setFullYear( 2011 );
    anotherDate.setHours( 23 );
    anotherDate.setMinutes( 59 );
    anotherDate.setSeconds( 59 );
    $ ("#setMethods" ).html("<p>Modified date: " + anotherDate + "</p>");

});;

</script> |
</body>
</html>
```

# Built-in Object-Date Object

Date and Time Methods

file:///C:/books/2011/TW3HTP5/e

## String representations and valueOf

toString: Thu Aug 11 2011 10:31:21 GMT-0400 (Eastern Daylight Time)  
toLocaleString: Thu Aug 11 2011 10:31:21 GMT-0400 (Eastern Daylight Time)  
toUTCString: Thu, 11 Aug 2011 14:31:21 GMT  
valueOf: 1313073081914

## Get methods for local time zone

getDate: 11  
getDay: 4  
getMonth: 7  
getFullYear: 2011  
getTime: 1313073081914  
getHours: 10  
getMinutes: 31  
getSeconds: 21  
getMilliseconds: 914  
getTimezoneOffset: 240

Fig. 11.11 | HTML document to demonstrate Date-object methods.

(Part 3 of 3)

## Specifying arguments for a new Date

Date: Fri Mar 18 2011 01:05:00 GMT-0400 (Eastern Daylight Time)

## Set methods for local time zone

Modified date: Sat Dec 31 2011 23:59:59 GMT-0500 (Eastern Standard Time)

Fig. 11.11 | HTML document to demonstrate Date-object methods.  
(Part 3 of 3.)



### Common Programming Error 11.1

Assuming that months are represented as numbers from 1 to 12 leads to off-by-one errors when you're processing Dates.

# UCCD2323 Front-End Web Development



## Chapter 3 JavaScript Object Part 2.

# Objects in JavaScript

- JavaScript is designed on a simple object-based paradigm. An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method.

## Objects in JavaScript: User-Defined Object

- In JavaScript there are several ways to create a custom object.
  1. Using the keyword new
  2. Constructor function
  3. Object Initializer/literal

# Object Creation – new keyword

OOP

- Objects can be created with a *new* expression
  - The most basic object is one that uses the Object constructor, as in  
`var myObject = new Object();`
- The new object has no properties - a blank object
- Properties can be added to any object, any time:  
`var myAirplane = new Object();  
myAirplane.make = "Cessna ";  
myAirplane.model = "Centurian";`
- The number of properties in a JavaScript object is dynamic
- Objects can be nested, so a property could be itself another object, created with new

# Object Creation – new keyword

- Properties can be accessed in two ways:

- by dot notation

```
var property1 = myAirplane.model;
```

- in array notation

```
var property1 = myAirplane["model"];
```

```
var obj = new Object();
```

```
obj["prop"] = 42;
```

=> obj.prop  
= 42

```
obj["0"] = "hello";
```

=> obj[0]  
= hello

- If you try to access a property that does not exist, you get undefined

# Object Creation-Using a constructor function

- JavaScript constructors are special methods that create and initialize the properties for newly created objects
  - When the constructor is called, this is a reference to the newly created object
- ```
var kenscar = new Car('Nissan', '300ZX', 1992);
```
- A constructor is useful when you want to **create multiple similar objects** with the same properties and methods. It's a convention to **capitalize the name** of constructors to distinguish them from regular functions.
  - For example, suppose you want to create an object type for cars. You want this type of object to be called Car, and you want it to have properties for make, model, and year. To do this, you would write the following function:

```
function Car(make, model, year) {  
    this.make = make;  
    this.model = model;  
    this.year = year; }
```

## Object Creation-Using a constructor function

- Notice the use of `this` to assign values to the object's properties based on the values passed to the function.
- In JavaScript, the thing called `this` is the object that "owns" the code. The value of `this`, when used in an object, is the object itself.
- In a constructor function `this` does not have a value. It is a substitute for the **new object**. The value of `this` will become the new object when a new object is created.
- Now you can create an object called mycar as follows:

```
var kenscar = new Car('Nissan', '300ZX', 1992);  
var vpgscar = new Car('Mazda', 'Miata', 1990);
```

- Properties can be accessed :
  - `kenscar.make`
  - `Kenscar["make"]`

# Object Creation-Using object initializers

- An object initializer is a comma-delimited list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}).
- The following example creates myHonda with three properties. Note that the engine property is also an object with its own properties.

```
var myHonda = {color: 'red', wheels: 4, engine: {cylinders: 4, size: 2.2}};
```

- Properties can be accessed :
  - myHonda.color
  - myHonda["color"]

# JavaScript Object Methods

```
var person = {  
    firstName: "John",  
    lastName : "Doe",  
    id      : 5566,  
    fullName : function() {  
        fullName // This refers to the object  
        return this.firstName + " " + this.lastName;  
    }  
};
```

- In a function definition, **this** refers to the "owner" of the function.
- In the example above, **this** is the person object that "owns" the **fullName** function.
- In other words, **this.firstName** means the **firstName** property of this object.

# JavaScript Object Methods

- JavaScript methods are actions that can be performed on objects.
- A JavaScript method is a property containing a function definition.

| Property  | Value                                                                  |
|-----------|------------------------------------------------------------------------|
| firstName | John                                                                   |
| lastName  | Doe                                                                    |
| age       | 50                                                                     |
| eyeColor  | blue                                                                   |
| fullName  | <code>function() {return this.firstName + " " + this.lastName;}</code> |

- You access an object method with the following syntax:

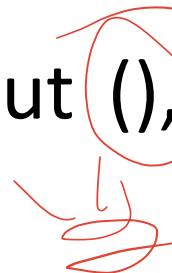
```
objectName.methodName()
```

- You will typically describe `fullName()` as a method of the `person` object, and `fullName` as a property.
- The `fullName` property will execute (as a function) when it is invoked with `()`.
- This example accesses the `fullName()` method of a `person` object:

```
name = person.fullName(); //John Doe
```

# JavaScript Object Methods

- If you access the fullName property, without `()`, it will return the function definition:



```
function () { return this.firstName + " " + this.lastName; }
```

# Deleting Properties

- The **delete** keyword deletes a property from an object:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

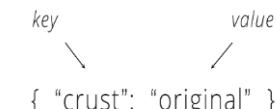
```
delete person.age; // or delete person["age"];
```

- The **delete** keyword deletes both the value of the property and the property itself.
- After deletion, the property cannot be used before it is added back again.
- The **delete** operator is designed to be used on object properties. It has no effect on variables or functions.
- The **delete** operator should not be used on predefined JavaScript object properties. It can crash your application.

# Using JSON to Represent Objects

- **JSON (JavaScript Object Notation)**
  - A simple way to represent JavaScript objects as strings
  - introduced as an alternative to XML as a data-exchange technique
- JSON has gained acclaim due to its simple format, making objects easy to read, create and parse.
- Each JSON object is represented as a list of property names and values contained in curly braces, in the following format:  
*{ propertyName1 : value1, propertyName2 : value2 }*
- Arrays are represented in JSON with square brackets in the following format:  
*[ value0, value1, value2 ]*
- Each value can be a string, a number, a JSON object, true, false or null.

```
{ "firstName": "John", "lastName": "Smith",
  "address": { "streetAddress": "21 Street", "city": "New York", "postal": 10021 }, "phoneNumbers": [ "212 555-1234",
  "646 555-4567" ]
}
```



# Using JSON to Represent Objects

- To appreciate the simplicity of JSON data, examine this representation of an array of address-book entries

```
[ {'first': 'Cheryl', 'last': 'Black' },  
  {'first': 'James', 'last': 'Blue' },  
  {'first': 'Mike', 'last': 'Brown' },  
  {'first': 'Meg', 'last': 'Gold' } ]
```

- JSON provides a straightforward way to manipulate objects in JavaScript, and many other programming languages now support this format.
- In addition to simplifying object creation, JSON allows programs to easily extract data and efficiently transmit it across the Internet.

# JSON Example – Object From String

- Create a JavaScript string containing JSON syntax:

```
var txt = '{ "employees" : [ '+  
    '{ "firstName":"John" , "lastName":"Doe" } , '+  
    '{ "firstName":"Anna" , "lastName":"Smith" } , '+  
    '{ "firstName":"Peter" , "lastName":"Jones" } ] }';
```

John Smith to  
be added

Can add  
to end

to end

| Explanation                                    | Examples                                                                                                                                                                           |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object with one member                         | { "course-name": "6.470" }                                                                                                                                                         |
| object with two members (separated by a comma) | { "course-name": "6.470", "units": 6 }                                                                                                                                             |
| array with 3 elements                          | ["foo", "bar", 12345]                                                                                                                                                              |
| object with an array as a value                | { "course-name": "6.470", "times": ["12-4","12-4", "12-4"] }                                                                                                                       |
| objects with objects and arrays                | { "John": { "label": "John", "data": [[1880,0.081536],[1881,0.08098],[1882,0.078308]] }, "James": { "label": "James", "data": [[1880,0.050053],[1881,0.05025],[1882,0.048278]] } } |

# Converting a JSON Text to a JavaScript Object

- Since JSON syntax is a subset of JavaScript syntax, the JSON.parse can be used to convert a JSON text into a JavaScript object.
- var obj = JSON.parse(txt);
- **Complete code:**

```
<script type="text/javascript">

var txt = '{"employees":[' +
  '{"firstName":"John","lastName":"Doe"},' +
  '{"firstName":"Anna","lastName":"Smith"},' +
  '{"firstName":"Peter","lastName":"Jones"}]';

var obj = JSON.parse(txt);

for (var i=0; i < obj.employees.length; i++)
{
  document.write(obj.employees[i].firstName + " " + obj.employees[i].lastName + "<br/>");
}

</script>
```

3 Val output

# JSON.stringify Function

- Converts a JavaScript value to a JavaScript Object Notation (JSON) string.
  - `JSON.stringify(value [, replacer] [, space])`

choose value to JSON String

➤ Parameters :-

➤ value

- Required. A JavaScript value, usually an object or array, to be converted.

➤ replacer

- Optional. A function or array that transforms the results.

• **Example:**

- This example uses `JSON.stringify` to convert the contact object to JSON text. The `memberfilter` array is defined so that only the `surname` and `phone` members are converted. The `firstname` member is omitted.

# JSON.stringify Function

- Example 1:

```
<script type="text/javascript">

var contact = new Object();
contact.firstname = "Jesper";
contact.surname = "Aaberg";
contact.phone = ["555-0100", "555-0120"];

var jsonText = JSON.stringify(contact);
document.write(jsonText);

</script>
```

*Y'all merge var your code*

// Output:  
// { "firstname": "Jesper", "surname": "Aaberg", "phone": [ "555-0100", "555-0120" ] }

# JSON.stringify Function

- Example 2:

```
<script>
    var contact = new Object();
    contact.firstname = "Jesper";
    contact.surname = "Aaberg";
    contact.phone = ["555-0100", "555-0120"];

    var memberfilter = new Array();
    memberfilter[0] = "surname";
    memberfilter[1] = "phone";
    var jsonText = JSON.stringify(contact, memberfilter, "\t");
    document.write(jsonText);

// Output:
// { "surname": "Aaberg", "phone": [ "555-0100", "555-0120" ] }

</script>
```

# UCCD2323 Front-End Web Development



## Chapter 4 Reading and Writing Data Locally.

*bel no db*

## Cookies

*Here Info*

- Before HTML5, websites could store only small amounts of text-based information on a user's computer using cookies.
- A **cookie** is a *key/value pair* in which each *key* has a corresponding *value*. Eg: `username = John Doe`
- The key and value are both strings.
- Cookies are stored by the browser on the user's computer to maintain client-specific information during and between browser sessions.
- A website might use a cookie to record user preferences or other information that it can retrieve during the client's subsequent visits.
- When a user visits a website, the browser locates any cookies written by that website and sends them to the server.
- *Cookies may be accessed only by the web server and scripts of the website from which the cookies originated*

*WA*

# Cookies

- Any of the following cookie attribute values can optionally follow the key-value pair, each preceded by a semicolon separator:
  - ;**expires=date-in-GMTString-format** – The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
  - ;**domain=domain** (e.g., 'example.com' or 'subdomain.example.com') – The domain name of your site.
  - ;**path=path** – The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.

# Cookies

- **;secure** – If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- **;samesite** - Prevents the browser from sending this cookie along with cross-site requests. Possible values are lax, strict or none.
  - The lax value will send the cookie for all same-site requests and top-level navigation GET requests. This is sufficient for user tracking, but it will prevent many Cross-Site Request Forgery (CSRF) attacks. This is the default value in modern browsers.
  - The strict value will prevent the cookie from being sent by the browser to the target site in all cross-site browsing contexts, even when following a regular link.
  - The none value explicitly states no restrictions will be applied. The cookie will be sent in all requests—both cross-site and same-site.

# Storing Cookies

- Assign a string value to the document.cookie object
  - Example:

```
document.cookie = "name=" + cookievalue1 + "SameSite=Strict" + ";expires=" + now.toUTCString();  
document.cookie = "age=" + cookievalue2 + "SameSite=Strict" + ";expires=" + now.toUTCString();
```

- Note:
  - Use the JavaScript encodeURIComponent() function to **encodes** special characters. In addition, it also encodes the following characters: , / ? : @ & = + \$ # before storing it in the cookie.
  - Use the corresponding decodeURIComponent() function when you read the cookie value.

del cookie +  
After you quit  
before here

# CookiesDemo.html

