HTML 1: Overview

Chapter 2

Randy Connolly and Ricardo Hoar

Fundamentals of Web Development

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

Section 1 of 6

HTML DEFINED + ITS HISTORY

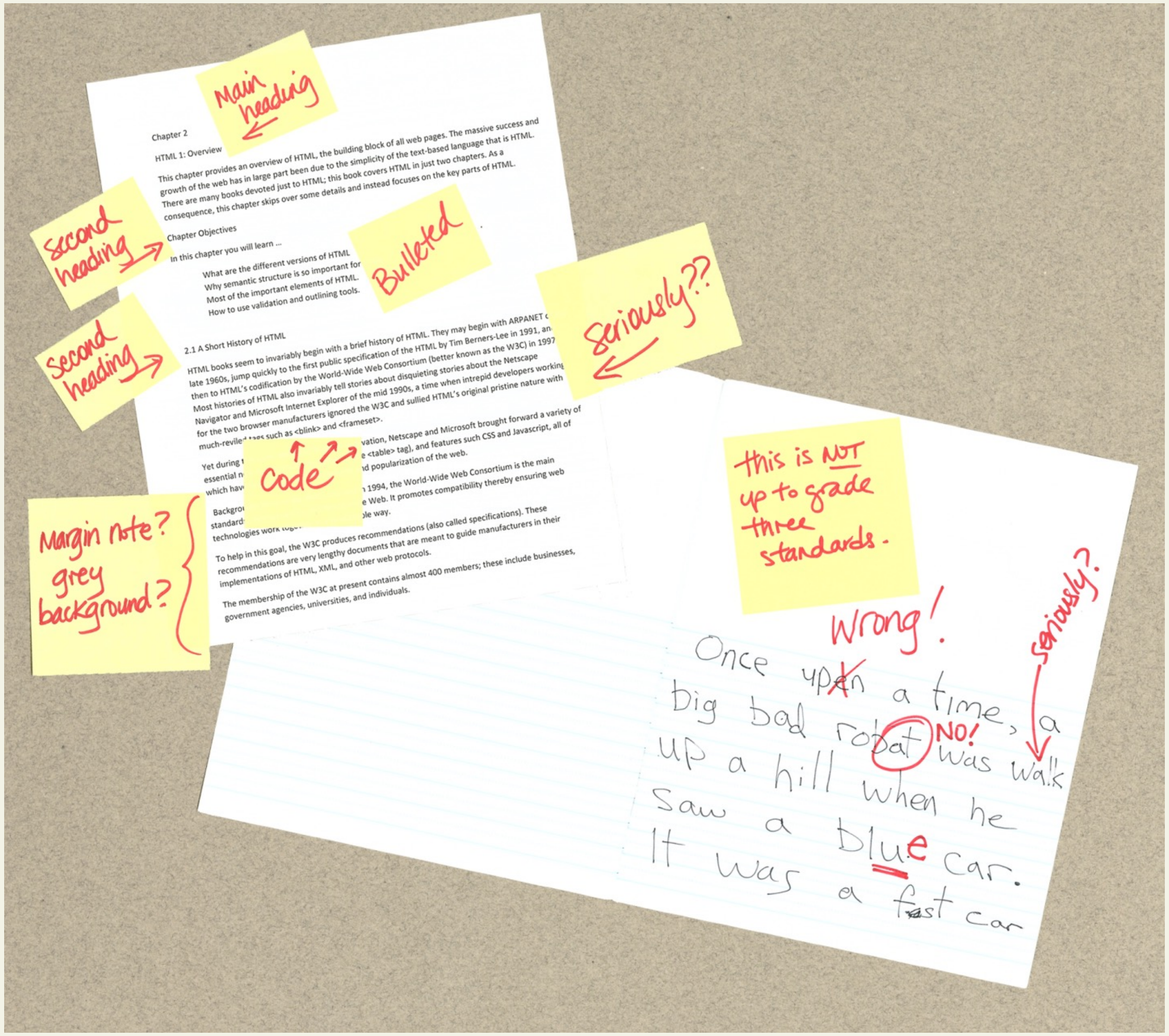
HTML Syntax

What is a markup language?

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

- markup is a way of indicate information about the content.
- annotations distinct from the text being annotated.
- The term comes from the days of print, when editors would write instructions on manuscript pages that might be revision instructions to the author or copy editor.

Sample ad hoc markup



Chapter 2

HTML 1: Overview

This chapter provides an overview of HTML, the building block of all web pages. The massive success and growth of the web has in large part been due to the simplicity of the text-based language that is HTML. There are many books devoted just to HTML; this book covers HTML in just two chapters. As a consequence, this chapter skips over some details and instead focuses on the key parts of HTML.

Chapter Objectives

In this chapter you will learn ...

What are the different versions of HTML
Why semantic structure is so important for
Most of the important elements of HTML.
How to use validation and outlining tools.

2.1 A Short History of HTML

HTML books seem to invariably begin with a brief history of HTML. They may begin with ARPANET c late 1960s, jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991, an then to HTML's codification by the World-Wide Web Consortium (better known as the W3C) in 1997 Most histories of HTML also invariably tell stories about disquieting stories about the Netscape Navigator and Microsoft Internet Explorer of the mid 1990s, a time when intrepid developers working for the two browser manufacturers ignored the W3C and sullied HTML's original pristine nature with much-revile tags such as <blink> and <frameset>.

Yet during vation, Netscape and Microsoft brought forward a variety of essential n e <table> tag), and features such CSS and Javascript, all of which hav d popularization of the web.

Backgro 1994, the World-Wide Web Consortium is the main standard e Web. It promotes compatibility thereby ensuring web technologies work ble way.

To help in this goal, the W3C produces recommendations (also called specifications). These recommendations are very lengthy documents that are meant to guide manufacturers in their implementations of HTML, XML, and other web protocols.

The membership of the W3C at present contains almost 400 members; these include businesses, government agencies, universities, and individuals.

Once up~~e~~n a time, a big bad robat was walk up a hill when he saw a blue car. It was a fast car

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.

XHTML

You too can be strict

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 XML-based syntax rules for XHTML are pretty easy to follow.

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**.

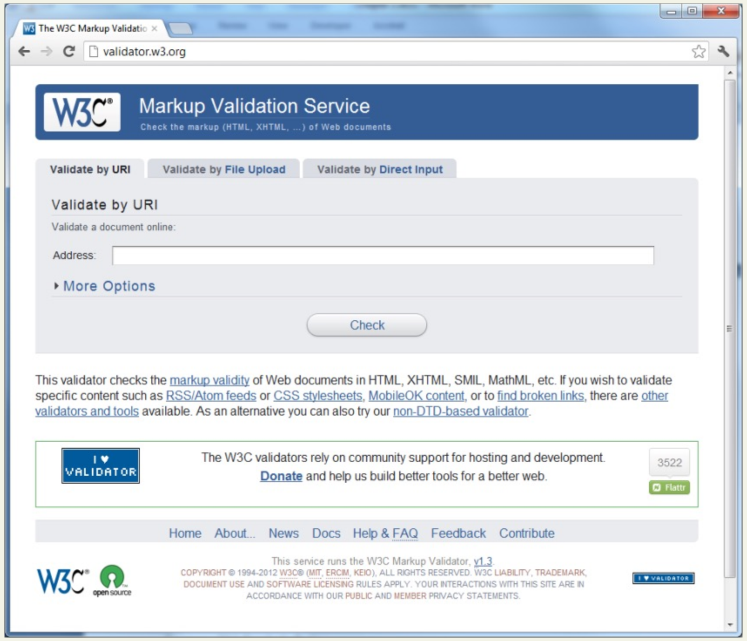
- The **strict** version was meant to be rendered by a browser using the strict syntax rules and tag support described by the W3C XHTML 1.0 Strict specification.
- The **transitional** recommendation is a more forgiving flavor of XHTML, and was meant to act as a temporary transition to the eventual global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

During much of the 2000s, the focus in the professional web development community was on standards: that is, on limiting oneself to the W3C specification for XHTML.

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.


The W3C Markup Validatio ×
validator.w3.org
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 URI
Validate a document online:
Address:
More Options
Check
I ♥ VALIDATOR
3522
Flattr
Home About... News Docs Help & FAQ Feedback Contribute
open source

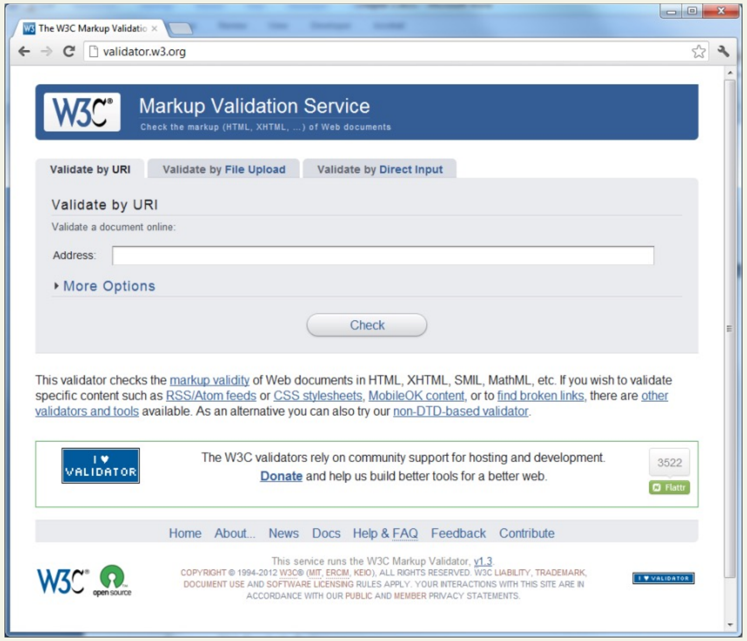
This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

The W3C validators rely on community support for hosting and development.
Donate and help us build better tools for a better web.

