

Fundamentals of Web Development

Third Edition by Randy Connolly and Ricardo Hoar



Chapter 2

(In the book this is chapter 3+4+5)

Part 1

HTML: Introduction

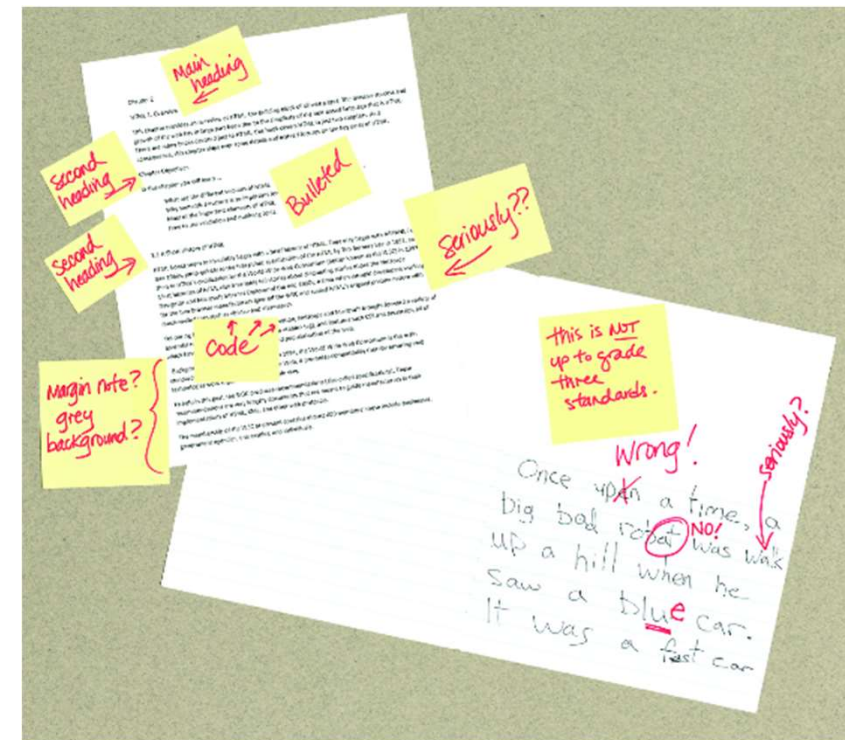
In this chapter you will learn . . .

- A very brief history of HTML
- The syntax of HTML
- Why semantic structure is so important for HTML
- How HTML documents are structured
- A tour of the main elements in HTML
- The semantic structure elements in HTML5

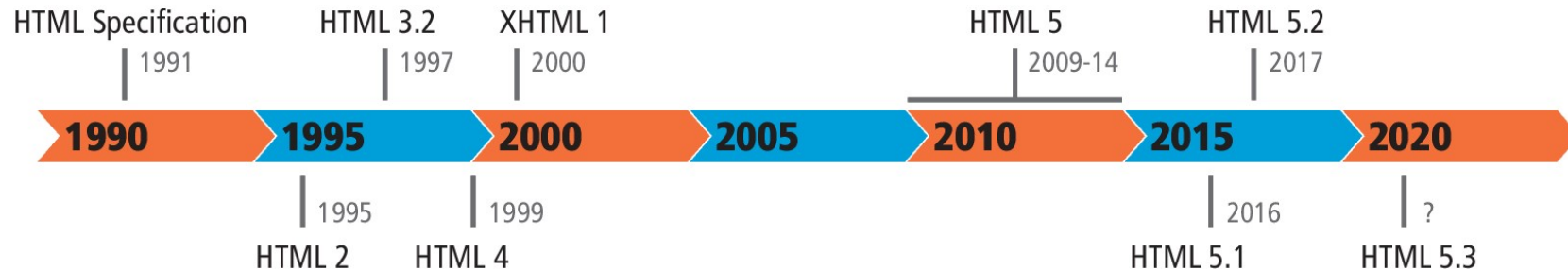
What Is HTML and Where Did It Come From?

- HTML is defined as a **markup language**. A markup language is simply a way of annotating a document in such a way as to make the annotations distinct from the text being annotated (using tags, now called HTML elements)

```
1 <body>
2 <!-- This is a comment .... -->
3 <!-- This is a heading of level 1 -->
4 <h1>A - Heading of Level 1:</h1>
5   <h2>1. Heading of Level 2:</h2>
6     <p>This is text about the subject presented in A.1.....</p>
7     <p>. more text about the subject presented in A.1....</p>
8
9     </img>
10    <!-- The image tag can be written as a self closing tag .... -->
11    
12
```



History: HTML, HTML4, XHTML, HTML5



- In the late 1990s the W3C developed a new specification called **XHTML 1.0**, which was a version of HTML that used stricter **XML**
- The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without syntax errors.
- Development on the XHTML 2.0 specification dragged on for many years.

HTML5

- While the XHTML 2.0 specification was being developed, a small group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group).
- By 2009, the W3C stopped work on XHTML 2.0 and instead adopted the work done by WHATWG and named it **HTML5**.
- There are three main aims to HTML5:
 1. Specify unambiguously how browsers should deal with invalid markup.
 2. Provide an open, nonproprietary programming framework (via JavaScript) for creating rich web applications.
 3. Be backward compatible with the existing web.

HTML Syntax

- **HTML documents** are composed of **content** and HTML elements (tags).
- An **HTML element** is identified in the HTML document by tags.
 - A **tag** consists of the element name within angle brackets.
- The **element name** appears in both the **opening tag** and the **closing tag**.
- HTML elements can also contain attributes. An **HTML attribute** is a **name=value** pair that provides more information about the element



Empty Element

- An **empty element** does not contain any text content; instead, it is an instruction to the browser to do something.
- Perhaps the most common empty element is ``, the image element.
- In HTML5, the trailing slash in empty elements is optional.

Example empty element ``

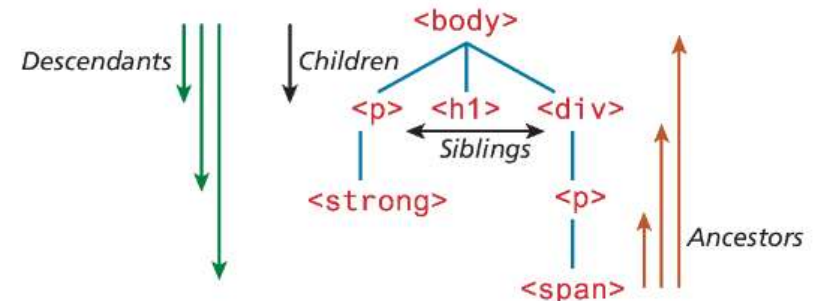
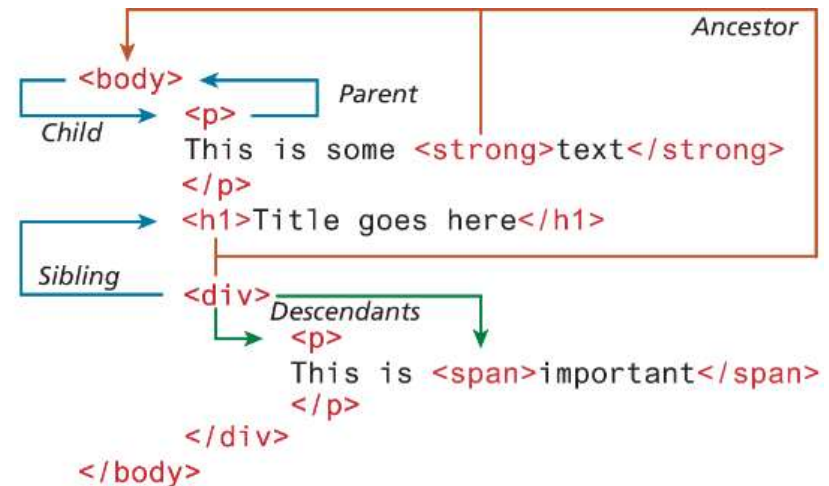
`img` is the Element name

`/>` is the Trailing slash (*optional*)

Google style guide: do not close empty elements.

Nesting HTML Elements


- Often an HTML element will contain other HTML elements. In such case, the container element is said to be a **parent** of the contained, or **child**, element.
- Any elements contained within the child are said to be **descendants** of the parent element; likewise, any given child element may have a variety of **ancestors**.
- This concept is called the **Document Object Model (DOM)**.



Correct Nesting

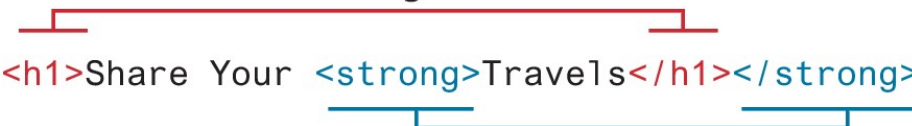
- In order to properly construct this hierarchy of elements, your browser expects each HTML nested element to be properly nested.
- A child's ending tag must occur before its parent's ending tag

Correct nesting



```
<h1>Share Your <strong>Travels</strong></h1>
```

Incorrect nesting



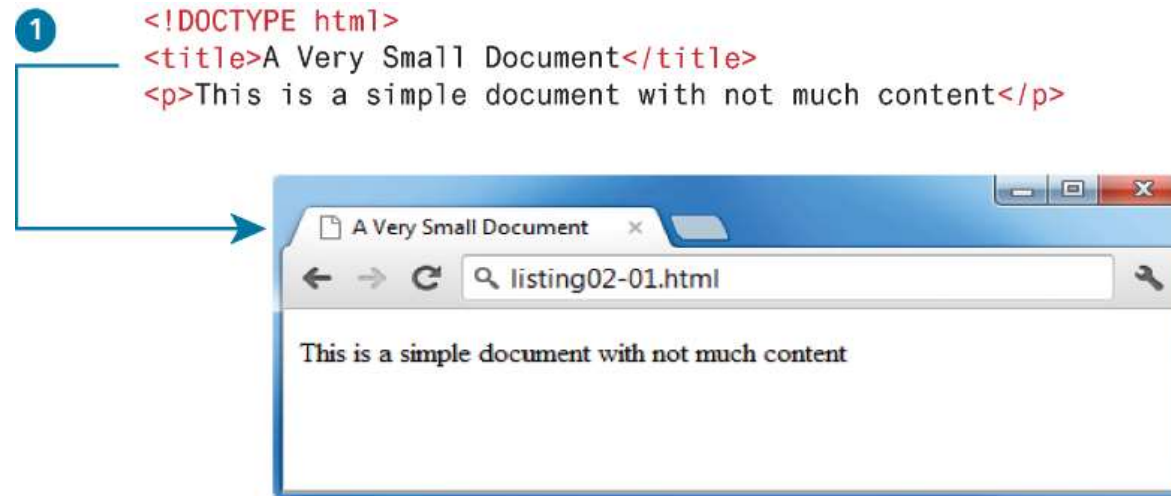
```
<h1>Share Your <strong>Travels</h1></strong>
```

Semantic Markup

- HTML documents should **only** focus on the structure of the document
- HTML document should not describe how to visually present content but only describe its content's structural semantics or meaning
- Information about how the content should look is left to CSS (Cascading Style Sheets).
- As a consequence, beginning HTML authors should create **semantic HTML** documents (such as: section, article, menu, etc.. Instead of div).

