# Comprehensive Web Design

## and Introductory Drupal

Preston So

New Media Denver
Preston So Design

2009 Special Edition for
Rocky Mountain Robotics

# Table of Contents

# Foreword

I wrote this text with the intention of providing a highly topical and profound analysis of web design and its mechanics. Students of introductory web design at both the high school and collegiate level are frequently exposed to obsolete course material that does not have currency in contemporary situations. Instead of utilizing a progression whereby the code is learned from the outset of the text, I have chosen to establish a brief yet firm foundation regarding web design and its state today, including a very detailed focus on web standards and accessibility. This text is written with the intention of retaining its relevance for a lengthy period of time, at the very least until the widespread adoption of HTML 5. To this end, a discussion of content management paradigms, semantics, and Drupal has been included. The instructor may choose to supplement the first three parts of the content with a different content management system, but Drupal has been selected due to its increasingly commonplace nature and overall powerful functionality. It is by no means an exhaustive study of Drupal, and focus is drawn much more to the underpinnings of content management and semantics rather than the structure of Drupal.

The actual content of this text moves very rapidly in order to introduce students to concepts quickly and supportively with a limited amount of wasted time. Students are expected to already be aware of web-based technologies (including social networks and search engines) and to have a general understanding of how to browse the Internet. Web design is a field filled with unnecessary jargon. Special attention has been devoted to simplifying certain terminology and its definitions to facilitate its absorption and retention. To this end, I have formulated this content with the following three objectives:

- to ensure that students have a solid and comprehensive understanding of web standards, web accessibility, and the state of web design today
- to provide a deep and highly specific treatment of XHTML and CSS as well as methods for embedding external material
- to elucidate the relevance of content management systems and semantics in contemporary web design, the decreasing importance of maintenance, and a treatment of Drupal on a conceptual rather than topical foundation

Students will find a relatively low number of examples in this book that treat each coding concept. This is due to the fact that this should be an independently guided learning process, with students applying and practicing their knowledge in their own projects.

**Preston So** has worked in web design since 2001. His blog *Seraphic Zephyr* was a vocal advocate of web standards and accessibility from 2005 to 2007 and was featured in the design gallery Screenspire. In 2008, he became creative production lead for Monarch Digital, a multimedia studio in Colorado Springs. He is a leading member of the Drupal community as founder and manager of the Southern Colorado Drupal User Group (SCUG) and founding member of Design in Drupal (D4D). His session "Theming Out of the Box: Designing in Drupal" was well-received at DrupalCamp Colorado 2008 in Denver and at DrupalCon DC 2009 in Washington, D.C. Currently, Preston is a developer at New Media Denver and manages Preston So Design, his freelance base.

# Note to the student

Although this book moves fairly rapidly, it's my intention to provide you with the resources and skills to pursue web design on your own time for your own needs. You may need to read the sections regarding code several times to understand the dynamics of what you are seeing. It's highly recommended that you have a monospace text editor (e.g. Notepad or Adobe Dreamweaver) handy to try these examples for yourself. There is a limited number of examples in this book in order to treat as many concepts as possible, so it's imperative that you practice independently on a regular basis and supplement the examples in the content with your own ideas and inventions. This is how I learned web design.