This service runs the W3C Markup Validator, v1.3.
COPYRIGHT © 1994-2012 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

How about an example

Only if you have an internet connection



Validate by URI

Validate a document online:

Address:

This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

The W3C validators rely on community support for hosting and development.
Donate and help us build better tools for a better web.

This service runs the W3C Markup Validator, v1.3.
COPYRIGHT © 1994-2012 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

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

All of the major browser manufacturers have 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

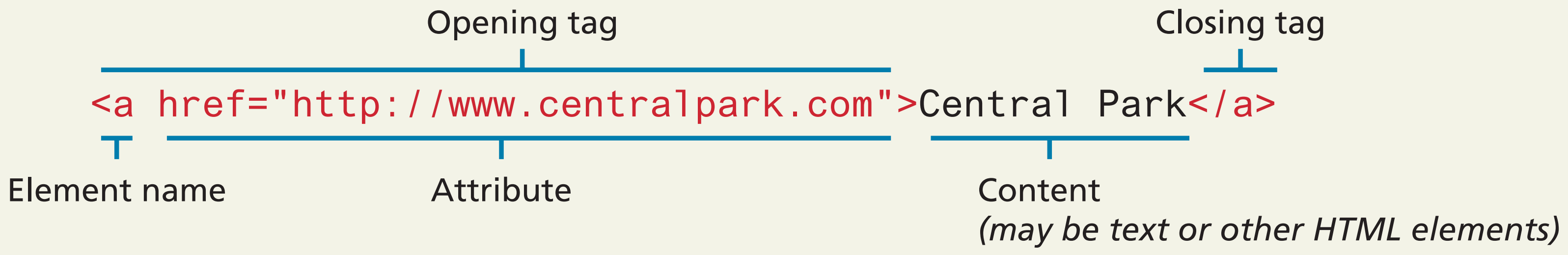
Section 2 of 6

HTML SYNTAX

HTML Syntax

Elements and Attributes

- **HTML documents** are composed of textual content and **HTML elements**
- **HTML element** encompasses
  - the **element name** within angle brackets (i.e., the **tag**) and
  - HTML elements can also contain **attributes**.
  - **the content** within the tag.

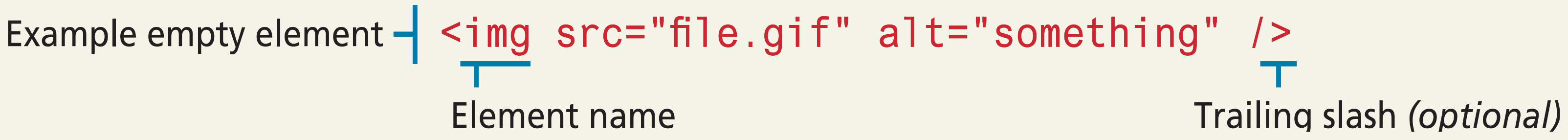

Opening tag
Closing tag
<a href="http://www.centralpark.com">Central Park</a>
Element name
Attribute
Content
(may be text or other HTML elements)

HTML Syntax

Elements and Attributes

An **empty element** does not contain any text content; instead, it is an instruction to the browser to do something.

- In XHTML, empty elements had to be terminated by a trailing slash.
- In HTML5, the trailing slash in empty elements is optional.



```
<img src="file.gif" alt="something" />
```

Nesting HTML elements

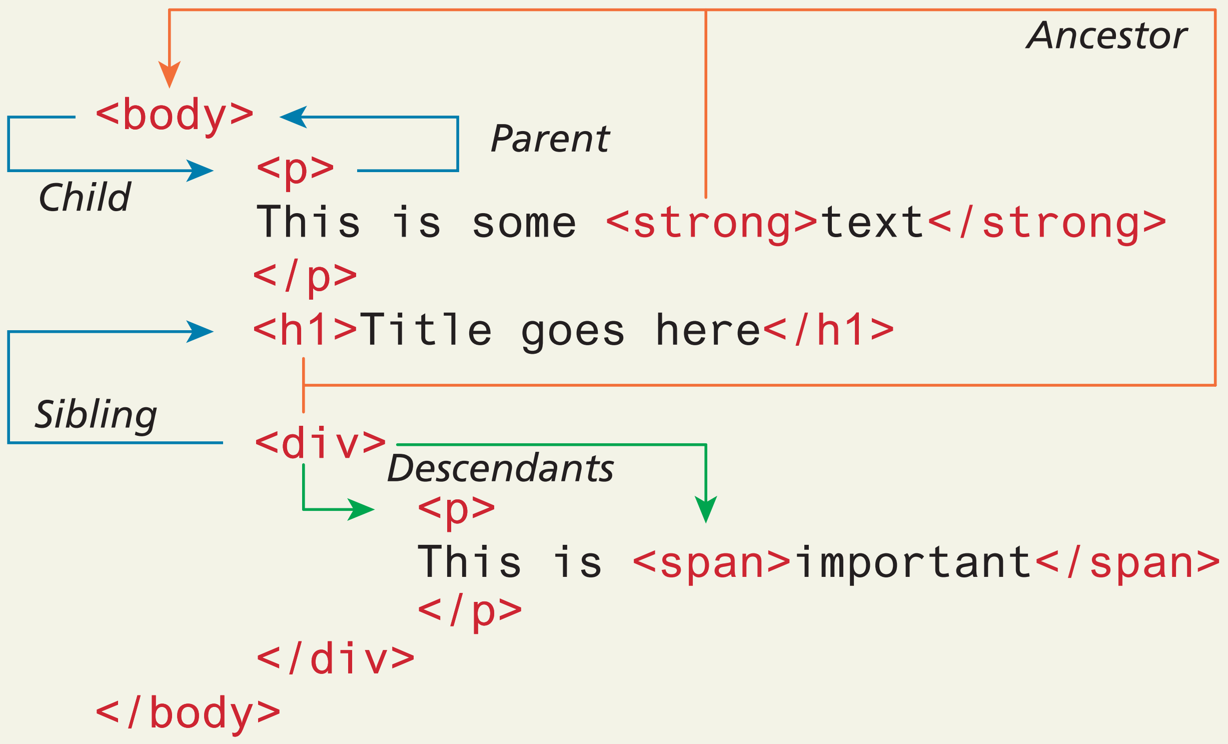
Often an HTML element will contain other HTML elements.

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.**

HTML Syntax

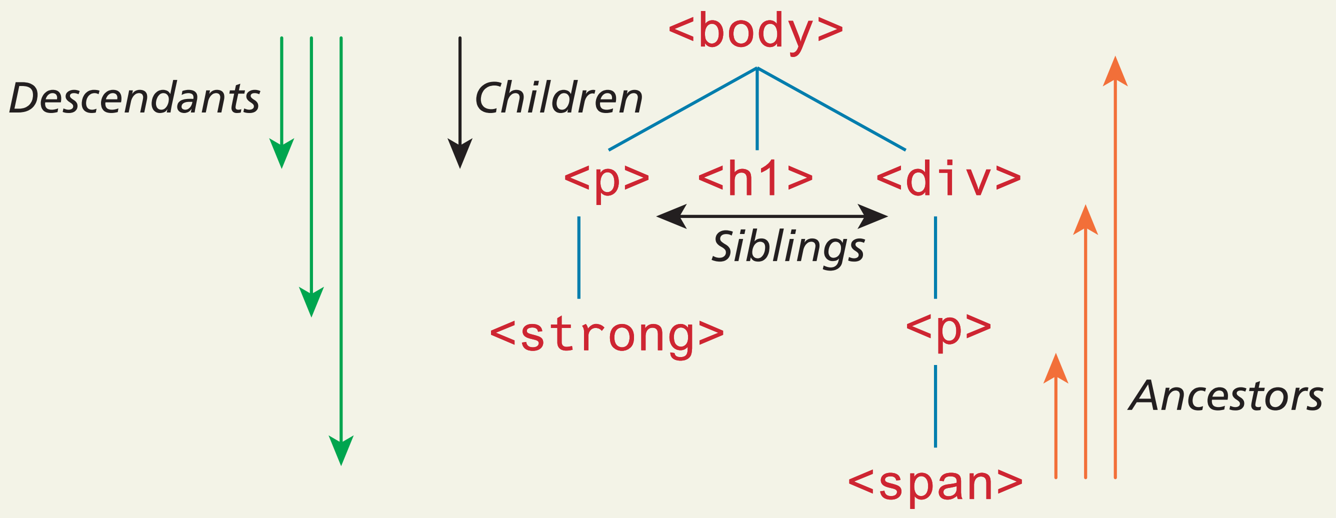
Nesting HTML Elements



```
<body>
    <p>
    This is some <strong>text</strong>
    </p>
    <h1>Title goes here</h1>
    <div>
          <p>
          This is <span>important</span>
          </p>
    </div>
</body>
```

HTML Syntax

Nesting HTML Elements

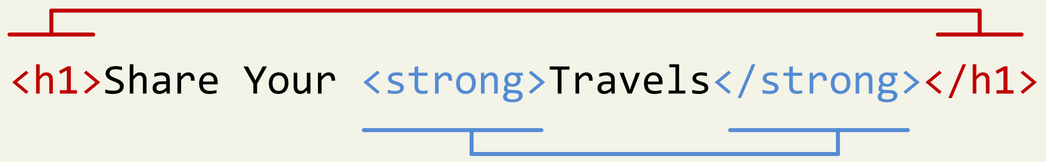


Nesting HTML elements

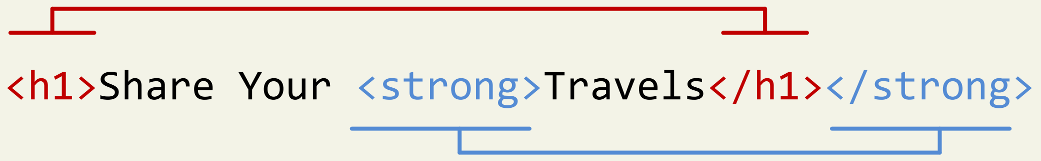
In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

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

Correct Nesting



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



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

Incorrect Nesting

Section 3 of 6

SEMANTIC MARKUP

Semantic Markup

What does it mean?

