HTML 1: Overview

Chapter 2

Objectives

1. HTML Defined and its History
2. HTML Syntax
3. Semantic Markup
4. Structure of HTML
5. Quick Tour of HTML
6. HTML Semantic Elements

Brief History of HTML

Did we mention that this will be brief?

- ARPANET of the late 1960s
- jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**) in 1997.

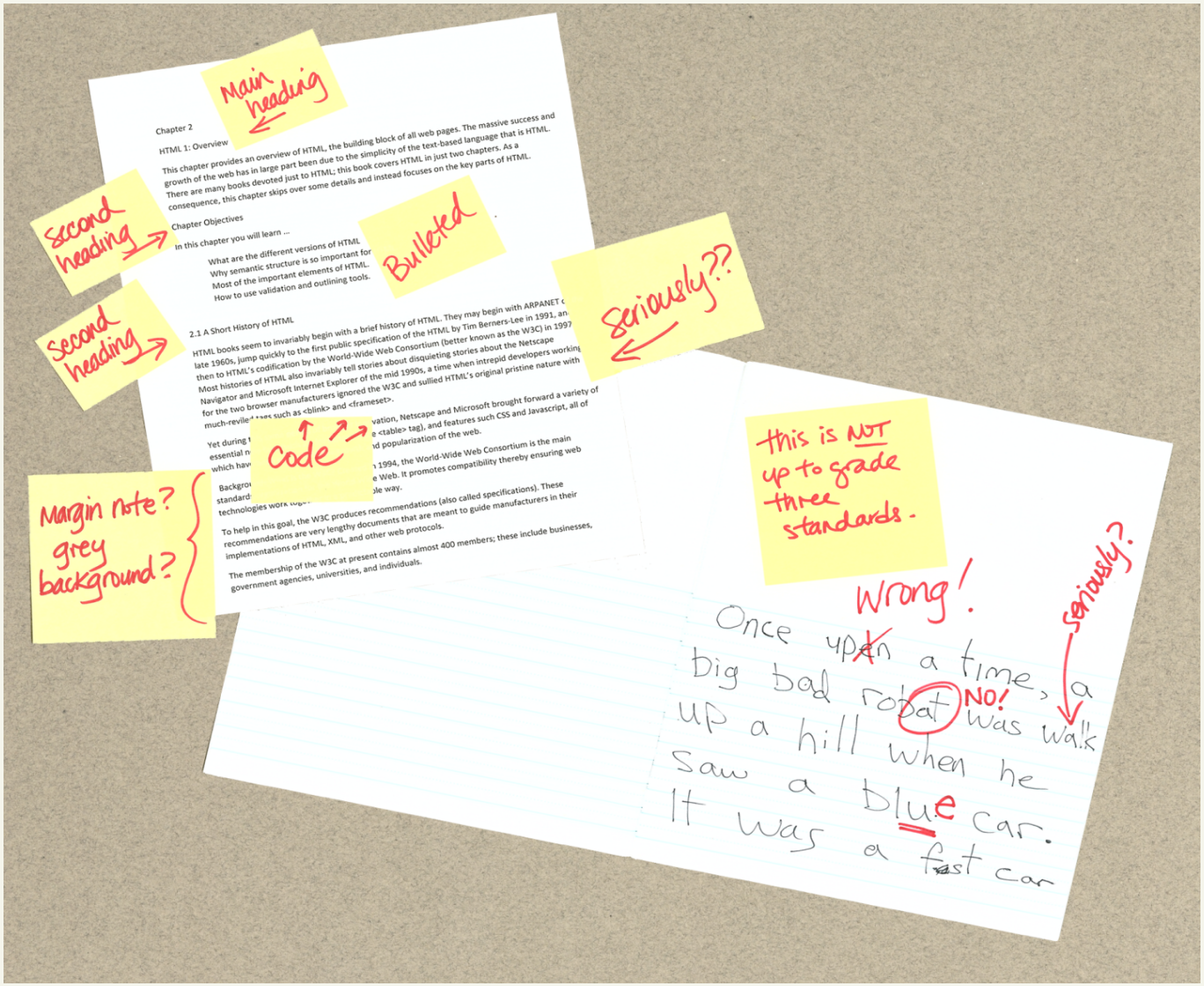
HTML Syntax

What is a markup language?

HTML is defined as a **markup language**.

- A markup language is simply a way of annotating a document in such a way to make the annotations distinct from the text being annotated.
- At its simplest, **markup** is a way to indicate *information about the content*
- This “information about content” in HTML is implemented via textual **tags** (aka elements).

Sample ad hoc markup



Markup

What is it again?

At its simplest, **markup** is a way to indicate *information about the content*

- This “information about content” in HTML is implemented via **tags** (aka elements).
- The markup in the previous slide consists of the red text and the various circles and arrows on the one page, and the little yellow sticky notes on the other.
- HTML does the same thing but uses textual tags.

What is the W3C?

Standards

The W3C is the main standards organization for the World Wide Web.

To promotes compatibility the W3C produces **recommendations** (also called **specifications**).

In 1998, the W3C turned its attention to a new specification called XHTML 1.0, which was a version of HTML that used stricter **XML** (Extensible Markup Language) syntax rules.

XML Overview

XML is a markup language, but unlike HTML, XML can be used to mark up any type of data.

One of the key benefits of XML data is that as plain text, it can be read and transferred between applications and different operating systems as well as being human-readable and understandable as well.

XML is not only used on the web server and to communicate asynchronously with the browser, but is also used as a data interchange format for moving information between systems

Well Formed XML

For a document to be **well-formed XML**, it must follow the syntax rules for XML:

- Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML.
- Element names can't start with a number.
- There must be a single-root element. A **root element** is one that contains all the other elements; for instance, in an HTML document, the root element is <html>.
- All elements must have a closing element (or be self-closing).
- Elements must be properly nested.
- Elements can contain attributes.
- Attribute values must always be within quotes.
- Element and attribute names are case sensitive.

Well Formed XML

Sample Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
  <painting id="192">
    <title>The Kiss</title>
    <artist>
      <name>Klimt</name>
      <nationality>Austria</nationality>
    </artist>
    <year>1907</year>
    <medium>Oil and gold on canvas</medium>
  </painting>
  <painting id="139">
    <title>The Oath of the Horatii</title>
    <artist>
      <name>David</name>
      <nationality>France</nationality>
    </artist>
    <year>1784</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

LISTING 17.1 Sample XML document

Valid XML

Requires a DTD

A **valid XML** document is one that is well formed and whose element and content conform to the rules of either its document type definition (DTD) or its schema.

A DTD tells the XML parser which elements and attributes to expect in the document as well as the order and nesting of those elements.

A DTD can be defined within an XML document or within an external file.