Structure of HTML Documents

- A very simple *valid* HTML5 document you can create:



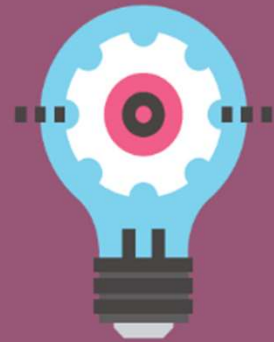
The title element

PRO TIP

The **<title>** element plays an important role in search engine optimization (SEO), that is, improving a page's rank (its position in the results page after a search) .

While each search engine uses different algorithms for determining a page's rank, the **title** (and the major headings) provides a key role in determining what a given page is about.

As a result, be sure that a page's title text briefly summarizes the document's content. As well, put the most important content first in the title. Most browsers limit the length of the title that is displayed in the tab or window title to about 60 characters.



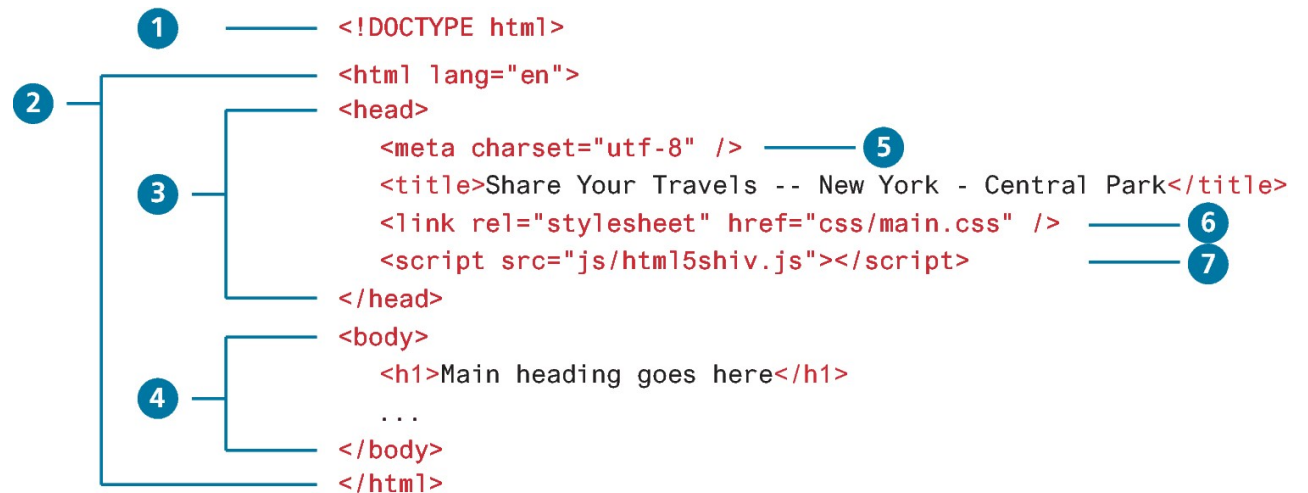
If you are curious to know more about SEO:

https://www.youtube.com/watch?v=MYE6T_gd7H0

Structure elements of an HTML5 document

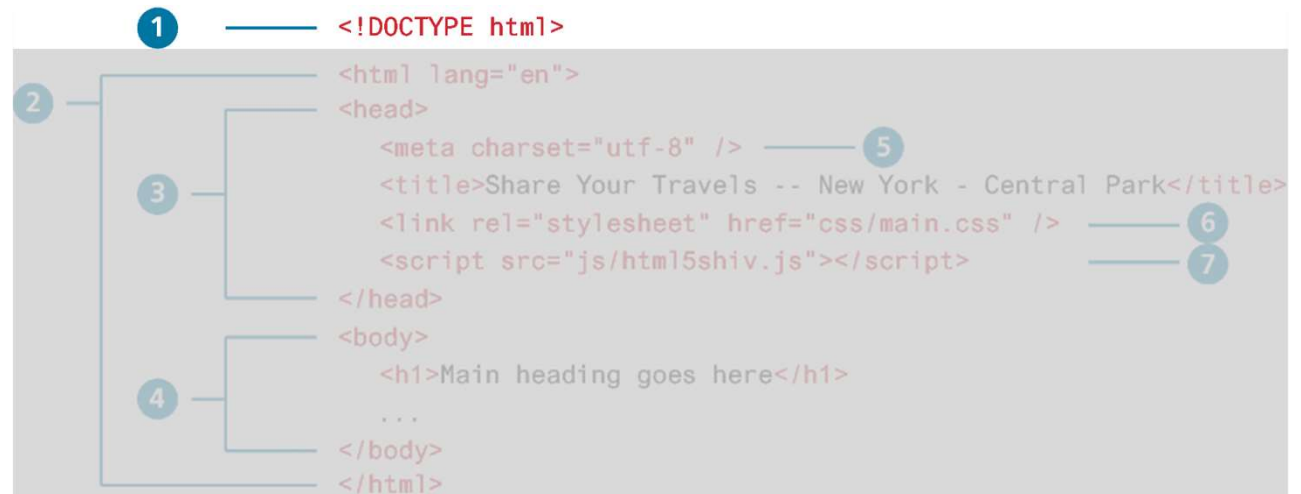
- Consider this more complete HTML5 document that includes structural elements as well as some other common HTML elements.

- Let's explore this page in detail



DOCTYPE

- The DOCTYPE declaration tells the browser what type of document it is about to process.
- It does not indicate what version of HTML is contained within the document



<Html> element

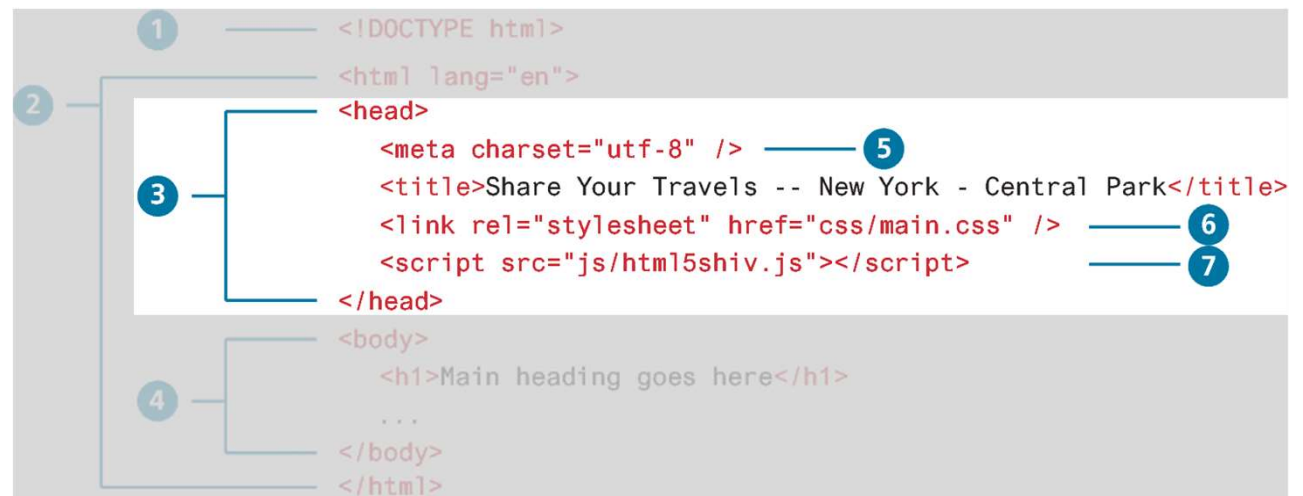
HTML5 does not require the use of the <html>, <head>, and <body> elements. However, they are used for readability.

The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document. The optional **lang** attribute tells the browser the language that is being used.



<Head> Element

The **head** contains descriptive elements *about* the document, such as its title, encoding, and any style sheets or JavaScript files it uses.



<Body> Element

The body contains content (both HTML elements and regular text) that will be displayed by the browser. The rest of this chapter and the next chapter will cover the HTML that will appear within the body.