Over the past decades, 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).

As a consequence, beginning HTML authors are often counseled 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.

Structure

Structure is a vital way of communicating information in paper and electronic documents.

All of the tags that we will examine in this presentation are used to describe the basic structural information in a document, such as articles, headings, lists, paragraphs, links, images, navigation, footers, and so on.

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.

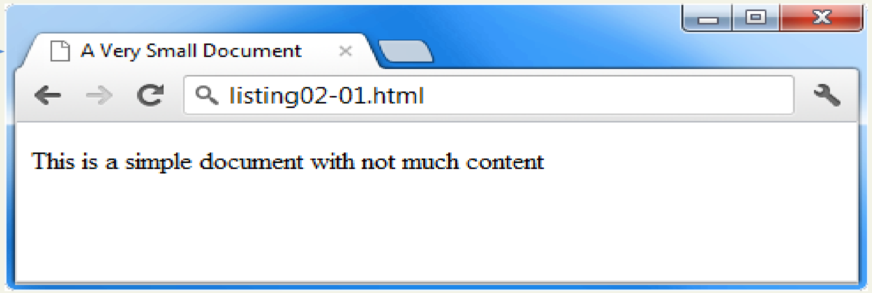
Section 4 of 6

STRUCTURE OF HTML

Simplest HTML document

1

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

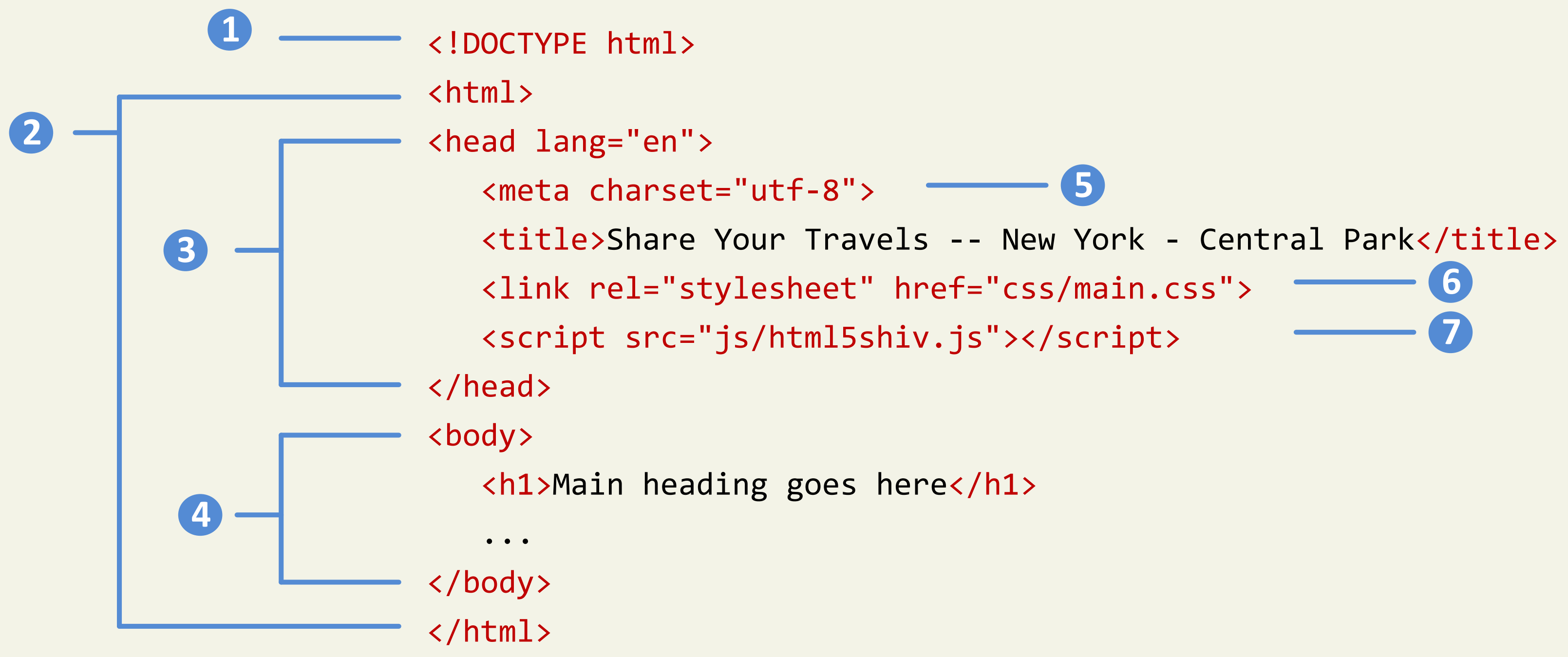


This is a simple document with not much content

DOCTYPE Short for **Document Type Definition** tells the browser what type of document it is about to process

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



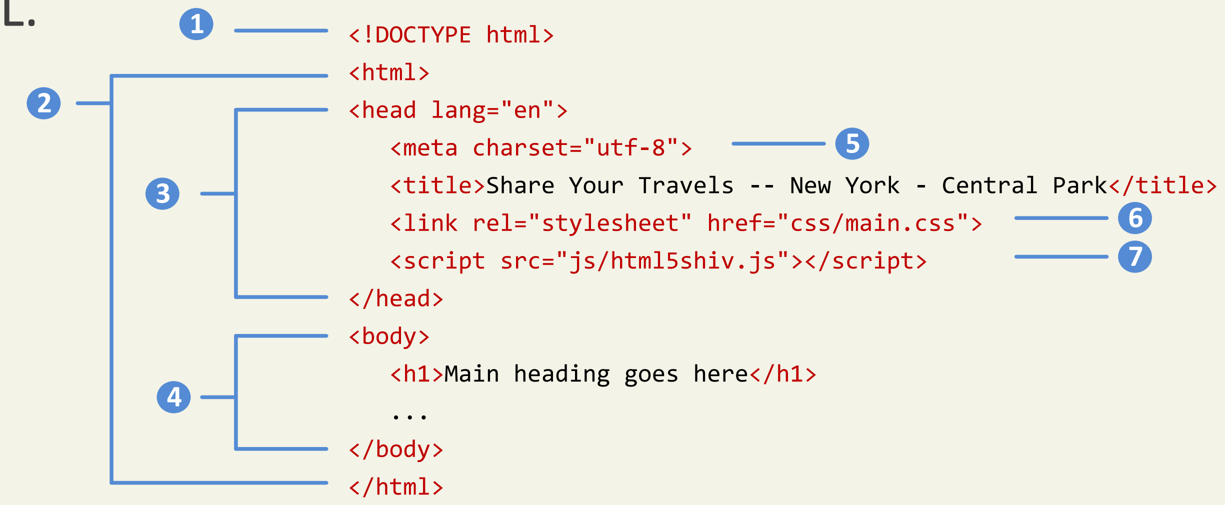
```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

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.



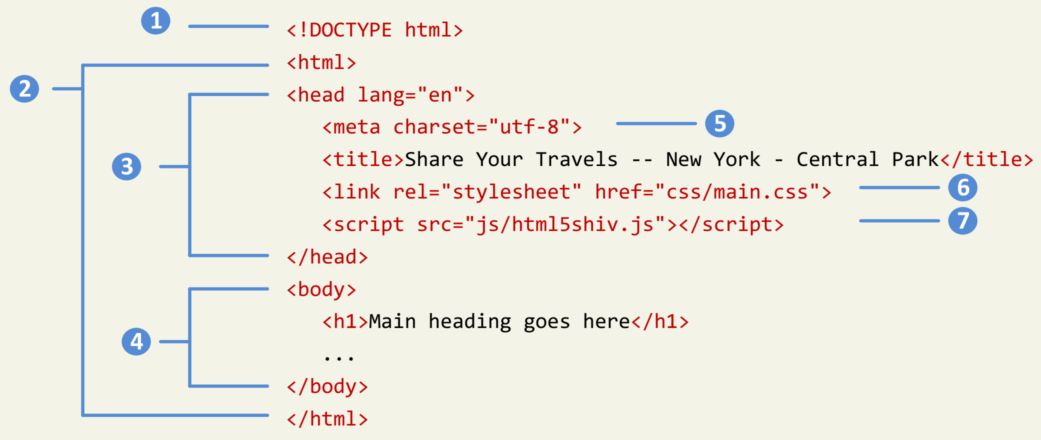
```html
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

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.



