School of Computer Science & Technology

Web Designing and Programming (UE25CS1105)

Chapter 1: Traditional HTML and XHTML

CHAPTER 1

Traditional HTML and XHTML

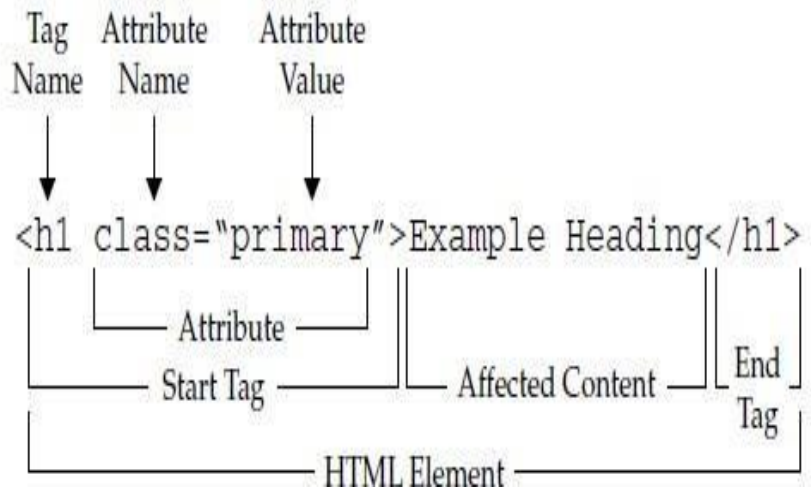
Markup languages are ubiquitous in everyday computing. Although you may not realize it, word processing documents are filled with markup directives indicating the structure and often presentation of the document. In the case of traditional word processing documents, these structural and presentational markup codes are more often than not behind the scenes. However, in the case of Web documents, markup in the form of traditional Hypertext Markup Language (HTML) and its Extensible Markup Language (XML)-focused variant, XHTML, is a little more obvious. These not-so-behind-the-scenes markup languages are used to inform Web browsers about page structure and, some might argue, presentation as well.

First Look at HTML and XHTML

In the case of HTML, markup instructions found within a Web page relay the structure of the document to the browser software. For example, if you want to emphasize a portion of text, you enclose it within the tags and , as shown here:

This is important text!

A graphical overview of the HTML markup syntax shown so far is presented here:



Hello HTML and XHTML World:

Given these basics of HTML syntax, it is best now to look at an example document to see its application. Our first complete example written in strict HTML4 is shown here:

A simple modification of the initial line is really all that is necessary to make this an HTML5 example, the comment and text is changed so you can keep the examples straight:

```
<!DOCTYPE html>

<html>

<head>

<meta http-equiv= "Content-Type" content="text/html; charset=utf-8">

<title>Hello HTML5 World</title>

</head>

<body>

<h1>Welcome to the Future World of HTML5</h1>

<hr>

<p>HTML5 <em>really</em> isn't so hard!</p>

<p>Soon you will &hearts; using HTML.</p>

<p>You can put lots of text here if you want. We could go on and on with fake text for you to read, but let's get back to the book.</p>
</body>

</html>
```

- The **<DOCTYPE!>** statement, which indicates the particular version of HTML or XHTML being used in the document. The above example uses the strict 4.01 specification.

- The **<html>, <head>, and <body>** tag pairs are used to specify the general structure of the document. The required inclusion of the **xmlns** attribute in the **<html>** tag is a small difference required by XHTML.

- The **<meta>** tag indicates the MIME type of the document and the character set in use.

- The **<title> </title>** tag pair specifies the title of the document, which generally appears in the title bar of the Web browser.

- A comment is specified by "**<!-- -- >" ,**allowing page authors to provide notes for future reference.

- The **<h1> </h1>** header tag pair indicates a headline specifying some importantinformation.

- The **<hr>** tag, which has a self-identifying end tag **(<hr />)** under XHTML, insertsa horizontal rule, or bar, across the screen.

- The ** ** paragraph tag pair indicates a paragraph of text.
  A special character is inserted using a named entity **(♥),** which in this caseinserts a heart dingbat character into the text.

- The ** ** tag pair surrounds a small piece of text to emphasize which a browser typically renders in italics.

HTML and XHTML: Version History

Since its initial introduction in late 1991, HTML (and later its XML-based cousin, XHTML) has undergone many changes. Interestingly, the first versions of HTML used to build the earliest Web pages lacked a rigorous definition. Fortunately, by 1993 the Internet Engineering. Task Force (IETF) began to standardize the language and later, in 1995, released the first real HTML standard in the form of HTML 2.0. You will likely encounter more than just the lateststyle of markup for many years to come, so Table 1 presents a brief summary of the version history of HTML and XHTML.