```
1 <!DOCTYPE html>
2 <html lang="en-us">
3
4 <head>
5   <meta charset="utf-8">
6   <script type="text/javascript">
7     function WriteCookie() {
8       if (document.myform.customer.value == "") {
9         alert("Enter some value!");
10        return;
11      }
12
13      var now = new Date();
14      now.setMinutes(now.getMinutes() + 1);
15
16      cookievalue1 = encodeURIComponent(document.myform.customer.value) + ";";
17      cookievalue2 = encodeURIComponent(document.myform.age.value) + ";";
18
19      //document.cookie = "name=" + cookievalue1 + "SameSite=Strict";
20      //document.cookie = "age=" + cookievalue2 + "SameSite=Strict";
21
22      document.cookie = "name=" + cookievalue1 + "SameSite=Strict" + ";expires=" + now.toUTCString();
23      document.cookie = "age=" + cookievalue2 + "SameSite=Strict" + ";expires=" + now.toUTCString();
24
25      document.write("Setting Cookies : " + "name=" + cookievalue1 + " age=" + cookievalue2);
26    }
27  </script>
28 </head>
29
30 <body>
31   <form name="myform" action="">
32     Enter name: <input type="text" name="customer" /> <br>
33     Age: <input type="number" name="age" />
34     <input type="button" value="Set Cookie" onclick="WriteCookie();"/>
35
36   </form>
37 </body>
38
39 </html>
```

## Output:

The screenshot shows two instances of a browser developer tools window, specifically the Application tab under Storage. In the top instance, there are no visible cookies. In the bottom instance, after entering 'asdf ghjk' into the 'Enter name:' field and '35' into the 'Age:' field, and clicking 'Set Cookie', a new cookie entry appears in the list:

| Name        | Value                                                      | Domain          | Path | Expires / ... | Size | HttpOnly | Secure | SameSite |
|-------------|------------------------------------------------------------|-----------------|------|---------------|------|----------|--------|----------|
| ARRAffinity | 34b30d867c2e7fc11cd7fec89c8eb26887f9e0566869daec081f213... | .uccd2223-we... | /    | Session       | 75   | ✓        |        |          |

<https://uccd2223-week10-demo.azurewebsites.net/CookiesDemo.html>

F12 →  
Application

SS/LS  
nothing

# CookiesDemo.html

## Output:

A screenshot of a browser developer tools window. The title bar shows the URL: uccd2223-week10-demo.azurewebsites.net/CookiesDemo.html. The main content area displays the message: "Setting Cookies : name=asdf%20ghjk; age=35;". Below this, the Application tab is selected in the toolbar. The Storage section shows a table of cookies. One cookie is listed: ARRAffinity, with the value 34b30d867c2e7fc11cd7fec89c8eb26887f9e0566869daec081f213... . The table includes columns for Name, Value, Domain, Path, Expires / ..., Size, HttpOnly, Secure, and SameSite.

- Press Ctrl-F5

A screenshot of a browser developer tools window, similar to the previous one but after a page refresh (Ctrl-F5). The title bar shows the URL: uccd2223-week10-demo.azurewebsites.net/CookiesDemo.html. The main content area displays the message: "Enter name: [ ] Age: [ ] Set Cookie". Below this, the Application tab is selected in the toolbar. The Storage section shows a table of cookies. Two cookies are listed: ARRAffinity (value 34b30d867c2e7fc11cd7fec89c8eb26887f9e0566869daec081f213...) and name (value asdf%20ghjk). The table includes columns for Name, Value, Domain, Path, Expires / ..., Size, HttpOnly, Secure, and SameSite. Two large blue arrows point from the text input fields in the browser to the corresponding cookie entries in the developer tools table.

# Reading Cookies

- Use strings' split() function to break a string into key and values.

## GetCookiesDemo.html

```
1 <!DOCTYPE html>
2 <html lang="en-us">
3
4 <head>
5   <meta charset="utf-8">
6   <script type="text/javascript">
7     function ReadCookie() {
8       var allcookies = document.cookie;
9       document.write("All Cookies : " + allcookies);
10
11     // Get all the cookies pairs in an array
12     cookiearray = allcookies.split(';");
13
14     // Now take key value pair out of this array
15     for (var i = 0; i < cookiearray.length; i++) {
16       name = cookiearray[i].split('=')[0];
17       value = decodeURIComponent(cookiearray[i].split('=')[1]);
18       document.write("<br /> Key is : " + name + " and Value is : " + value);
19     }
20   }
21 </script>
22 </head>
23
24 <body>
25   <form name="myform" action="">
26     <p> click the following button and see the result:</p>
27     <input type="button" value="Get Cookie" onclick="ReadCookie()" />
28   </form>
29 </body>
30
31 </html>
```

## Output:

click the following button and see the result:

Get Cookie

All Cookies : age=12; name=sun%20teik%20heng%20!%40%23%24%25%5E%26\*()

Key is : age and Value is : 12

Key is : name and Value is : sun teik heng !@#\$%^&\*()

<https://uccd2223-week10-demo.azurewebsites.net/GetCookiesDemo.html>

# Setting Cookies Expiry Date

- Extend the life of a cookie beyond the current browser session by setting an expiration date and saving the expiry date within the cookie.

```
var now = new Date();
now.setMinutes( now.getMinutes() + 1 );

cookievalue1 = escape(document.myform.customer.value) + ";";
cookievalue2 = escape(document.myform.age.value) + ";";
document.cookie = "name=" + cookievalue1 +
"SameSite=Strict";
document.cookie = "name=" + cookievalue1 + "SameSite=Strict;
" + "expires=" + now.toUTCString();
document.cookie = "age=" + cookievalue2 + "SameSite=Strict";
document.write ("Setting Cookies : " + "name=" +
cookievalue1 + " age=" + cookievalue2);
```

## Storing Cookies: Create Expiration Date

- To set an expiration date for a cookie:

## ➤ 24 hours:

```
let date = new Date();
date.setTime(date.getTime() + (24 * 60 * 60 * 1000)); // forward
let expires = "expires=" + date.toUTCString();
```

➤ 7 days:

```
let date = new Date();
date.setTime(date.getTime() + (7 * 24 * 60 * 60 * 1000));
let expires = "expires=" + date.toUTCString();
```

## ➤ 30 days:

```
let date = new Date();
date.setTime(date.getTime() + (30 * 24 * 60 * 60 * 1000));
let expires = "expires=" + date.toUTCString();
```

# Storing Cookies: Create Expiration Date Output:

The screenshot shows a browser window with a form for entering name and age, and a 'Set Cookie' button. Below the browser is the Chrome DevTools Application tab, which displays a table of cookies. A blue arrow points to the 'Expires' column for the 'name' cookie, which is highlighted with a red circle.

| Name        | Value                                                       | Domain          | Path | Expires                  | Max-Age | Size | H.. | Secure | SameSite |
|-------------|-------------------------------------------------------------|-----------------|------|--------------------------|---------|------|-----|--------|----------|
| ARRAffinity | 34b30d867c2e7fcb11cd7fec89c8eb26887f9e0566869daec081f213... | .uccd2223-we... | /    | Session                  | 75      | ✓    |     |        |          |
| age         | 25                                                          | uccd2223-we...  | /    | Session                  | 5       |      |     | Strict |          |
| name        | asdf%20ghjk                                                 | uccd2223-we...  | /    | 2020-01-29T07:07:43.000Z | 15      |      |     | Strict |          |

Select a cookie to preview its value

## Example: Set a cookie for 24 hours

- The function sets a cookie by adding together the cookie name, the cookie value, and the expires string.

```
function setCookie(name, value, days) {  
    let date = new Date();  
    date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));  
    let expires = "expires=" + date.toUTCString();  
    document.cookie = name + "=" + value + ";" + expires + ";path=/";  
}
```

- The parameters of the function above are the name of the cookie (name), the value of the cookie (value), and the number of days until the cookie should expire (days).
- Example: Set a cookie named “username” for 24 hours with user input

```
let username = document.getElementById('usernameInput').value;  
setCookie("username", username, 1);
```

- In this example, 1 is the number of days until the cookie expires. This means the cookie will be valid for 24 hours.

# Function to Check a Cookie

- To create the function that checks if a cookie is set.
- If the cookie is set it will display a greeting.
- If the cookie is not set, it will display a prompt box, asking for the name of the user, and stores the username cookie for 365 days, by calling the setCookie function:

```
function checkCookie() {  
    let username = getCookie("username");  
    if (username != "") {  
        alert("Welcome again " + username);  
    } else {  
        username = prompt("Please enter your name:", "");  
        if (username != "" && username != null) {  
            setCookie("username", username, 365); expired  
        }  
    }  
}
```

## Function to Delete a Cookie

- The function deleteCookie takes the name of the cookie as a parameter (name).
- It sets the cookie's value to an empty string and its expiration date to a past date, effectively deleting the cookie.

```
function deleteCookie(name) {  
    document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";  
}  
  
deleteCookie("username");
```

# Advantages of using cookies

- Cookies are simple to use and implement.
- Occupies less memory, do not require any server resources and are stored on the user's computer so no extra burden on server.
- We can configure cookies to expire when the browser session ends (session cookies) or they can exist for a specified length of time on the client's computer (persistent cookies).

# Problems with Cookies

- They're extremely limited in size (limited to about 4 KB of data).
- Cookies cannot store entire documents.
- Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
- User has the option of disabling cookies on his computer from browser's setting.
- Users can delete a cookie.
- Complex type of data not allowed (e.g. dataset etc). It allows only plain text (i.e. cookie allows only string content)

# localStorage and sessionStorage

- ***Introducing localStorage and sessionStorage***
  - ▶ As of HTML5, there are two new mechanisms for storing key/value pairs that help eliminate some of the problems with cookies.
    - Web applications can use the window object's **localStorage** property to **store up to several megabytes** of key/value-pair string data on the user's computer and can access that data across browsing sessions and browser tabs. Stores data with **no expiration date** *permanently*
    - Web applications that need access to data for **only** a browsing session and that must keep that data separate among multiple tabs can use the window object's **sessionStorage** property. There's a separate sessionStorage object for every browsing session, including separate tabs that are accessing the same website. Stores data for one session (**data is lost when the tab is closed**) *+ temporary*

# localStorage and sessionStorage

- Check browser support

```
if (typeof(Storage) != "undefined") {
```

```
}
```

- setItem

➤ Set a key/value pair.

- getItem

➤ Retrieves the current value associated with the key

- removeItem

➤ Delete a key/value pair from the storage collection

## Example: localStorage

- localStorage
- The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
// Store
localStorage.setItem("lastname", "Smith");

// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

- The example above:
  - Create a localStorage name/value pair with name="lastname" and value="Smith"
  - Retrieve the value of "lastname" and insert it into the element with id="result"

flexible

## Example: sessionStorage

- sessionStorage
- The sessionStorage object stores data for one session. The data will be deleted when the browser or tab is closed but will be available as long as the browser or tab is open.

```
// Store
sessionStorage.setItem("lastname", "Smith");

// Retrieve
document.getElementById("result").innerHTML = sessionStorage.getItem("lastname");
```

- The example above:
- Creates a sessionStorage name/value pair with name="lastname" and value="Smith".
- Retrieves the value of "lastname" and inserts it into the element with id="result".

# setItem and getItem

```
1 <!DOCTYPE html>
2 <html lang="en-us">
3
4 <head>
5   <title>Local Storage and Session Storage Demo</title>
6   <meta charset="utf-8">
7   <script src="https://code.jquery.com/jquery-3.4.1.min.js"
8     integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbyMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>
9 </head>
10
11 <body>
12   <div id="result"></div>
13   <script>
14     if (typeof (storage) != "undefined") /* Check browser support* {
15
16       // Store
17       localStorage.setItem("lastname", "Smith");
18       sessionStorage.setItem("firstname", "John");
19
20       let newSpan1 = document.createElement("span");
21       let newSpan2 = document.createElement("span");
22       let $lineBreak = $("<br />");
23
24       // Retrieve
25       newSpan1.append("Local Storage " + localStorage.getItem("lastname"));
26       newSpan2.append("Session Storage " + sessionStorage.getItem("firstname"));
27
28       $("#result").append(newSpan1, $lineBreak, newSpan2);
29
30     } else {
31       $("#result").html("Sorry, your browser does not support Web Storage...");}
32   </script>
33 </body>
34 </html>
```

## Output:

The screenshot shows a browser window displaying the output of the provided JavaScript code. The page content is:

```
Local Storage Smith
Session Storage John
```

Below the browser window is the Microsoft Edge developer tools interface, specifically the Application tab. The storage data is listed as follows:

| Key       | Value |
|-----------|-------|
| firstname | John  |

Under the Storage section, there are two entries:

- Local Storage (https://uccd2223-week1): lastname (Value: Smith)
- Session Storage (https://uccd2223-week1): firstname (Value: John)

<https://uccd2223-week10-demo.azurewebsites.net/LocalStorageAndSessionStorageDemo.html>

# GetFromWebStorage

A screenshot of a browser window showing two tabs: "Local Storage and Session Storage" and "Local Storage and Session Storage". The URL in the address bar is "uccd2223-week10-demo.azurewebsites.net/GetFromWebStorage.html". The page content displays "LocalStorage Smith" and "SessionStorage null". Handwritten red annotations include "proxy" above the table and "cause" with an arrow pointing to the "Value" column. The browser's developer tools Application tab is selected, showing the Local Storage table:

| Key      | Value |
|----------|-------|
| lastname | Smith |

The sidebar shows "Manifest", "Service Workers", and "Storage" sections, with "LocalStorage" expanded to show entries for "https://uccd2223-week1" and "https://uccd2223-week1" under "lastname". A red arrow points from the "cause" annotation to the "Value" column of the "lastname" row.

Open <https://uccd2223-week10-demo.azurewebsites.net/GetFromWebStorage.html> in new tab

# removeItem