Data Type Definition

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE art [
<!ELEMENT art (painting*)>
<!ELEMENT painting (title,artist,year,medium)>
<!ATTLIST painting id CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (name,nationality)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nationality (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT medium (#PCDATA)>
]>
<art>
...
</art>
```

LISTING 17.2 Example DTD

XHTML

Partying like it's 1999

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.

The main rules are:

- lowercase tag names,
- attributes always within quotes,
- and all elements must have a closing element (or be self-closing).

XHTML

Two versions

To help web authors, two versions of XHTML were created:

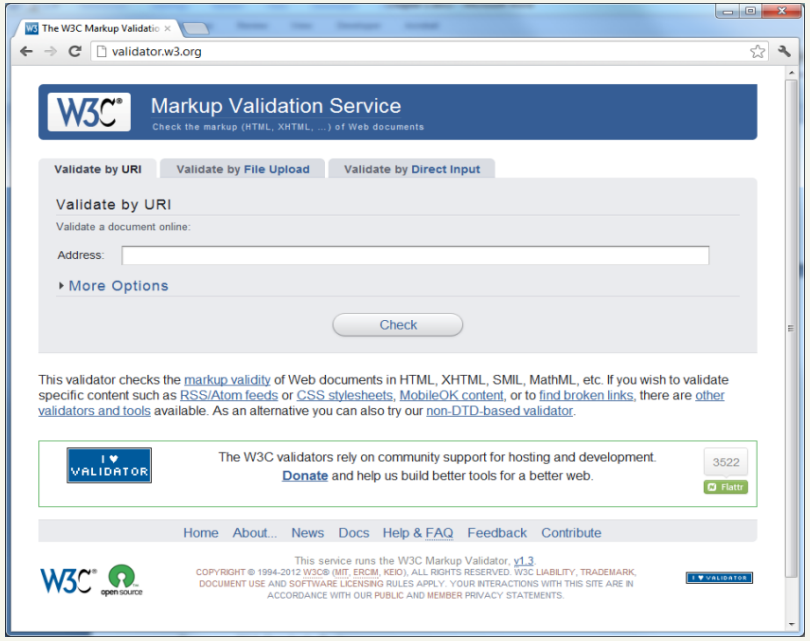
XHTML 1.0 Strict and **XHTML 1.0 Transitional**.

- The **strict** version was meant to be rendered by a browser using the strict syntax rules and tag support described by the W3C XHTML 1.0 Strict specification.
- The **transitional** recommendation is a more forgiving flavor of XHTML, and was meant to act as a temporary transition to the eventual global adoption of XHTML Strict.

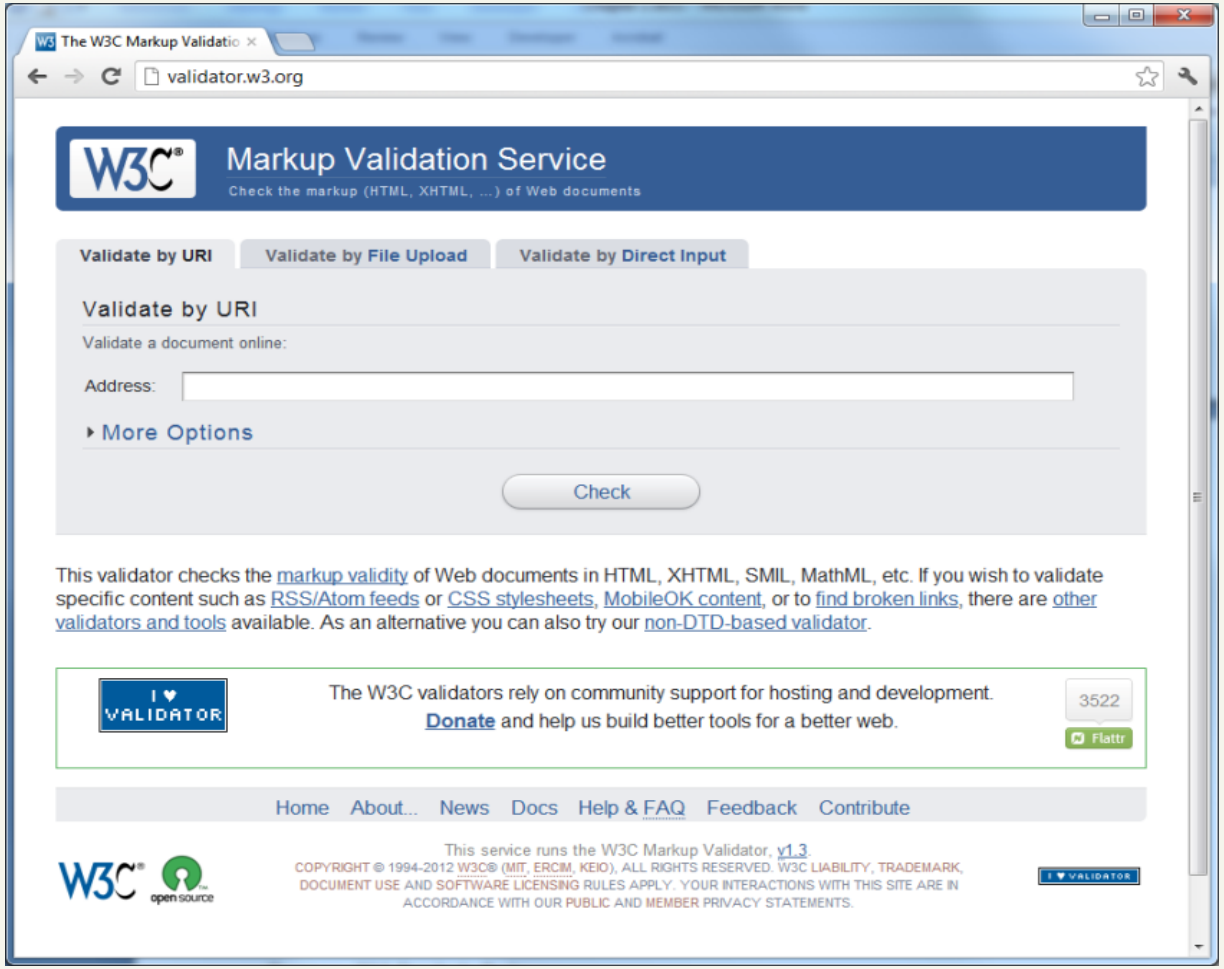
Validators

How to ensure your pages follow a standard

A key part of the standards movement in the web development community of the 2000s was the use of **HTML Validators** as a means of verifying that a web page's markup followed the rules for XHTML transitional or strict.



How about an example



Open a web browser to the W3C validator and find a few websites to test.

Type the URL into the bar, and you can check if the home page is valid against various standards (or auto-detect)

XHTML 2.0 and WHATWG