Quick Tour of HTML Elements

1. **Headings.** Define paragraph titles/sub-titles. There are six levels of headings.
2. **Paragraphs.** The basic unit of text in HTML. As block-level elements, browsers typically add newlines before and after the element.
3. **Link.** Hyperlinks are essential feature of all web pages and can reference another page or another location in same page.
4. **Inline Text Elements.** These do not change the flow of text and provide more information about text.
5. **Image.** Used to display an image by specifying a filename or URL.

```
<body>
1 | <h1>Share Your Travels</h1>
2 | <h2>Venice - Grand Canal</h2>
  | <p>Photo by Randy Connolly</p>
  | <p>This view of the Grand Canal in
  |   <a href="https://en.wikipedia.org/wiki/Venice">Venice</a>
  |   was taken from the <strong>Ponte di Rialto</strong>.
  | </p>
5 | 
6 | <ul>
  |   <li>Photo by <em>Randy Connolly</em></li>
  |   <li>Take on June 23, 2017</li>
  | </ul>
  | <h3>Reviews</h3>
7 | <div>
  |   <p>By Hypatia on <time>2019-10-23</time></p>
  |   <p>I love Venice in the morning.</p>
  | </div>
  | <hr>
  | <div>
  |   <p>By Curia on <time>2019-12-11</time></p>
  |   <p>I want to visit Venice!</p>
  | </div>
  | <footer>Copyright &copy; 2020 Share Your Travels</footer>
</body>
```

Quick Tour of HTML Elements (cont)

6. **Unordered List.** Used to display a bulleted list. Within a list is a collection of list item elements.
7. **Division.** Container for text or other HTML elements. Like paragraphs, they are also block-level elements.
8. **Horizontal Rule.** Indicates a thematic break in the text. Usually displayed as a horizontal line.
9. **Character Entity.** The mechanism for including special symbols (such as ©) or characters that have a reserved meaning in HTML.
10. **Semantic Block Element.** Special containers in HTML5 for describing structural elements in a document.

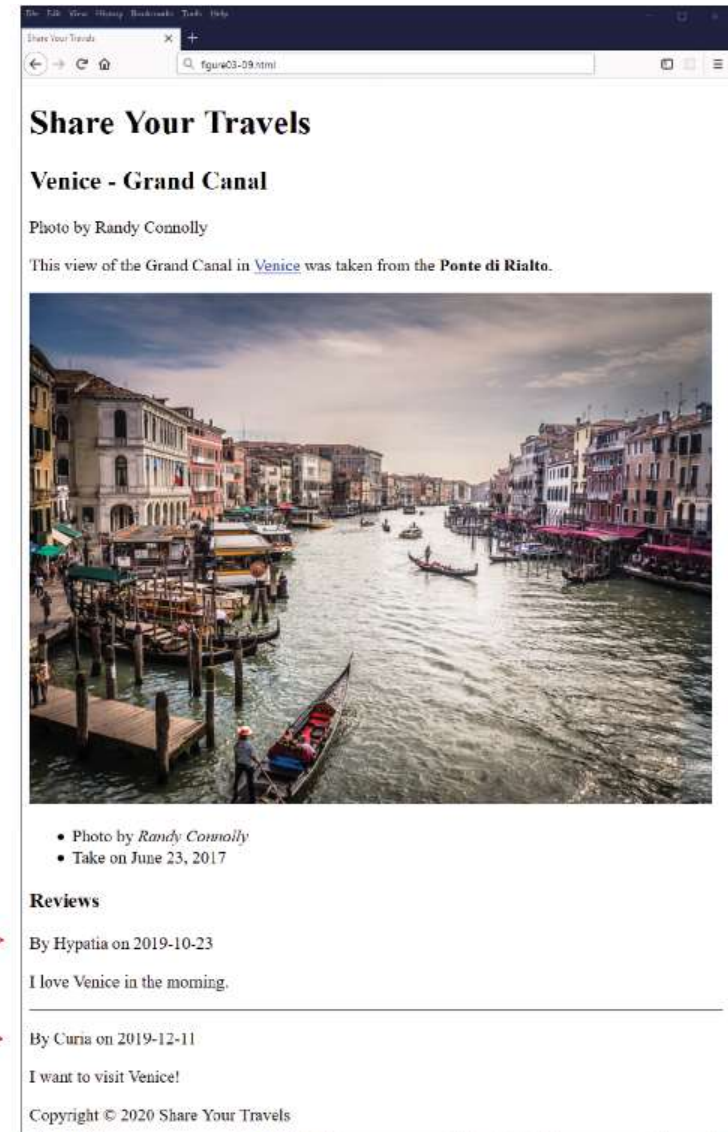
```
<body>
1  <h1>Share Your Travels</h1>
2  <h2>Venice - Grand Canal</h2>
   <p>Photo by Randy Connolly</p>
   <p>This view of the Grand Canal in
      <a href="https://en.wikipedia.org/wiki/Venice">Venice</a>
      was taken from the <strong>Ponte di Rialto</strong>.
   </p>
5  
6  <ul>
   <li>Photo by <em>Randy Connolly</em></li>
   <li>Take on June 23, 2017</li>
</ul>
   <h3>Reviews</h3>
7  <div>
   <p>By Hypatia on <time>2019-10-23</time></p>
   <p>I love Venice in the morning.</p>
</div>
   <hr>
   <div>
   <p>By Curia on <time>2019-12-11</time></p>
   <p>I want to visit Venice!</p>
</div>
   <footer>Copyright &copy; 2020 Share Your Travels</footer>
</body>
```

Diagram illustrating the structure of an HTML document with numbered annotations (1-10) pointing to specific elements:

- 1: `<h1>` tag
- 2: `<h2>` tag
- 3: `` tag
- 4: `<div>` tag
- 5: `` tag
- 6: `` tag
- 7: `<div>` tag
- 8: `<hr>` tag
- 9: `<div>` tag
- 10: `<body>` tag

In the browser

```
<body>
├── <h1>
├── <h2>
├── <p>
├── <p>
│   ├── <a>
│   └── <strong>
├── <img>
├── <ul>
│   ├── <li> ── <em>
│   └── <li>
├── <h3>
├── <div>
│   ├── <p> ── <time>
│   └── <p>
├── <hr>
├── <div>
│   ├── <p> ── <time>
│   └── <p>
└── <footer>
```

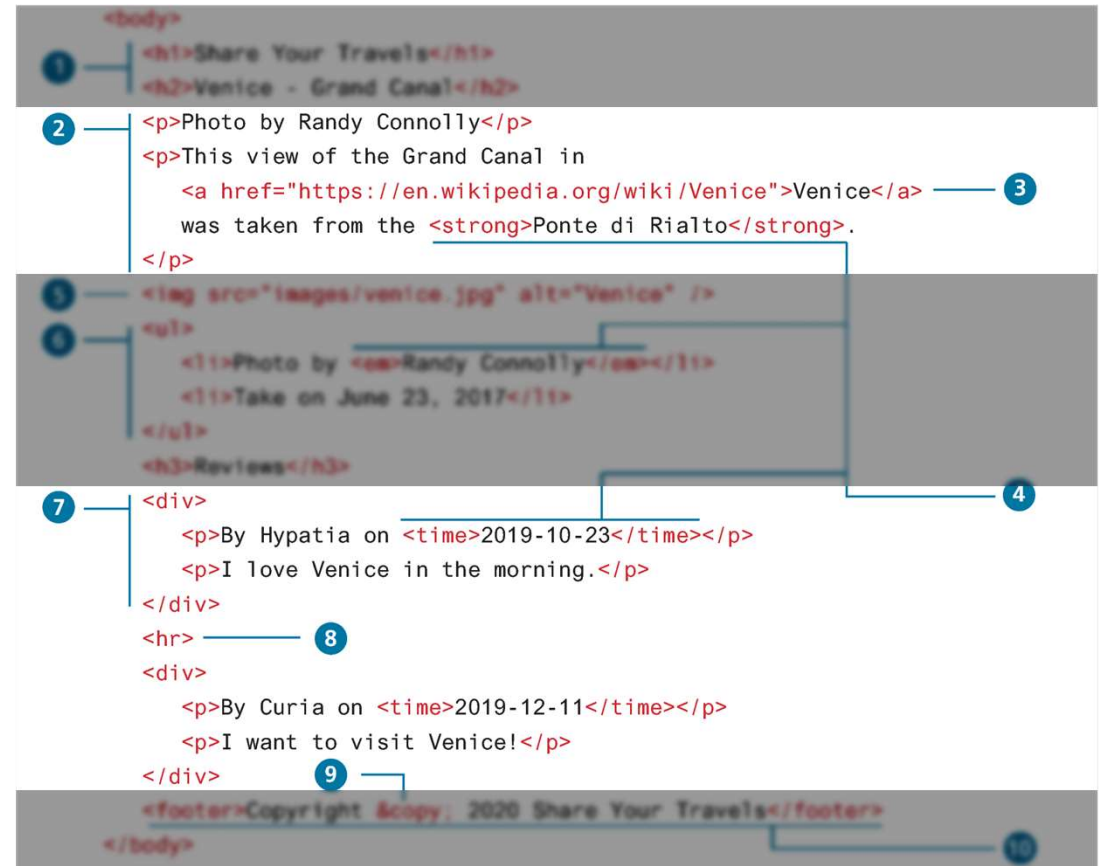


Headings

- HTML provides six levels of heading (h1 through h6)
- They are an essential way for document authors to show their readers the structure of the document
- Headings are also used by the browser to create a **document outline** for the page.
- Choose the heading level because it is appropriate semantically NOT because of its default presentation (e.g., choosing <h3> because you want your text to be bold and 16pt).

Paragraphs and Divisions

- The `<p>` tag is a container. It can contain HTML and other **inline HTML elements**
- `<div>` is also a container element. The `<div>` element has no intrinsic presentation or semantic value;
- `<hr>` element is used to add a **semantic** “break” between paragraphs or `<div>` elements.