```
1 <!DOCTYPE html>
2 <html lang="en-us">
3
4 <head>
5   <title>Local Storage and Session Storage Demo</title>
6   <meta charset="utf-8">
7   <script src="https://code.jquery.com/jquery-3.4.1.min.js"
8     |   integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>
9 </head>
10
11 <body>
12
13   <div id="result"></div>
14
15   <script>
16
17     // Check browser support
18     if (typeof (Storage) != "undefined") {
19
20       //Remove
21       localStorage.removeItem("lastname");
22       sessionStorage.removeItem("firstname");
23
24       let newSpan1 = document.createElement("span");
25       let newSpan2 = document.createElement("span");
26       let $lineBreak = $("<br />");
27
28       // Retrieve
29       newSpan1.append("Local Storage " + localStorage.getItem("lastname"));
30       newSpan2.append("Session Storage " + sessionStorage.getItem("firstname"));
31
32       $("#result").append(newSpan1, $lineBreak, newSpan2);
33
34     } else {
35       $("#result").html("Sorry, your browser does not support Web Storage...");
36     }
37
38   </script>
39 </body>
40
41 </html>
```

<https://uccd2223-week10-demo.azurewebsites.net/RemoveItemFromSessionStorageAndLocalStorage.html>

## Output:

The screenshot shows a browser window with the URL [uccd2223-week10-demo.azurewebsites.net/RemoveItemFromSessionStorageAndLocalStorage.html](https://uccd2223-week10-demo.azurewebsites.net/RemoveItemFromSessionStorageAndLocalStorage.html). The page content displays "Local Storage null" and "Session Storage null". Below the page content, the browser's developer tools are open, specifically the Application tab. The Application tab shows the storage hierarchy: Manifest, Service Workers, and Clear storage under Application; and Local Storage and Session Storage under Storage. Under Local Storage, there is an entry for the URL <https://uccd2223-week10-demo.azurewebsites.net/>. A tooltip "Select a value to preview" is visible near the bottom right of the storage list.

# localStorage and sessionStorage

|               | Session Storage                                                                                                                                                                                                     | Local Storage                                                                                                                                                                      |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Storage       | Stored on the user's browser for a specific session.                                                                                                                                                                | Stored on the user's browser and persists across sessions                                                                                                                          |
| Expiration    | Cleared at the end of a session.<br><br>~~~~~                                                                                                                                                                       | Persists until manually cleared.<br><br>~~~~~                                                                                                                                      |
| Accessibility | Accessible only within the same browsing session.<br><br>~~~~~                                                                                                                                                      | Accessible only within the same <u>browsing session</u> .                                                                                                                          |
| Use Cases     | <ul style="list-style-type: none"><li>Temporary data storage during navigation within the website.</li><li>Multi-step form submission</li><li>Storing data required for a specific user session.</li></ul><br>~~~~~ | <ul style="list-style-type: none"><li>Caching data to improve performance.</li><li>Storing user preferences.</li><li>Building offline-capable web applications.</li></ul><br>~~~~~ |

# Summary: localStorage and sessionStorage

- Pros:
  - Persistent Data: localStorage allows data to persist across sessions.
  - Temporary Data: sessionStorage keeps data for the duration of the session.
  - Faster Access: Reduces server requests, improving load times.
  - User Personalization: Stores user preferences and state.
    - Eg: Stores preferences, settings, login info, and shopping cart items.
- Cons:
  - Size Limitations: Both local and session storages are limited to about 5-10 MB.
  - Security Concerns: Data is accessible via JavaScript, which can lead to security risks.
    - Eg: Sensitive information can be exposed, unauthorized access
  - Data Loss: sessionStorage data is lost when the session ends.
  - Compatibility: Not all browsers support these features consistently.

# UCCD2323 Front-End Web Development



## Chapter 5 Responsive Web.

# Introduction to Bootstrap

- Is a front-end framework.
- For creating common user interface components like forms, buttons, navigations, dropdowns, alerts, modals, tabs, accordions, carousels, tooltips, and so on.
- Provides **flexible and responsive web layouts.**
- Reference: <https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/>

# Things you can do with Bootstrap

- Create responsive websites.
- Create multi-column layout with pre-defined classes.
- Create different types of form layouts.
- Create different variation of navigation bar.
- Create components like accordions, modals, etc. without writing any JS code.
- Create dynamic tabs to manage large amount of content.
- Create tooltips and popovers to show hint text.
- Create carousel or image slider to showcase your content.
- Create different types of alert boxes.

# Advantages of Using Bootstrap

- Save design time
- Responsive features
- Consistent design
- Easy to use
- Compatible with browsers (Google Chrome, Firefox, Safari, Internet Explorer 10 and above, etc.)
- Open Source  
(Bootstrap 5 is the latest and most stable version of the Bootstrap.)

# Getting Started with Bootstrap

- Include the Bootstrap CSS and JS files via CDN.
- Bootstrap requires a third-party library Popper.js for some of its components like popovers and tooltips.
  - `bootstrap.bundle.min.js` includes Popper.

- CDN:

| Description | URL                                                                                                                                                                     |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CSS         | <a href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css</a>           |
| JS          | <a href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js">https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js</a> |

# Creating your first web page with Bootstrap

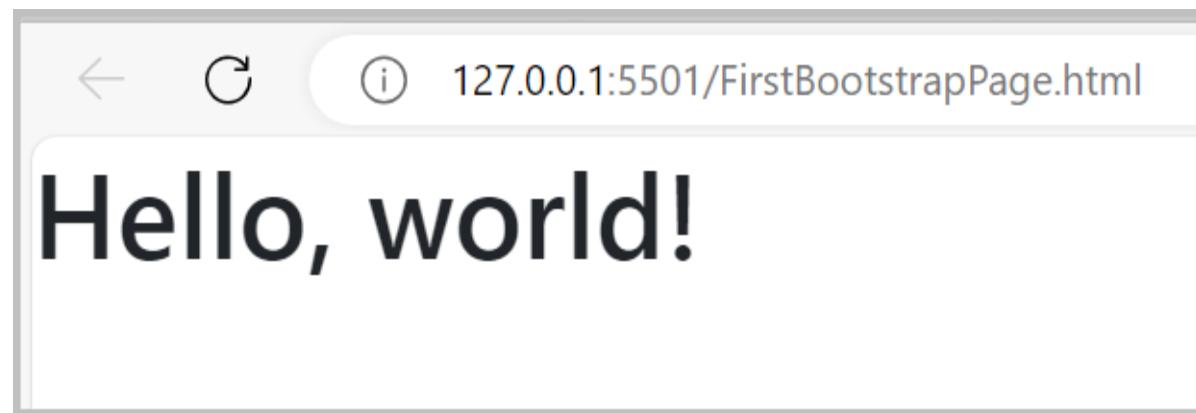
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Basic Bootstrap Template</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6Coli6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">
</head>

<body>
  <h1>Hello, world!</h1>
  <!-- Bootstrap JS Bundle with Popper -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>
</body>

</html>
```

## Output in Web Browser



# Bootstrap Containers

- Containers are used to wrap content with some padding and to align the content horizontally center on the page for fixed width layout.

- Three different types containers:**

**.container (max-width at each responsive breakpoint)**

**.container-fluid (100% width at all breakpoints)**

**.container-{breakpoint} (100% width until the specified breakpoint)**

Classes	X-Small	Small	Medium	Large	X-Large	XX-Large
Bootstrap Grid System	<576px	≥576px	≥768px	≥992px	≥1200px	≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

# Bootstrap Containers

- **Responsive Fixed-width Containers**

```
<div class="container">
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
</div>
```

- **Fluid Containers**

```
<div class="container-fluid">
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
</div>
```

# Bootstrap Containers

- Responsive Breakpoints for Containers
- <div class="container-sm">100% wide until screen size less than 576px</div>
- <div class="container-md">100% wide until screen size less than 768px</div>
- <div class="container-lg">100% wide until screen size less than 992px</div>
- <div class="container-xl">100% wide until screen size less than 1200px</div>
- Use the Bootstrap background-color and border utility classes to add background and borders to containers.

```
<!-- Container with dark background and white text color -->
<div class="container bg-dark text-white">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>

<!-- Container with light background -->
<div class="container bg-light">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>

<!-- Container with border -->
<div class="container border">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>
```

# Bootstrap Containers

- Containers have padding of 12px on the left and right sides, and no padding on the top and bottom sides by default.
- Use the [spacing utility classes](#) to apply extra padding and margins.
- Spacing utility classes, [Bootstrap 5 Helper Classes - Tutorial Republic](https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-helper-classes.php),  
<https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-helper-classes.php>
- Note:
- Avoid setting left and right margin on fixed and responsive containers.
- Bootstrap set the value of margin-left and margin-right properties to auto by default.

```
<!-- Container with border, extra paddings and margins -->
<div class="container border py-3 my-3">
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
</div>
```

# Bootstrap Grid System

- It is fully responsive and uses twelve column system (12 columns available per row) and six default responsive tiers.
- Use the Bootstrap's predefined grid classes to make the layouts for different types of devices like mobile phones, tablets, laptops, desktops, and so on.
- For example, use the .col-\* classes to create grid columns for extra small devices like mobile phones in portrait mode, and the .col-sm-\* classes for mobile phones in landscape mode.
- Use the .col-md-\* classes to create grid columns for medium screen devices like tablets, the .col-lg-\* classes for devices like small laptops, the .col-xl-\* classes for laptops and desktops, and the .col-xxl-\* classes for large desktop screens.

# Key features of the Bootstrap's grid system

Features	X-Small (xs)	Small (sm)	Medium (md)	Large (lg)	X-Large (xl)	XX-Large (xxl)
Bootstrap Grid System	<576px	≥576px	≥768px	≥992px	≥1200px	≥1400px
Container max-width	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-	.col-xxl-
Number of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
Nestable	Yes					
Column ordering	Yes					

# Create rows and columns using the 12 column responsive grid system

- Step 1
  - Create a container that acts as a wrapper for your rows and columns using any container classes such as .container.
- Step 2
  - Create rows inside the container using the .row class.
- Step 3
  - Create columns inside any row you can use the .col-\* , .col-sm-\* , .col-md-\* , .col-lg-\* , .col-xl-\* and .col-xxl-\* classes.

# Creating Two Column Layouts

- Two column layouts for medium, large and extra large devices like tables, laptops and desktops etc.
- The sum of the grid column numbers within a single row should not be greater than 12.
- i.e. col-md-\* add up to twelve (6+6, 4+8 and 3+9) for every row

```
<div class="container">
    <!--Row with two equal columns-->
    <div class="row">
        <div class="col-md-6">Column left</div>
        <div class="col-md-6">Column right</div>
    </div>

    <!--Row with two columns divided in 1:2 ratio-->
    <div class="row">
        <div class="col-md-4">Column left</div>
        <div class="col-md-8">Column right</div>
    </div>

    <!--Row with two columns divided in 1:3 ratio-->
    <div class="row">
        <div class="col-md-3">Column left</div>
        <div class="col-md-9">Column right</div>
    </div>
</div>
```

# Creating Three Column Layouts

```
<div class="container">
    <!--Row with three equal columns-->
    <div class="row">
        <div class="col-lg-4">Column left</div>
        <div class="col-lg-4">Column middle</div>
        <div class="col-lg-4">Column right</div>
    </div>

    <!--Row with three columns divided in 1:4:1 ratio-->
    <div class="row">
        <div class="col-lg-2">Column left</div>
        <div class="col-lg-8">Column middle</div>
        <div class="col-lg-2">Column right</div>
    </div>

    <!--Row with three columns divided unevenly-->
    <div class="row">
        <div class="col-lg-3">Column left</div>
        <div class="col-lg-7">Column middle</div>
        <div class="col-lg-2">Column right</div>
    </div>
</div>
```

# Bootstrap Auto-Layout Columns

- Create equal width columns for all devices (x-small, small, medium, large, x-large, and xx-large) by using the class .col, without specifying any column number.

```
<div class="container">  
    <!--Row with two equal columns-->  
    <div class="row">  
        <div class="col">Column one</div>  
        <div class="col">Column two</div>  
    </div>  
  
    <!--Row with three equal columns-->  
    <div class="row">  
        <div class="col">Column one</div>  
        <div class="col">Column two</div>  
        <div class="col">Column three</div>  
    </div>  
</div>
```

# Bootstrap Auto-Layout Columns

- set the width of one column and let the sibling columns automatically resize around it equally.

```
<div class="container">  
    <!--Row with two equal columns-->  
    <div class="row">  
        <div class="col">Column one</div>  
        <div class="col">Column two</div>  
    </div>  
  
    <!--Row with three columns divided in 1:2:1 ratio-->  
    <div class="row">  
        <div class="col">Column one</div>  
        <div class="col-sm-6">Column two</div>  
        <div class="col">Column three</div>  
    </div>  
</div>
```

# Column Wrapping Behavior

- Create a three column layout on large devices like laptops and desktops, as well as on tablets (e.g. Apple iPad) in landscape mode, but on medium devices like tablets in portrait mode ( $768\text{px} \leq \text{screen width} < 992\text{px}$ ), it will change into a two column layout.
- The sum of the medium grid column numbers (i.e. col-md-\*) is  $3 + 9 + 12 = 24 > 12$ , therefore the third `<div>` element with the class .col-md-12 that is adding the extra columns beyond the maximum 12 columns in a .row, gets wrapped onto a new line as one contiguous unit on the medium screen size devices.

```
<div class="container">
```

```
<div class="row">
```

```
<div class="col-md-4 col-lg-3">Column one</div>
```

```
<div class="col-md-8 col-lg-6">Column two</div>
```

```
<div class="col-md-12 col-lg-3">Column three</div>
```

```
</div>
```

```
</div>
```

# Bootstrap 5 Grid Examples

- [Bootstrap 5 Grid Examples - Tutorial Republic](#),
- <https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-grid-examples.php>

## Nesting of Grid Columns

- You can put rows and columns inside an existing column.

```
<div class="container">
  <div class="row">
    <div class="col-sm-8">Column left</div>
    <div class="col-sm-4">
      <!--Column right with nested rows and columns--&gt;
      &lt;div class="row"&gt;
        &lt;div class="col-12"&gt;&lt;/div&gt;
      &lt;/div&gt;
      &lt;div class="row"&gt;
        &lt;div class="col-6"&gt;&lt;/div&gt;
        &lt;div class="col-6"&gt;&lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  &lt;/div&gt;
&lt;/div&gt;</pre>
```

# Creating Variable Width Columns

- Use the col-{breakpoint}-auto classes to size columns based on the natural width of their content.

```
<div class="container">  
  <div class="row justify-content-md-center">  
    <div class="col-md-3">Column left</div>  
    <div class="col-md-auto">Variable width column</div>  
    <div class="col-md-3">Column right</div>  
  </div>  
  <div class="row">  
    <div class="col">Column left</div>  
    <div class="col-auto">Variable width column</div>  
    <div class="col">Column right</div>  
  </div>  
</div>
```

# Alignment of Grid Columns

- Use the flexbox alignment utilities to vertically and horizontally align grid columns inside a container.
- Use the classes .align-items-start, .align-items-center, and .align-items-end to align the grid columns vertically at the top, middle and bottom of a container.
- Use the classes .justify-content-start, .justify-content-center, and .justify-content-end to align the grid columns horizontally at the left, center and right of a container, respectively.