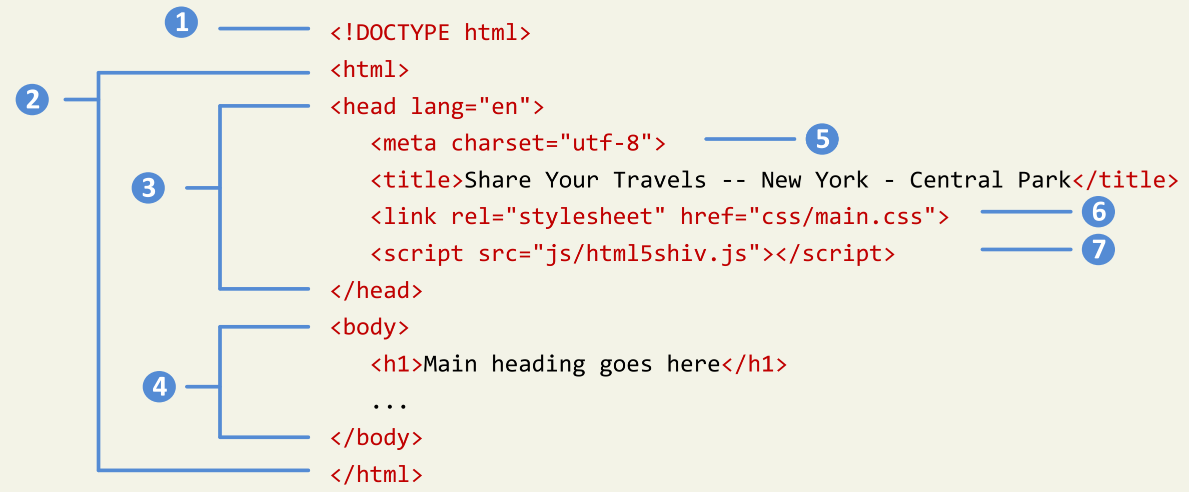
```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

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 (title, style sheets, JavaScript files etc.)

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



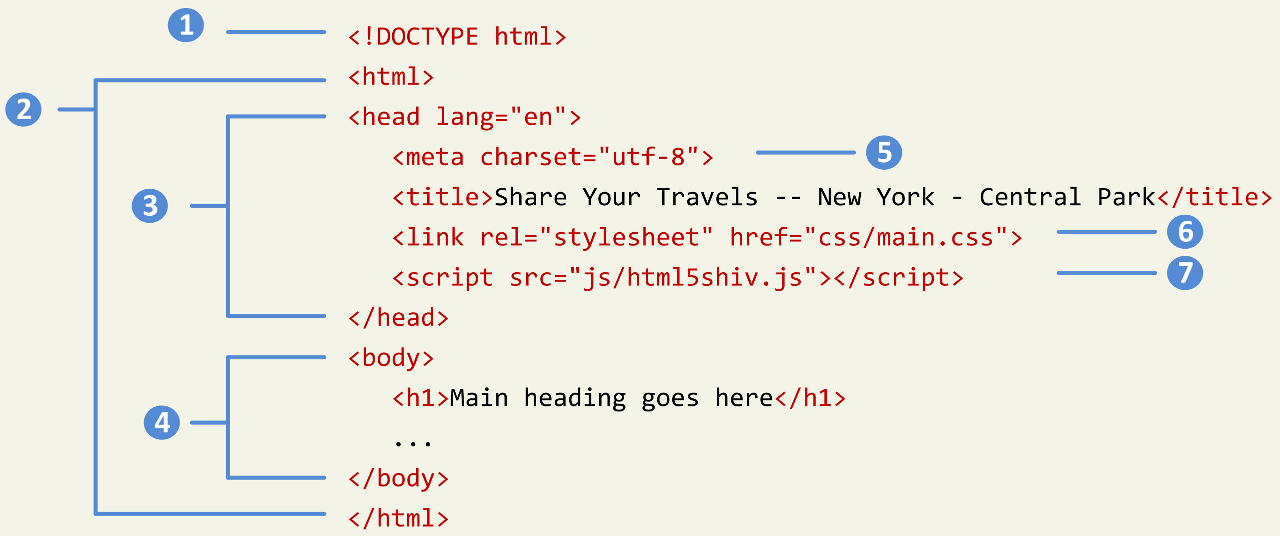
```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

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.



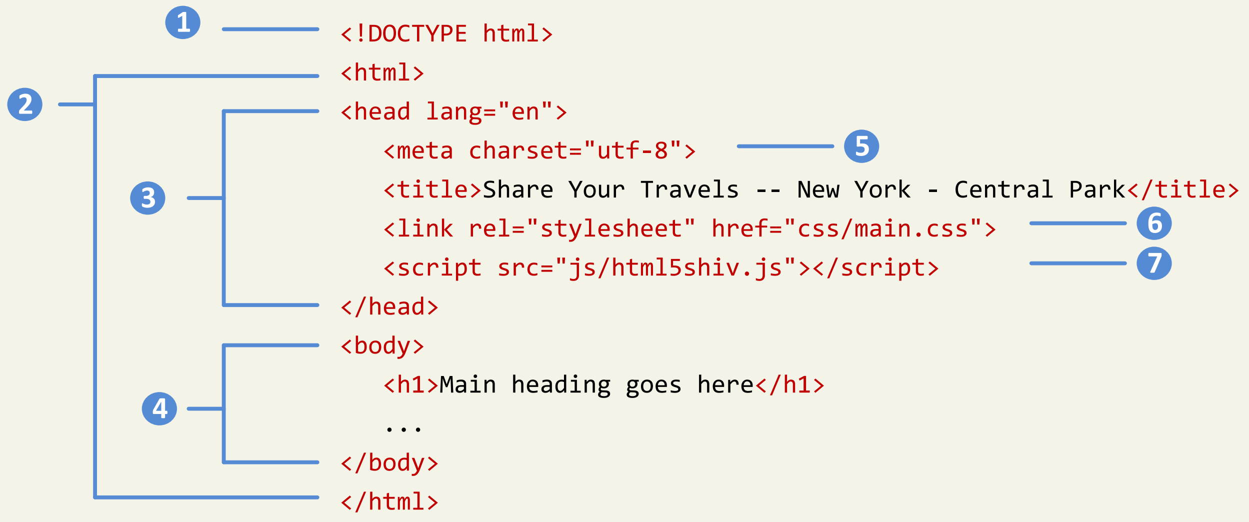
```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

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.



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

Section 5 of 6

QUICK TOUR OF HTML

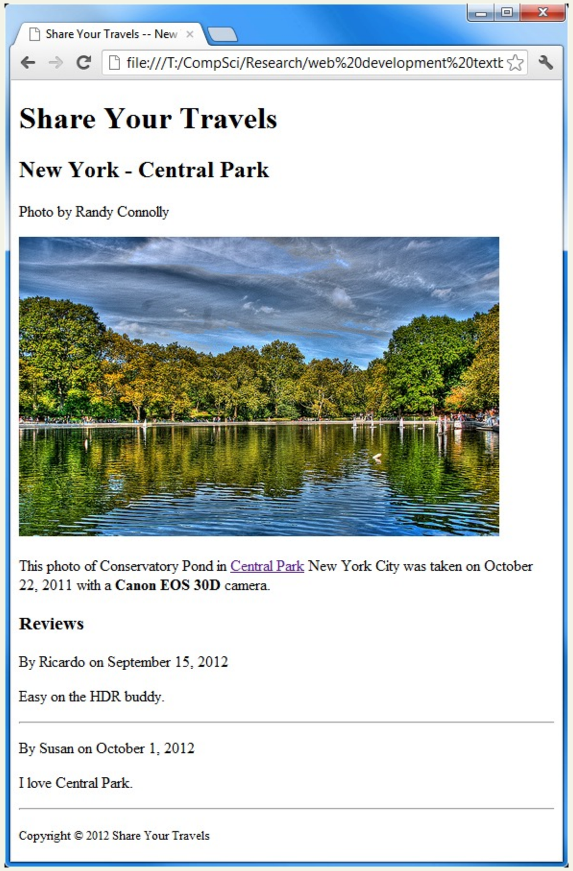
Why a quick tour?

HTML5 contains many structural 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

```
   <body>
1     <h1>Share Your Travels</h1>
      <h2>New York - Central Park</h2>
2     <p>Photo by Randy Connolly</p>
      <p>This photo of Conservatory Pond in
         <a href="http://www.centralpark.com/">Central Park</a>   3
         New York City was taken on October 22, 2011 with a
         <strong>Canon EOS 30D</strong> camera.
      </p>                                                 4
5     <img src="images/central-park.jpg" alt="Central Park" />

      <h3>Reviews</h3>
6     <div>                                                       7
         <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>    8