PRESTON SO
Colorado Springs

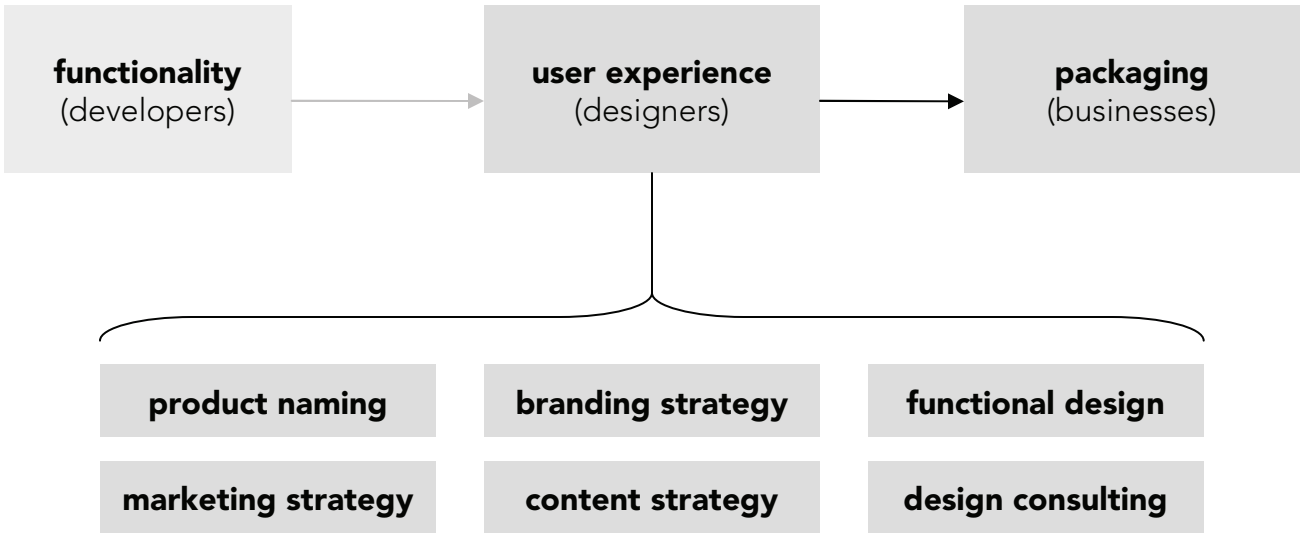


**Fig. 1.3. Where designers reside in project continuums.** Designers may be considered the middlemen for a project, adding creative insight to a project that already yields significant functionality, or they may be responsible for the project in its entirety.

How do web designers work in the real world? To answer this question, we must consider the examples of designers in the past and present. The web design community is a vibrant and flourishing group of individuals who collaborate on various issues and voice their concerns about weaknesses in design or development practices. Traditionally, a web designer can either be a middleman between a developer and business interests or the only individual present who creates a product *(Fig. 1.3)*. In either case, web designers are responsible for maintaining the *creative integrity* of any project.

You do not need to have any knowledge to be a designer. According to Norman Potter, "Every human being is a designer. Many also earn their living by design—in every field that warrants pause, and careful consideration, between the conceiving of an action and the fashioning of the means to carry it out, and an estimation of the effects."[1]

**Checkpoint, 1.1**

1. Define the term **web designer**.
2. Find an example of poor functional design (e.g. the door). How would you improve it to ensure that its end users ascertain an ideal user experience?
3. Sketch a rough structure for a website that sells T-shirts. What elements would you include? How would you position these elements to maximize the user experience?

**RECOMMENDED READING**
*Taking Your Talent to the Web*, **Jeffrey Zeldman**. A classic guide to the new medium known as the web and how you can harness your creativity in online endeavors.

**Fig. 1.4. Sir Tim Berners-Lee.** The inventor of the World Wide Web, whose independent contract work at CERN led him to knighthood. (Photo by Uldis Bojars.)

## 1.2 Web standards and accessibility

The World Wide Web (namely, the web) as we know it is the brainchild and invention of Sir Tim Berners-Lee *(Fig. 1.4)*. Websites as they are referred to today first began to appear as a popular medium in the mid-1990s. Through the remainder of the decade those who created websites did so using a loose set of guidelines supplemented by proprietary additions from Microsoft (the creator of Internet Explorer) and Netscape (now essentially nonexistent).

Berners-Lee founded the *World Wide Web Consortium* (W3C), a group of individuals committed to bringing standardization to the web design and development communities. Its initial recommendations revolved around improving HTML (Hypertext Markup Language), which eventually adopted various forms.

[1] Potter, Norman. *What is a designer*, 4th ed. London: Hyphen Press, 2009.

**web standards** solidified guidelines that dictate how languages used in web design should be written

The World Wide Web Consortium works to disseminate the benefits of **web standards**, solidified guidelines that dictate how languages used in web design should be written. They are the equivalents of English grammar books, and due to the efforts of the Consortium and standards advocates, they are well on the way to complete adoption among all newly created websites. There exist many benefits to using web standards. One unified, consistent set of rules allows many people working on the same system to adopt a common methodology. Much in the same manner that international networks of telecommuters use English as a *lingua franca*, the concept of web standards dispels inconsistencies from the language used by all involved parties.