# Vertical Alignment of Grid Columns

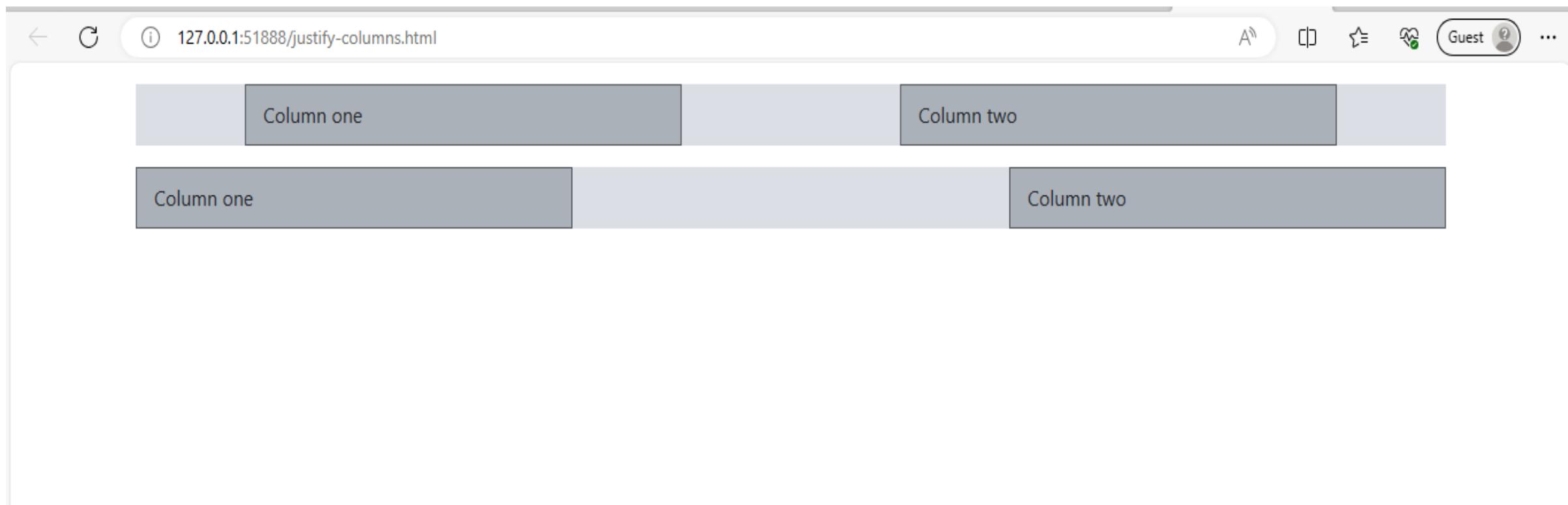
```
<div class="container">
  <div class="row align-items-start">
    <div class="col">Column one</div>
    <div class="col">Column two</div>
    <div class="col">Column three</div>
  </div>
  <div class="row align-items-center">
    <div class="col">Column one</div>
    <div class="col">Column two</div>
    <div class="col">Column three</div>
  </div>
  <div class="row align-items-end">
    <div class="col">Column one</div>
    <div class="col">Column two</div>
    <div class="col">Column three</div>
  </div>
</div>
```

# Horizontal Alignment of Grid Columns

```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">Column one</div>
    <div class="col-4">Column two</div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">Column one</div>
    <div class="col-4">Column two</div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">Column one</div>
    <div class="col-4">Column two</div>
  </div>
</div>
```

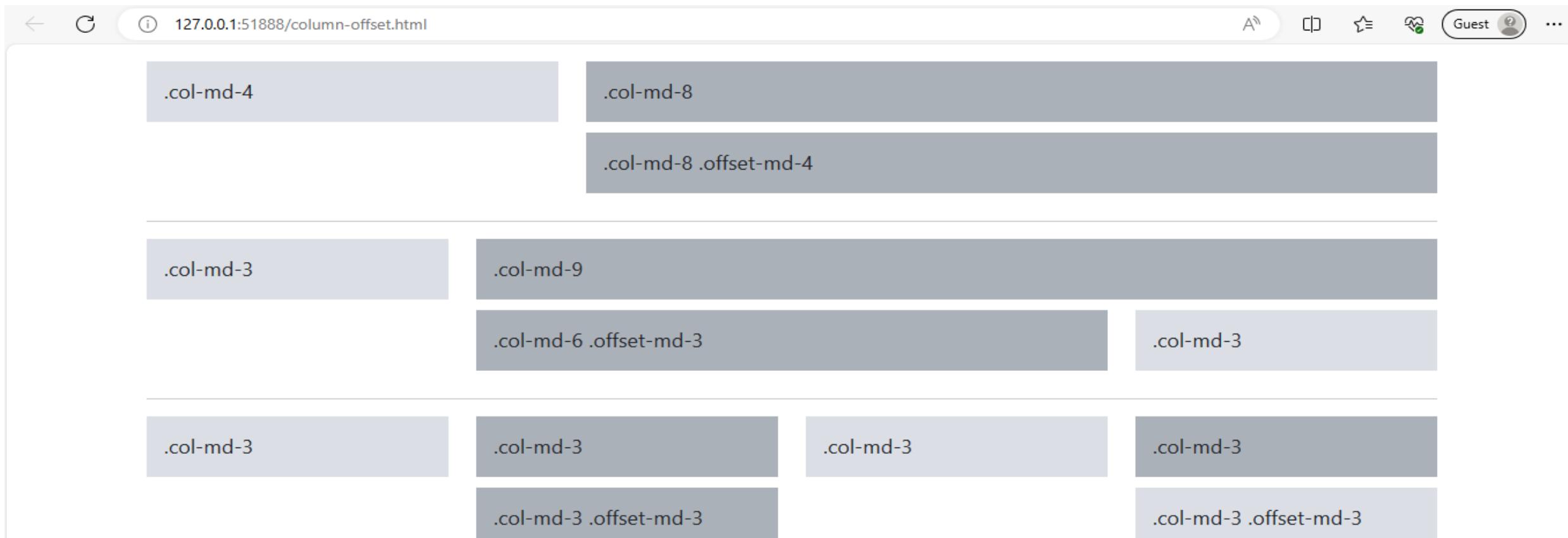
# Justify columns

- Use the class `.justify-content-around` to distribute grid columns evenly with half-size spaces on either end, whereas you can use the class `.justify-content-between` to distribute the grid columns evenly where the first column placed at the start and the last column placed at the end.



# Offsetting the Grid Columns

- Move grid columns to the right for alignment purpose using the column offset classes like .offset-sm-\*, .offset-md-\*, .offset-lg-\*, and so on.



# Creating Compact Columns

- Remove the default gutters between columns to create compact layouts by adding the class .g-0 on .row.

The screenshot shows a web browser window with the URL 127.0.0.1:51888/compact-columns.html. The page displays two sections: "Columns with Gutters" and "Columns without Gutters". Both sections contain three grey rectangular boxes labeled ".col-4". In the "Columns with Gutters" section, there are thin white spaces (gutters) between the boxes. In the "Columns without Gutters" section, the boxes are placed directly next to each other with no gaps. A cursor arrow is visible at the bottom center of the page.

127.0.0.1:51888/compact-columns.html

Guest

Columns with Gutters

.col-4 .col-4 .col-4

Columns without Gutters

.col-4 .col-4 .col-4

# Breaking Columns to a New Line

The screenshot shows a web browser window with the URL `127.0.0.1:51888/break-columns-to-a-new-line.html`. The page content is as follows:

**Breaking columns on all devices**

.col	.col
.col	.col

**Breaking columns on all devices except extra large devices**

.col	.col	.col	.col
------	------	------	------

**Note:** Resize the browser window to understand how it works.

# Bootstrap Layout

- Bootstrap Fixed Layout
  - The process of creating the fixed yet responsive layout basically starts with the .container class. After that you can create rows with the .row class to wrap the horizontal groups of columns. Rows must be placed within a .container for proper alignment and padding.
- Bootstrap Fluid Layout
  - Use the class .container-fluid to create fluid layouts to utilize the 100% width of the viewport across all devices (extra small, small, medium, large, extra large, and extra-extra large). The class .container-fluid simply applies the width: 100% instead of different width for different viewport sizes. However, the layout will still responsive and you can use the grid classes as usual.
- Bootstrap Responsive Layout
  - Responsive layouts automatically adjust and adapts to any device screen size, whether it is a desktop, a laptop, a tablet, or a mobile phone.

# Bootstrap Typography

- Bootstrap uses the browser's default root font-size (typically 16px) so visitors can customize their browser defaults as needed.
- The font-size is defined in 'rem' and 1rem is equal to 16px.

## Headings

- You can define all HTML headings, `<h1>` through `<h6>` — In the same way you define in simple HTML document. You can also utilize the heading classes `.h1` through `.h6` on other elements, if you want to apply the style on element's text same as headings.

```
<p class="h1">h1. Bootstrap heading</p>
<p class="h2">h2. Bootstrap heading</p>
<p class="h3">h3. Bootstrap heading</p>
<p class="h4">h4. Bootstrap heading</p>
<p class="h5">h5. Bootstrap heading</p>
<p class="h6">h6. Bootstrap heading</p>
```

# Customizing Headings

- E.g. Use the <small> tag with .text-muted class to display the secondary text of any heading in a smaller and lighter variation.

```
<h2>
  Fancy display heading
  <small class="text-muted">With faded secondary text</small>
</h2>
```

Fancy display heading With faded secondary text

# Display Headings

- Can be used when you need a heading to stand out. Display headings are displayed in larger font-size but lighter font-weight.

```
<h1 class="display-1">Display Heading 1</h1>
<h1 class="display-2">Display Heading 2</h1>
<h1 class="display-3">Display Heading 3</h1>
<h1 class="display-4">Display Heading 4</h1>
<h1 class="display-5">Display Heading 5</h1>
<h1 class="display-6">Display Heading 6</h1>
```

Display Heading 1  
Display Heading 2  
Display Heading 3  
Display Heading 4  
Display Heading 5  
Display Heading 6

# Working with Paragraphs

- Bootstrap's global default font-size is 1rem (typically 16px), with a line-height of 1.5 (typically 24px), which is applied to the <body> element as well as all the paragraphs i.e. the <p> elements. In addition to that margin-bottom of 1rem is also applied to all the paragraphs.
- You can also make a paragraph stand out by adding the class .lead on it.

```
<p>This is how a normal paragraph looks like in Bootstrap.</p>
<p class="lead">This is how a paragraph stands out in Bootstrap.</p>
```

This is how a normal paragraph looks like in Bootstrap.

This is how a paragraph stands out in Bootstrap.

# Text Alignment

- Align text to left, right, and center using the text alignment classes.

```
<p class="text-start">Left aligned text on all viewport sizes.</p>
<p class="text-center">Center aligned text on all viewport sizes.</p>
<p class="text-end">Right aligned text on all viewport sizes.</p>
```

Left aligned text on all viewport sizes.

Center aligned text on all viewport sizes.

Right aligned text on all viewport sizes.

- Align text based on screen size using the responsive text alignment classes.

```
<p class="text-sm-center">Text will be center aligned on small sized (sm) viewports and up.</p>
<p class="text-md-center">Text will be center aligned on medium sized (md) viewports and up.</p>
<p class="text-lg-center">Text will be center aligned on large sized (lg) viewports and up.</p>
<p class="text-xl-center">Text will be center aligned on extra-large sized (xl) viewports and up.</p>
```

# Text Formatting

- Use text formatting tags like `<strong>`, `<i>`, `<small>` to make your text bold, italic, small and so on, in the same way you do in simple HTML page.

```
<p><b>This is bold text</b></p>
<p><code>This is computer code</code></p>
<p><em>This is emphasized text</em></p>
<p><i>This is italic text</i></p>
<p><mark>This is highlighted text</mark></p>
<p><small>This is small text</small></p>
<p><strong>This is strongly emphasized text</strong></p>
<p>This is <sub>subscript</sub> and <sup>superscript</sup></p>
<p><ins>This text is inserted to the document</ins></p>
<p><del>This text is deleted from the document</del></p>
```

This is bold text  
This is computer code  
This is emphasized text  
This is italic text  
This is highlighted text  
This is small text  
This is strongly emphasized text  
This is subscript and superscript  
This text is inserted to the document  
This text is deleted from the document

## Text Transformation

- Transform the text to lowercase, uppercase or make them capitalize.

```
<p class="text-lowercase">The quick brown fox jumps over the lazy dog.</p>
```

```
<p class="text-uppercase">The quick brown fox jumps over the lazy dog.</p>
```

```
<p class="text-capitalize">The quick brown fox jumps over the lazy dog.</p>
```

the quick brown fox jumps over the lazy dog.

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

The Quick Brown Fox Jumps Over The Lazy Dog.

# Text Coloring

```
<p class="text-primary">Primary: Please read the instructions carefully  
before proceeding.</p>  
  
<p class="text-secondary">Secondary: This is featured has been removed from  
the latest version.</p>  
  
<p class="text-success">Success: Your message has been sent successfully.</p>  
  
<p class="text-info">Info: You must agree with the terms and conditions to  
complete the sign up process.</p>  
  
<p class="text-warning">Warning: There was a problem with your network  
connection.</p>  
  
<p class="text-danger">Danger: An error has been occurred while submitting  
your data.</p>  
  
<p class="text-muted">Muted: This paragraph of text is grayed out.</p>
```

Primary: Please read the instructions carefully before proceeding.

Secondary: This is featured has been removed from the latest version.

Success: Your message has been sent successfully.

Info: You must agree with the terms and conditions to complete the sign up process.

Warning: There was a problem with your network connection.

Danger: An error has been occurred while submitting your data.

Muted: This paragraph of text is grayed out.