HTML or XHTML Version	Description
HTML 2.0	Classic HTML dialect supported by browsers such as Mosaic. This form of HTML supports core HTML elements and features such as tables and forms, but does not consider any of the browser innovations of advanced features such as style sheets, scripting, or frames.
HTML 3.0	The proposed replacement for HTML 2.0 that was never widely adopted, most likely due to the heavy use of browser-specific markup.
HTML 3.2	An HTML finalized by the W3C in early 1997 that standardized most of the HTML features introduced in browsers such as Netscape 3. This version of HTML supports many presentation-focused elements such as **font**, as well as early support for some scripting features.
HTML 4.0 Transitional	The 4.0 transitional form finalized by the W3C in December of 1997 preserves most of the presentational elements of HTML 3.2. It provides a basis of transition to Cascading Style Sheets (CSS) as well as a base set of elements and attributes for multiple-language support, accessibility, and scripting.
HTML 4.0 Strict	The strict version of HTML 4.0 removes most of the presentation elements from the HTML specification, such as **font**, in favor of using CSS for page formatting.
4.0 Frameset	The frameset specification provides a rigorous syntax for framed documents that was lacking in previous versions of HTML.
HTML 4.01 Transitional/ Strict/Frameset	A minor update to the 4.0 standard that corrects some of the errors in the original specification.
HTML5	Addressing the lack of acceptance of the XML reformulation of HTML by the mass of Web page authors, the emerging HTML5 standard originally started by the WHATWG[3] group and later rolled into a W3C effort aimed to rekindle the acceptance of traditional HTML and extend it to address Web application development, multimedia, and the ambiguities found in browser parsers. Since 2005, features now part of this HTML specification have begun to appear in Web browsers, muddying the future of XHTML in Web browsers.
XHTML 1.0 Transitional	A reformulation of HTML as an XML application. The transitional form preserves many of the basic presentation features of HTML 4.0 transitional but applies the strict syntax rules of XML to HTML.
XHTML 1.0 Strict	A reformulation of HTML 4.0 Strict using XML. This language is rule enforcing and leaves all presentation duties to technologies like CSS.
XHTML 1.1	A restructuring of XHTML 1.0 that modularizes the language for easy extension and reduction. It is not commonly used at the time of this writing and offers minor gains over strict XHTML 1.0.

Table 1 : Description of Common HTML Versions.

HTML and XHTML DTDs: The Specifications Up Close:

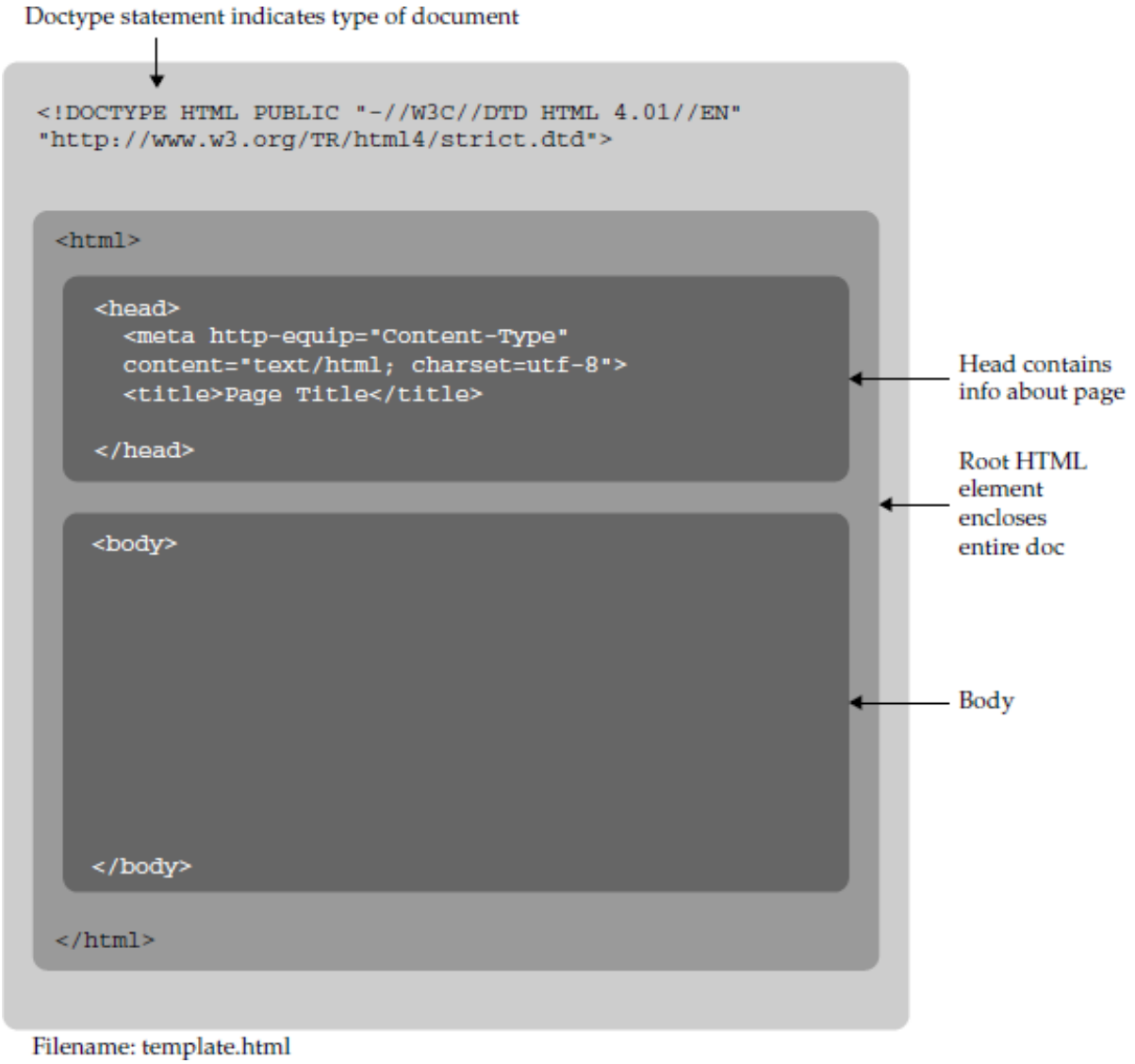
- All (X)HTML documents should follow a formal structure defined by the World Wide Web Consortium (W3C; www.w3.org), which is the primary organization that defines Web standards.
- Traditionally, the W3C defined HTML as an application of the Standard Generalized Markup Language (SGML). SGML is a technology used to define markup languages by specifying the allowed document structure in the form of a document type definition (DTD).
- A DTD indicates the syntax that can be used for the various elements of a language such as HTML.