Where did it go?

In the mid 2000s, XHTML 2.0 proposed a revolutionary and substantial change to HTML.

- backwards compatibility with HTML and XHTML 1.0 was dropped.
- Browsers would become significantly less forgiving of invalid markup.

At around the same time, a group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within the W3C.

This group was not convinced that the W3C's embrace of XML and its abandonment of backwards-compatibility was the best way forward for the web.

HTML5

Three main aims

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:

- Specify unambiguously how browsers should deal with invalid markup.
- Provide an open, non-proprietary programming framework (via Javascript) for creating rich web applications.
- Be backwards compatible with the existing web.

HTML5

It evolves

While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5.

Certainly not all browsers and all versions support every feature of HTML5.

This is in fact by design. HTML in HTML5 is now a living language: that is, it is a language that evolves and develops over time.

As such, every browser will support a gradually increasing subset of HTML5 capabilities

Section 2 of 6

HTML SYNTAX

Elements and Attributes

More syntax

HTML documents are composed of textual content and HTML elements.

An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.

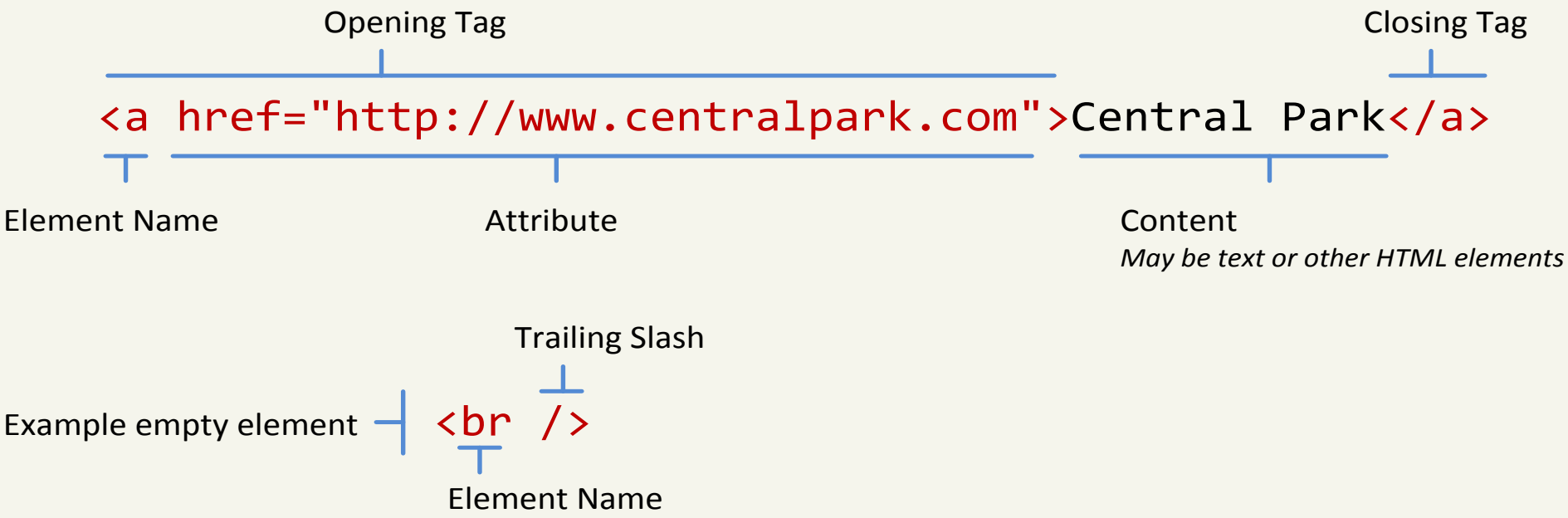
HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.

In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional.

What HTML lets you do

- Insert images using the **<img>** tag
- Create links with the **<a>** tag
- Create lists with the **<ul>, ** and **<li>** tags
- Create headings with **<H1>, <H2>, …, <H6>**
- Define metatdata with **<meta>** tag