Although the W3C was responsible for the initiation of the grassroots web standards movement, the adoption of standards was stagnant for many years. Microsoft and Netscape, the two companies with the largest shares of the web browser market, were slow to adopt the standards released by the W3C and continued to advocate the utilization of their own respective proprietary methodologies. Standards advocates fought to ensure that browser companies accepted the legitimacy of the W3C standards, and in a series of hugely influential steps, people such as Jeffrey Zeldman created websites that were dysfunctional in Internet Explorer and Netscape but followed good practices. The gradual acceptance of web standards came in the early 2000s as designers worldwide began to understand the importance of a consistent way of coding their websites.

**IN DEPTH**
Apart from XHTML 1.0 Strict, there exist many other standards that the W3C has released. Among these are HTML 4.01 Transitional, likely the most utilized on the web, and XHTML 1.0 Transitional, which represents a "middle ground" between HTML and XHTML (see 1.3).

In this book, we will not be treating proprietary code proliferated by the likes of Microsoft and Netscape as being *valid*. **Validation** involves ensuring that produced code follows the standards released by the W3C. All of the code that is presented in the examples within this book is valid and follows the XHTML 1.0 Strict standard that the W3C maintains. Although there are less stringent standards available, XHTML 1.0 Strict is the most forward-looking and recommended form of the language.

**APPLY IT!**
There are many tools for validation on the web today, including the W3C's own, located at ***validator.w3.org***. Try validating several websites to see what standard they use and if they have committed any errors.

By 2003, close to when the dot-com bubble burst, a majority of web designers began to tout the benefits of web standards. There remained, however, one additional battle. Although standards provided for a universal interpretation of code, the methods that browsers used to interpret code for disabled users were wildly disparate or even nonexistent. In the United States, the Americans with Disabilities Act (ADA) dictates how public facilities should be, *without exception*, accessible to everyone.

**validation** the confirmation that produced code follows the standards released by the W3C

The same concept applies to the web. Around this time, advocates for disabled end users of websites began clamoring for increased **web accessibility**, the idealist philosophy that every user of every condition should have access to have the same experience as every other user. In the W3C, a working group known as the WCAG works to facilitate the same user experience for every user regardless of ability or disability. As access to the web becomes less and less selective, the concept of web accessibility and its contemporary relevance becomes increasingly important. Web designers must take the disabled into consideration when working on their own designs; it is only fair.

**web accessibility** the idealist philosophy that every user of every condition should have access to the same experience as every other user

But how do the disabled browse websites? There are many invisible mechanisms that are in place to assist the disabled as they peruse the web. Able-bodied users of the Internet are seldom aware of the variety of available options that attempt to provide the same experience for everyone. *Screen readers* are devices that read the text of websites aloud for users who have difficulty seeing. A feature that you may have seen in your browser of choice is a *text zoom* capability, which allows for legally blind users to increase the size of text on the screen for improved legibility. Many conventions in web standards have been included by the W3C to anticipate disabled users, such as alternative text for images and provisions for captioning audio files and videos. Although these are excellent tools for content, embedded objects such as Flash animations, Java applets, and others are unattainable for users who are legally blind. Web accessibility remains a significant obstacle to experiential universality, and there are few permanent solutions that even approach this trait.

The code examples provided in this book are meticulously created with a high level of attention toward web accessibility. It is important that as you consider your designs you respect the rights of the disabled. The two ways of treating web accessibility are:

1. *Use web standards and conventions for accessibility*. NEVER deviate from web standards, and your website will almost invariably be accessible.

2. *Use accessibility tools and validators*. There are many tools and validators that allow you to simulate how disabled users will see your website, including free screen readers and online validation tools.

## **Essay** Legislating Web Accessibility

THE NATURE OF THE WEB THESE DAYS IS DAUNTING TO THOSE PEOPLE who require assistance when they browse from other programs or sources. It is for these disabled people that we need to fix the problems we have created and allow a fully accessible and universal web, open to everyone, to thrive.