## Styling Blockquotes

- To give a pretty look to your blockquotes — Just define the blockquotes using the standard `<blockquote>` element and bootstrap's style sheet will do the rest.

```
<blockquote class="blockquote">  
  <p>Imagination is more important than knowledge.</p>  
</blockquote>
```

Imagination is more important than knowledge.

- Can also align blockquotes to the right or center by applying the text alignment classes `.text-end` or `.text-center` on the `<blockquote>` or `<figure>` element.

# Truncating Long Text

- Use the class .text-truncate to truncate the text with an ellipsis. The display property value of the element must be inline-block or block.

```
<!-- Block level element -->
<div class="row">
    <div class="col-2 text-truncate">
        The quick brown fox jumps over the lazy dog.
    </div>
</div>

<!-- Inline level element -->
<span class="d-inline-block text-truncate" style="max-width: 100px;">
    The quick brown fox jumps over the lazy dog.
</span>
```

## Text wrapping and Overflow

- Use the class .text-wrap to wrap the text within an element by overwriting its white-space property if it is set to pre or nowrap, such as Bootstrap badge components.
- Use the class .text nowrap to prevent text from wrapping within an element.

```
<div class="badge bg-primary text-wrap" style="width: 6rem;">  
    This text will wrap.  
</div>  
  
<div class="bg-warning text nowrap" style="width: 6rem;">  
    This text will overflow the element's box.  
</div>
```

## Wrapping Long Word

- Use the class .text-break to prevent long word from breaking your layout.

```
<div class="row">  
  <div class="col-2">  
    <p class="text-break">veryveryveryveryveryverylongword</p>  
  </div>  
</div>
```

# Bootstrap Tables

- Create table with basic styling with horizontal dividers and small cell padding (8px by default) by adding the Bootstrap's class .table to the <table> element.

```
<table class="table">
  <thead>
    <tr>
      <th>#</th>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Clark</td>
      <td>Kent</td>
      <td>clarkkent@mail.com</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Peter</td>
      <td>Parker</td>
      <td>peterparker@mail.com</td>
    </tr>
    <tr>
      <td>3</td>
      <td>John</td>
      <td>Carter</td>
      <td>johncarter@mail.com</td>
    </tr>
  </tbody>
</table>
```

#	First Name	Last Name	Email
1	Clark	Kent	clarkkent@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Carter	johncarter@mail.com

# Creating Accented Tables

- Apply the contextual classes such as .table-primary, .table-secondary, .table-success, .table-danger, .table-warning, .table-info, .table-light and .table-dark to color tables, table rows or individual cells.

#	First Name	Last Name	Email
1	Clark	Kent	clarkkent@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Carter	johncarter@mail.com

```
<table class="table table-dark">
  <thead>
    <tr>
      <th>#</th>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Clark</td>
      <td>Kent</td>
      <td>clarkkent@mail.com</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Peter</td>
      <td>Parker</td>
      <td>peterparker@mail.com</td>
    </tr>
    <tr>
      <td>3</td>
      <td>John</td>
      <td>Carter</td>
      <td>johncarter@mail.com</td>
    </tr>
  </tbody>
</table>
```

# Tables with Striped Rows

- Add zebra-striping to the table rows within the <tbody> by adding an additional class .table-striped to the .table base class.

#	First Name	Last Name	Email
1	Clark	Kent	clarkkent@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Carter	johncarter@mail.com

```
<table class="table table-striped">
  <thead>
    <tr>
      <th>#</th>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Clark</td>
      <td>Kent</td>
      <td>clarkkent@mail.com</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Peter</td>
      <td>Parker</td>
      <td>peterparker@mail.com</td>
    </tr>
    <tr>
      <td>3</td>
      <td>John</td>
      <td>Carter</td>
      <td>johncarter@mail.com</td>
    </tr>
  </tbody>
</table>
```

## More Examples:

- Create borderless tables by using the class .table-borderless on the .table element.
- Enable a hover state on table rows within a <tbody> element by adding the modifier class .table-hover to the .table base class.
- Make the tables more compact and save the space through adding the modifier class .table-sm to the .table base class. All the cell padding are cut in half.
- Use the modifier classes .table-light or .table-dark on the <thead> element to make it appear in light or dark gray.

# Creating Responsive Tables with Bootstrap

- To make any table responsive just place it inside a <div> element and apply the .table-responsive class on it.
- You can also specify when the table should have a scrollbar, based on the viewport width (i.e. breakpoints), using the classes .table-responsive{-sm |-md |-lg |-xl}.

## Example

```
1 <div class="table-responsive">
2   <table class="table">
3     <thead>
4       <tr>
5         <th>#</th>
6         <th>First Name</th>
7         <th>Last Name</th>
8         <th>Email</th>
9         <th>Biography</th>
10      </tr>
11    </thead>
12    <tbody>
13      <tr>
```

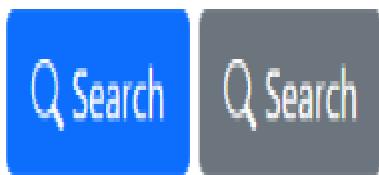
# Using Icons in Bootstrap 5

- Over 1,300 high quality icons, which are available in SVGs, SVG sprite, or web fonts format.
- Create icons of any color just through applying the CSS color property.
- Use the CSS font-size property to change the size of icons
- Use the CDN link to include Bootstrap icons in a web page.
- To use Bootstrap icons in your code use the *<i>* tag with an individual icon class `.bi-*` applied on it.
- [Bootstrap Icons Class List](https://tutorialrepublic.com/bootstrap-icons-class-list)  
[\(tutorialrepublic.com\)](https://tutorialrepublic.com)

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <title>Including Bootstrap Icons in HTML</title>
7   <!-- Bootstrap CSS -->
8   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css">
9   <!-- Bootstrap Font Icon CSS -->
10  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">
11 </head>
12 <body>
13   <h1><i class="bi-globe"></i> Hello, world!</h1>
14
15   <!-- Bootstrap JS Bundle with Popper -->
16   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></script>
17 </body>
18 </html>
```

## Placing a search icon inside a button

```
<button type="submit" class="btn btn-primary"><span class="bi-search"></span>  
Search</button>  
  
<button type="submit" class="btn btn-secondary"><span class="bi-search"></span>  
Search</button>
```



# Font-Awesome

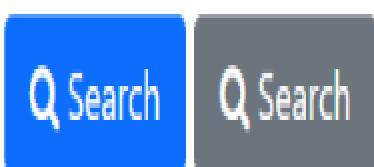
- Can also use external icon libraries in Bootstrap such as Font Awesome.
- Use an *<i>* tag along with a base class .fa and an individual icon class .fa-\*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Including Font Awesome Icons in Bootstrap</title>
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css">
  <!-- Font Awesome CSS -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
  <h1><i class="fa fa-globe"></i> Hello, world!</h1>

  <!-- Bootstrap JS Bundle with Popper -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

# Font-Awesome

```
<button type="submit" class="btn btn-primary"><span class="fa fa-search">  
/</span> Search</button>  
  
<button type="submit" class="btn btn-secondary"><span class="fa fa-search">  
/</span> Search</button>
```

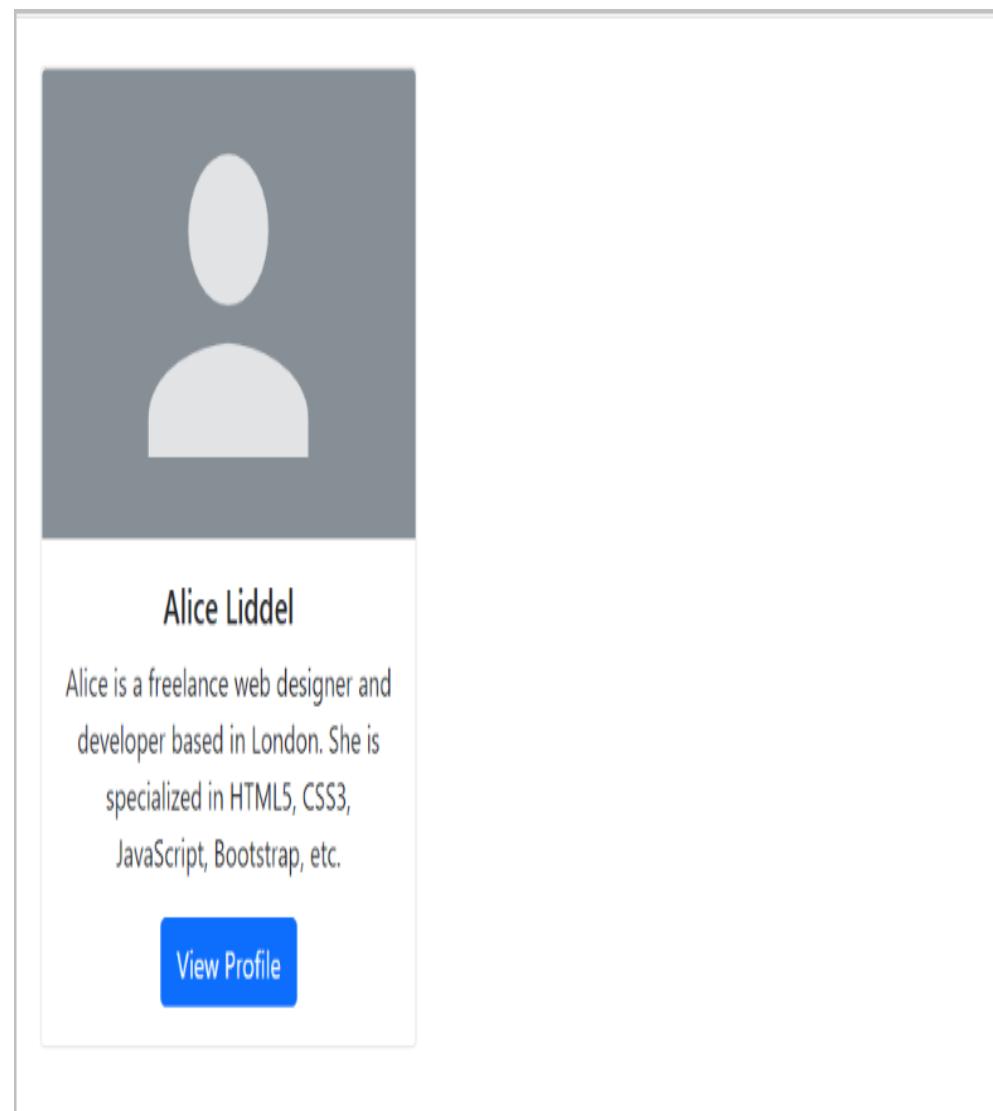


# Bootstrap Cards

- Is a flexible and extensible content container.
- It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options.
- The outer wrapper require the base class .card, whereas content can be placed inside the .card-body element.

# Creating a basic card

```
<div class="card" style="width: 300px;">  
    
  <div class="card-body text-center">  
    <h5 class="card-title">Alice Liddel</h5>  
    <p class="card-text">Alice is a freelance web designer and developer  
based in London. She is specialized in HTML5, CSS3, JavaScript, Bootstrap, etc.  
  </p>  
    <a href="#" class="btn btn-primary">View Profile</a>  
  </div>  
</div>
```



# Content Types for Card Component

- Body Only Card
  - use .card with .card-body within