- And much more…

Elements and Attributes



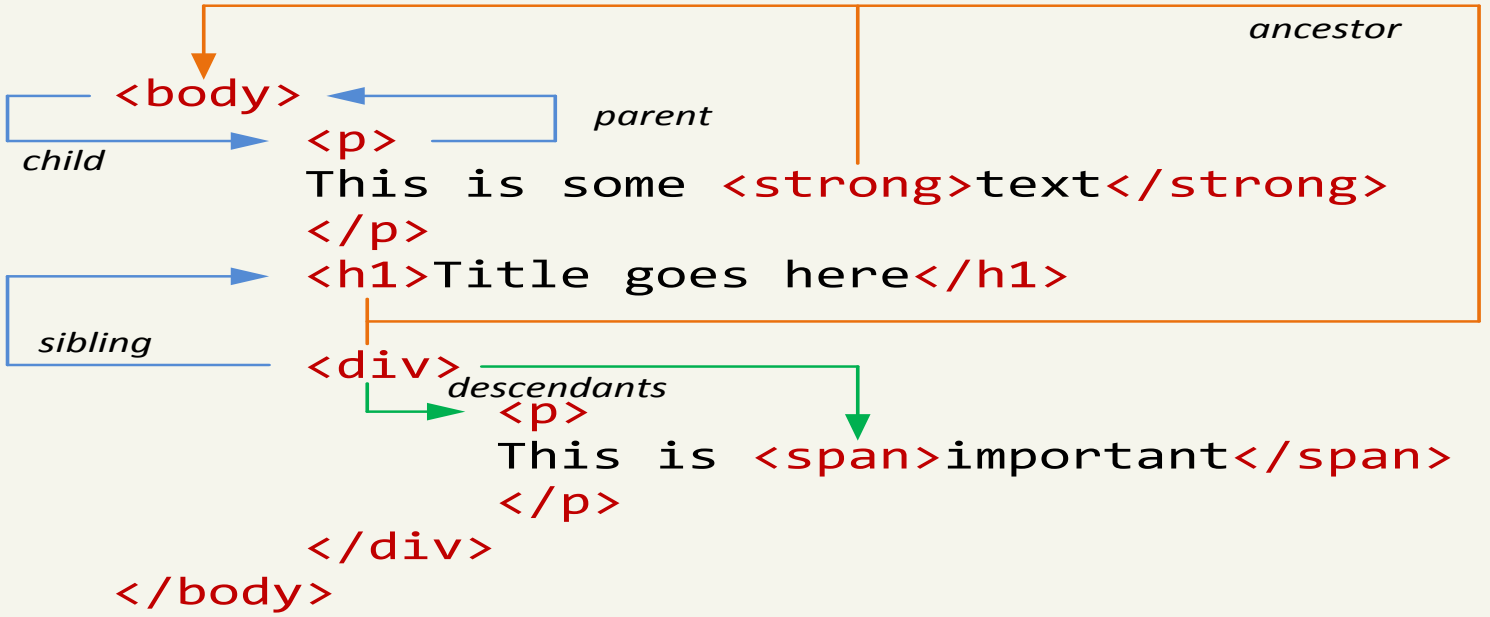
Nesting HTML elements

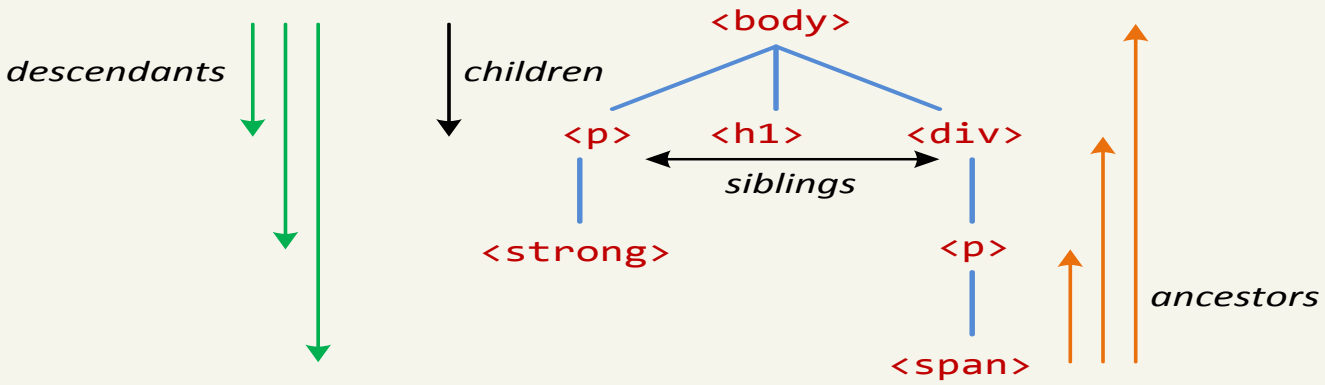
Often an HTML element will contain other HTML elements.

In such a 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 **descendents** of the parent element; likewise, any given child element, may have a variety of **ancestors.**

Hierarchy of elements



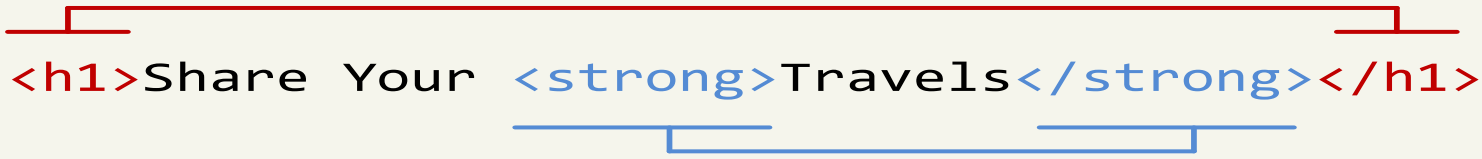


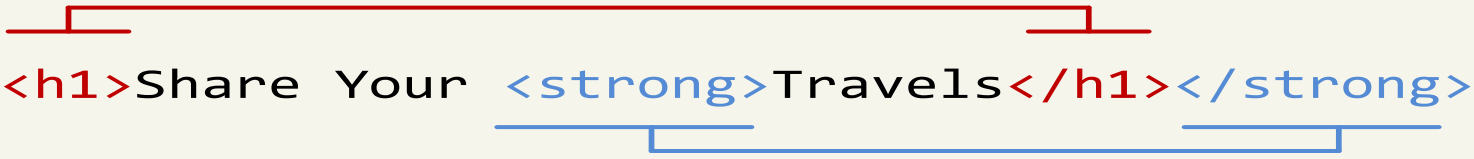
Nesting HTML elements

In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

That is, a child's ending tag must occur before its parent's ending tag.

Correct Nesting





Incorrect Nesting

Section 3 of 6

SEMANTIC MARKUP

Semantic Markup

What does it mean?

Over the past decade, a strong and broad consensus has grown around the belief that HTML documents should **only** focus on the structure of the document.

Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).

Semantic Markup

As a consequence, beginning HTML authors are often advised to create **semantic HTML** documents.

That is, an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.

Semantic Markup

Its advantages

Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

Maintainability. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup.

Faster. Semantic web pages are typically quicker to author and faster to download.

Accessibility. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.

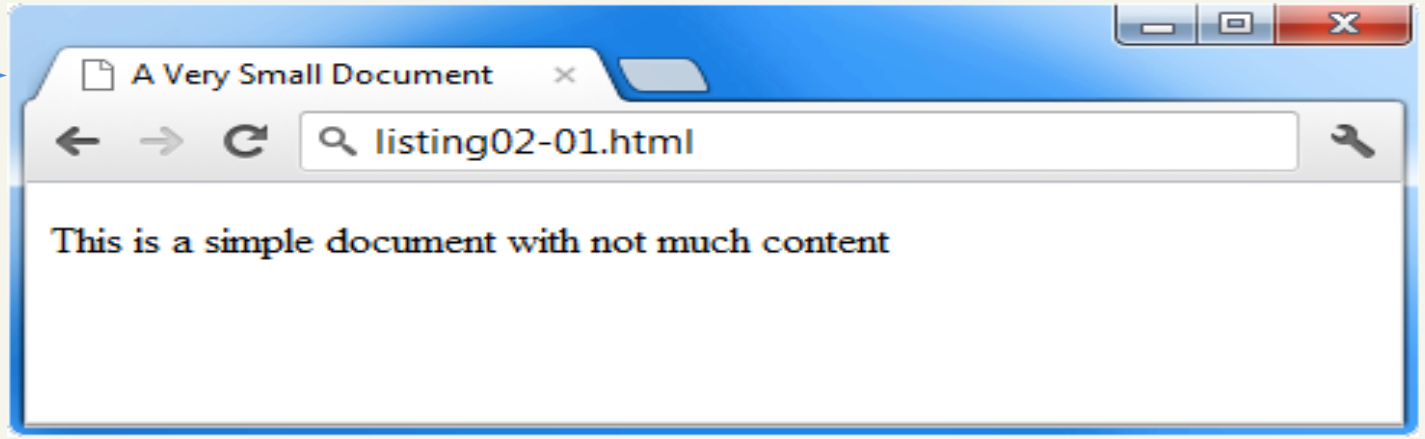
Search engine optimization. Semantic markup provides better instructions for search engines: it tells them what things are important content on the site.

Section 4 of 6

STRUCTURE OF HTML

Simplest HTML document