One prominent case regarding accessibility is *Maguire vs. Sydney Organizing Committee for the Olympic Games* (SOCOG) in 1999. Joe Clark writes about it in an article from 2001 (these times give you an idea of how early accessibility has been notable) and offers: "This case teaches us that the legal need for accessibility is so clear-cut, and the means of achieving basic accessibility so straightforward, that even an unspeakably wealthy and powerful international organization can lose in a judicial proceeding."

Bruce Maguire, in June 1999, lodged a complaint with the Australian Human Rights and Equal Opportunity Commission (HCEOG), stating that the SOCOG's site violated the Australian Disability Discrimination Act (DDA). Maguire stated that, as he is legally blind, the site of SOCOG should be entirely accessible. He does not use a screen reader, but rather, a "refreshable Braille display." However, images lacking an `alt` attribute were unable to be identified with his technology. SOCOG's site, apparently, had many, and Maguire alleged that significant amounts of the site were unavailable for his use.

On August 24, 2000, the HCEOG announced that it would support Maguire and ordered SOCOG to provide, by September 15, an accessible version of its site, under the following guidelines:

1. that SOCOG include alt text on all images and image map links on the site
2. that SOCOG ensure access from the Schedule page to the Index of Sports
3. that SOCOG ensure access to the Results Tables on the site during the Olympic Games

SOCOG ignored this order, and was consequently fined AUS$20,000.

SOCOG then made the mistake of saying that "access to the Index of Sports from the Schedule was available and had always been available by a different route; namely, by entering the URL for each sport directly into the Web browser." Now, where would SOCOG be able to provide a list of said URLs? Would visitors be expected to know potentially lengthy URLs offhand?

In this case, Maguire won, and Clark states, "In any event, in the Maguire case we now have a firm worldwide precedent that inaccessible Web sites can be and are illegal." *Maguire vs. SOCOG* was decided in Australia, but the United States and Canada are very close in nature. And this happened in 1999 and 2000. Where is the change? Why is there no increase in Web accessibility?

According to Mazen Basrawi of the Disability Rights Advocates in Berkeley, the addition of websites to the ADA will not come until a Supreme Court ruling is made.

The Baltimore-based National Federation of the Blind (NFB) and California student Bruce Sexton sued the retail store giant Target for providing an effectively inaccessible website in February 2006, citing the following motives:

- alt-text is missing from images, preventing screen readers from describing them to blind users
- purchases cannot be completed without a mouse because keyboard controls do not work
- image maps are inaccessible
- headings are missing that are needed to navigate

As opposed to Bruce Maguire, whom we heard about previously, the NFB took a different approach, attacking the fact that Target offered a store locator, printable coupons, and an online pharmacy of sorts where customers could order drugs and then pick them up later at Target stores. The NFB alleged that the stores were accessible public accommodations and therefore they should be accessible on the website as well.

Judge Marilyn Hall Patel disagreed with Target's argument that websites were not included by the ADA at all, saying, "The statute [in the ADA] applies to the services of a place of public accommodation, not services in a place of public accommodation. To limit the ADA to discrimination in the provision of services occurring on the premises of a public accommodation would contradict the plain language of the statute." Judge Patel ultimately decided that the case could proceed, but did not clarify whether the ADA also blanketed websites.

In the United Kingdom, however, web accessibility is required on every government site. They must be usable and accessible. There is no argument for such a law in the United States. After all, the United Nations recently published a draft International Convention on the Rights of People with Disabilities, which includes provisions for web accessibility.

One clause in the aforementioned Convention requires that countries "take appropriate measures to ensure to persons with disabilities access, on an equal basis with others, to ... information, communications and other services, including electronic services."

Another clause reads, "Private entities that provide services to the general public, including through the Internet, [should] provide information and services in accessible and usable formats for persons with disabilities."

Shouldn't this be included immediately in the ADA? Not so fast, says a spokesman for the U.S. State Department, who stated that the ADA provides adequate protection and that they would refuse to sign the UN Convention. Forms of accessibility have been around since 1994. Why does the U.S. not endorse Web accessibility?

It should be entirely evident that the United States and the world need to provide effective provisions for disabled people on websites. It's time to realize that we need web accessibility.