      <p><small>Copyright &copy; 2012 Share Your Travels</small></p>
   </body>                  9
```


Share Your Travels -- New
file:///T:/CompSci/Research/web%20development%20textt

Share Your Travels

New York - Central Park

Photo by Randy Connolly

This photo of Conservatory Pond in Central Park New York City was taken on October 22, 2011 with a **Canon EOS 30D** camera.

Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

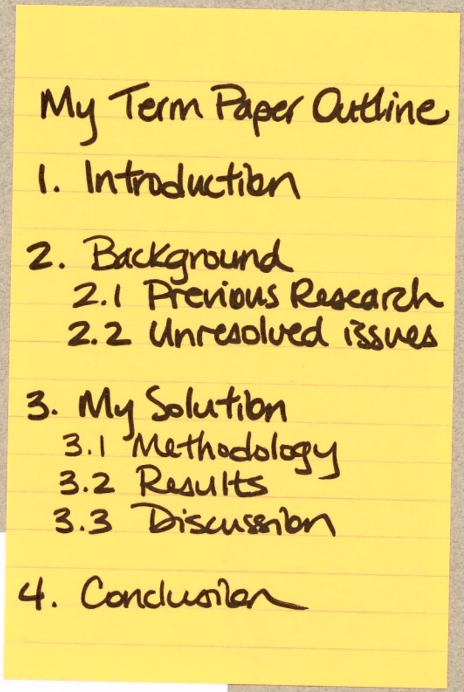
Copyright © 2012 Share Your Travels

1 Headings

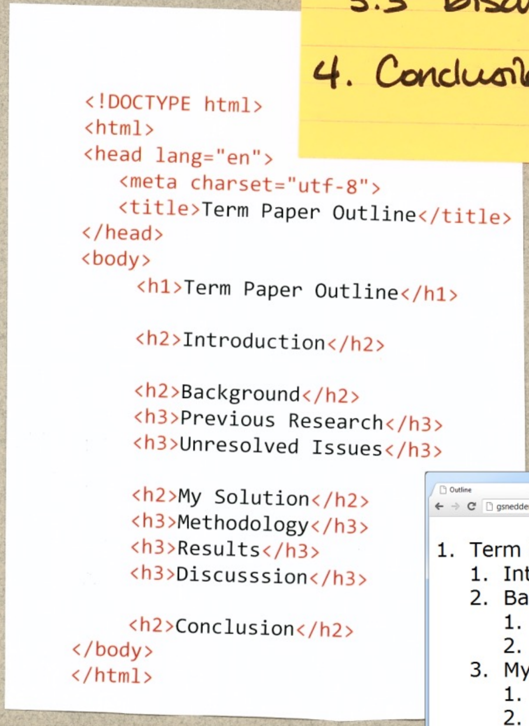
<h1>, <h2>, <h3>, etc

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

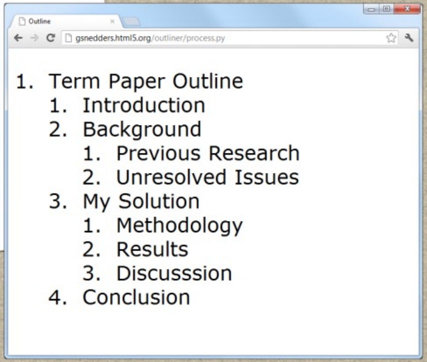
Headings are an essential way for document authors use to show their readers the structure of the document.



My Term Paper Outline
1. Introduction
2. Background
   2.1 Previous Research
   2.2 Unresolved issues
3. My Solution
   3.1 Methodology
   3.2 Results
   3.3 Discussion
4. Conclusion



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Term Paper Outline</title>
</head>
<body>
      <h1>Term Paper Outline</h1>

      <h2>Introduction</h2>

      <h2>Background</h2>
      <h3>Previous Research</h3>
      <h3>Unresolved Issues</h3>

      <h2>My Solution</h2>
      <h3>Methodology</h3>
      <h3>Results</h3>
      <h3>Discusssion</h3>

      <h2>Conclusion</h2>
</body>
</html>
```



Outline

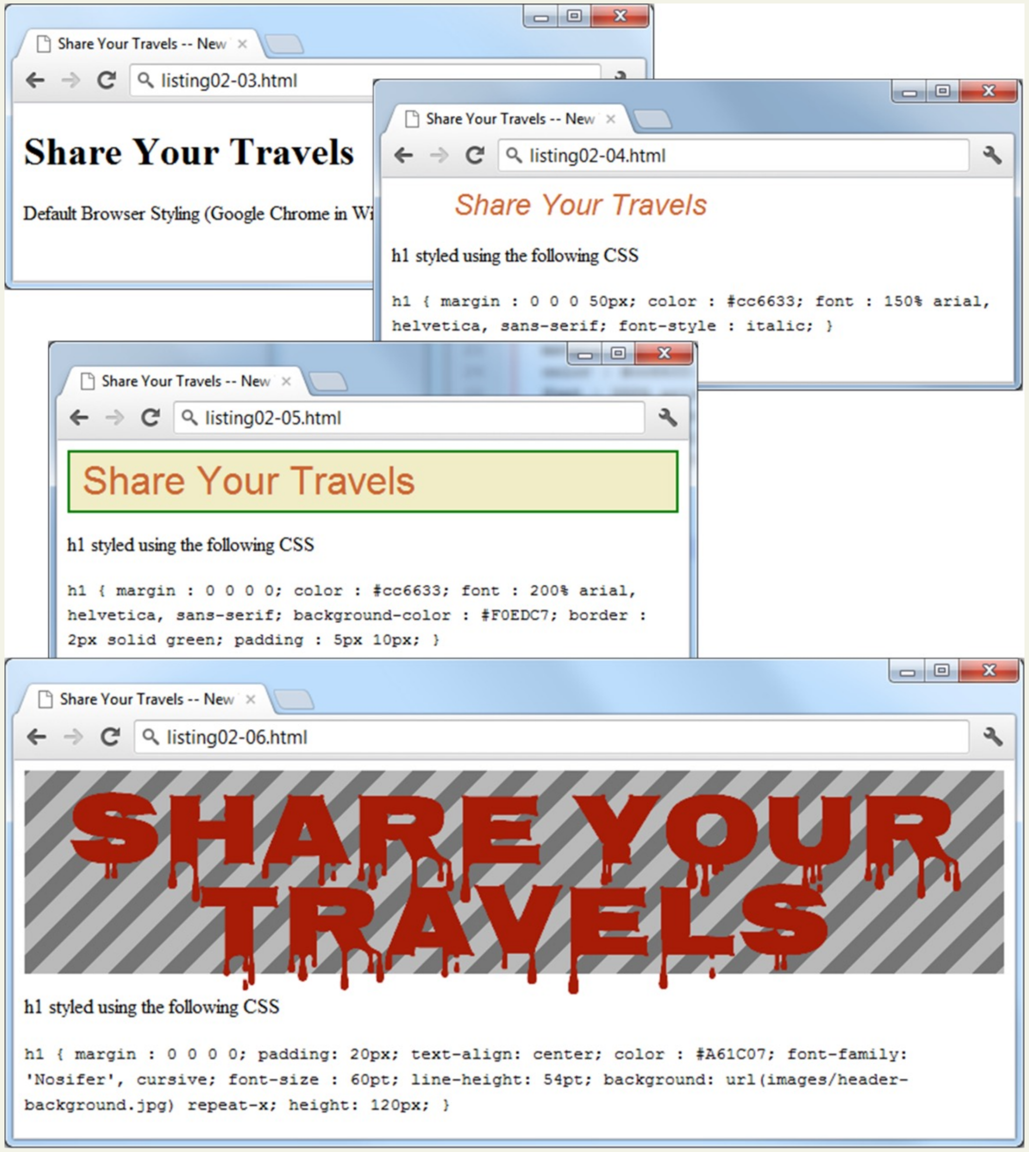
gsnedders.html5.org/outliner/process.py

1. Term Paper Outline
   1. Introduction
   2. Background
      1. Previous Research
      2. Unresolved Issues
   3. My Solution
      1. Methodology
      2. Results
      3. Discusssion
   4. Conclusion

Headings

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

However, these are easily modified and customized via CSS.



Share Your Travels

Default Browser Styling (Google Chrome in Wi

Share Your Travels

h1 styled using the following CSS

```
h1 { margin : 0 0 0 50px; color : #cc6633; font : 150% arial,
helvetica, sans-serif; font-style : italic; }
```

Share Your Travels

h1 styled using the following CSS

```
h1 { margin : 0 0 0 0; color : #cc6633; font : 200% arial,
helvetica, sans-serif; background-color : #F0EDC7; border :
2px solid green; padding : 5px 10px; }
```

h1 styled using the following CSS

```
h1 { margin : 0 0 0 0; padding: 20px; text-align: center; color : #A61C07; font-family:
'Nosifer', cursive; font-size : 60pt; line-height: 54pt; background: url(images/header-
background.jpg) repeat-x; height: 120px; }
```

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.