```
<!DOCTYPE html>
<title>A Very Small Document</title>
<p>This is a simple document with not much content</p>
```



The <title> element (Item 1) is used to provide a broad description of the content. The title is not displayed within the browser window. Instead, the title is typically displayed by the browser in its window and/or tab.

A more complete document

1 DOCTYPE

(short for **Document Type Definition**)

Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.

Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.

HTML, Head, and Body

HTML5 does not require the use of the <html>, <head>, and <body>.

However, in XHTML they were required, and most web authors continue to use them.

2 The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.

Head and Body

HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.

3. The head contains descriptive elements *about* the document

4. The body contains content that will be displayed by the browser.

Inside the head

There are no brains

You will notice that the <head> element contains a variety of additional elements.

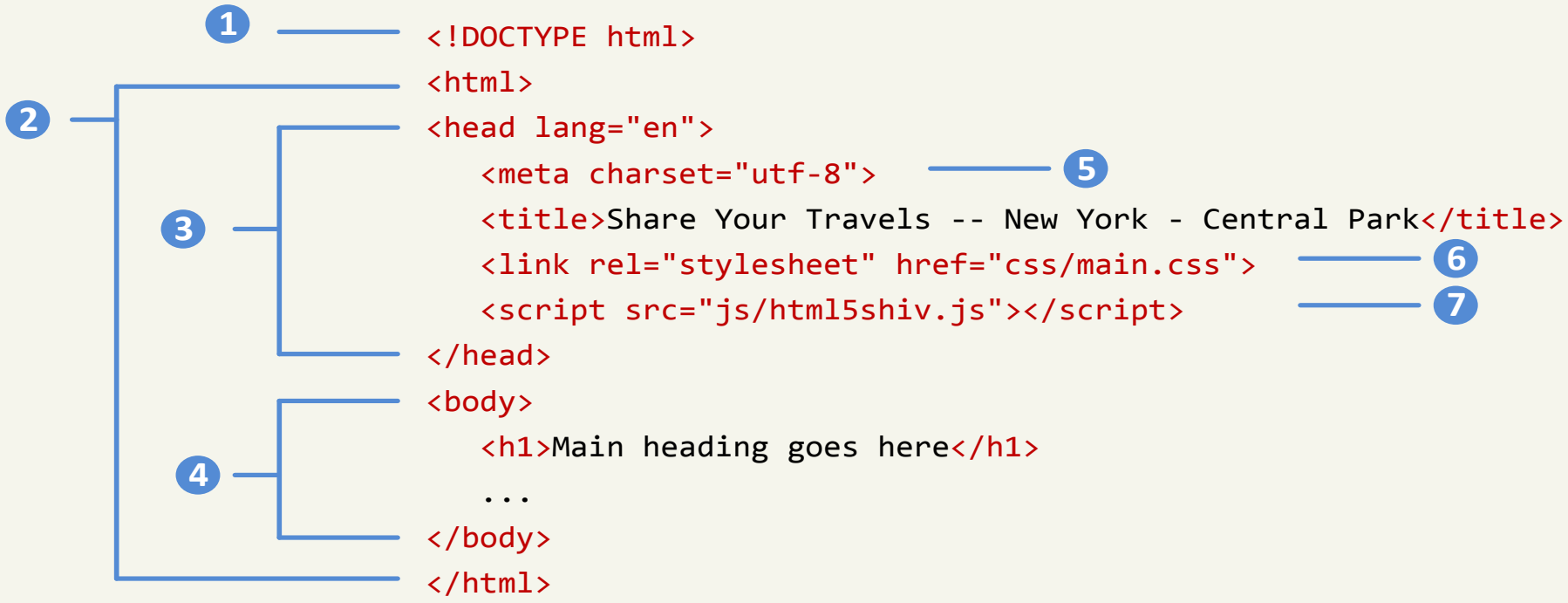
5. The first of these is the <meta> element. Our example declares that the character encoding for the document is UTF-8.

Inside the head

No brains but metas, styles and javascripts

6. Our example specifies an external CSS style sheet file that is used with this document.

7. It also references an external Javascript file.



Section 5 of 6

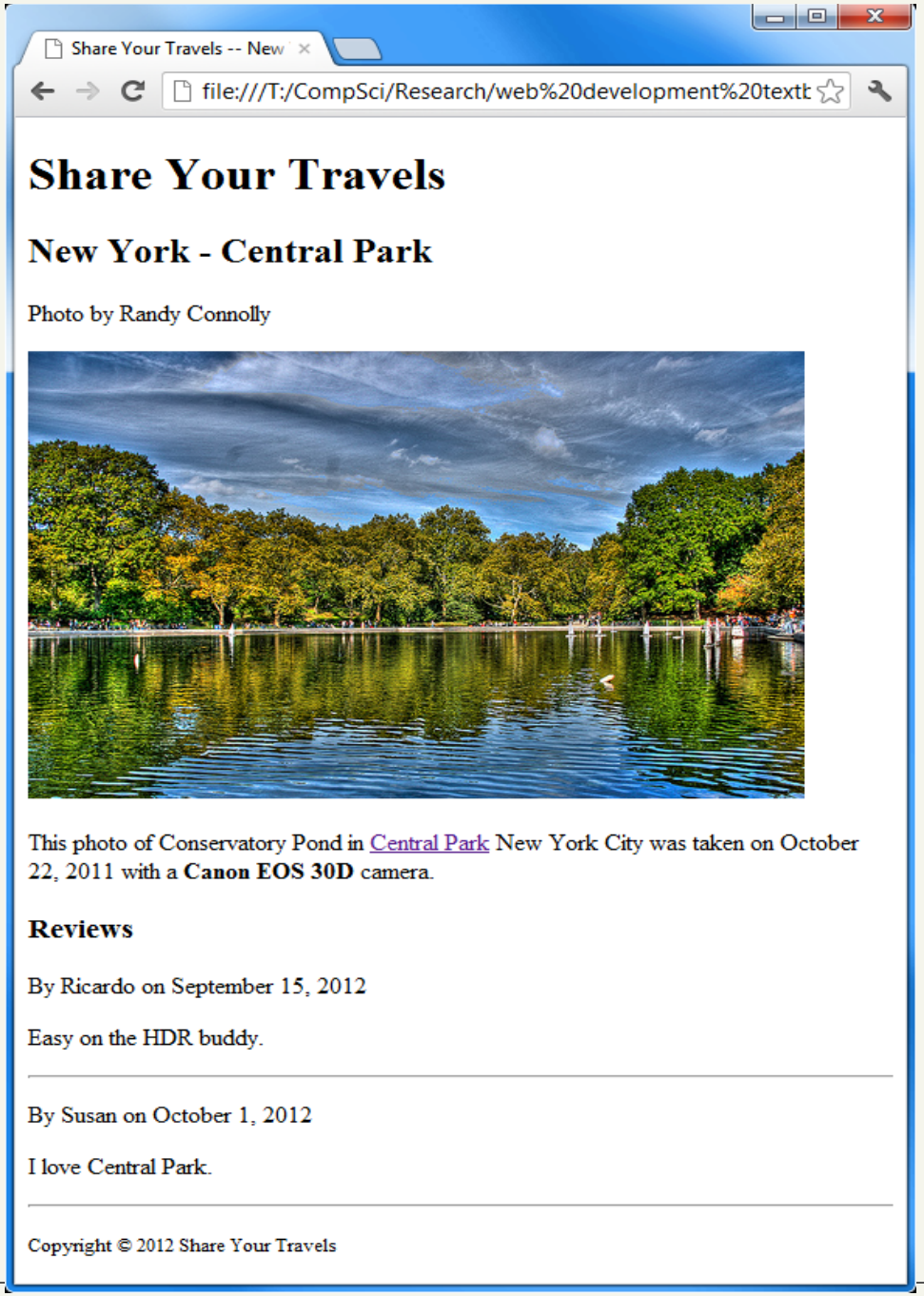
QUICK TOUR OF HTML

Why a quick tour?

HTML5 contains many structural and presentation elements – too many to completely cover in this presentation.

Rather than comprehensively cover all these elements, this presentation will provide a quick overview of the most common elements.

Sample Document



Sample Document