Do we consider disabled people to be less than ourselves, us privileged users who have the ability to actually see the screen, rather than having it read for us? If we want to initiate inferiority once more in the United States, then let us ignore these rulings. Let us ignore the UN Convention. Let us ignore the proof. Let us ignore the fact that in a UK study, 97% of the sites under the jurisdiction of the European Union provide little to no concessions for accessibility. Let us ignore that blind European users are restricted to 3% of all EU sites.

Why are we so slow in opening the web to the people who may truly need it most yet find themselves against the largest of obstacles?

*Revised edition of an article from November 4, 2006.*

**Checkpoint, 1.2**

1. Why are web standards important?
2. Why is web accessibility important?
3. Download a free screen reader and browse the Internet for one day using the tool. What do you notice? Does this change your perception of web accessibility?

## 1.3 Elements in XHTML

We have discussed the foundational concepts of modern web design in the context of how web designers interpret the web. Now let us apply this knowledge to coding.

**XHTML** a language used in web design that focuses on the value of elements within a structure

**XHTML** (Extensible Hypertext Markup Language) is a language used in web design that focuses on the value of elements within a structure. Web designers often use the term XHTML interchangeably with HTML; the distinction between the two lies in several nuances in coding and differences in how the standards are described by the W3C. Files written in XHTML are usually saved with the extension *.htm* or *.html*. Editing XHTML is done using a plain-text monospace editor (such as Notepad or Adobe Dreamweaver), although other methods exist, namely WYSIWYG (pronounced *wiz-ee-wig*) editors.

**IN DEPTH**
WYSIWYG (What You See Is What You Get) is the blanket name for a category of HTML editors that allow the user to manipulate elements on a web page without having to see any code. The editor's screen updates continuously with a live generated preview.

XHTML is comprised of **elements** (also known as **tags**), definitions of specific contextual areas that adopt a set of characteristics based on their respective names. Elements in XHTML take the form

```
<element> [content] </element>
```
EX. 1.3.1 *

**SYMBOLOGY**
Examples shown hereafter with an asterisk (*), e.g. "EX. 1.3.1 *", are not correct code and will not yield a functional output if saved. Examples shown hereafter with × are not correct code in any case. Examples without asterisks or ×'s are functional. "RES." denotes the result of some code in the browser.

The element begins with its name within angle brackets. This is the *initial tag*, and it denotes that the element is beginning at this point. Elements more often than not have *content* that follows the initial tag. This can take the form of plain text or other tags, as seen in EX. 1.3.2. After the content is the *terminal tag*, which denotes that the element is ending at this point. As you can see, the slash represents that the element ends. Forgetting to include the slash in the terminal tag will tell the browser interpreting your code that you wish to begin another element of the same name.

```
<element1> <element2> </element2> </element1>
```
EX. 1.3.2 *

**element** a definition of a specific contextual area that adopts a set of characteristics based on its name; also known as a **tag**

In this example, the elements are *nested*, meaning that they are within each other. *Element1* surrounds *element2*, which *must* be terminated before *element1* can be terminated. The following is incorrect:

```
<element1> <element2> </element1> </element2>
```
EX. 1.3.3 ×

There are two basic types of elements in XHTML: *block-level* and *inline*. It is imperative that you understand this distinction. **Block-level elements** are elements that occupy a "block" of space. For example, the examples of code above occupy a block of space; they are *block-level*. **Inline elements** are elements that are located within an area of text or *contextually*. For example, the italicized word *contextually* at left is within a "line" of text; it is *inline*.

**block-level element** an element that occupies a "block" of space

**inline element** an element that is located contextually within an area of text

Some elements in XHTML are *presentational*, meaning they affect how the content looks in the browser. For example, the *strong* tag boldfaces the content as you can see here:

**NOTE**
In EXS. 1.3.1-3, spaces were included between the tags. This is not necessary in practice. The spacing shown in EXS. 1.3.4-8 is valid, correct, and ubiquitously understood.

```
<strong>bold</strong>
```
EX. 1.3.4

**bold**
RES. 1.3.4

On the contrary, the *em* tag italicizes the content as follows:

```
<em>italic</em>
```
EX. 1.3.5

*italic*

RES. 1.3.5

Both the *strong* and *em* tags are *inline*. Remember that tags can be *nested*:

```
<strong><em>bold and italic</em></strong>
```
EX. 1.3.6

***bold and italic***

RES. 1.3.6

Here we introduce our first *block-level* element, the *p* tag. *p* denotes a paragraph. It is logically block-level because it occupies a block of space.

```
<p>Hello, world!</p>
```
EX. 1.3.7

Hello, world!