```
<div class="card">  
  <div class="card-body">This is some text within a padded box.</div>  
</div>
```

This is some text within a padded box.

# Card with Titles, Text, and Links

- Place title and links inside the card along with text.

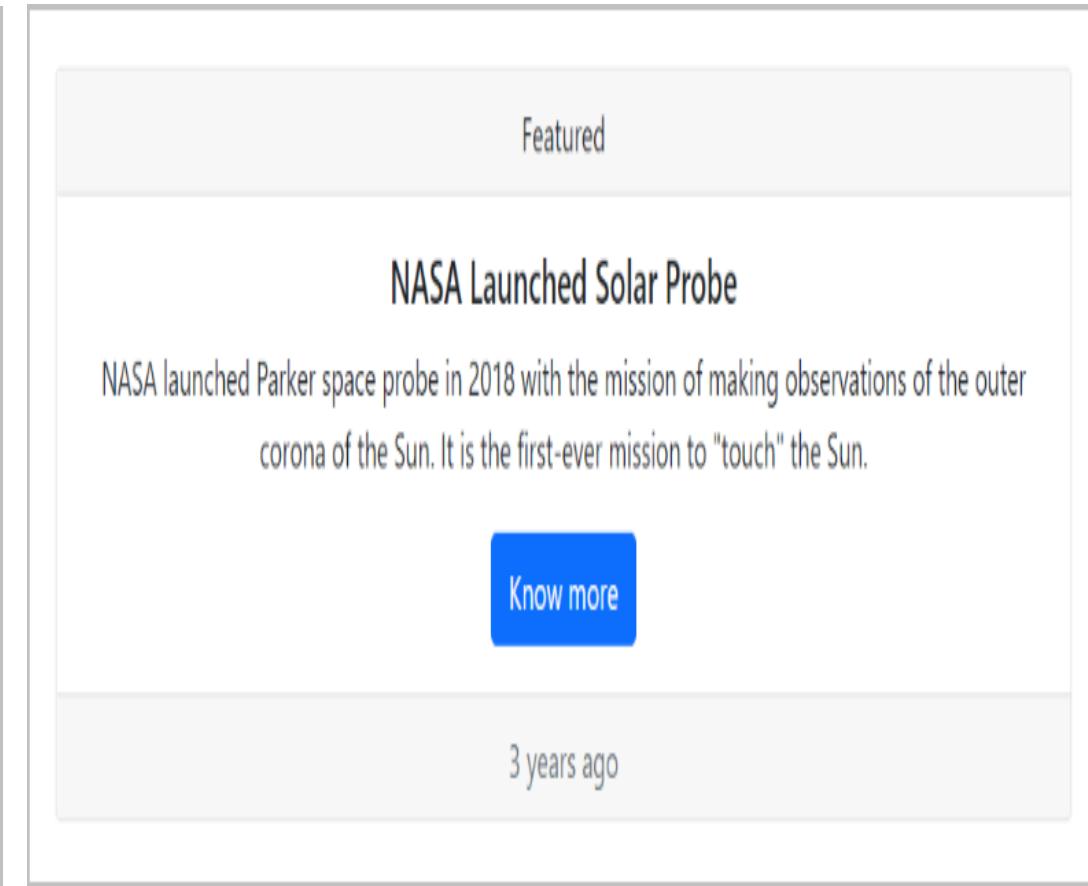
```
<div class="card" style="width: 300px;">  
  <div class="card-body">  
    <h5 class="card-title">Eiffel Tower</h5>  
    <h6 class="card-subtitle mb-3 text-muted">Champ de Mars, Paris,  
    France</h6>  
    <p class="card-text">Built in 1889 Eiffel Tower is one of the most  
    iconic landmarks in the world.</p>  
    <a href="#" class="card-link">View pictures</a>  
    <a href="#" class="card-link">Discover history</a>  
  </div>  
</div>
```

Eiffel Tower  
Champ de Mars, Paris, France  
  
Built in 1889 Eiffel Tower is one of the  
most iconic landmarks in the world.  
  
[View pictures](#) [Discover history](#)

# Card with Header and Footer

- Add header and footer within your cards using the .card-header and .card-footer class.

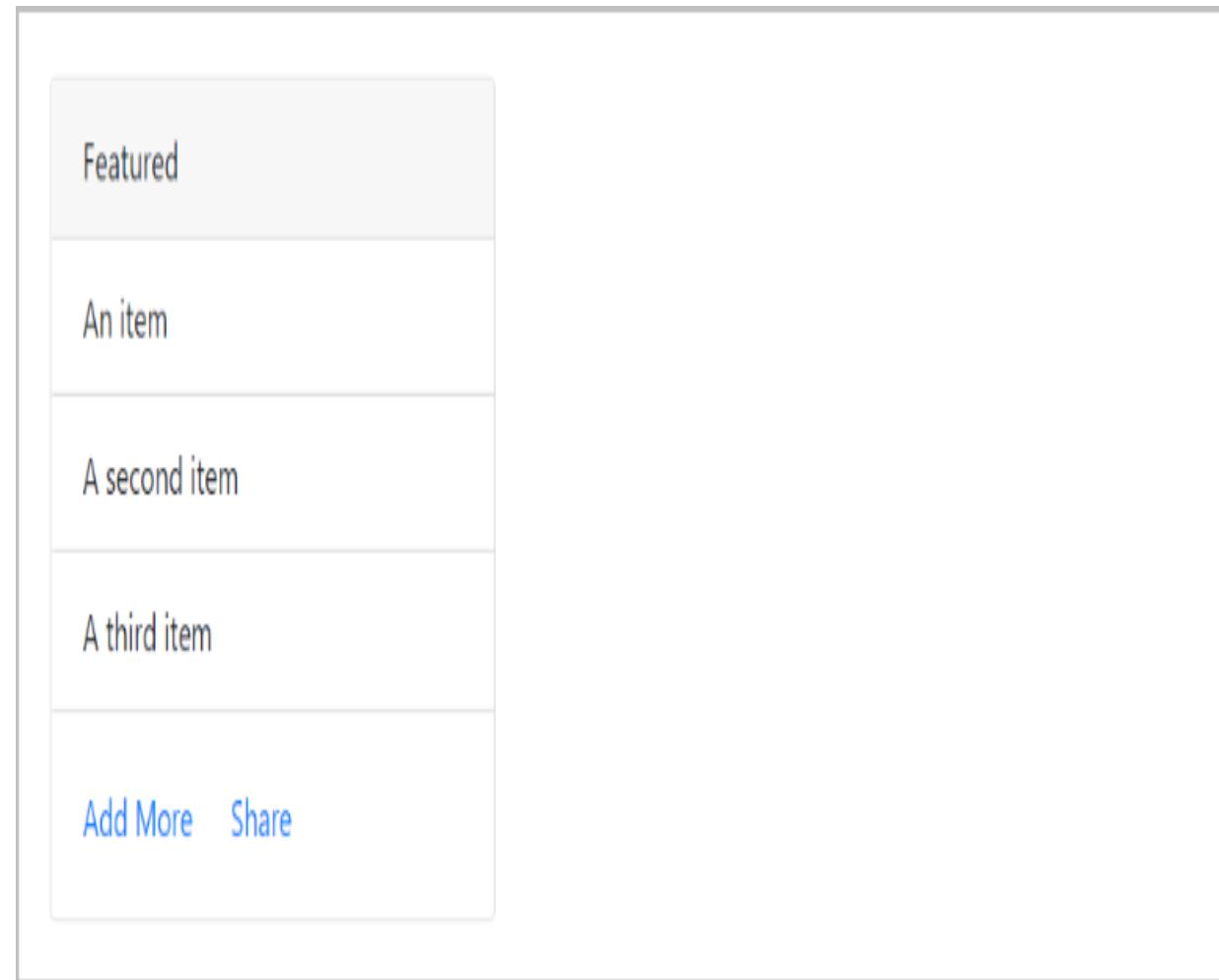
```
<div class="card text-center">  
  <div class="card-header">Featured</div>  
  <div class="card-body">  
    <h5 class="card-title">NASA Launched Solar Probe</h5>  
    <p class="card-text">NASA launched Parker space probe in 2018 with the  
mission of making observations of the outer corona of the Sun. It is the first-  
ever mission to "touch" the Sun.</p>  
    <a href="#" class="btn btn-primary">Know more</a>  
  </div>  
  <div class="card-footer text-muted">3 years ago</div>  
</div>
```



# Placing List Groups within Card

- Can place list groups inside the card along with other content types.

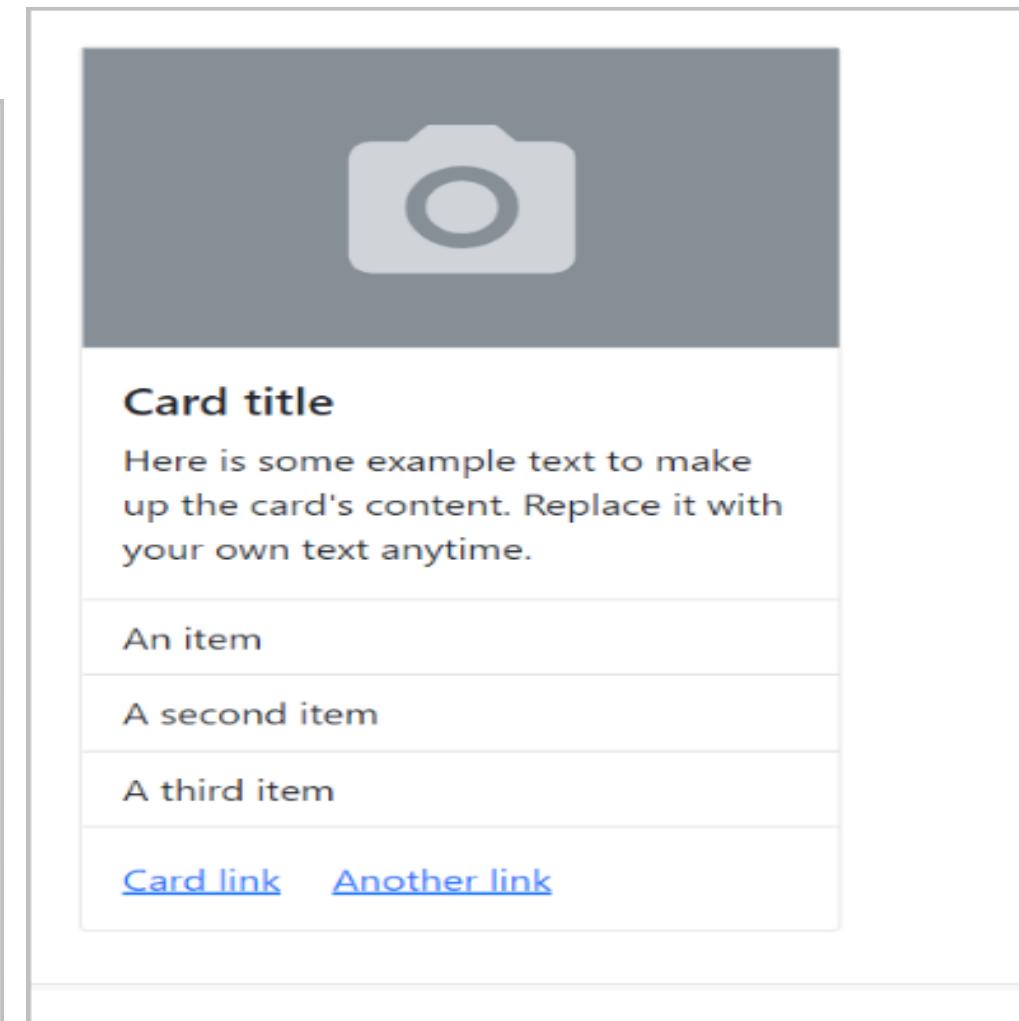
```
<div class="card" style="width: 300px;">
    <div class="card-header">Featured</div>
    <ul class="list-group list-group-flush">
        <li class="list-group-item">An item</li>
        <li class="list-group-item">A second item</li>
        <li class="list-group-item">A third item</li>
    </ul>
    <div class="card-body">
        <a href="#" class="card-link">Add More</a>
        <a href="#" class="card-link">Share</a>
    </div>
</div>
```



# Mix and Match Multiple Content Types within Card

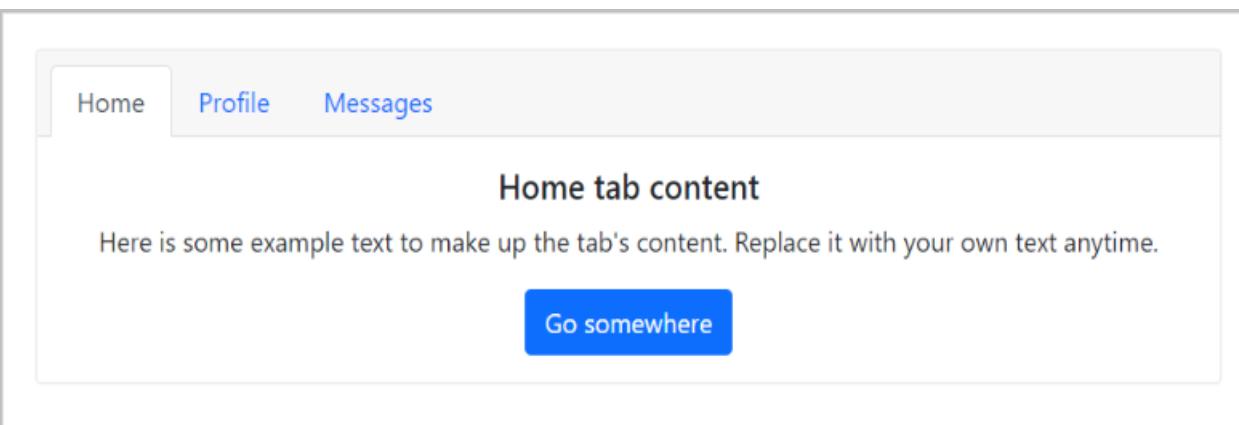
- Example: Create a fixed-width card with an image, text, list group, and hyperlinks.

```
<div class="card" style="width: 300px;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Here is some example text to make up the card's content. Replace it with your own text anytime.</p>
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">An item</li>
    <li class="list-group-item">A second item</li>
    <li class="list-group-item">A third item</li>
  </ul>
  <div class="card-body">
    <a href="#" class="card-link">Card link</a>
    <a href="#" class="card-link">Another link</a>
  </div>
</div>
```



# Adding Navigation to Cards

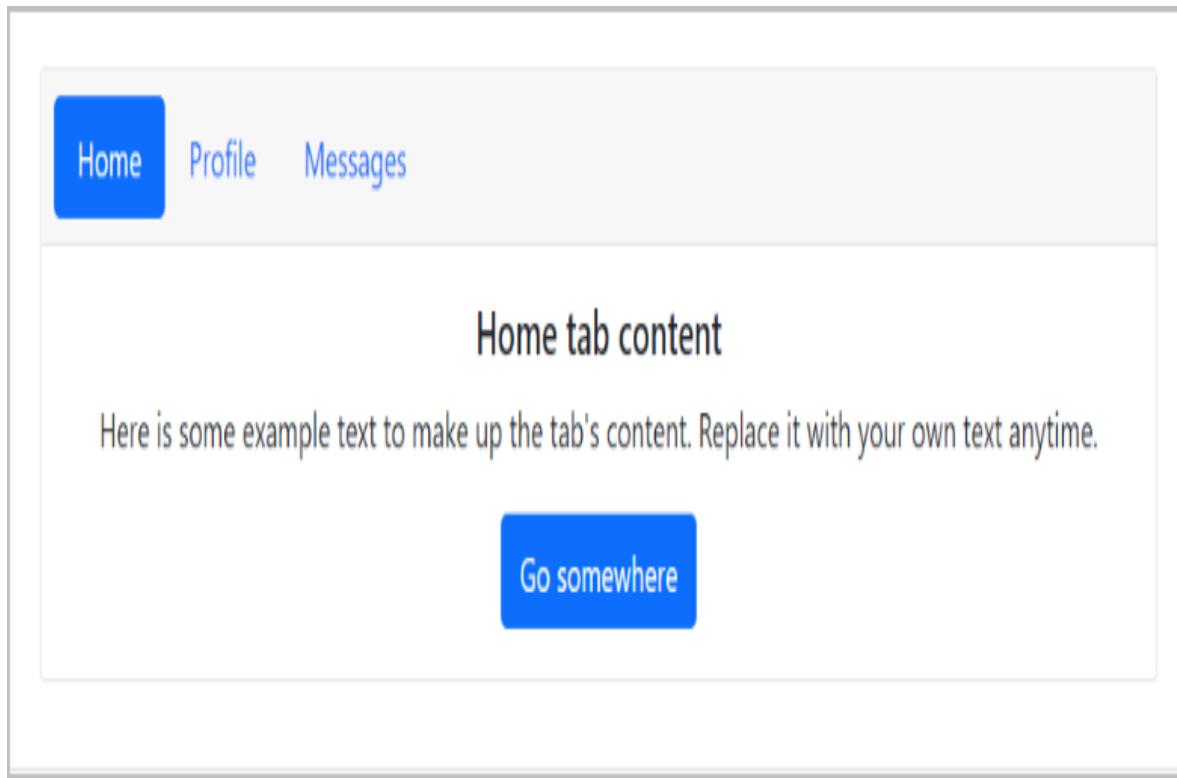
- Add Bootstrap's nav components such as tabs and pills to the card header.
- To add tabs navigation to a card, place the tabs markup inside the card header, and the tabs content inside the card body. Use an additional class `.card-header-tabs` on the `.nav` element along with the class `.nav-tabs` for proper alignment.