A snippet of the HTML 4.01 DTD defining the **P** element, which indicates a paragraph, is shown here:

```
<!--=================== Paragraphs =================================-->
<!ELEMENT P - O (%inline;)*              -- paragraph -->
<!ATTLIST P
  %attrs;                                -- %coreattrs, %i18n, %events --
 >
```

- The first line is a comment indicating what is below it.
- The second line defines the P element, indicating that it has a start tag (<P>), as shown by the dash, and an optional close tag (</P>), as indicated by the O.
- The type of content that is allowed to be placed within a **P** element is defined by the entity %inline, which acts here as a shorthand for various other elements and content.
- This idea of only allowing some types of elements within other elements is called the *content* model.
- The final line defines the attributes for a **<P>** tag as indicated by the entity %attrs which then expands to a number of entities like %core,%i18n, and %coreevents which finally expand into a variety of attributes like **id**, **class**, **style**, **title**, **lang**, **dir**, **onclick**, **ondblclick**, and many more.

HTML Document Structure:



- In this graphical representation, the **<!DOCTYPE>** indicator, shows the particular version of HTML being used, in this case Transitional.
- Within a root **html** element, the basic structure of a document reveals two elements: the **head** and the **body**.
- The **head** element contains information and tags describing the document, such as its **title**, while the **body** element houses the document itself, with associated markup required to specify its structure. HTML5 follows the same core structure.

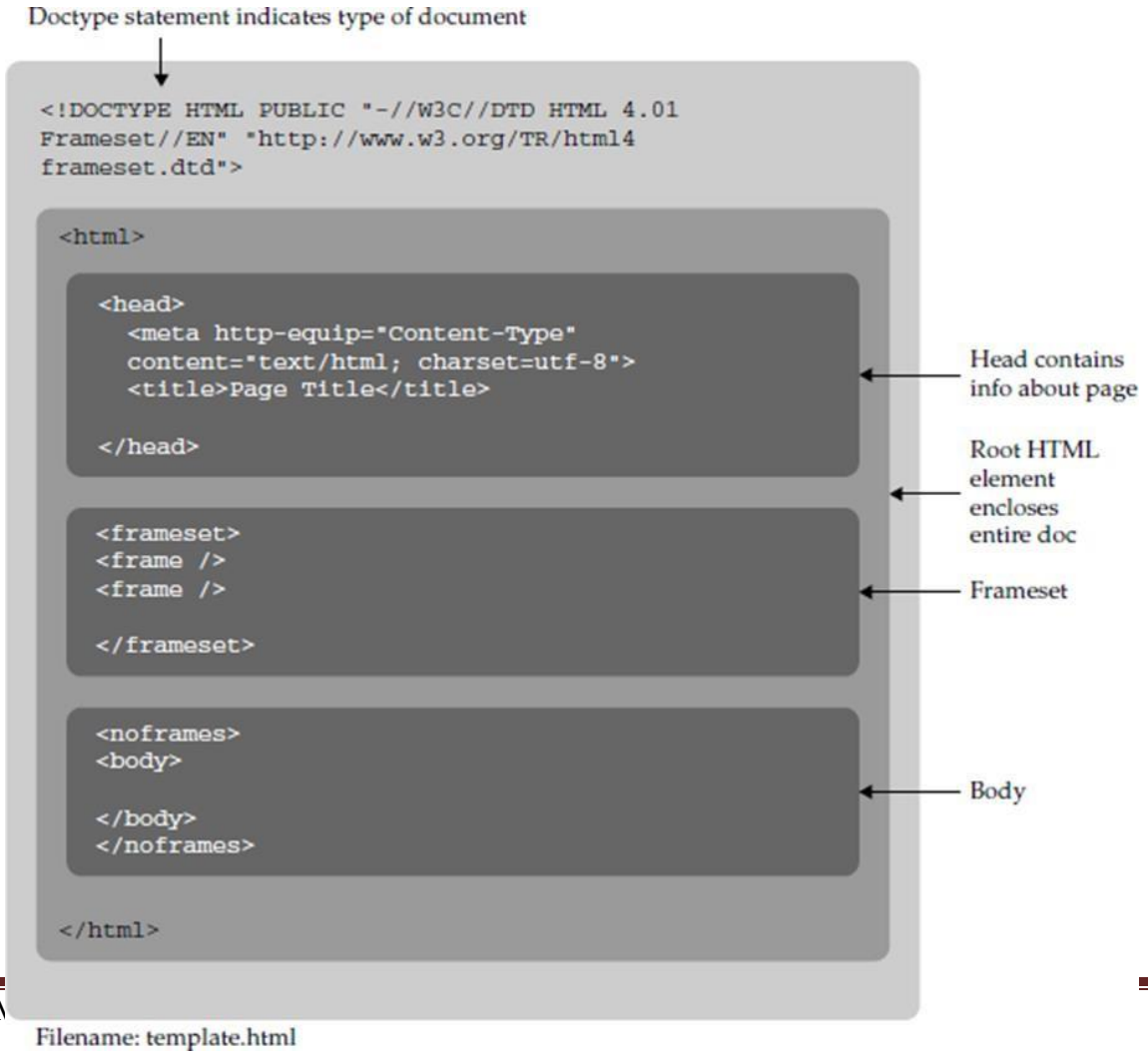
The <!DOCTYPE> :