HyperLinks

- Links are created using the `<a>` element (the “a” stands for anchor).
- A link has two main parts: the **destination** and the **label**.

```
<a href="http://www.centralpark.com">Central Park</a>
```

Destination

Label (text)

```
<a href="index.html"></a>
```

Label (image)

Kinds of Links

- Links to external sites (or to individual resources, such as images or videos on an external site).
- Links to other pages or resources within the current site.
- Links to other places within the current page.
- Links to particular locations on another page (whether on the same site or on an external site).
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to to execute a JavaScript function: ``
- Links that are instructions to the mobile browser to make a phone call: ``
- Links that are instructions to other programs (e.g., Skype, FaceTime, FaceBook Messenger):
`<a href="http://m.me/<FACEBOOK_PAGE_USERNAME">`

Absolute and Relative URLs

When referencing a page or resource on an **external site**, a full **absolute URL reference** is required

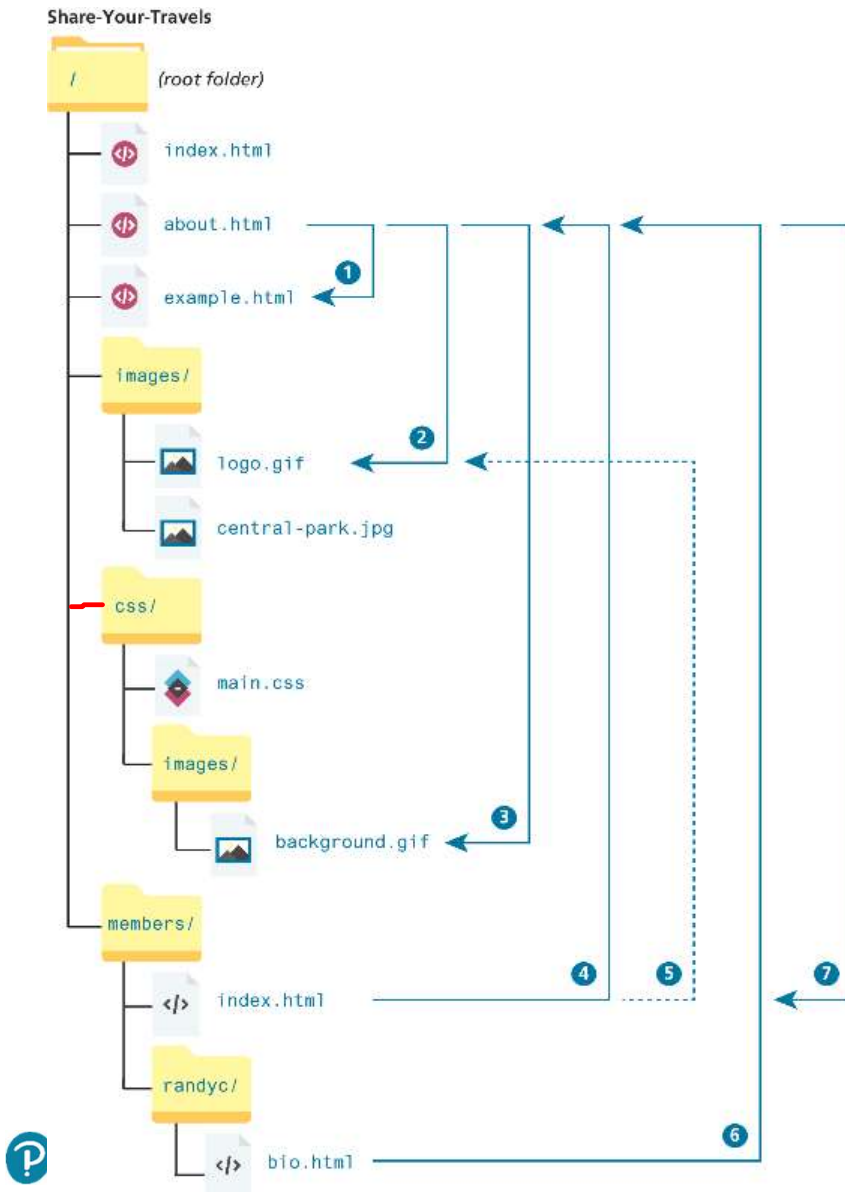
- Full URL with a protocol (typically, http:// or https://), the domain name, any paths, and the file name of the desired resource.

When referencing a resource that is on the **same server**, you can use **relative referencing**.

- If the URL does not include the “**http://**” then the browser will request the current server for the file.

Relative URLs

1. **Same Directory** To link to a file within the same folder: simply use the file name.
2. **Child Directory** To link to a file within a subdirectory: use the name of the subdirectory and a slash before the file name.
3. **Grandchild/Descendant Directory** To link to a file that is multiple subdirectories *below* the current one: construct the full path by including each subdirectory name (separated by slashes) before the file name.
4. **Parent/Ancessor Directory** Use “../” to reference a folder *above* the current one. If trying to reference a file several levels above the current one, simply string together multiple “../”.
5. **Sibling Directory** Use “../” to move up to the appropriate level, and then use the same technique as for child or grandchild directories.
6. **Root Reference** In this approach, begin the reference with the root reference (the “/”), and then use the same technique as for child or grandchild directories.



Inline Text Elements

Inline elements do not disrupt the flow of text (i.e., cause a line break).

- **<a>** Anchor used for hyperlinks.
- **<abbr>** An abbreviation
- **
** Line break
- **<cite>** Citation (i.e., a reference to another work)
- **<code>** Used for displaying code, such as markup or programming code
- **** Emphasis
- **<small>** For displaying the fine-print, that is, “nonvital” text, such as copyright or legal notices
- **** The inline equivalent of the <div> element. It is generally used to mark text that will receive special formatting using CSS
- **** For content that is strongly important
- **<time>** For displaying time and date data

Images

- Note the key attributes of the `` element.
- Attributes such as `title`, `width`, and `height` are optional

Specifies the URL of the image to display
(note: uses standard relative referencing).

Text in `title` attribute will be displayed in a pop-up
tool tip when user moves mouse over image (optional).

```

```

Text in `alt` attribute provides a brief
description of image's content for users who
are unable to see it.

Specifies the width and height of
image in pixels (discouraged)

Character Entities

Character entities are special characters for symbols for which there is either no easy way to type them via a keyboard or which have a reserved meaning in HTML (for instance the “<” or “>” symbols).

They can be used in an HTML document by using the entity name or the entity number

Entity Name	Entity Number	Description
&nbsp;	&#160;	Non-breakable space.
&lt;	&#60;	Less than symbol (“<”).
&gt;	&#62;	Greater than symbol (“>”).
&copy;	&#169;	The © copyright symbol
&euro;	&#8364;	The € euro symbol.
&trade;	&#8482;	The ™ trademark symbol.

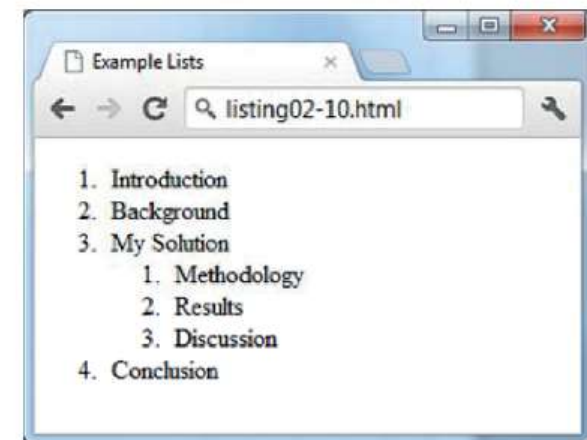
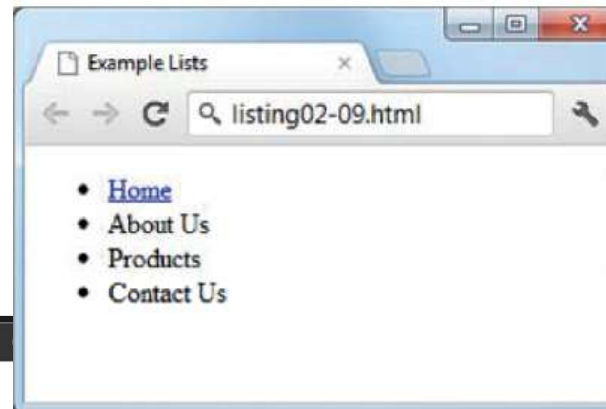
Lists

- **Ordered lists** Collections of items that have a set order
`` ``
- **Unordered Lists** Collections of items in no particular order
`` ``
- **Description Lists** Collection of name and description/definition pairs.
`<dl>` `<dt>``<dd>`

Notice that the list item element can contain other HTML elements.

```
<ul>
  <li><a href="index.html">Home</a></li>
  <li>About Us</li>
  <li>Products</li>
  <li>Contact Us</li>
</ul>
```

```
<ol>
  <li>Introduction</li>
  <li>Background</li>
  <li>My Solution</li>
  <li>
    <ol>
      <li>Methodology</li>
      <li>Results</li>
      <li>Discussion</li>
    </ol>
  </li>
  <li>Conclusion</li>
</ol>
```



HTML	CSS
1 <p>Cryptids of Cornwall:</p>	
2	
3 <dl>	
4 <dt>Beast of Bodmin</dt>	
5 <dd>A large feline inhabiting Bodmin Moor.</dd>	
6	
7 <dt>Morgawr</dt>	
8 <dd>A sea serpent.</dd>	
9	
10 <dt>Owlman</dt>	
11 <dd>A giant owl-like creature.</dd>	
12 </dl>	
13	

Cryptids of Cornwall:

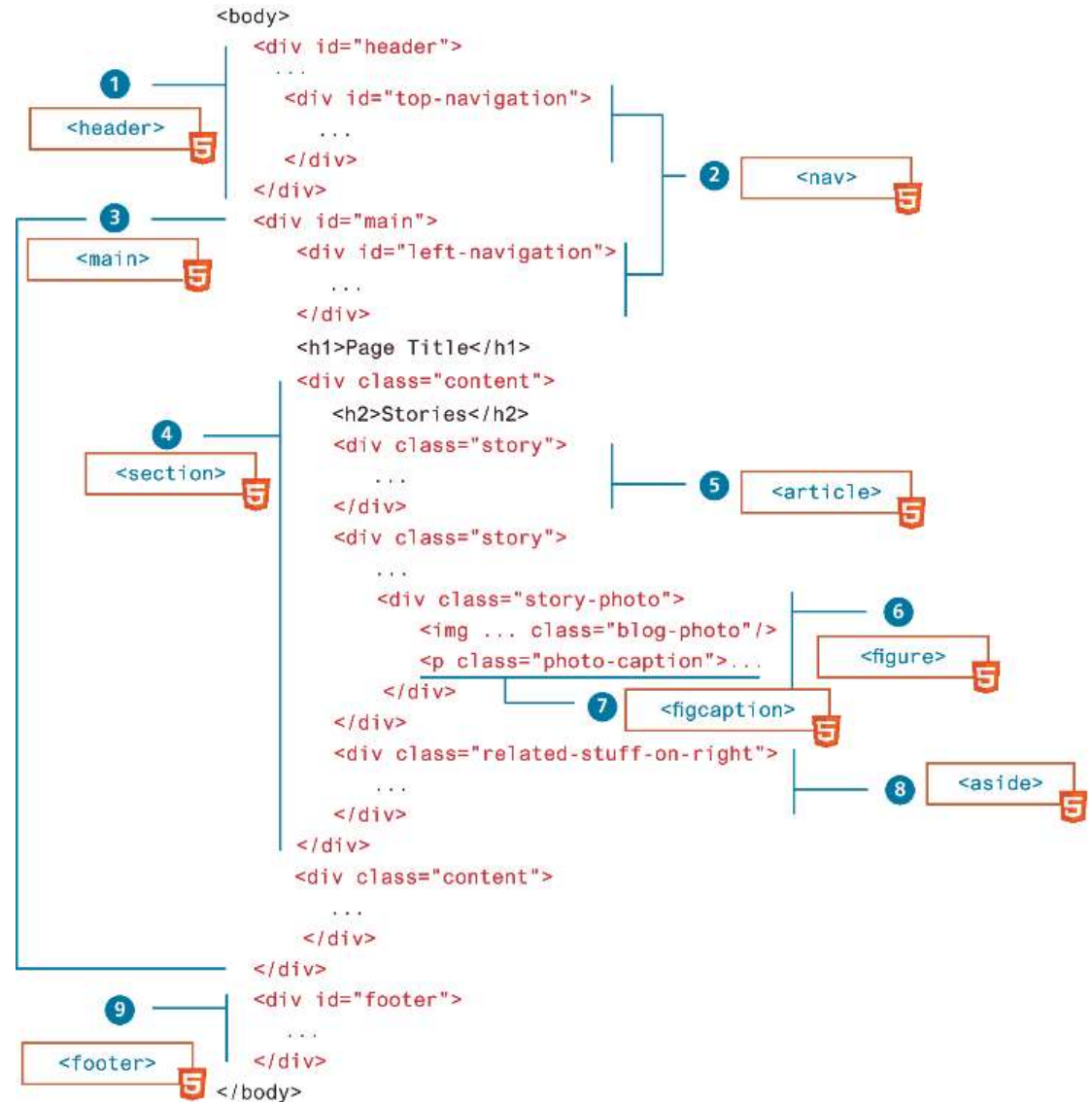
Beast of Bodmin
A large feline inhabiting Bodmin Moor.

Morgawr
A sea serpent.

Owlman
A giant owl-like creature.

HTML5 Semantic Structure

- So far, the main semantic elements you have seen are headings, paragraphs, lists, some inline elements
- HTML5 semantic elements allow to replace some of your `<div>` with cleaner and more self-explanatory elements



HTML5 Semantic Structure Elements

- Header
- Nav
- Main
- Section
- Article
- Figure
- Figcaption
- Aside
- Footer

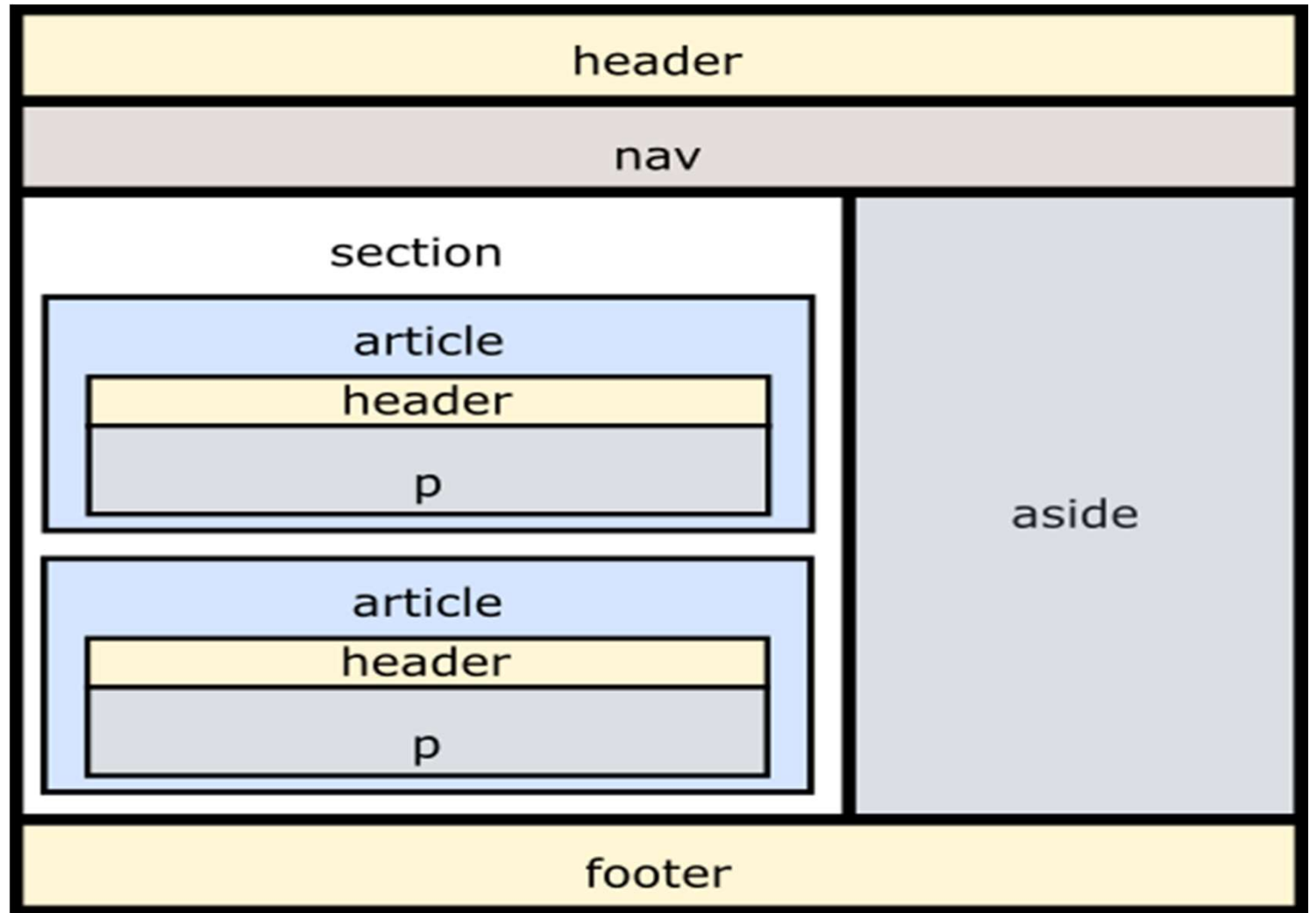


Figure and figcaption

The `<figure>` element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document, and the rest of the document would still make sense.

Figure could be moved to a different location in document ...

But it has to exist in the document (i.e., the figure isn't optional).

```
<p>This photo was taken on October 22, 2011 with a Canon EOS 30D camera.</p>
<figure>
  <br/>
  <figcaption>Conservatory Pond in Central Park</figcaption>
</figure>
<p>
  It was a wonderfully beautiful autumn Sunday, with strong sunlight and
  expressive clouds. I was very fortunate that my one day in New York was
  blessed with such weather!
</p>
```



Fundamentals of Web Development

Third Edition by Randy Connolly and Ricardo Hoar



Chapter 2: Part2

HTML Tables and Forms

In this chapter you will learn . . .

- What HTML tables are and how to create them
- What forms are and how they work
- What the different form controls are and how to use them
- How to improve the accessibility of your websites

HTML Tables

A **table** in HTML is created using the `<table>` element and can be used to represent information that exists in a two-dimensional grid.

Just like a real-world table, an HTML table can contain any type of data: not just numbers, but text, images, forms, even other tables

`<table>` contains any number of rows (`<tr>`); each row contains any number of table data cells (`<td>`)

#	TEAMS	P	W	D	L	F	A	GD	PTS
1	Liverpool	8	6	2	0	22	8	14	20
2	Manchester City	8	5	1	2	20	10	10	16
3	West Ham United	8	4	3	1	7	4	3	15
4	Arsenal	8	4	2	2	15	10	5	14
5	Leicester City	8	5	3	0	16	8	8	14
6	Chelsea								
7	AFC Bournemouth								
8	Tottenham Hotspur								
9	Crystal Palace								
10	Manchester United								

Name	Project	Start Date	End Date	Status
Thomas Aquinas	Divine Justice	01/01/1225	03/07/1274	Completed
Baruch Spinoza	Ethics	02/16/1632	02/21/1632	Completed
David Hume	Epistemological principles	05/07/1711	08/25/1776	Started
John Rawls	Justice as fairness	02/21/1921	11/24/2002	Pending
Michel Foucault	Power	11/15/1926	06/25/1984	Pending
Martin Heidegger	Undear	09/26/1889	05/26/1976	Cancelled

SU	MO	TU	WE	TH	FR	SA
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Basic table structure

Month	Savings
January	\$100
February	\$80

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
</table>
```

- All content must appear within the `<td>` or `<th>` container.
- If you want a given cell to cover several columns or rows, then you can do so by using the **colspan** or **rowspan** attributes

Spanning columns

<table>	Title<th>	Artist<th>	Year<th>	Size (width x height)<th colspan=2>	
<tr>	The Death of Marat<td>	Jacques-Louis David<td>	1793<td>	162cm<td>	128cm<td>
<tr>	Burial at Ornans<td>	Gustave Courbet<td>	1849<td>	314cm<td>	663cm<td>

Notice that this row now only has four cell elements.