6 Divisions

<div>

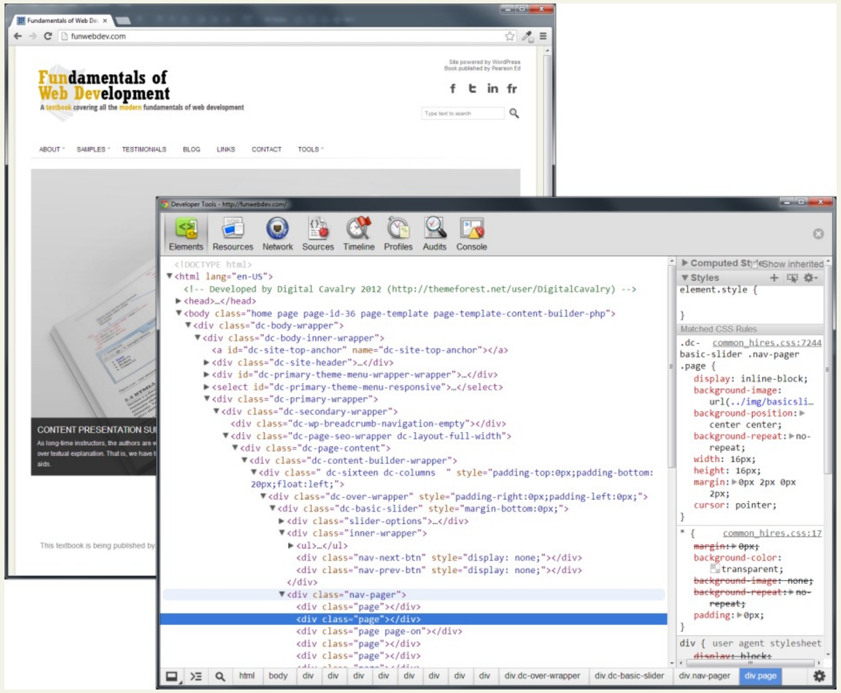
This **<div>** tag is also a container element and is used to create a logical grouping of content

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

Using div elements

Can you say “div-tastic”

HTML5 has a variety of new semantic elements (which we will examine later) that can be used to reduce somewhat the confusing mass of div within divs within divs that is so typical of contemporary web design.



```
<!DOCTYPE html>
▼<html lang="en-US">
   <!-- Developed by Digital Cavalry 2012 (http://themeforest.net/user/DigitalCavalry) -->
  ▶<head>…</head>
  ▼<body class="home page page-id-36 page-template page-template-content-builder-php">
    ▼<div class="dc-body-wrapper">
      ▼<div class="dc-body-inner-wrapper">
         <a id="dc-site-top-anchor" name="dc-site-top-anchor"></a>
        ▶<div class="dc-site-header">…</div>
        ▶<div id="dc-primary-theme-menu-wrapper-wrapper">…</div>
        ▶<select id="dc-primary-theme-menu-responsive">…</select>
        ▼<div class="dc-primary-wrapper">
          ▼<div class="dc-secondary-wrapper">
             <div class="dc-wp-breadcrumb-navigation-empty"></div>
            ▼<div class="dc-page-seo-wrapper dc-layout-full-width">
              ▼<div class="dc-page-content">
                ▼<div class="dc-content-builder-wrapper">
                  ▼<div class=" dc-sixteen dc-columns  " style="padding-top:0px;padding-bottom:
                  20px;float:left;">
                    ▼<div class="dc-over-wrapper" style="padding-right:0px;padding-left:0px;">
                      ▼<div class="dc-basic-slider" style="margin-bottom:0px;">
                        ▶<div class="slider-options">…</div>
                        ▼<div class="inner-wrapper">
                          ▶<ul>…</ul>
                           <div class="nav-next-btn" style="display: none;"></div>
                           <div class="nav-prev-btn" style="display: none;"></div>
                         </div>
                        ▼<div class="nav-pager">
                           <div class="page"></div>
                           <div class="page"></div>
                           <div class="page page-on"></div>
                           <div class="page"></div>
                           <div class="page"></div>
```

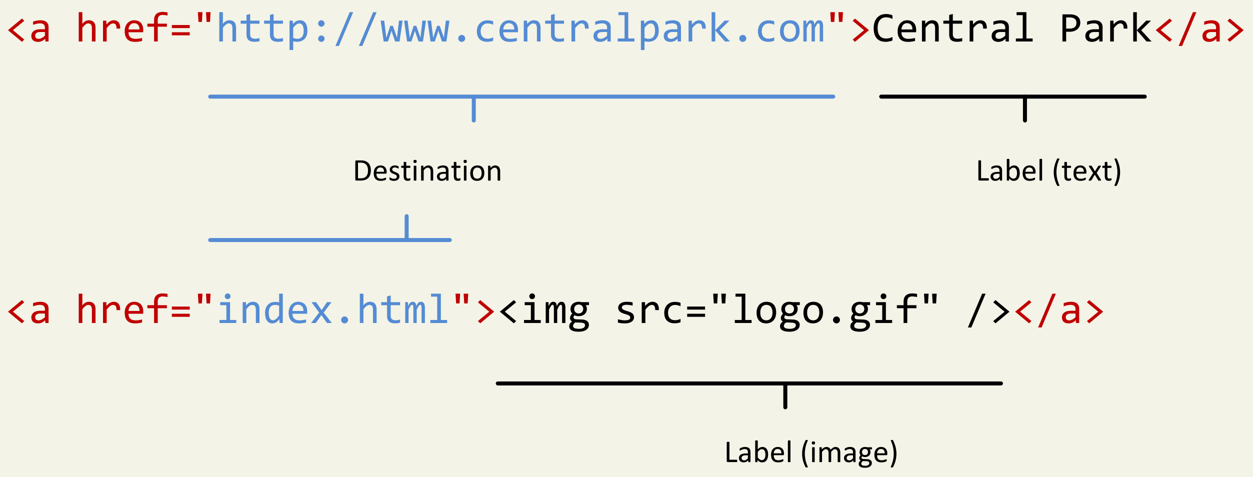
```
element.style {

}
.dc-   common_hires.css:7244
basic-slider .nav-pager
.page {
   display: inline-block;
   background-image:
      url(../img/basicsli…
   background-position: ▶
      center center;
   background-repeat: ▶ no-
      repeat;
   width: 16px;
   height: 16px;
   margin: ▶ 0px 2px 0px
      2px;
   cursor: pointer;
}
* {      common_hires.css:17
   ~~margin: ▶ 0px;~~
   background-color:
      transparent;
   ~~background-image: none;~~
   ~~background-repeat: ▶ no-~~
      ~~repeat;~~
   padding: ▶ 0px;
}
div { user agent stylesheet
   ~~display: block;~~
```

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.



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

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

Types of Links

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

- Links to external sites (or to individual resources such as images or movies 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.
- Links that are instructions to the browser to start the user’s email program.
- Links that are instructions to the browser to execute a Javascript function.

Quick Tour of HTML Elements

Links

Link to external site

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

Link to resource on external site

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

Link to another page on same site as this page

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

Link to another place on the same page

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

Defines anchor for a link to another place on same page

Quick Tour of HTML Elements

Links (continued)