- In The structure of an XHTML document is pretty much the same with the exception of a different **<!DOCTYPE>** indicator and an **xmlns** (XML name space) attribute added to the **html** tag so that it is possible to intermix XML more easily into the XHTML document.

- Alternatively, in either HTML or XHTML (but not in HTML5), we can replace the <body> tag with a <frameset> tag, which encloses potentially numerous <frame> tags corresponding to individual portions of the browser window, termed frames.

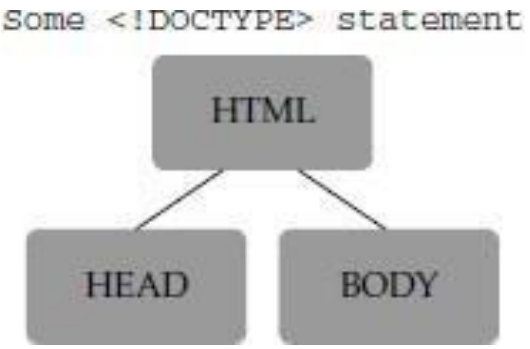
- Each frame in turn would reference another HTML/XHTML document containing either a standard document, complete with <html>, <head>, and <body> tags, or perhaps yet another framed document.

- The <frameset> tag also should include a noframes element that provides a version of the page for browsers that do not support frames. Within this element, a <body> tag should be found for browsers that do not support frames.



Filename: template.html

The structure of a non-framed (X)HTML document breaks out like so:



The Document Head:

- The information in the **head** element of an (X)HTML is used to describe the content of the document.
- The element acts like the front matter or cover page of a document.
- **head** element is information about the page that is useful for visual styling, defining interactivity, setting the page title, and providing other useful information that describes or controls the document.

The title Element:

- A single **title** element is required in the **head** element and is used to set the text to display in their title bar.
- The value within a **title** is used in a browser's history system, recorded when the page is bookmarked.
- In short, it is pretty important to have a syntactically correct, descriptive, and appropriate page title. Thus, given :

```
<title>Simple HTML Title Example</title>
```

When a `title` is not specified, most browsers display the URL path or filename instead:

<meta>: Specifying Content Type, Character Set, and More:

- A **<meta>** tag has a number of uses. For example, it can be used to specify values that are equivalent to HTTP response headers.
- For example, if you want to make sure that your MIME type and character set for an English-based HTML document is set, you could use other elements in the head- includes **base**, **link**, **object**, **script**, and **style**. Comments are also allowed.

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

- **<base>:** A **<base>** tag specifies an absolute URL address that is used to provide server and directory information to *relative links*

  <base href= "http://htmlref.com/basexeample" >

- **<script>:** A **<script>** tag allows scripting language code to be either directly embedded within

  ```
  <script type="text/javascript"> alert("Hi
  from JavaScript!");
  /* more code below */
  </script>
  ```

- **<link> :** A **<link>** tag specifies a special relationship between the current document and another document. Most commonly, it is used to specify a style sheet used by the document.

  <link rel="stylesheet" media="screen“ href="global.css"type="text/css" >

- **<object>:** An **<object>** tag allows programs and other binary objects to be directly embedded in

a Web page.