```
<body>
   <h1>Share Your Travels</h1>
   <h2>New York - Central Park</h2>
   <p>Photo by Randy Connolly</p>
   <p>This photo of Conservatory Pond in
      <a href="http://www.centralpark.com/">Central Park</a>
      New York City was taken on October 22, 2011 with a
      <strong>Canon EOS 30D</strong> camera.
   </p>
   <img src="images/central-park.jpg" alt="Central Park" />

   <h3>Reviews</h3>
   <div>
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>

   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>

   <p><small>Copyright &copy; 2012 Share Your Travels</small></p>
</body>
```

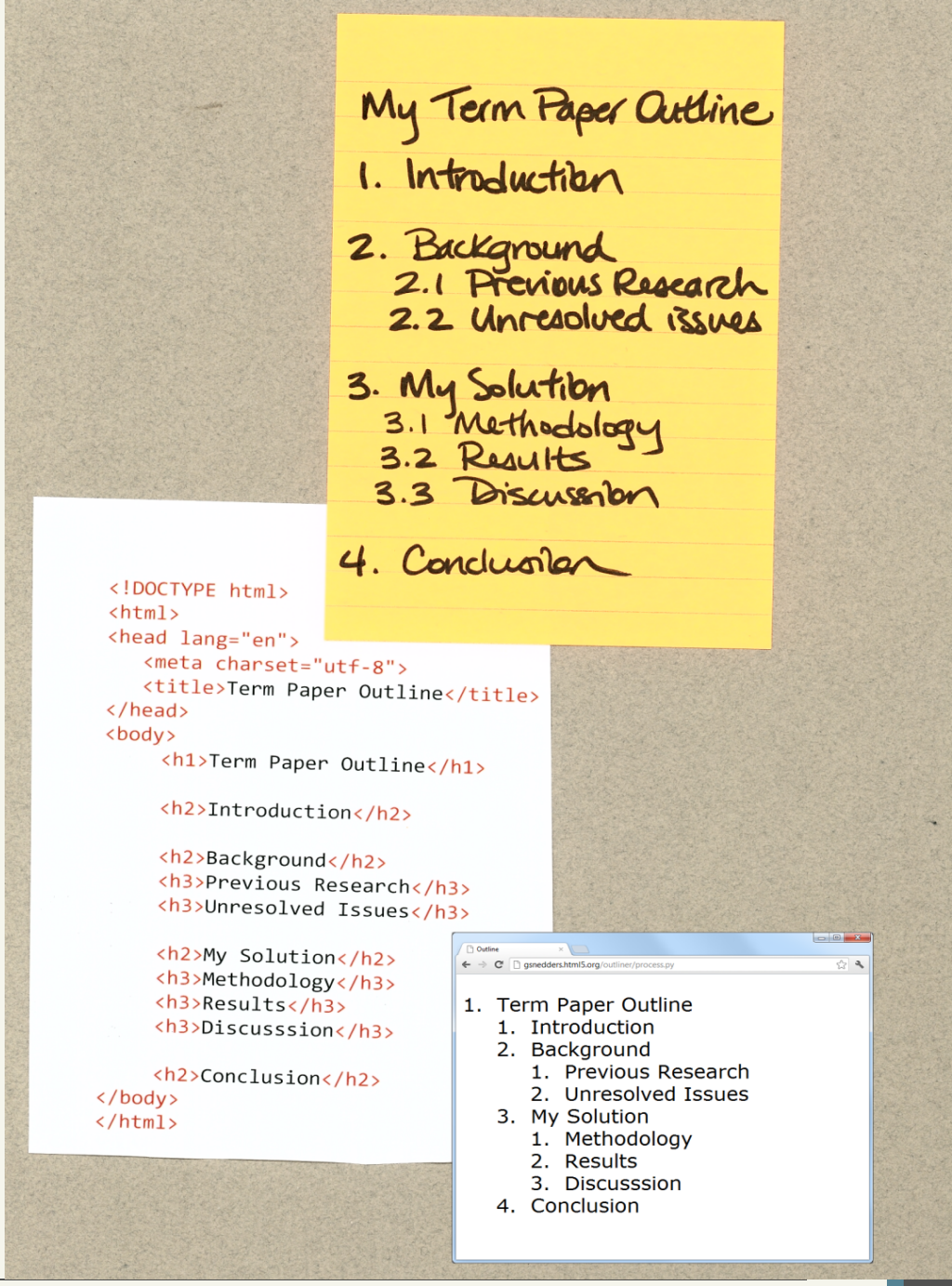
1 2 3 4 5 6 7 8 9

1 Headings

<h1>, <h2>, <h3>, etc

HTML provides six levels of heading (**h1, h2, h3**, …), with the higher heading number indicating a heading of less importance.

Headings are an essential way for document authors use to show their readers the structure of the document.



Headings

The browser has its own default styling for each heading level.

However, these are easily modified and customized via CSS.

Headings

Be semantically accurate

In practice, specify a heading level that is semantically accurate.

Do not choose a heading level because of its default presentation

- e.g., choosing <h3> because you want your text to be bold and 16pt

Rather, choose the heading level because it is appropriate

- e.g., choosing <h3> because it is a third level heading and not a primary or secondary heading

2 Paragraphs

<p>

Paragraphs are the most basic unit of text in an HTML document.

Notice that the **<p>** tag is a container and can contain HTML and other **inline HTML elements**

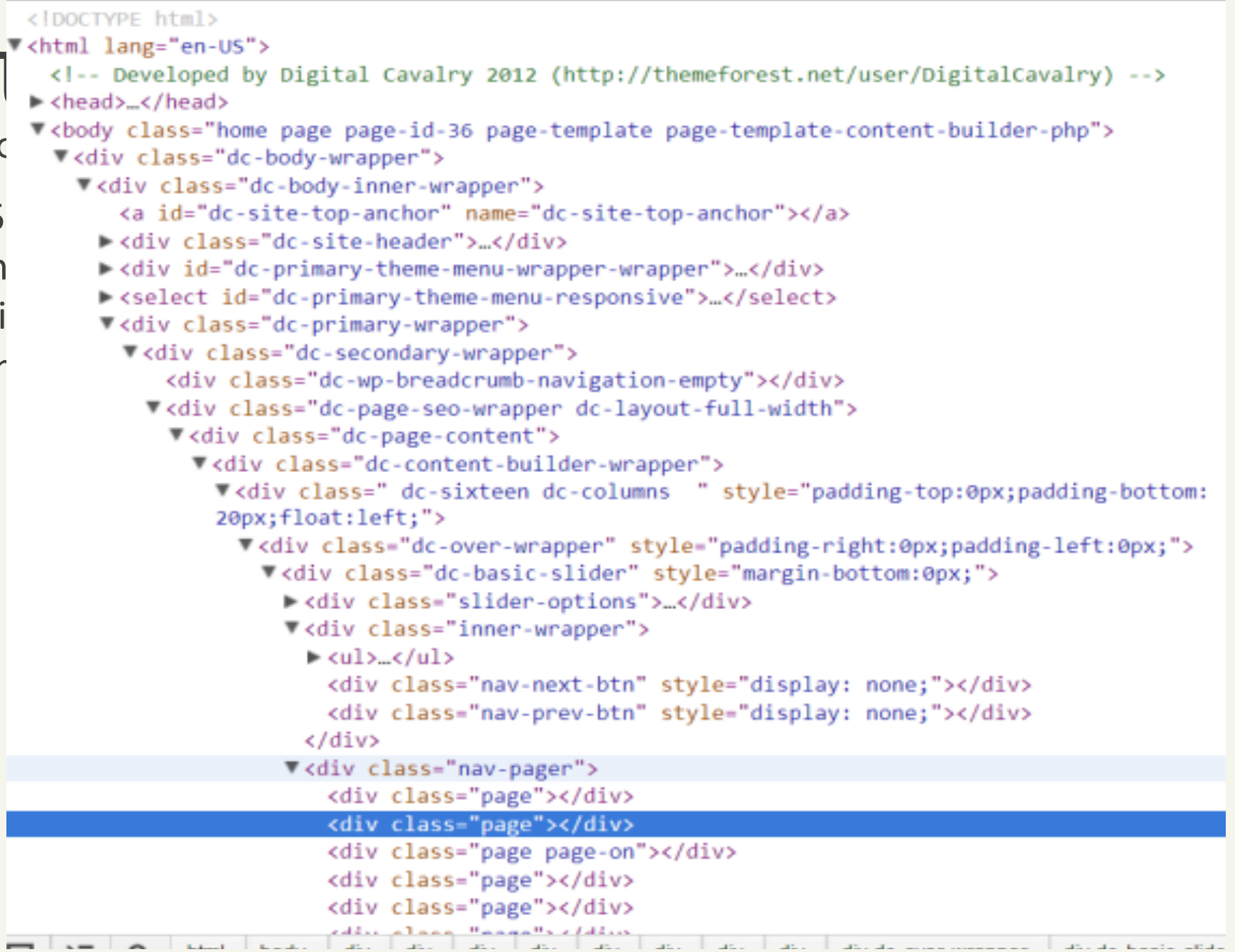
- **inline HTML elements** refers to HTML elements that do not cause a paragraph break but are part of the regular “flow” of the text.
- **Block HTML elements** build a block around themselves, causing a break in the flow oh the text (e.g., paragraph, table)
- Composition: Block elements can contain block and inline elements, while inline elements can contain only inline elements

6 Divisions

This ** ** tag is also a container element and is used to create a logical grouping of content

- The element has no intrinsic presentation.
- It is frequently used in contemporary CSS-based layouts to mark out sections.

HTML5
examin
confusi
contem



3 Links

<a>

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

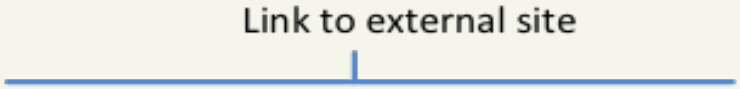
Label (text)

```
<a href="index.html"><img src="logo.gif" /></a>
```

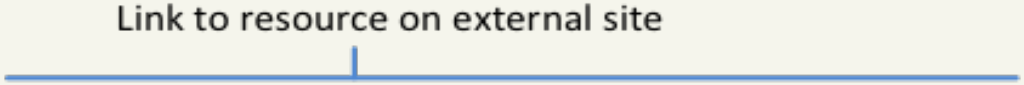
Label (image)

Different link destinations

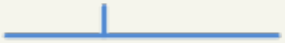
You can use the anchor element to create a wide range of links:




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




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




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