```
<table>
<tr>
  <th>Title</th>
  <th>Artist</th>
  <th>Year</th>
  <th colspan="2">Size (width x height)</th>
</tr>
<tr>
  <td>The Death of Marat</td>
  <td>Jacques-Louis David</td>
  <td>1793</td>
  <td>162cm</td>
  <td>128cm</td>
</tr>
  ...
</table>
```

Additional table elements

An HTML table with a <thead>, <tbody>, and a <tfoot> **semantic** elements:

```
*lab01.html X
1 <table>
2 <thead>
3   <tr> <th>Month</th> <th>Savings</th> </tr>
4 </thead>
5 <tbody>
6   <tr> <td>January</td> <td>$100</td> </tr>
7   <tr> <td>February</td> <td>$80</td> </tr>
8 </tbody>
9 <tfoot>
10   <tr> <td>Sum</td> <td>$180</td> </tr>
11 </tfoot>
12 </table>
```

Using Tables for Layout

Prior to the broad support for CSS in browsers, HTML tables were frequently used to create page layouts. Unfortunately, this practice of using tables for layout had some problems:

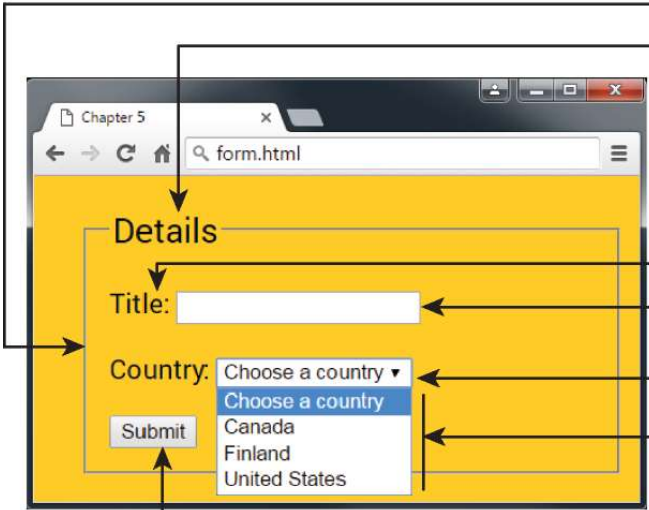
1. This approach tended to increase the size of the HTML document
2. The resulting markup is not semantic.

Introducing Forms

Forms provide the user with an alternative way to interact with a web server.

- Up to now, clicking hyperlinks was the only mechanism available to the user
- Using a form, the user can enter text, choose items from lists, and click buttons
- Typically, programs running on the server will take the input from HTML forms and do something with it,
 - such as save it in a database,
 - interact with an external web service, or
 - customize subsequent HTML based on that input.
- HTML5 has added a number of new controls and more customization options

Form Structure



```
<form method="post" action="process.php">
  <fieldset>
    <legend>Details</legend>
    <div>
      <label>Title: </label>
      <input type="text" name="title" />
    </div>
    <div>
      <label>Country: </label>
      <select name="where">
        <option>Choose a country</option>
        <option>Canada</option>
        <option>Finland</option>
        <option>United States</option>
      </select>
    </div>
    <input type="submit" />
  </fieldset>
</form>
```

The <form> Element

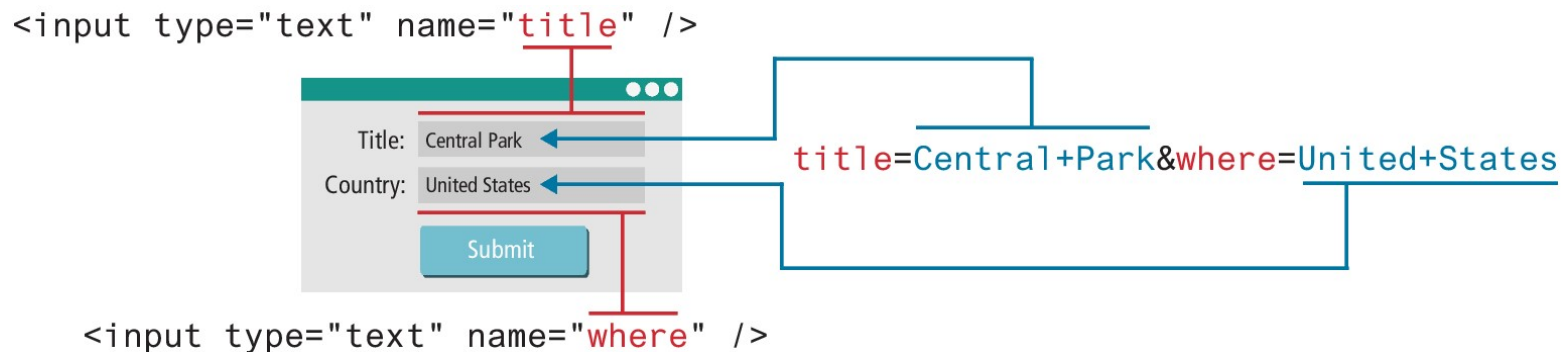
Two important attributes that are essential features of any <form> are the **action** and the **method** attributes.

- The **action** attribute specifies the URL of the server-side resource that will process the form data.
- The **method** attribute specifies how the query string data will be transmitted from the browser to the server (GET, POST, DELETE, UPDATE).

Query Strings

The browser “sends” the data to the server via an HTTP request using a query string.

A **query string** is a series of **name=value** pairs separated by ampersands (the & character). Special symbols must be **URL encoded**



GET

- Data can be clearly seen in the address bar. This may be an advantage during development but a disadvantage in production.
- Data remains in browser history and cache. Again this may be beneficial to some users, but it is a security risk on public computers.
- Data can be bookmarked (also an advantage and a disadvantage).
- There is a limit on the number of characters in the returned form data.

POST

POST

- Data can contain binary data.
- Data is hidden from user (*open Network/Payload tab in devtools to see it*).
- Submitted data is not stored in cache, history, or bookmarks.

NOTE

It should be noted that while the POST method “hides” form data, any user could easily inspect the HTTP header. As a result, the POST method is NOT sufficient from a security standpoint.



Form Control Elements

<button> Defines a clickable button.

<datalist> An HTML5 element that defines lists of pre-defined values to use with input fields.

<fieldset> Groups related elements in a form together.

<form> Defines the form container.

<input> Defines an input field. HTML5 defines over 20 different types of input.

<label> Defines a label for a form input element.

<legend> Defines the label for a fieldset group.

<optgroup> Defines a group of related options in a multi-item list.

<option> Defines an option in a multi-item list.

<output> Defines the result of a calculation.

<select> Defines a multi-item list.

<textarea> Defines a multiline text entry box.

Text Input Controls

Most forms need to gather text information from the user. Whether it is a search box or a login form or a user registration form, some type of text input is usually necessary.

```
<input type="text" ... />
```

Text: |

```
<textarea>enter some text</textarea>
<textarea placeholder="enter some text">
</textarea>
```

TextArea: TextArea:

```
<input type="password" ... />
```

Password: Password:

```
<input type="search" placeholder="enter search text" ... />
```

Search: Search:

```
<input type="email" ... />
```

Email: In Opera

Please enter a valid email address

Email: In Chrome

Please enter an email address.

```
<input type="url" ... />
```

url:

Please enter a URL.

```
<input type="tel" ... />
```

Tel: |

Choice Controls

Forms often need the user to select an option from a group of choices. HTML provides several ways to do this.

- Select Lists
- Radio Buttons
- Checkboxes

Select Lists

- The **<select>** element is used to create a drop-down list.
- The multiple attribute allows more than one element to be selectable.
- The selected attribute in the **<option>** makes it a default value.
- The value attribute is optional; if it is not specified, then the text within the container is sent instead

Select:

Select:

Second
First
Second
Third

```
<select name="choices">  
  <option>First</option>  
  <option selected>Second</option>  
  <option>Third</option>  
</select>
```

Select Lists (ii)

- Option items can be grouped together via the **<optgroup>** element.



```
<select ... >  
  <optgroup label="North America">  
    <option>Calgary</option>  
    <option>Los Angeles</option>  
  </optgroup>  
  <optgroup label="Europe">  
    <option>London</option>  
    <option>Paris</option>  
    <option>Prague</option>  
  </optgroup>  
</select>
```

Radio Buttons

Radio buttons are useful when you want the user to select a single item from a small list of choices that are visible.

radio buttons are added via the **<input type="radio">** element

The checked attribute is used to indicate the default choice.

The value attribute is given to the name variable created and sent as form data: here, city=1 if Riyadh is selected.

```
<fieldset>
  <legend>Radio input:</legend>
  <div>
    <input type="radio" id="radio1" name="city" value="1" checked>
    <label for="radio1">Riyadh</label>
  </div>
  <div>
    <input type="radio" id="radio2" name="city" value="2">
    <label for="radio2">Jeddah</label>
  </div>
  <div>
    <input type="radio" id="radio3" name="city" value="3">
    <label for="radio3">Damman</label>
  </div>
</fieldset>
```

Radio input:

☒ Riyadh

☐ Jeddah

☐ Damman

Checkboxes

A **checkbox** is used for obtaining a yes/no or on/off response from the user.

Checkboxes are added via the **<input type="checkbox">**

The **checked** attribute can be used to set the default value of a checkbox.

Each checked checkbox will have its value sent to the server.



Button Controls

<input type="submit"> Creates a button that submits the form data to the server.

<input type="reset"> Creates a button that clears any of the user's already entered form data.

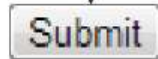
<input type="button"> Creates a custom button. This button may require JavaScript for it to actually perform any action.

<input type="image"> Creates a custom submit button that uses an image for its display.

<button> Creates a custom button.

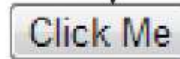
Example button elements

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

A rectangular button with a light gray gradient and a thin border, containing the text "Submit" in a black serif font.A rectangular button with a light gray gradient and a thin border, containing the text "Reset" in a black serif font.