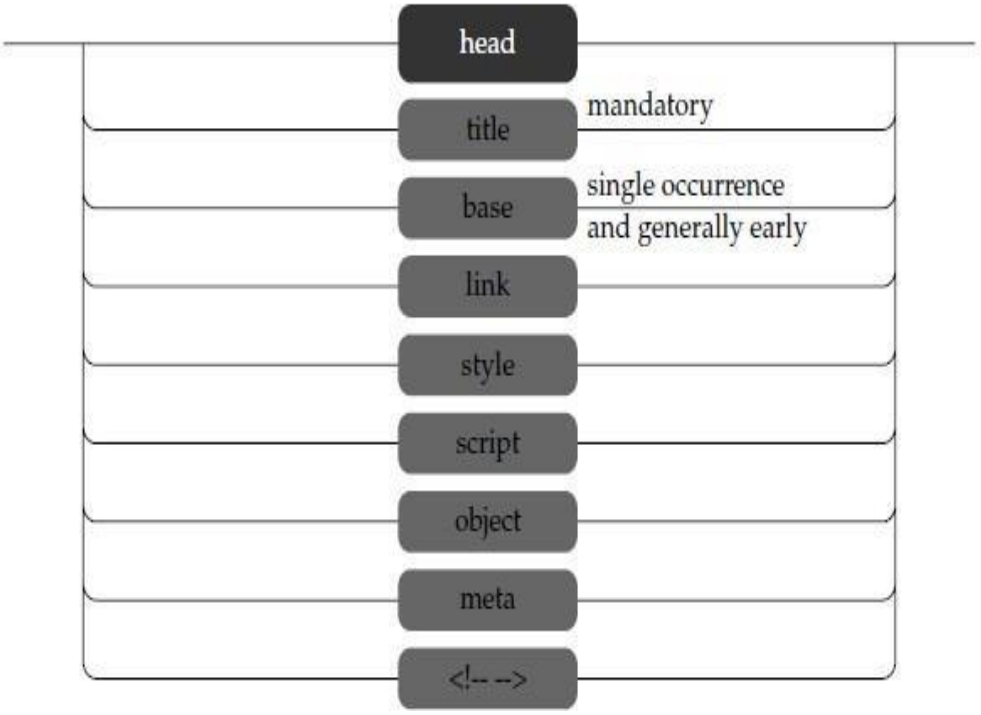
```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
width="0" height="0" id="HiddenFlash" >
<param name="movie" value="flashlib.swf" />
</object>
```

- **<style> :** A <style> tag is used to enclose document- wide style specifications, typically in Cascading StyleSheet (CSS) format, relating to fonts, colors, positioning, and other aspects of content presentation:

```
<style type="text/css" media="screen">
h1 {font-size: xx-large; color: red; font-style: italic;}
/* all h1 elements render as big, red and italic */
</style>
```

The complete syntax of the markup allowed in the head element under strict (X)HTML is shown here:

nd



The Document Body:

- The body of a document is delimited by <body>and </body>.
  Only one body element can appear per document.
- Within the body of a Web document is a variety of types of elements.
- For example, block level *elements* define structural content blocks such as paragraphs (p) or headings (h1-h6).
- Block-level elements generally introduce line breaks visually. Special forms of blocks, such as unordered lists (ul), can be used to create lists of information.
- Within non empty blocks, inline elements are found. There are numerous inline elements, such as bold (b), italic (i), strong (strong), emphasis (em).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<title>Hello HTML World</title>
<!-- Simple hello world in HTML 4.01 strict example -->
</head>
<body>
<h1>Welcome to the World of HTML</h1>
<hr>
<p>HTML <em>really</em> isn't so hard!</p>
<p>Soon you will &hearts; using HTML.</p>
<p>You can put lots of text here if you want.
We could go on and on with fake text for you
to read, but let's get back to the book.</p>
</body>
</html>
```

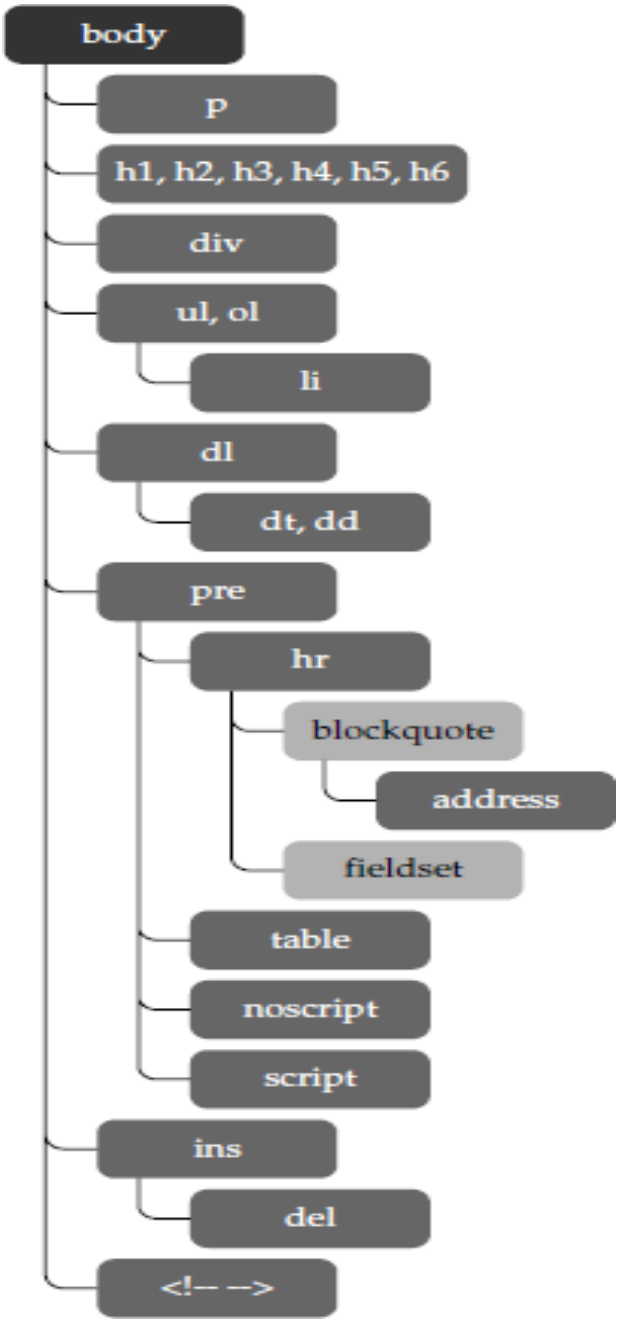
Comment

Block Elements

Inline Elements

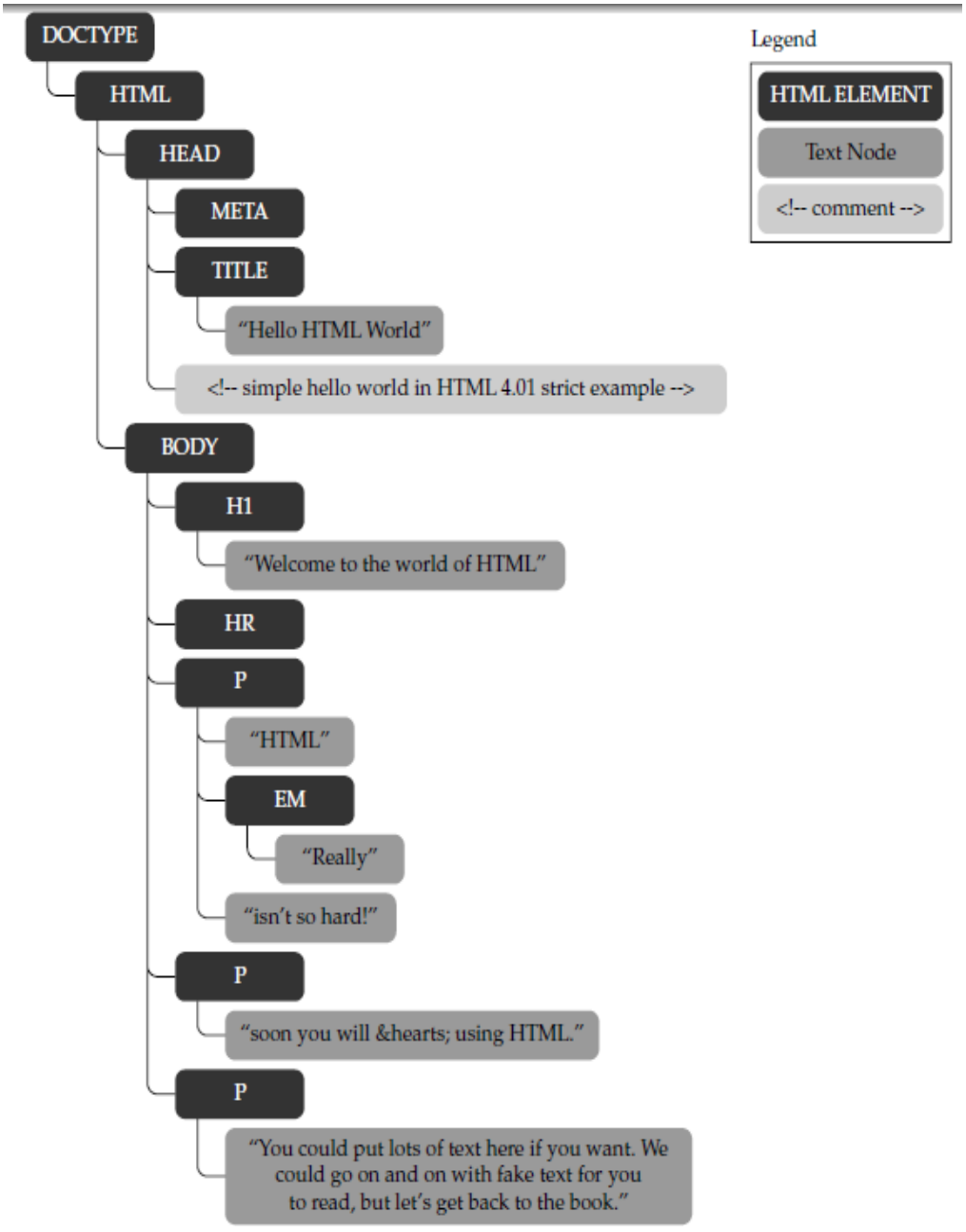
Character Entity

A visual overview of all the items presented in the body is shown here:



The full syntax of the elements allowed in the body element is a bit more involved than the full syntax of the head. This diagram shows what is directly included in the body:

Browsers and (X)HTML:



When a browser reads a marked-up document, such as the "hello world" ex it builds a parse tree to interpret the structure of the document, like this:

Example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hello HTML 4 World</title>
<!-- Simple hello world in HTML 4.01 strict example -->
</head>
<body>
<h1>Welcome to the World of HTML</h1>
<hr>
<p>HTML <em>really</em> isn't so hard!</p>
<p>Soon you will &hearts; using HTML.</p>
<p>You can put lots of text here if you want.
We could go on and on with fake text for you
to read, but let's get back to the book.</p>
</body>
</html>
```