```
<a href="#top">Go to Top of Document</a>
```

Different link destinations

Link to specific place on another page

```
<a href="productX.html#reviews">Reviews for product X</a>
```

Link to email

```
<a href="mailto://person@somewhere.com">Someone</a>
```

Link to javascript function

```
<a href="javascript://OpenAnnoyingPopup();">See This</a>
```

Link to telephone (automatically dials the number when user clicks on it using a smartphone browser)

```
<a href="tel:+18009220579">Call toll free (800) 922-0579</a>
```

URL Absolute Referencing

For external resources

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

- the protocol (typically, http://),
- the domain name,
- any paths, and then finally
- the file name of the desired resource.

URL Relative Referencing

An essential skill

We also need to be able to successfully reference files within our site.

This requires learning the syntax for so-called **relative referencing**.

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

For these situations, a relative pathname (following UNIXi conventions) is required along with the filename.

Inline Text Elements

Do not disrupt the flow

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

HTML5 defines over 30 of these elements.

e.g., <a>,
, ,

Images

While the **<img>** tag is the oldest method for displaying an image, it is not the only way.

For purely decorative images, such as background gradients and patterns, logos, border art, and so on, it makes semantic sense to keep such images out of the markup and in CSS where they more rightly belong.

But when the images are content, such as in the images in a gallery or the image of a product in a product details page, then the <img> tag is the semantically appropriate approach.

Images

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

Text in title attribute will be displayed in a popup tool tip when user moves mouse over image.

```
<img src="images/central-park.jpg" alt="Central Park"  title="Central Park" width="80" height="40" />
```

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.

Lists

HTML provides three types of lists

**Unordered lists **. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

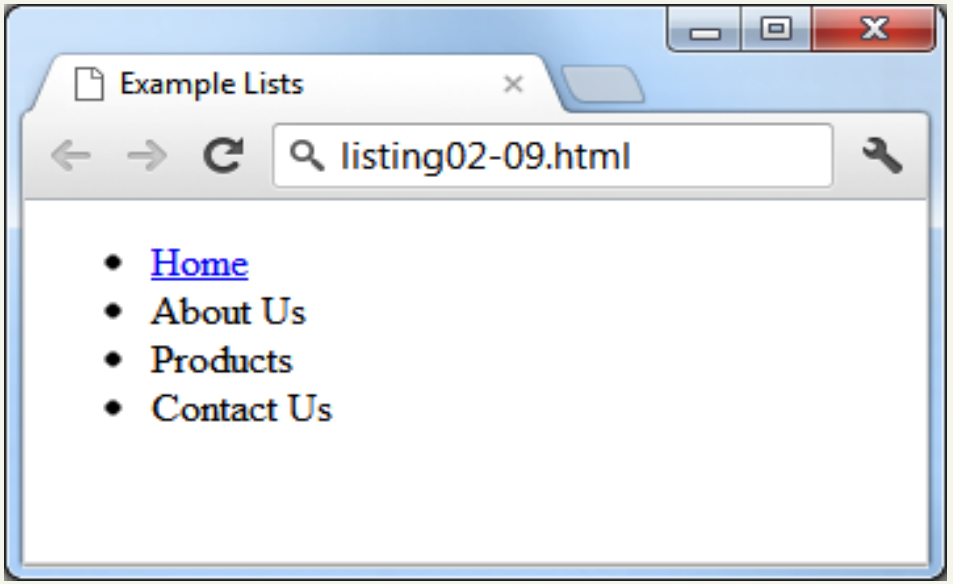
**Ordered lists **. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

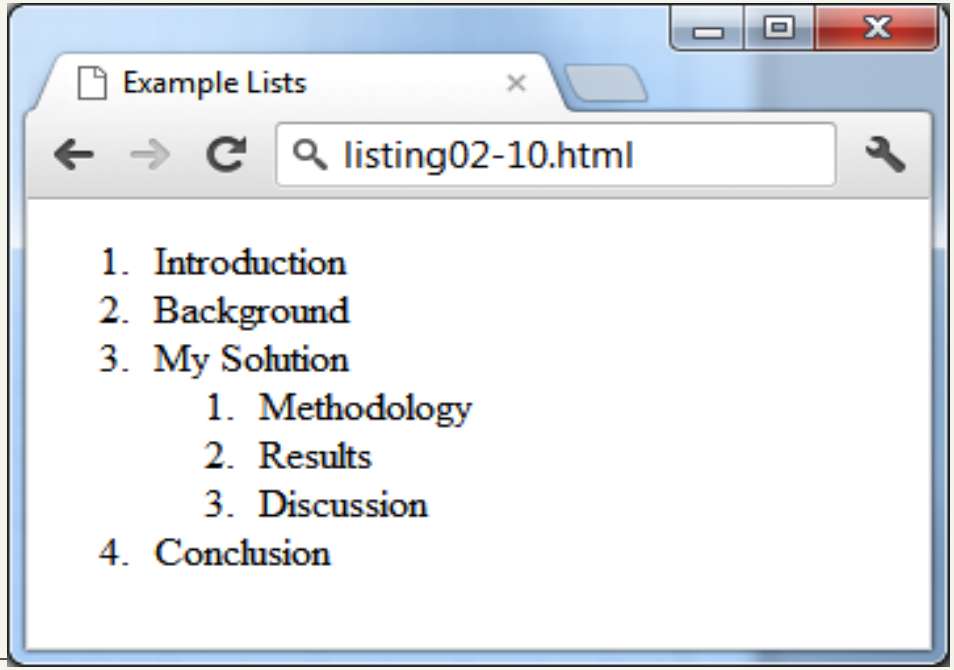
Definition lists <dt>. Collection of name <dt> and definition <dd> pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

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>
```





Character Entities

These are special characters for symbols for which there is either no way easy way to type in via a keyboard (such as the copyright symbol or accented characters) 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.

e.g.,   and ©

Section 6 of 6

HTML SEMANTIC ELEMENTS

HTML5 Semantic Elements

Why are they needed?