```
<input type="reset" />
```

```
<input type="button" value="Click Me" />
```

A rectangular button with a light gray gradient and a thin border, containing the text "Click Me" in a black serif font.

```
<input type="image" src="appointment.png" />
```

A rectangular button with a light gray gradient and a thin border. It contains a small icon of a document with a pencil on the left and the text "Edit" on the right.A rectangular button with a light gray gradient and a thin border. It contains a small icon of an envelope on the left and the text "Email" on the right.

```
<button>
  <a href="email.html">
    
    Email
  </a>
</button>
```

```
<button type="submit" >
  
  Edit
</button>
```

<button> allows for more customization

Number and Range

- HTML5 introduced the number and range controls, they provide a way to input numeric values that reduces the need for client-side numeric validation (for security reasons you would still check the numbers for validity on the server)

Rate this photo:

```
<label>Rate this photo: <br />
```

```
<input type="number" min="1" max="5" name="rate" />
```

Grumpy



Ecstatic

Grumpy

```
<input type="range" min="0" max="10" step="1" name="happiness" />
```

Ecstatic

Rate this photo:

Grumpy

Ecstatic

Controls as they appear in browser
that doesn't support these input types

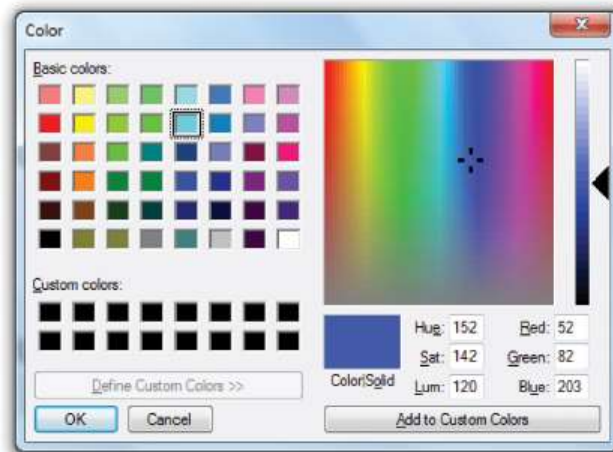
Color

When it is necessary, the HTML5 color control provides a convenient interface for the user

Background Color:



```
<label>Background Color; <br />
<input type="color" name="back" />
```



Background Color:

Control as it appears in browser that
doesn't support this input type

Date and Time Controls

date Creates a general date input control. The format for the date is “yyyy-mm-dd.”

time Creates a time input control. The format for the time is “HH:MM:SS,” for hours:minutes:seconds.

datetime Creates a control in which the user can enter a date and time.

datetime-local Creates a control in which the user can enter a date and time without specifying a time zone.

month Creates a control in which the user can enter a month in a year. The format is “yyyy-mm.”

week Creates a control in which the user can specify a week in a year. The format is “yyyy-W##.”

Date and time controls Figure

Date:



A date picker control showing a calendar for March 2013. The calendar has a header with 'March' and '2013'. The days of the week are listed as Mon, Tue, Wed, Thu, Fri, Sat, Sun. The dates are displayed in a grid, with the 8th of March highlighted. A 'Today' button is at the bottom right.

```
<label>Date: <br/>
<input type="date" ... />
```


Month:



A month picker control showing a calendar for March 2013. The calendar has a header with 'March, 2013'. The days of the week are listed as Sun, Mon, Tue, Wed, Thu, Fri, Sat. The dates are displayed in a grid, with the 8th of March highlighted. A 'This month' button and a 'Clear' button are at the bottom.

```
<input type="month" ... />
```


Time:



A time picker control showing the time 02:02 AM. The time is displayed in a text box with a dropdown arrow on the right.

```
<input type="time" ... />
```

DateTime:



A datetime picker control showing the date and time 2013-03-08 05:46 UTC. The date and time are displayed in a text box with a dropdown arrow on the right.

```
<input type="datetime" ... />
```

DateTime Local:



A datetime local picker control showing the date and time 2013-03-13 12:02. The date and time are displayed in a text box with a dropdown arrow on the right.

```
<input type="datetime-local" ... />
```

Week:



A week picker control showing a calendar for March 2013. The calendar has a header with 'March' and '2013'. The days of the week are listed as Mon, Tue, Wed, Thu, Fri, Sat, Sun. The dates are displayed in a grid, with the 10th of March highlighted. A 'Today' button is at the bottom right.