```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-tabs card-header-tabs">
      <li class="nav-item">
        <a href="#" class="nav-link active" data-bs-
        toggle="tab">Home</a>
      </li>
      <li class="nav-item">
        <a href="#" class="nav-link" data-bs-
        toggle="tab">Profile</a>
      </li>
      <li class="nav-item">
        <a href="#" class="nav-link" data-bs-
        toggle="tab">Messages</a>
      </li>
    </ul>
  </div>
  <div class="card-body">
    <div class="tab-content">
      <div class="tab-pane fade show active" id="home">
        <h5 class="card-title">Home tab content</h5>
        <p class="card-text">Here is some example text to make up the
        tab's content. Replace it with your own text anytime.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
      <div class="tab-pane fade" id="profile">
        <h5 class="card-title">Profile tab content</h5>
        <p class="card-text">Here is some example text to make up the
        tab's content. Replace it with your own text anytime.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
      <div class="tab-pane fade" id="messages">
        <h5 class="card-title">Messages tab content</h5>
        <p class="card-text">Here is some example text to make up the
        tab's content. Replace it with your own text anytime.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
</div>
```

# Adding Navigation to Cards

- add pills nav to the card by using an additional class .card-header-pills along with the class .nav-pills on the .nav element.



```
<div class="card text-center">
  <div class="card-header">
    <ul class="nav nav-pills card-header-pills">
      <li class="nav-item">
        <a href="#home" class="nav-link active" data-bs-
        toggle="tab">Home</a>
      </li>
      <li class="nav-item">
        <a href="#profile" class="nav-link" data-bs-
        toggle="tab">Profile</a>
      </li>
      <li class="nav-item">
        <a href="#messages" class="nav-link" data-bs-
        toggle="tab">Messages</a>
      </li>
    </ul>
  </div>
  <div class="card-body">
    <div class="tab-content">
      <div class="tab-pane fade show active" id="home">
        <h5 class="card-title">Home tab content</h5>
        <p class="card-text">Here is some example text to make up the
        tab's content. Replace it with your own text anytime.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
      <div class="tab-pane fade" id="profile">
        <h5 class="card-title">Profile tab content</h5>
        <p class="card-text">Here is some example text to make up the
        tab's content. Replace it with your own text anytime.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
      <div class="tab-pane fade" id="messages">
        <h5 class="card-title">Messages tab content</h5>
        <p class="card-text">Here is some example text to make up the
        tab's content. Replace it with your own text anytime.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
    </div>
  </div>
</div>
```

# Customizing the Card Styles

- Use the background and color utility classes to change the appearance of a card.
- Customize the border and text color of any card using the text and border utility classes. Apply these classes on the .card or its child elements.

## Card Layout Options

- Use card groups to render cards as a single, attached element with equal width and height columns.
- Cards inside a card group become horizontally stacked on extra small devices (i.e. viewport width <576px).

# Creating the Card Groups

```
<div class="card-group">
  <div class="card">
    
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">Some dummy text to make up the card's content.
      You can replace it anytime.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 5 mins ago</small>
    </div>
  </div>
  <div class="card">
    
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">Some dummy text to make up the card's content.
      You can replace it anytime.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 5 mins ago</small>
    </div>
  </div>
  <div class="card">
    
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">Some dummy text to make up the card's content.
      You can replace it anytime.</p>
    </div>
    <div class="card-footer">
      <small class="text-muted">Last updated 5 mins ago</small>
    </div>
  </div>
</div>
```

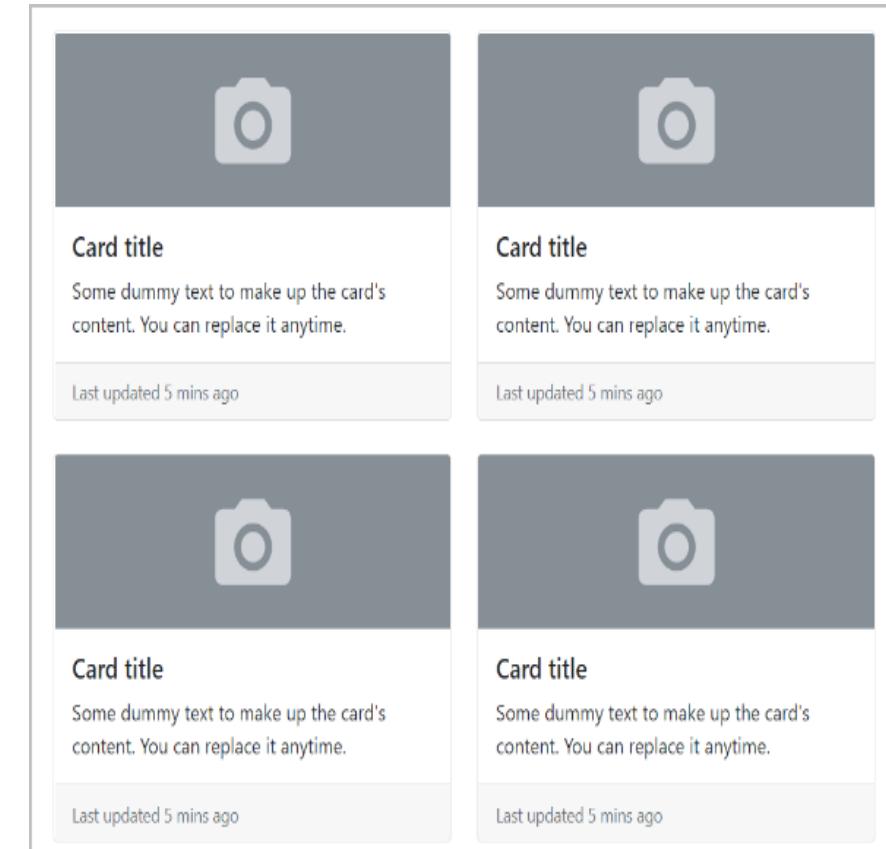
		
Card title  Some dummy text to make up the card's content. You can replace it anytime.	Card title  Some dummy text to make up the card's content. You can replace it anytime.	Card title  Some dummy text to make up the card's content. You can replace it anytime.
Last updated 5 mins ago	Last updated 5 mins ago	Last updated 5 mins ago

## Creating the Card Grids

- Use the Bootstrap grid system and its .row-cols-\* classes to control how many grid columns (wrapped around your cards) to show per row.
- For example, you can use the class .row-cols-1 to show one card per row, similarly you can use the class .row-cols-md-2 to show two cards per row, from the medium breakpoint up (i.e. viewport width  $\geq 768\text{px}$ ).

# Creating the Card Grids

```
<div class="row row-cols-1 row-cols-md-2 g-4">
  <div class="col">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">Some dummy text to make up the card's content. You can replace it anytime.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 5 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">Some dummy text to make up the card's content. You can replace it anytime.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 5 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">Some dummy text to make up the card's content. You can replace it anytime.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 5 mins ago</small>
      </div>
    </div>
  </div>
  <div class="col">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">Some dummy text to make up the card's content. You can replace it anytime.</p>
      </div>
      <div class="card-footer">
        <small class="text-muted">Last updated 5 mins ago</small>
      </div>
    </div>
  </div>
</div>
```



## Bootstrap Carousel

- The carousel also known as slideshow or image slider.
- The Bootstrap carousel generally has three components — carousel indicators (small rectangles), carousel controls (previous and next arrows) and the carousel items or slides.
- It is required to add the class `.active` to one of the carousel slides (i.e. on the `.carousel-item` element), otherwise carousel will not be visible.
- The `.slide` class on the `.carousel` element adds CSS slide transition animation to the carousel that makes the carousel items slide when showing the new item.

# Bootstrap Carousel

```
<div id="myCarousel" class="carousel slide" data-bs-ride="carousel">
    <!-- Carousel indicators -->
    <ol class="carousel-indicators">
        <li data-bs-target="#myCarousel" data-bs-slide-to="0" class="active">
    </li>
        <li data-bs-target="#myCarousel" data-bs-slide-to="1"></li>
        <li data-bs-target="#myCarousel" data-bs-slide-to="2"></li>
    </ol>

    <!-- Wrapper for carousel items -->
    <div class="carousel-inner">
        <div class="carousel-item active">
            
        </div>
        <div class="carousel-item">
            
        </div>
        <div class="carousel-item">
            
        </div>
    </div>

    <!-- Carousel controls -->
    <a class="carousel-control-prev" href="#myCarousel" data-bs-slide="prev">
        <span class="carousel-control-prev-icon"></span>
    </a>
    <a class="carousel-control-next" href="#myCarousel" data-bs-slide="next">
        <span class="carousel-control-next-icon"></span>
    </a>
</div>
```



# Bootstrap Carousel

- Explanation of Code
  - The outermost container of every carousel (i.e. the .carousel element) requires a unique id (in our case id="myCarousel") so that it can be targeted by the carousel indicators (line no-4,5,6) and carousel controls (line no-23,26) to function properly.
  - The data-bs-ride="carousel" attribute of the .carousel element tells the Bootstrap to start animating the carousel immediately when the page load.
  - The data-bs-slide-to attribute (line no-4,5,6) move the slide position to a particular item (index beginning with 0) when clicking on the specific carousel indicator.
  - The slides are specified within the .carousel-inner (line no-10) and the content of each slide is defined within the .carousel-item element that can be text and images.
  - The data-bs-slide attribute on carousel controls (line no-23,26) accepts the keywords prev or next, which alters the slide position relative to its current position.

## Activate Carousels via JavaScript

```
<script>

document.addEventListener("DOMContentLoaded", function(){

    var element = document.getElementById("myCarousel");

    var myCarousel = new bootstrap.Carousel(element);

});

</script>
```

## Bootstrap Forms

- Bootstrap greatly simplifies the process of styling and alignment of form controls like labels, input fields, selectboxes, textareas, buttons, etc. through predefined set of classes.
- Bootstrap provides three different types of form layouts:
  - Vertical Form (default form layout)
  - Horizontal Form
  - Inline Form

# Creating Vertical Form Layout

- Use the predefined margin utility classes for grouping the labels, form controls, optional form text, and form validation messages.

Email  
Email  
Password  
Password  
 Remember me  
**Sign in**

```
<form>
  <div class="mb-3">
    <label class="form-label" for="inputEmail">Email</label>
    <input type="email" class="form-control" id="inputEmail"
placeholder="Email">
  </div>
  <div class="mb-3">
    <label class="form-label" for="inputPassword">Password</label>
    <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
  </div>
  <div class="mb-3">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="checkRemember">
      <label class="form-check-label" for="checkRemember">Remember
me</label>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

## Creating Vertical Form Layout

- All textual form controls, such as <input> and <textarea> requires the class .form-control, while <select> requires the class .form-select for general styling. These classes also makes the forms controls 100% wide. To change their width or use them inline, you can utilize the Bootstrap's predefined grid classes.
- It is recommended to use margin-bottom utility classes (e.g., mb-2, mb-3, etc.) to add vertical spacing between the form groups. Using single direction margin throughout in the form prevent margin collapsing and create more consistent form.

# Creating Horizontal Form Layout

- Create horizontal form layouts where labels and form controls are aligned side-by-side using the Bootstrap grid classes.
- Add the class `.row` on form groups and use the `.col-*-*` grid classes to specify the width of your labels and controls.
- Apply the class `.col-form-label` on the `<label>` elements, so that they're vertically centered with their associated form controls.

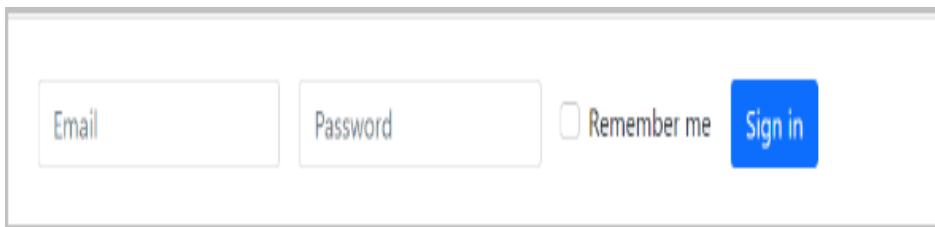
# Creating Horizontal Form Layout

```
<form>
  <div class="row mb-3">
    <label for="inputEmail" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail"
placeholder="Email">
    </div>
  </div>
  <div class="row mb-3">
    <label for="inputPassword" class="col-sm-2 col-form-
label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
    </div>
  </div>
  <div class="row mb-3">
    <div class="col-sm-10 offset-sm-2">
      <div class="form-check">
        <input class="form-check-input" type="checkbox"
id="checkRemember">
        <label class="form-check-label" for="checkRemember">Remember
me</label>
      </div>
    </div>
  </div>
  <div class="row">
    <div class="col-sm-10 offset-sm-2">
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
  </div>
</form>
```

The image shows a user interface for a sign-in form. It features a light gray background with a white rectangular input area. On the left side, there are labels for 'Email' and 'Password'. To the right of each label is a corresponding input field. Below the input fields is a checkbox labeled 'Remember me'. At the bottom right is a large blue button with the text 'Sign in'.