The Rules of (X)HTML:

- HTML Is Not Case Sensitive, XHTML Is .
- Attribute Values May Be Case Sensitive.
- (X)HTML Is Sensitive to a Single Whitespace Character.
- (X)HTML Follows a Content Model.
- Elements Should Have Close Tags Unless Empty.
- Unused Elements May Minimize.
- Elements Should Nest.
- Attributes Should Be Quoted.

- Entities Should Be Used for Special Characters.
- Browsers Ignore Unknown Attributes and Elements.

1. HTML Is Not Case Sensitive, XHTML Is:

- These markup examples are all equivalent under traditional HTML:

  <B>Go boldly**</B>**

  <B>Go boldly****

  ****Go boldly**</B>**

  ****Go boldly****

2. Attribute Values May Be Case Sensitive:

- Consider **<img SRC="test.gif">** and **<IMG src="test.gif">**.
  Under traditional HTML, these are equivalent because the **<img>** tag and the **src** attribute are not case sensitive.

```
<strong>T e s t o f s p a c e s</strong><br>
<strong>T   e   s   t   o f   s p a c e s </strong><br>
<strong>T
e s
t o f s p          a c e s</strong><br>
```

As shown here, all the spaces, tabs, and returns are collapsed to a single element:

Testofspaces
Testofspaces
Testofspaces

3. (X)HTML Is Sensitive to a Single Whitespace Character:

- Any white space between characters displays as a single space. This includes all tabs, line breaks, and carriage returns.

4. (X)HTML Follows a Content Model:

- All forms of markup support a content model that specifies that certain elements are supposed to occur only within other elements. For example, markup like this

```
<ul>
   <p>What a simple way to break the content model!</p>
</ul>
```

5. Elements Should Have Close Tags Unless Empty:

- Under traditional HTML, some elements have optional close tags. For example, both of the paragraphs here are allowed, although the second one is better:

   This isn't closed

  ** This is **

- A few elements, like the horizontal rule (hr) and line break (br), do not have close tags because they do not enclose any content.

6. Unused Elements May Minimize:

- Sometimes tags may not appear to have any effect in a document. Consider, for example, the tag, which specifies a paragraph.
- As a block tag, it induces a return by default, but when used repeatedly, like so,

  does this produce numerous blank lines? No, since the browser minimizes the empty p elements.

7. Elements Should Nest:

- A simple rule states that tags should nest, not cross;

Thus is in error as tags cross

is not since tags nest

and this is syntactically correct.

- All forms of markup, traditional HTML, XHTML, and HTML5, follow this rule, and while crossing tags may seem harmless, it does introduce some ambiguity in parse trees.

- To be a well-formed markup, proper nesting is mandatory.

8. Attributes Should Be Quoted:

- Under traditional HTML as well as under HTML5, simple attribute values do not need to be quoted. If the attribute contains only alphanumeric content, dashes, and periods, then the quotes can safely be removed; so would work fine in most browsers and would validate.
  <img src=robot.gif height=10 width=10 alt=robot>
- However, the lack of quotes can lead to trouble, especially when scripting is involved. Quotes should be used under transitional markup forms and are required under strict forms like XHTML; so,

  would be the correct form of the tag.

9. Entities Should Be Used for Special Characters:

- Markup parsers are sensitive to special characters used for the markup itself, like < and >.
- Instead of writing these potentially parse-dangerous characters in the document, they should be escaped out using a character entity.
- For example, instead of <, use **<** or the numeric equivalent **<**.Instead of >, use **>** or **>**.

10. Browsers Ignore Unknown Attributes and Elements:

- For better or worse, keep in mind that browsers will ignore unknown elements and attributes; so,

<bogus>this text will display on screen**</bogus>**

Major Themes of (X)HTML:

- Logical and Physical
- Standards vs. Practice
- Myths and Misconceptions About HTML and XHTML

1. Logical and Physical:

- Physical markup refers to using a markup language such as (X)HTML to make pages look a particular way
- logical markup refers to using (X)HTML to specify the structure or meaning of content while using another technology, such as CSS, to designate the look of the page.
- Physical markup is obvious; if you want to highlight something that is important to the reader, you might embolden it by enclosing it within a tag:
  This is important!
- This simple approach fits with the WYSIWYG (*what you see is what you get*) world of programs such as Microsoft Word.
- Logical markup is a little less obvious; to indicate the importance of the phrase, it should be enclosed in the logical strong element:
  This is important.
- Whether you subscribe to the physical (specific) or logical (general) viewpoint, traditional HTML is neither purely physical *nor* purely logical, at least not yet.
- In other words, currently used HTML elements come in both flavors, physical and logical, though users nearly always think of them as physical.