```
<input type="week" ... />
```

Associating labels and input elements

```
<label for="f-title">Title: </label>
```

```
<input type="text" name="title" id="f-title"/>
```

```
<label for="f-country">Country: </label>
```

```
<select name="where" id="f-country">  
  <option>Choose a country</option>  
  <option>Canada</option>  
  <option>Finland</option>  
  <option>United States</option>  
</select>
```


Validating User Input

- User input must never be trusted.
- It could be missing.
- It might be in the wrong format.
- It might even contain JavaScript or SQL as a means to causing some type of attack/hack.
- Thus, almost always user input must be tested for validity.

Where to Perform Validation

Validation can be performed at three different levels.

Client-side	<ol style="list-style-type: none">1. With HTML5, the browser can perform basic validation.2. JavaScript validation dramatically improves the user experience of data-entry forms, and is an essential feature of any real-world web site that uses forms. Unfortunately, JavaScript validation is not sufficient.
Server-side	<ol style="list-style-type: none">3. server-side validation is arguably the most important since it is the only validation that is guaranteed to run. Server-side functionality should be developed as if no validation occurred on the client-side.

Types of Input Validation

- **Required information.** Some data fields just cannot be left empty.
- **Correct data type.** Some fields such as numbers or dates, must follow the rules for its data type in order to be considered valid.
- **Correct format.** Some information, such as postal codes, credit card numbers, and social security numbers have to follow certain pattern rules.
- **Comparison.** Some user-entered fields are considered correct or not in relation to an already inputted value (password confirm, end date > start date, etc.)
- **Range check.**
- **Custom.**

Notifying the User

- **What is the problem?** Users do not want to read lengthy messages to determine what needs to be changed.
- **Where is the problem?** Some type of error indication should be located near the field that generated the problem.
- **If appropriate, how do I fix it?** For instance, don't just tell the user that a date is in the wrong format; tell him or her what format you are expecting

Notifying the User (Figures)

Sample Form

Form Validation Examples

The following data input errors must be corrected:

- The year must be a valid number between 500 and 2014
- The painting height must be valid number larger than 0

Title

Year The year must be a valid number between 500 and 2014

Medium

Width

Height The painting height must be valid number larger than 0

Link

Add

How to Reduce Validation Errors

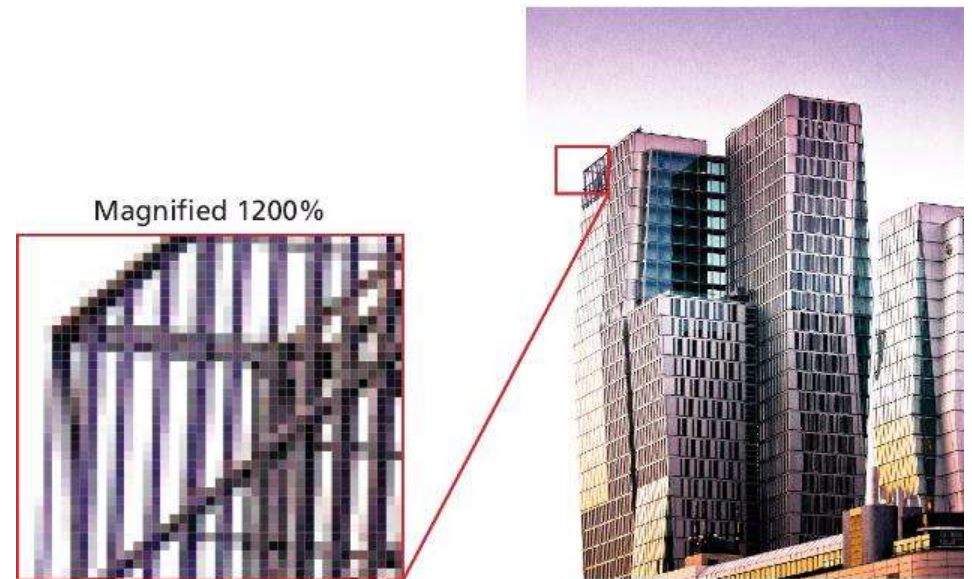
- Provide textual hints to the user on the form itself.
- Using tool tips or pop-overs to display context-sensitive help about the expected input (CSS or the title attribute)
- Another technique for helping the user understand the correct format for an input field is to provide a JavaScript-based mask **(999)-999-9999**
- Providing well chosen default values for text fields can reduce validation errors
- Finally, many user input errors can be eliminated by choosing a better data entry type than the standard `<input type="text">`.

Raster images

In a **raster image** (also called a **bitmap image**) the smaller components are pixels.

Each colored square uses a number that represents its color value.

Raster images can be manipulated on a pixel-by-pixel basis by programs such as Adobe Photoshop, Apple Aperture, Microsoft Paint, or the opensource GIMP

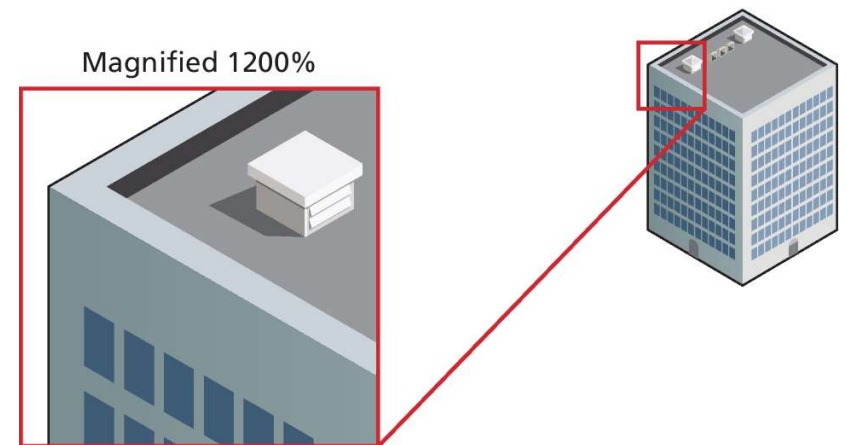


Vector images

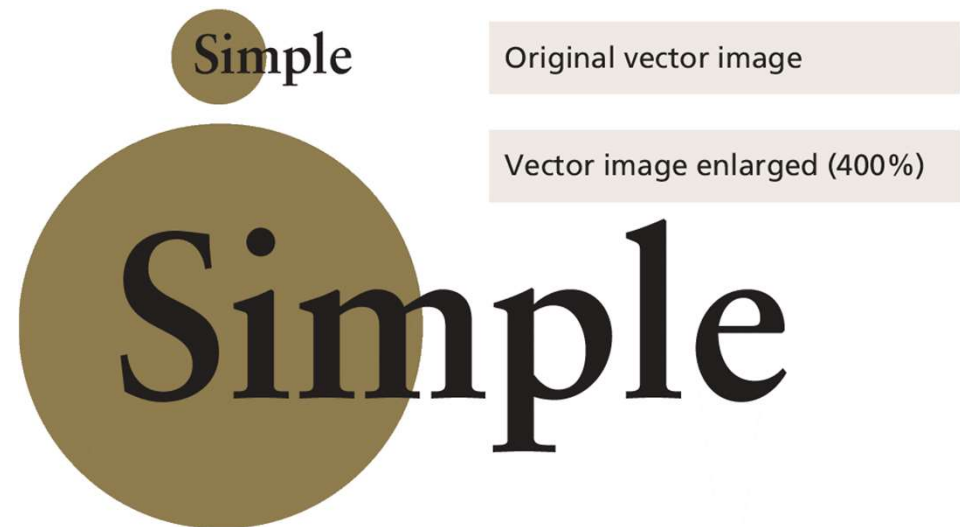
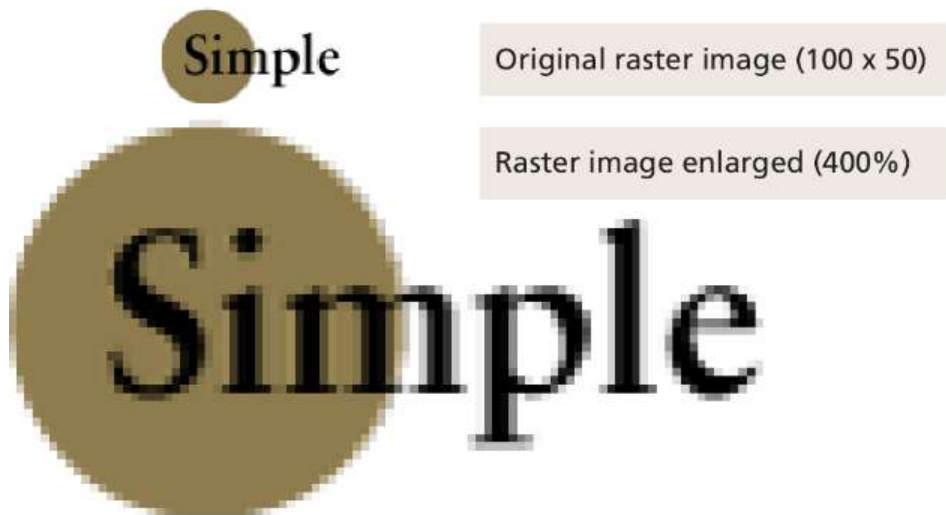
A **vector image** is composed of objects such as lines, circles, curves, and polygons.

Vector images are resolution independent and can be shrunk or enlarged without loss of quality

Software includes Adobe Illustrator, Microsoft Visio, Adobe Animate (formerly Adobe Flash), Affinity Designer (Mac only), and the open-source Inkscape



Resizing raster images versus vector images

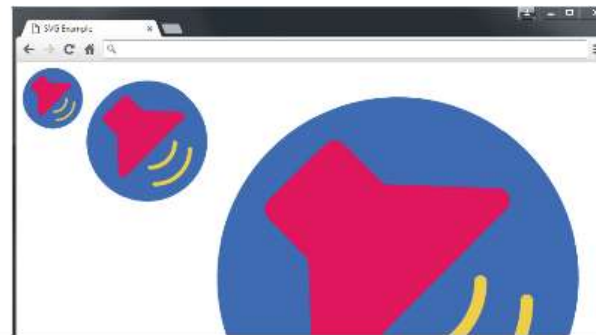


SVG Format

SVG (Scalable Vector Graphics) is a *vector* format

Like all vector formats, SVG graphics do not lose quality when enlarged or reduced.

The files are actually XML files



```
  
  

```

Because SVG is a vector format, there is no loss of quality when it is resized

```
<?xml version="1.0" encoding="utf-8"?>  
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"  
  viewBox="0 0 95 94" style="enable-background:new 0 0 95 94;" xml:space="preserve">  
  <style type="text/css">  
    .st0{fill:#366BC9;} .st1{fill:#E0105B;} .st2{fill:#EFCE4A;}  
  </style>  
  <path class="st0" d="M92.7,46.9c0.25,1.2,0.4,45.5-45.5,45.5C22.1,92.4,1.7,72.1,7.46,9c0-25.1,20.4-45.5,45.5-45.5  
    C72.3,1.4,92.7,21.8,92.7,46.9L92.7,46.9z M92.7,46.9"/>  
  <path class="st1" d="M42.8,22.5l-9.2-9.2c-1.3-1.3-3.4-1.3-4.7,0L14.7,27.4c-1.3,1.3-1.3,3.4,0,4.7l9.2,9.2c0.4,0.4,0.7,0.9,1.5  
    11,28.6c0.6,2.5,3.7,3.3,5.5,1.5l43-43c1.8-1.8,1.4,9-1.5-5.5l-28.6-10c43,7,23,2.43,2,22.9,42.8,22.5L42.8,22.5z M42.8,22.5"/>  
  <path class="st2" d="M51.7,80.3c-0.3-0.3-0.5-0.7-0.5-1.1c0-0.9,0.7-1.6,1.6-1.6c66.7,77.7,78,66.4,78,52.6c0-0.9,0.7-1.6,1.6-1.6  
    c0.9,0,1.6,0.7,1.6,1.6c0,15.6-12.7,28.2-28.2,28.2c52.4,80.8,52,80.6,51.7,80.3L51.7,80.3z M51.7,80.3"/>  
  <path class="st2" d="M48.1,67.8c-0.3-0.3-0.5-0.7-0.5-1.1c0-0.9,0.7-1.6,1.6-1.6c9.5,0,17.3-7.8,17.3-17.3c0-0.9,0.7-1.6,1.6-1.6  
    c0.9,0,1.6,0.7,1.6,1.6c0,11.3-9.2,20.4-20.4,20.4c48.8,68.2,48.4,68.4,48.1,67.8L48.1,67.8z M48.1,67.8"/>  
</svg>
```

SVG is compressed XML

Color Models

There are many ways to describe color in web development.

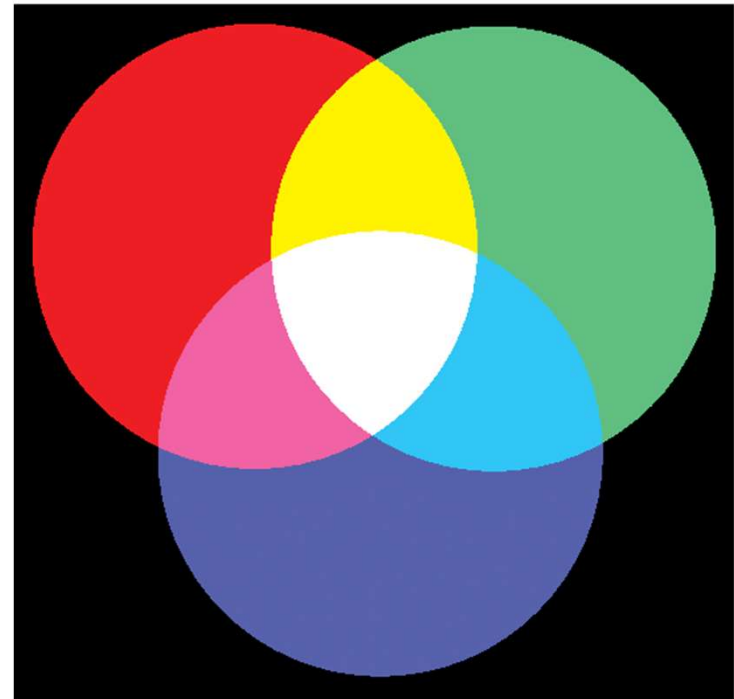
- Color Names
- RGB (Red Green Blue)
- HSL (Hue-Saturation-Lightness)

RGB Color Model

The **RGB color model** (Red Green Blue) relies on the fact that human visible color spectrum can be displayed using a combination of red, green, and blue lights

Each pixel is composed of tiny red, green, and blue subpixels.

Because the RGB colors combine to create white, they are also called **additive colors**.



A

Display Resolution

The **display resolution** refers to how many pixels a device can display.

Some common display resolutions include 1920×1600 px, 1280×1024 px, 1024×768 px, and 320×480 px

A web page will appear smaller on a high-resolution system (and larger on a low-resolution system)