RES. 1.3.7

Another extremely important distinction can be made between block-level and inline elements: Inline elements may be nested inside block-level elements, but block-level elements can never be nested inside inline elements. In EX. 1.3.8, notice that the inline *strong* and *em* elements are nested within the block-level *p* element.

```
<p><strong>Imagination</strong> is more important
than <em>knowledge</em>.</p>
```
EX. 1.3.8

**Imagination** is more important than *knowledge*.

RES. 1.3.8

You may try EXS. 1.3.4-8 in your own *.html* file and save it. Although the code will not be valid (some key components are absent), modern browsers will still interpret the code correctly. Notice that in RES. 1.3.4-8, the text is rendered in 12-point Times New Roman. This is the default font and size for most contemporary browsers.

**Checkpoint, 1.3**

1. What is the difference between block-level and inline elements?
2. What does it mean when elements are nested?
3. Give the code for the following results.

I *think*, therefore I **am**.
**Tim Berners-Lee** invented the *web*.
***Ciao!***
*The New York **Times***

**IN DEPTH**
You may have worked with HTML before in which the tags for boldfacing and italicizing were *b* and *i*. This is technically correct and valid in HTML 4, but XHTML attaches semantic value to the names for elements (also see 4.1). *strong* and *em* have much more meaning to the external viewer than do *b* and *i*.

## 1.4 Attributes and values in XHTML

We now have the capability of defining elements in XHTML. But what if we want to assign characteristics to these elements? And what if we want to be able to identify elements that have the same name? In this case, we must use attributes and values. An **attribute** is a trait of an element that changes its character in some fashion. In XHTML 1.0 Strict, every attribute must have a **value**, which is a denotation that shows the degree to which the attribute is true. Values may be boolean (true or false), infinitely numerical (0 to infinity), or descriptive (red, blue, etc.).

**attribute** a trait of an element that changes its character in some way

**value** a denotation that shows the degree to which an attribute is true

Consider the following structural example:

```
<element attribute="value"> [content] </element>
```
EX. 1.4.1 *

**NOTE**
It is always a good idea, in the course of your own web design efforts, to have multiple browsers handy and a screen reader to preview code in various situations. This is highly useful in debugging problems. Firefox, Internet Explorer, Opera, and Safari are among the major browsers that you should retain in your arsenal.

- Internet Explorer (www.microsoft.com/ie)
- Firefox (www.firefox.com)
- Opera (www.opera.com)
- Safari (www.apple.com/safari)

Notice that an equal sign separates the attribute from the value. This is logical, as a value is meant to demonstrate the extent of the attribute. Also note the quotation marks around the value. Always include these quotation marks, as they are part of the standard.

An element may have multiple attributes, which each have their own respective values. A perfect example of this is the *a* tag, which defines a hyperlink.

```
<a href="http://www.example.com" title="Link to
example.com">Link</a>
```
EX. 1.4.2

Link
RES. 1.4.2

The *a* element in this case has two attributes: *href* and *title*. The *href* attribute states the destination of the link, in this case a URL. The value of the *title* attribute as applied to hyperlinks is the text displayed in a box when you move your cursor over a link, also known in Microsoft as a *tool tip*. Tool tips in XHTML are always title attributes. Notice in EX. 1.4.2 that the attribute-value couplings are separated by single spaces.

Another important element in XHTML is the *img* element, which follows:

```
<img src="http://www.example.com/image.jpg"
alt="An African elephant" title="They have good
memories!" />
```
EX. 1.4.3


RES. 1.4.3a

An African elephant
RES. 1.4.3b

The *src* attribute uses a URL, just like the *href* attribute. In this case, we link directly to an image file (such as a JPEG). The *alt* attribute is used when the image cannot be displayed. This can be for one of several reasons: *a)* the image file is unavailable or inaccessible for whatever reason; *b)* the technology being used cannot display images (increasingly rare); or *c)* a screen reader or other accessibility device is being used and must read the text.