2. Standards vs. Practice:

- Just because a standard is defined doesn't necessarily mean that it will be embraced. Many Web developers simply do not know or care about standards.

- As long as their page looks right in their favorite browser, they are happy and will continue to go on abusing HTML tags like <table> and using various tricks and proprietary elements.

- Without standards, the modern world wouldn't work well.
- The Web needs standards, but standards have to acknowledge what people actually do.
- Web standards and development practices provide an interesting study of the difference between what theorists say and what people want and do.
- HTML5 seems a step in the right direction. The specification acknowledges that, for better or worse, traditional HTML practices are here for now, and thus attempts to make them solid while continuing to move technology forward and encourage correct usage.

3. Myths and Misconceptions About HTML and XHTML:

i. Misconception: WYSIWYG Works on the Web

- HTML isn't a specific, screen- or printer-precise formatting language like PostScript.
- Many people struggle with HTML on a daily basis, trying to create perfect layouts using (X)HTML elements inappropriately or using images to make up for HTML's lack of screen and font-handling features.

ii. Misconception: HTML Is a Programming Language

- Many people think that making HTML pages is similar to programming. However, HTML is unlike programming in that it does not specify logic.
- It specifies the structure of a document.

iii. Misconception: XHTML Is the Only Future

- Approaching its tenth birthday, XHTML still has yet to make much inroads in the widespread building of Web pages.
- Most documents are not authored in XHTML, and many of those that are, are done incorrectly.

iv. Misconception: XHTML Is Dead

- Although XHTML hasn't taken Web development by storm, the potential rise of HTML5 does not spell the end of XHTML.

- In fact, you can write XML-style markup in HTML, which most developers dub XHTML 5.

v. Myth: Traditional HTML Is Going Away

- HTML is the foundation of the Web; with literally billions of pages in existence, not every document is going to be upgraded anytime soon.
- The "legacy" Web will continue for years, and traditional non standardized HTML will always be lurking around underneath even the most advanced Web page years from now.

vi. Myth: Someday Standards Will Alleviate All Our Problems

- Standards are important. Standards should help. Standards likely won't fix everything.
- From varying interpretations of standards, proprietary additions, and plain old bugs, there is likely never going to be a day where Web development, even at the level of (X)HTML markup, doesn't have its quirks and oddities.

vii. Myth: Hand-Coding of HTML Will Continue Indefinitely

- Although some people will continue to craft pages in a manner similar to mechanical typesetting, as Web editors improve and produce standard markup perfectly, the need to hand-tweak HTML documents will diminish.

Viii. Myth: (X)HTML Is the Most Important Technology Needed to Create Web Pages

- Whereas (X)HTML is the basis for Web pages, you need to know a lot more than markup to build useful Web pages (unless the page is very simple).
- However, don't underestimate markup, because it can become a bit of a challenge itself.
- Based on the simple examples presented in this chapter, you might surmise that mastering Web page creation is merely a matter of learning the multitude of markup tags, such as **<h1>**, **<p>**, ****, and so on, that specify the structure of Web documents to browsers.

The Future of Markup—Two Paths?

.

Path 1: Evolution and Enhancement of HTML

1. HTML5 and Beyond: HTML5 has already transformed the web with its rich set of features for multimedia, semantic elements, and improved APIs. Future versions of HTML are likely to continue this trend, focusing on

- Enhanced Semantics: Further development of semantic elements to improve accessibility and SEO.
- Integration with JavaScript: More powerful and standardized APIs to enhance interactivity and functionality.
- Improved Performance: Optimizations for faster rendering and better handling of complex content.

2. Web Components: The adoption of Web Components (custom elements, shadow DOM, and HTML templates) is set to grow. These allow developers to create reusable and encapsulated components, which can lead to more modular and maintainable code.

3. Accessibility and Inclusivity: There's a continuous push towards improving accessibility and inclusivity in web development, ensuring that markup languages and web technologies cater to all users.

Path 2: Specialized and Emerging Markup Languages

1. **XML and Derivatives:** XML's influence continues through various specialized languages like SVG for graphics, XSLT for transformations, and custom XML-based formats. While XML is less prominent in new web projects compared to HTML, its role in data interchange and complex document structures remains significant.
2. **JSON and YAML:** JSON (JavaScript Object Notation) and YAML (YAML Ain't Markup Language) have become popular alternatives to XML for data serialization due to their simplicity and readability. They are heavily used in configurations, APIs, and data interchange formats.
3. **Markdown:** Markdown has gained popularity for its simplicity in formatting text. It's widely used in documentation, content management systems, and static site generators. Extensions and variations (like CommonMark and GitHub Flavored Markdown) continue to evolve, improving its functionality and versatility.

4. **New Formats and Standards:** The future might see the emergence of new markup languages or standards that address specific needs. For instance:
    - WebAssembly: Although not a markup language per se, WebAssembly might influence how we think about integrating different kinds of content and functionality.
    - Rich Data Formats: New formats could emerge for richer data representation, combining aspects of traditional markup with modern needs.

5. Declarative Languages: The rise of declarative programming paradigms might lead to new markup languages that focus on defining "what" should be done rather than "how" to do it. This can simplify complex interactions and data structures