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>
```

Link Text

Some guidance … or … don't "Click Here"

Links with the label "Click Here" were once a staple of the web.

Today, such links are frowned upon, since:

- they do not tell users where the link will take them
- as a verb "click" is becoming increasingly inaccurate when one takes into account the growth of mobile browsers.

Instead, textual link labels should be descriptive.

~~"Click here to see the race results"~~

"Race Results" or **"See Race Results".**

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**.

When referencing a resource that is on the same server as your HTML document, then you can use briefer relative referencing. If the URL does not include the “http://” then the browser will request the current server for the file.

URL Relative Referencing

If all the resources for the site reside within the same **directory** (also referred to as a **folder**), then you can reference those other resources simply via their filename.

However, most real-world sites contain too many files to put them all within a single directory.

For these situations, a relative pathname is required along with the filename.

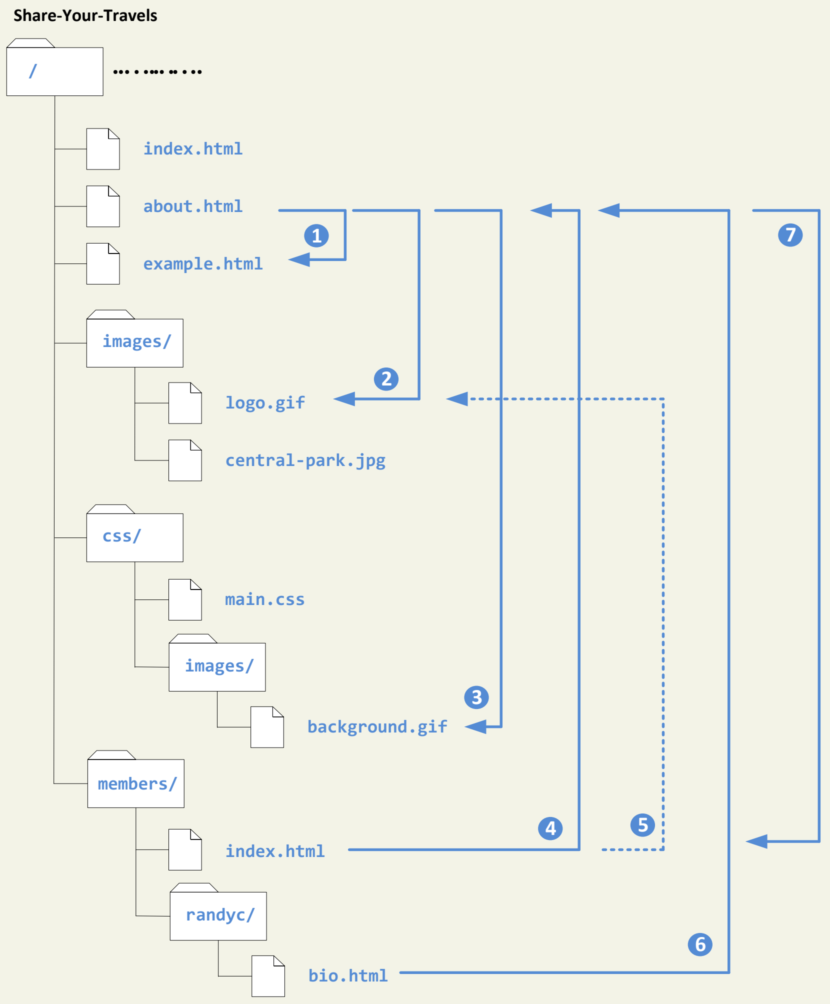
The **pathname** tells the browser where to locate the file on the server.

Pathnames

Pathnames on the web follow Unix conventions.

- Forward slashes ("/") are used to separate directory names from each other and from file names.
- Double-periods ("..") are used to reference a directory "above" the current one in the directory tree.

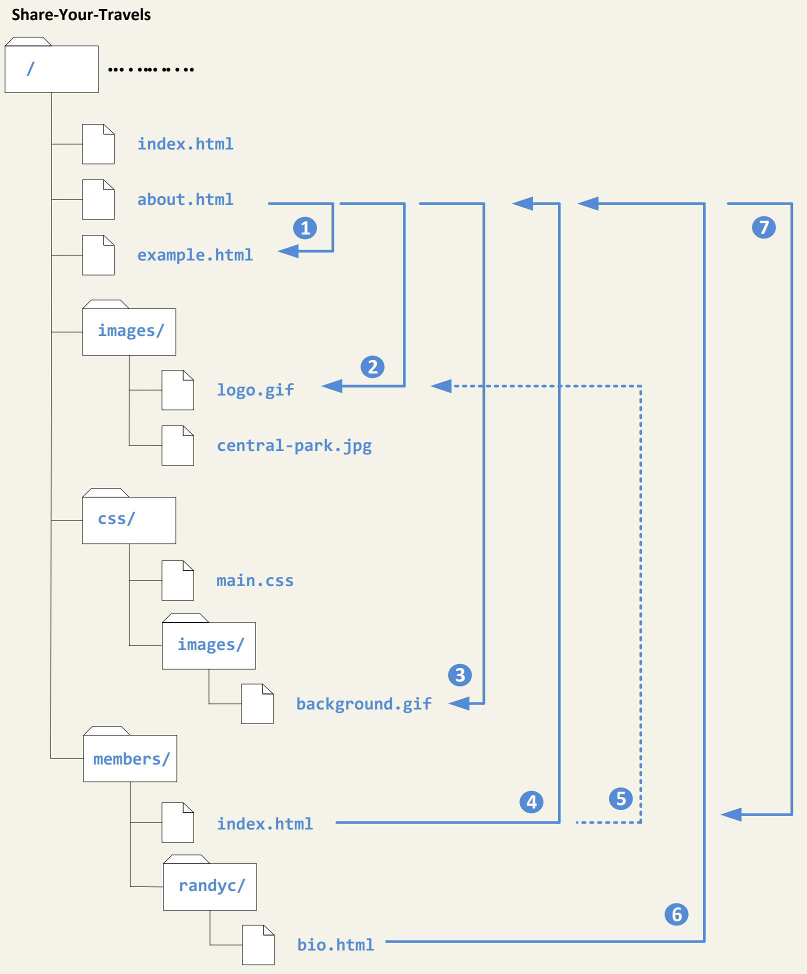
URL Relative Referencing



	Relative Link Type	Example
1	**Same Directory** To link to a file within the same folder, simply use the file name.	To link to example.html from about.html (in Figure 2.18), use: `<a href="example.html">`
2	**Child Directory** To link to a file within a subdirectory, use the name of the subdirectory and a slash before the file name.	To link to logo.gif from about.html, use: `<a href="images/logo.gif">`
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.	To link to background.gif from about.html, use: `<a href="css/images/background.gif">`

4	**Parent/Ancestor 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 “../”.
		To link to about.html from index.html in members, use:
		
		`<a href="../about.html">`
		
		To link to about.html from bio.html, use:
		
		`<a href="../../about.html">`

URL Relative Referencing



5 Sibling Directory

Use "../"to move up to the appropriate level, and then use the same technique as for child or grandchild directories.

To link to logo.gif from index.html in members, use:

```
<a href="../images/logo.gif">
```

To link to background.gif from bio.html, use:

```
<a
href="../../css/images/background.gif">
```

6 Root Reference

An alternative approach for ancestor and sibling references is to use the so-called **root reference** approach. In this approach, begin the reference with the root reference (the "/") and then use the same technique as for child or grandchild directories. **Note that these will only work on the server! That is, they will not work when you test it out on your local machine.**

To link to about.html from bio.html, use:

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

To link to background.gif from bio.html, use:

```
<a href="/images/background.gif">
```

7 Default Document

Web servers allow references to directory names without file names. In such a case, the web server will serve the default document, which is usually a file called index.html (apache) or default.html (IIS). **Again, this will only generally work on the web server.**

To link to index.html in members from about.html, use either:

```
<a href="members">
```

Or

```
<a href="/members">
```

Inline Text Elements

Do not disrupt the flow

Inline elements do not disrupt the flow of text (i.e., go to a new block).

HTML5 defines over 30 of these elements.

- **<a>**
- **<abbr>**
- **<br>**
- **<cite>**
- **<code>**
- **<em>**
- **<mark>**
- **<small>**
- **<span>**
- **<strong>**
- **<time>**

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 pop-up 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.

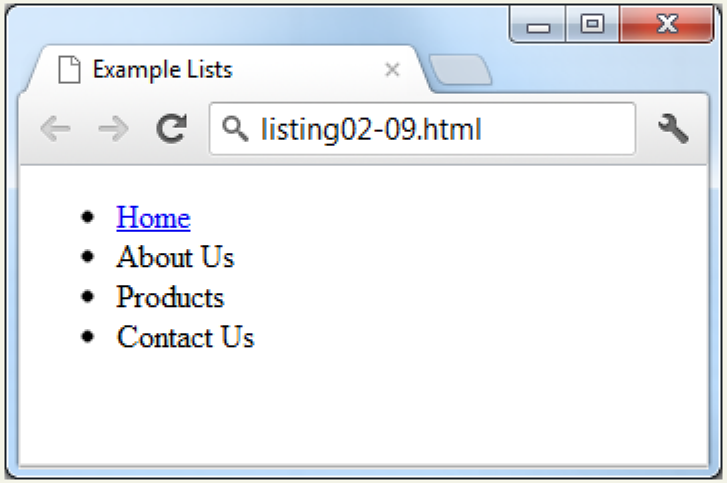
Definition lists. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

Lists

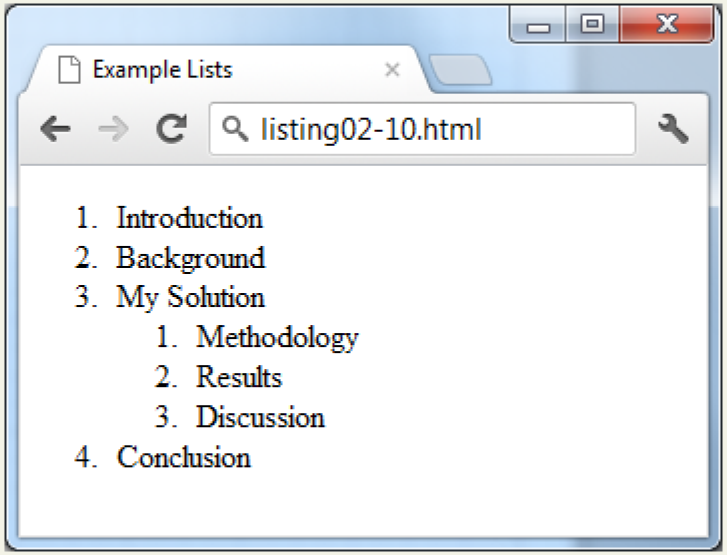
```
<ol>
   <li>Introduction</li>
   <li>Background</li>
   <li>My Solution</li>
   <li>
     <ol>
        <li>Methodology</li>
        <li>Results</li>
        <li>Discussion</li>
     </ol>
   </li>
   <li>Conclusion</li>
</ol>
```

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>
```



- Home
- About Us
- Products
- Contact Us



1. Introduction
2. Background
3. My Solution
   1. Methodology
   2. Results
   3. Discussion
4. Conclusion

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.

Entity	Description
&nbsp;	Nonbreakable space
&lt;	<
&gt;	>
&copy;	©
&trade;	™

Section 6 of 6

HTML SEMANTIC ELEMENTS

HTML5 Semantic Elements

Why are they needed?