# Creating Inline Form Layout

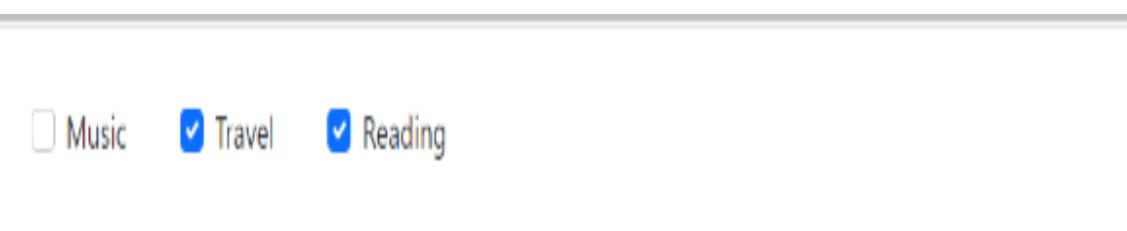
- Display a series of form controls, and buttons in a single horizontal row.
- It is recommended to include a label for every form inputs otherwise screen readers will have trouble with your forms.
- can hide the labels using the .visually-hidden class, so that only screen readers can read it.



```
<form>
  <div class="row align-items-center g-3">
    <div class="col-auto">
      <label class="visually-hidden" for="inputEmail">Email</label>
      <input type="email" class="form-control" id="inputEmail"
placeholder="Email">
    </div>
    <div class="col-auto">
      <label class="visually-hidden" for="inputPassword">Password</label>
      <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
    </div>
    <div class="col-auto">
      <div class="form-check">
        <input class="form-check-input" type="checkbox"
id="checkRemember">
        <label class="form-check-label" for="checkRemember">Remember
me</label>
      </div>
    </div>
    <div class="col-auto">
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
  </div>
</form>
```

# Placing Checkboxes and Radio Buttons Inline

- By default, any number of custom checkboxes and radio buttons that are immediate sibling will be vertically stacked and appropriately spaced with .form-check class.
- To place custom checkboxes and radio buttons inline (i.e., in the same line), add the class .form-check-inline to .form-check element.
- Similarly, you can place the radio buttons inline.



Music    Travel    Reading

```
<div class="row">
  <div class="col-12">
    <div class="form-check form-check-inline">
      <input type="checkbox" class="form-check-input" name="hobbies" id="checkMusic">
      <label class="form-check-label" for="checkMusic">Music</label>
    </div>
    <div class="form-check form-check-inline ms-3">
      <input type="checkbox" class="form-check-input" name="hobbies" id="checkTravel" checked>
      <label class="form-check-label" for="checkTravel">Travel</label>
    </div>
    <div class="form-check form-check-inline ms-3">
      <input type="checkbox" class="form-check-input" name="hobbies" id="checkReading" checked>
      <label class="form-check-label" for="checkReading">Reading</label>
    </div>
  </div>
</div>
```

# Adding Help Text to Form Controls

- Place block level help text for a form control using the class .form-text.
- The block help text is typically displayed at the bottom of the control.

```
<label class="form-label" for="inputPassword">Password</label>
<input type="password" class="form-control" id="inputPassword">
<div class="form-text">
    Must be 8-20 characters long, contain letters, numbers and special
    characters, but must not contain spaces.
</div>
```

Password

Must be 8-20 characters long, contain letters, numbers and special characters, but must not contain spaces.

# Height Sizing of Form Controls

- Can change the height of your text inputs and select boxes to match the button sizes.
- Use the form control height sizing classes such as .form-control-lg and .form-control-sm on the text inputs to create it's larger or smaller sizes.
- Be sure to apply the class .col-form-label-lg or .col-form-label-sm on the <label> or <legend> elements to correctly resize the label according to the form controls.

Email	<input class="form-control form-control-lg" type="text" value="Large input"/>
Email	<input class="form-control" type="text" value="Default input"/>
Email	<input class="form-control form-control-sm" type="text" value="Small input"/>

```
<div class="row mb-3">
    <label class="col-sm-2 col-form-label col-form-label-lg">Email</label>
    <div class="col-sm-10">
        <input type="email" class="form-control form-control-lg"
placeholder="Large input">
    </div>
</div>
<div class="row mb-3">
    <label class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
        <input type="email" class="form-control" placeholder="Default input">
    </div>
</div>
<div class="row">
    <label class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
        <input type="email" class="form-control form-control-sm"
placeholder="Small input">
    </div>
</div>
```

# Bootstrap Form Validation

- It uses browser's native form validation API to validate the form.
- Form validation styles are applied via CSS :invalid and :valid pseudo-classes.
- It applies to <input>, <select>, and <textarea> elements.
- For custom Bootstrap form validation messages, you'll need to disable the browser default feedback tooltips by adding the novalidate boolean attribute to the <form> element.

```
<form class="needs-validation" novalidate>
  <div class="mb-3">
    <label class="form-label" for="inputEmail">Email</label>
    <input type="email" class="form-control" id="inputEmail"
placeholder="Email" required>
    <div class="invalid-feedback">Please enter a valid email address.</div>
  </div>
  <div class="mb-3">
    <label class="form-label" for="inputPassword">Password</label>
    <input type="password" class="form-control" id="inputPassword"
placeholder="Password" required>
    <div class="invalid-feedback">Please enter your password to continue.</div>
  </div>
  <div class="mb-3">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="checkRemember">
      <label class="form-check-label" for="checkRemember">Remember me</label>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

# JavaScript code that displays error messages and disables form submission if there are invalid fields

```
<script>
  // Self-executing function
  (function() {
    'use strict';
    window.addEventListener('load', function() {
      // Fetch all the forms we want to apply custom Bootstrap validation
      // styles to
      var forms = document.getElementsByClassName('needs-validation');

      // Loop over them and prevent submission
      var validation = Array.prototype.filter.call(forms, function(form) {
        form.addEventListener('submit', function(event) {
          if (form.checkValidity() === false) {
            event.preventDefault();
            event.stopPropagation();
          }
          form.classList.add('was-validated');
        }, false);
      });
    }, false);
  })();
</script>
```

Email  
peterparker@example.com ✓

Password  
Password ⓘ  
Please enter your password to continue.

Remember me

Sign in

# Bootstrap Buttons

- Button styles can be applied to any element. However, it is applied normally to the <a>, <input>, and <button> elements for the best rendering.

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-link">Link</button>
```

Primary Secondary Success Danger Warning Info Dark Light Link

# Bootstrap Dropdowns

- The dropdown menu is typically used inside the navigation header to display a list of related links when a user mouse hover or click on the trigger element.
- You can use the Bootstrap dropdown plugin to add toggleable dropdown menus (i.e. open and close on click) to almost anything such as links, buttons or button groups, navbar, tabs and pills nav etc. without even writing a single line of JavaScript code.

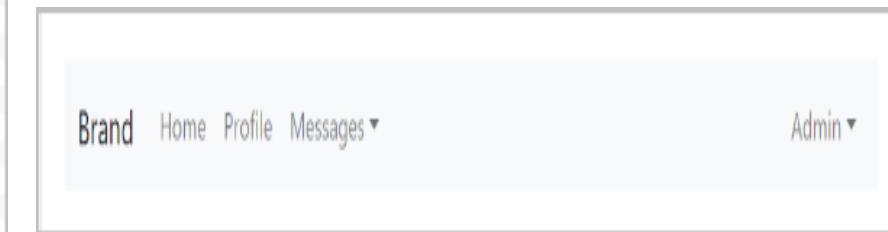
# Adding Dropdowns via Data Attributes

- The Bootstrap dropdown has basically two components — the dropdown trigger element which can be a hyperlink or button, and the dropdown menu itself.
- The .dropdown class specifies a dropdown menu.
- The .dropdown-toggle class defines the trigger element, which is a hyperlink in our case, whereas the attribute data-bs-toggle="dropdown" is required on the trigger element to toggle the dropdown menu.
- The <div> element with the class .dropdown-menu is actually building the dropdown menu that typically contains the related links or actions.

```
<div class="dropdown">
    <a href="#" class="dropdown-toggle" data-bs-toggle="dropdown">Dropdown</a>
    <div class="dropdown-menu">
        <a href="#" class="dropdown-item">Action</a>
        <a href="#" class="dropdown-item">Another action</a>
    </div>
</div>
```

# Dropdowns within a Navbar

```
<nav class="navbar navbar-expand-sm navbar-light bg-light">
    <div class="container-fluid">
        <a href="#" class="navbar-brand">Brand</a>
        <button type="button" class="navbar-toggler" data-bs-toggle="collapse"
data-bs-target="#navbarCollapse">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div id="navbarCollapse" class="collapse navbar-collapse">
            <ul class="nav navbar-nav">
                <li class="nav-item">
                    <a href="#" class="nav-link">Home</a>
                </li>
                <li class="nav-item">
                    <a href="#" class="nav-link">Profile</a>
                </li>
                <li class="nav-item dropdown">
                    <a href="#" class="nav-link dropdown-toggle" data-bs-
toggle="dropdown">Messages</a>
                    <div class="dropdown-menu">
                        <a href="#" class="dropdown-item">Inbox</a>
                        <a href="#" class="dropdown-item">Drafts</a>
                        <a href="#" class="dropdown-item">Sent Items</a>
                        <div class="dropdown-divider"></div>
                        <a href="#" class="dropdown-item">Trash</a>
                    </div>
                </li>
            </ul>
            <ul class="nav navbar-nav ms-auto">
                <li class="nav-item dropdown">
                    <a href="#" class="nav-link dropdown-toggle" data-bs-
toggle="dropdown">Admin</a>
                    <div class="dropdown-menu dropdown-menu-end">
                        <a href="#" class="dropdown-item">Reports</a>
                        <a href="#" class="dropdown-item">Settings</a>
                        <div class="dropdown-divider"></div>
                        <a href="#" class="dropdown-item">Logout</a>
                    </div>
                </li>
            </ul>
        </div>
    </div>
</nav>
```



# Dropdowns within Buttons

```
<div class="btn-group">
    <button type="button" class="btn btn-primary dropdown-toggle" data-bs-
toggle="dropdown">Primary</button>
    <div class="dropdown-menu">
        <a href="#" class="dropdown-item">Action</a>
        <a href="#" class="dropdown-item">Another action</a>
        <div class="dropdown-divider"></div>
        <a href="#" class="dropdown-item">Separated link</a>
    </div>
</div>
```



# Bootstrap Nav: Tabs and Pills

- All the Bootstrap's nav components, including tabs and pills, share the same base markup and styles through the base .nav class.
- Use the Bootstrap .nav class to create a basic navigation menu.

```
<nav class="nav">
    <a href="#" class="nav-item nav-link active">Home</a>
    <a href="#" class="nav-item nav-link">Profile</a>
    <a href="#" class="nav-item nav-link">Messages</a>
    <a href="#" class="nav-item nav-link disabled" tabindex="-1">Reports</a>
</nav>
```

Home      Profile      Messages      Reports

# Alignment of Nav Items

- By default, navs are left-aligned, but you can easily align them to center or right using flexbox utilities.
- For example, uses the class `.justify-content-center` to align nav items to center.

```
<nav class="nav justify-content-center">
  <a href="#" class="nav-item nav-link active">Home</a>
  <a href="#" class="nav-item nav-link">Profile</a>
  <a href="#" class="nav-item nav-link">Messages</a>
  <a href="#" class="nav-item nav-link disabled" tabindex="-1">Reports</a>
</nav>
```

Home      Profile      Messages      Reports

# Alignment of Nav Items

- Vertically stack nav items by changing the flex item direction with the class .flex-column.
- To stack nav vertically on smaller viewports but not on others, use it with responsive breakpoint (e.g., .flex-sm-column).

```
<nav class="nav flex-column">  
    <a href="#" class="nav-item nav-link active">Home</a>  
    <a href="#" class="nav-item nav-link">Profile</a>  
    <a href="#" class="nav-item nav-link">Messages</a>  
    <a href="#" class="nav-item nav-link disabled" tabindex="-1">Reports</a>  
</nav>
```

Home

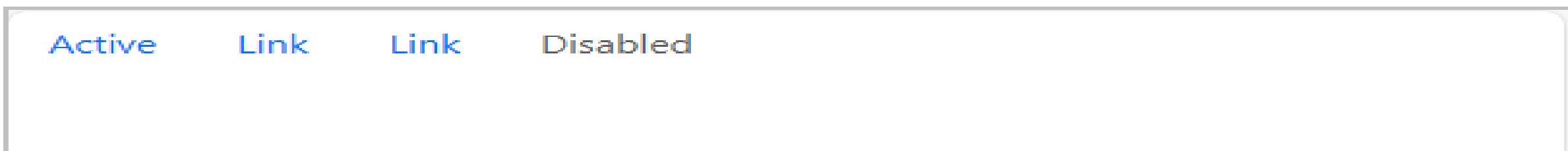
Profile

Messages

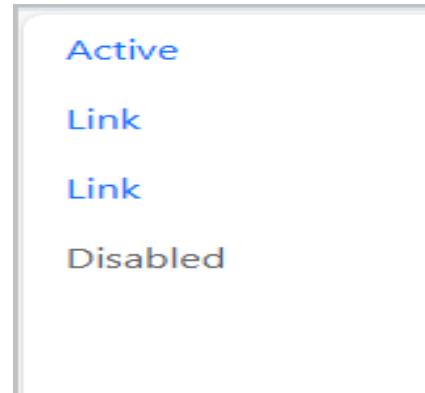
Reports

## Example

- Use flex-sm-row and flex-column classes for responsiveness. You could use flex-sm-row to keep the row horizontal on sm and up, then flex-column to stack vertically on screen widths (<576px).
- For sm and up,



- For screen widths (<576px)



# Bootstrap Pagination

```
<nav>
  <ul class="pagination">
    <li class="page-item"><a href="#" class="page-link">Previous</a></li>
    <li class="page-item"><a href="#" class="page-link">1</a></li>
    <li class="page-item"><a href="#" class="page-link">2</a></li>
    <li class="page-item"><a href="#" class="page-link">3</a></li>
    <li class="page-item"><a href="#" class="page-link">4</a></li>
    <li class="page-item"><a href="#" class="page-link">5</a></li>
    <li class="page-item"><a href="#" class="page-link">Next</a></li>
  </ul>
</nav>
```

Previous 1 2 3 4 5 Next