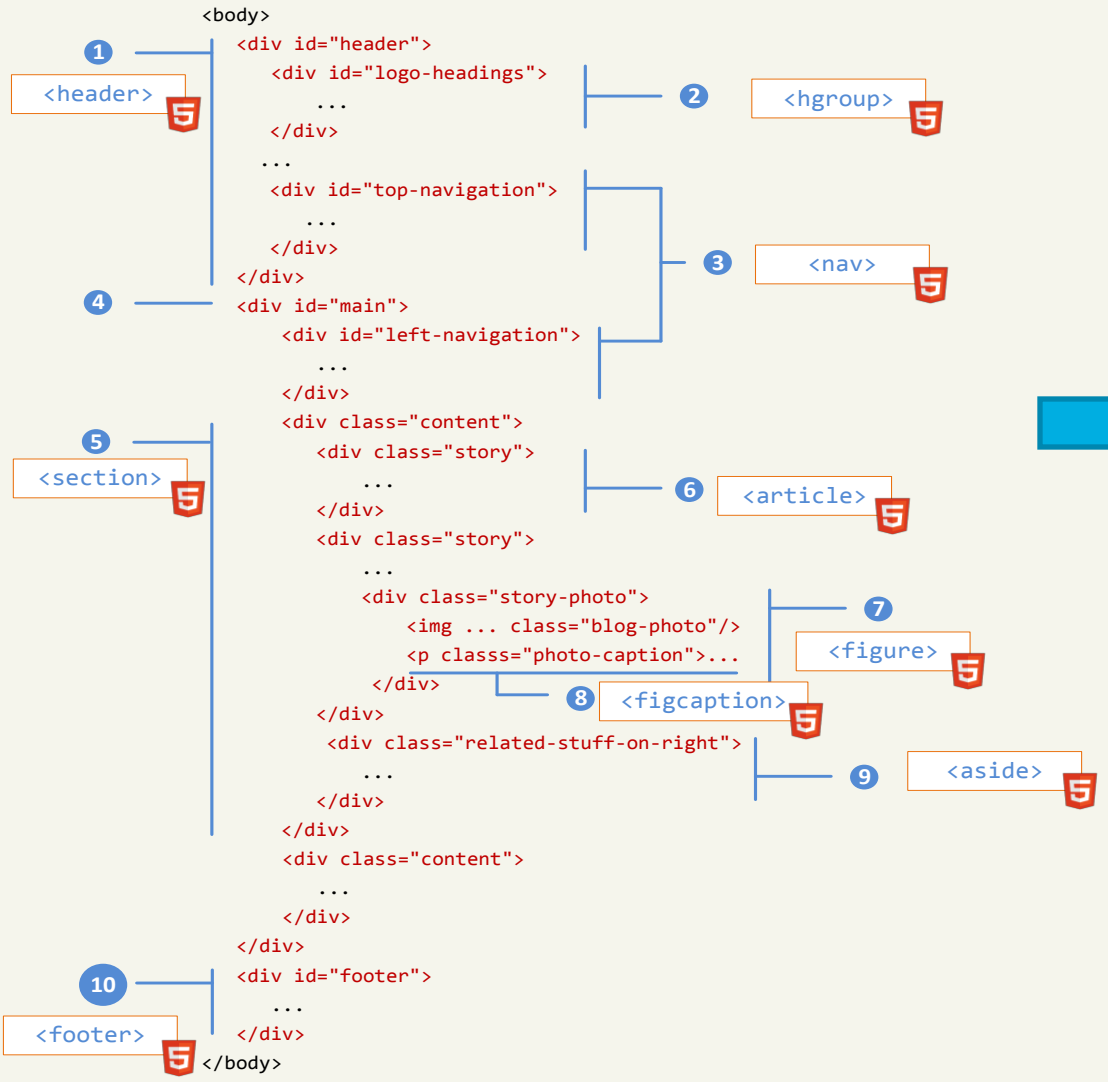
One substantial problem with modern, pre-HTML5 semantic markup:

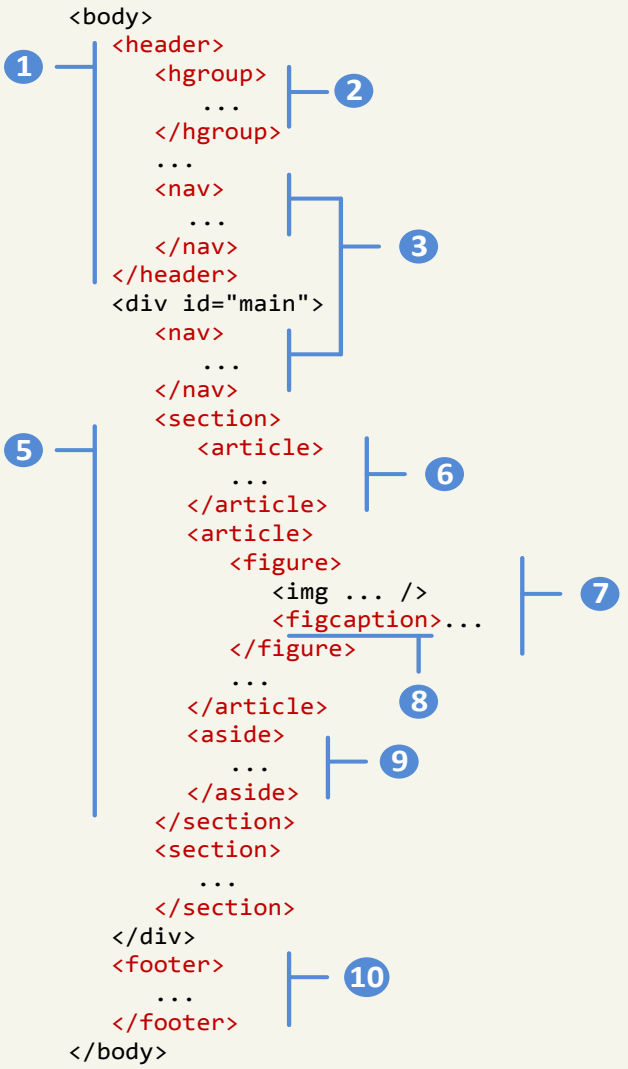
> most complex web sites are absolutely packed solid with elements.

Unfortunately, all these elements can make the resulting markup confusing and hard to modify.

Developers typically try to bring some sense and order to the chaos by using id or class names that provide some clue as to their meaning.

XHTML versus HTML5





Header and Footer

<header> <footer>

Most web site pages have a recognizable header and footer section.

Typically the **header** contains

- the site logo
- title (and perhaps additional subtitles or taglines)
- horizontal navigation links, and
- perhaps one or two horizontal banners.

Header and Footer

<header> <footer>

The typical footer contains less important material, such as

- smaller text versions of the navigation,
- copyright notices,
- information about the site's privacy policy, and
- perhaps twitter feeds or links to other social sites.

Header and Footer

Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as <article> or <section>.

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Introduction to HTML5</h1>
   ...
</header>
<article>
   <header>
       <h2>HTML5 Semantic Structure Elements </h2>
      <p>By <em>Mike Fennelly</em></p>
      <p><time>September 30, 2012</time></p>
   </header>
   ...
</article>
```

3 Navigation

<nav>

The **<nav>** element represents a section of a page that contains links to other pages or to other parts within the same page.

Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the <nav> element.

The <nav> element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems.

Navigation

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Fundamentals of Web Development</h1>
   <nav role="navigation">
      <ul>
         <li><a href="index.html">Home</a></li>
         <li><a href="about.html">About Us</a></li>
         <li><a href="browse.html">Browse</a></li>
      </ul>
   </nav>
</header>
```

Articles and Sections

<article> <section>

The **<article>** element represents a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.

The **<section>** element represents a section of a document, typically with a title or heading.

Articles and Sections

According to the W3C, **<section>** is a much broader element, while the **<article>** element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page.

Sections versus Divs

How to decide which to use

The WHATWG specification warns readers that the <section> element is **not** a generic container element. HTML already has the element for such uses.

When an element is needed only for styling purposes or as a convenience for scripting, it makes sense to use the element instead.

Another way to help you decide whether or not to use the <section> element is to ask yourself if it is appropriate for the element's contents to be listed explicitly in the document's outline.

If so, then use a <section>; otherwise use a .

Figure and Figure Captions

<figure> <figcaption>

The W3C Recommendation indicates that 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 and Figure Captions

Note however …

The **<figure>** element should **not** be used to wrap every image.

For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within <figure> elements.

Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.

Figure and Figure Captions

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>
  <img src="images/central-park.jpg" alt="Central Park" /><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>
```

9

Aside

<aside>

The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

But while the **<figure>** element was used to indicate important information whose location on the page is somewhat unimportant, the **<aside>** element “represents a section of a page that consists of content that is tangentially related to the content around the aside element.”

The **<aside>** element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.