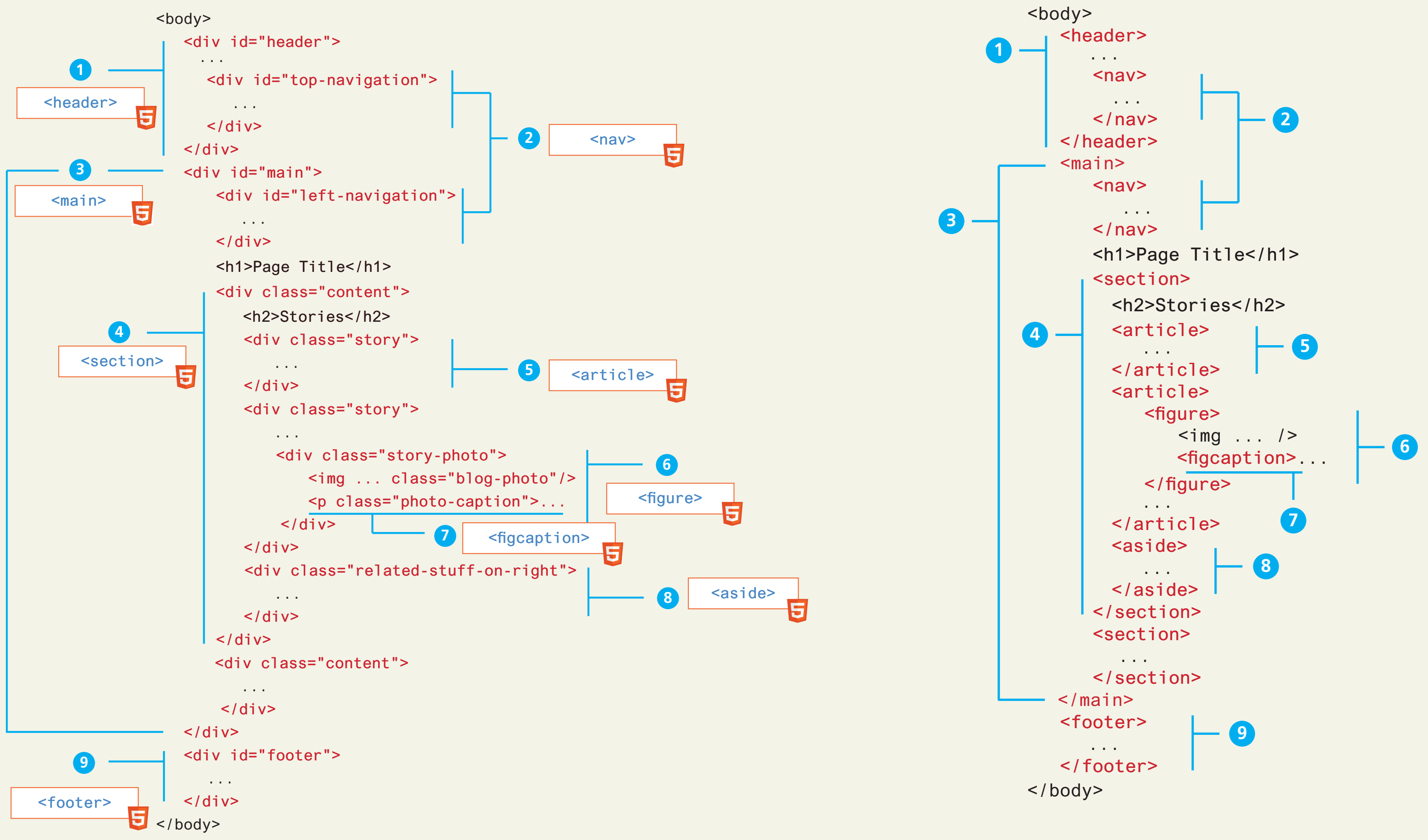
One substantial problem with modern, pre-HTML5 semantic markup:

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

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

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

HTML5 Semantic Structure Elements



```
<body>
   <div id="header">
     ...
      <div id="top-navigation">
         ...
      </div>
   </div>
   <div id="main">
      <div id="left-navigation">
         ...
      </div>
      <h1>Page Title</h1>
      <div class="content">
         <h2>Stories</h2>
         <div class="story">
            ...
         </div>
         <div class="story">
            ...
            <div class="story-photo">
               <img ... class="blog-photo"/>
               <p class="photo-caption">...
            </div>
         </div>
         <div class="related-stuff-on-right">
            ...
         </div>
      </div>
      <div class="content">
         ...
      </div>
   </div>
   <div id="footer">
     ...
   </div>
</body>
```

```
<body>
   <header>
      ...
      <nav>
         ...
      </nav>
   </header>
   <main>
      <nav>
         ...
      </nav>
      <h1>Page Title</h1>
      <section>
         <h2>Stories</h2>
         <article>
            ...
         </article>
         <article>
            <figure>
               <img ... />
               <figcaption>...
            </figure>
            ...
         </article>
         <aside>
            ...
         </aside>
      </section>
      <section>
         ...
      </section>
   </main>
   <footer>
      ...
   </footer>
</body>
```

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>Fundamentals of Web Development</h1>
   ...
</header>
<article>
   <header>
       <h2>HTML5 Semantic Structure Elements </h2>
        <p>By <em>Randy Connolly</em></p>
        <p><time>September 30, 2012</time></p>
   </header>
   ...
</article>
```

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 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>
      <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>
```

HTML5 Semantic Structure Elements

Main

- **<main>** is meant to contain the main unique content of the document.
- <main> provides a semantic replacement for markup such as <div id="main"> or <div id="main-content">
- The **<section>** element represents a section of a document, typically with a title or heading.
- The **<article>** element represents a section of content that forms an independent part of a document or site (could be read independently of other content); for example, a magazine or newspaper article, or a blog entry.

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 <div> 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 <div> 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 <div>.

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.

However...

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

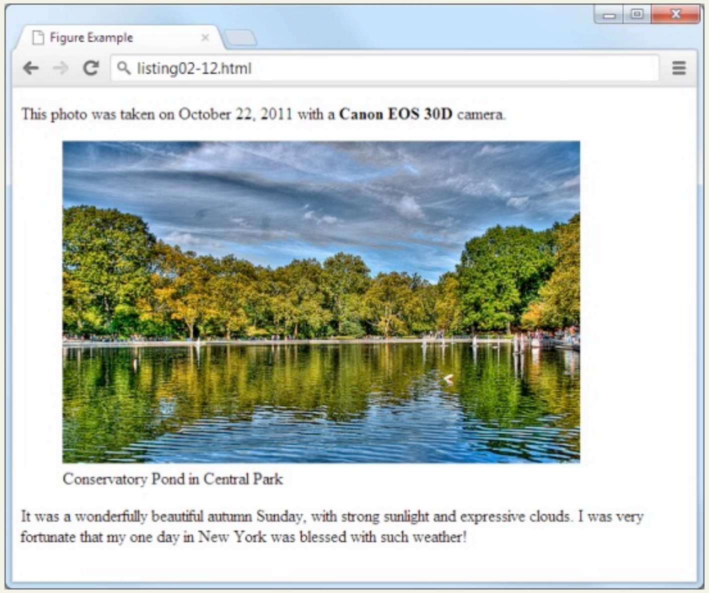
For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within <figure> elements.

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>
```



This photo was taken on October 22, 2011 with a **Canon EOS 30D** camera.

Conservatory Pond in Central Park

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!

9

Aside

<aside>

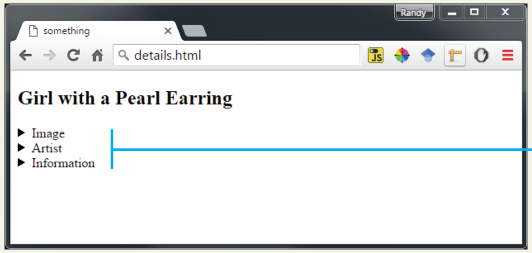
The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

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.

HTML5 Semantic Structure Elements

Details and Summary


something

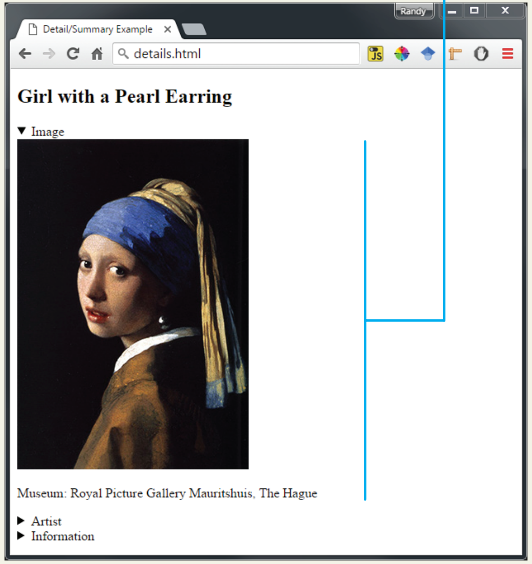
details.html

Girl with a Pearl Earring

- ▶ Image
- ▶ Artist
- ▶ Information

```
<body>
   <h2>Girl with a Pearl Earring</h2>
   <details>
     <summary>Image</summary>
     <img src="images/106020.jpg"><br>
     <p>Museum: Royal Picture Gallery Mauritshuis ...
   </details>
   <details>
     <summary>Artist</summary>
     <p><strong>Jan Vermeer</strong> was a Dutch ...
   </details>
   <details>
     <summary>Information</summary>
     <p>
        Date: 1665<br>
        Medium: Oil on Canvas
     </p>
  </details>
</body>
```

Clicking on the summary label reveals the rest of the content with the <details> container


Detail/Summary Example

details.html

Girl with a Pearl Earring

▼ Image

Museum: Royal Picture Gallery Mauritshuis, The Hague

- ▶ Artist
- ▶ Information