You *must* include *alt* text, as the *alt* attribute-value pairing is commonly known, in every single *img* element you use. As you can see in RES. 1.4.3a, the image is displayed as normal, but in RES. 1.4.3b, the *alt* text replaces the image. Try it for yourself by saving EX. 1.4.3 as a *.html* file, opening it in your browser, and turning off image display in your browser's preferences.

Notice also that the structure for this element is different. In the case of elements that lack any content contained within the tags, the slash comes before the final angle bracket. We will be returning to this essential discussion when we introduce later elements.

**phrase element** an inline element that modifies how its textual content is interpreted

There is a subset of inline elements that we will call **phrase elements** in this book. Phrase elements are solely inline elements that modify how its textual content is interpreted, whether that be by emphasis (*strong*, *em*) or clarification of the contextual importance of some portion of text. Let us now treat the phrase elements *abbr*, *acronym*, *sup*, *sub*, and *q*:

**NOTE**
The *q* tag is not supported by Internet Explorer, while the *abbr* tag is not recognized by Internet Explorer 6 and earlier.

```
<p>The <acronym title="Organization of Petroleum
Exporting Countries">OPEC</acronym> reported that it
had found a 24,000 km<sup>2</sup> area of oil
deposits with high levels of <abbr title="carbon
dioxide">CO<sub>2</sub></abbr>. <q
cite="http://www.un.org">This discovery is one that
is unprecedented in the history of petroleum,</q>
said a spokesman.</p>
```
EX. 1.4.4

The OPEC reported that it has found a 24,000 $km^2$ area of oil deposits with high levels of $CO_2$. "This discovery is unprecedented in the history of petroleum," said a spokesman. RES. 1.4.4

This example demonstrates a sample length of code that might be seen in an XHTML file. Many element names are self-explanatory, as shown by *acronym* and *abbr*. The *sup* and *sub* tags denote superscript and subscript respectively, while *q* shows a brief inline quotation with its attribute *cite* containing the URL from which the quote was taken. Browsers automatically insert the quotation marks.

Attributes and their values are highly useful in specifying how certain text should be interpreted. They are also effective in helping designers identify elements based on a quality they wish to implement later. The *class* and *id* attributes serve to provide "names" for the elements they describe. Consider a long passage of text that has multiple paragraphs:

```
<p><strong>African elephants</strong> are species of
elephants of the genus <em>Loxodonta</em>.</p>

<p>African elephants are larger than Asian
elephants.</p>

<p>Elephants' tusks are teeth; the second set of
incisors become the tusks. They are used for digging
for roots.</p>

<p>Poaching significantly reduced the population of
African elephants in certain regions during the
twentieth century.</p>

<p>In conclusion, I love elephants!</p>
```

EX. 1.4.5

This passage of text has the result:

**African elephants** are species of elephants of the genus *Loxodonta*.

African elephants are larger than Asian elephants.

Elephants' tusks are teeth; the second set of incisors become the tusks. They are used for digging for roots.

Poaching significantly reduced the population of African elephants in certain regions during the twentieth century.

In conclusion, I love elephants! RES. 1.4.5

To help ourselves identify paragraphs that we may want to be concerned with later, we can use *class* and *id* to accomplish this. **Classes** are identifiers that can be used multiple times. This makes sense, as the definition of the word *class* is a category or common group of individuals. **Ids** (pronounced *eye-dees*), meanwhile, are identifiers that can only be used once in a document. They are unique to the element to which they are attributed.

As mentioned before, values can be descriptive. In the case of classes and ids, they are names that serve as reference markers for designers. In RES. 1.4.6, you can see that the text remains precisely the same, even though each paragraph in the code now has an identifier.


**Fig. 1.5. Illustrating usage of classes and ids.** The topmost elephant is the only African elephant in this menagerie and thus can have the id "african". Elements can have both a class and an id at the same time, as illustrated by the far bottom elephant, which can have the class "asian" (taking into account the one immediately above it) but the id "borneo", reflective of its subcategory.

**class** an identifier that can be used multiple times

**id** an identifier that is unique and can only be used once in a document

**A LOOK AHEAD**
Classes and ids have extremely important applications in creating layout (see 3.5), so try to develop a solid understanding of how they function now.

```
<p id="intro"><strong>African elephants</strong> are species
of elephants of the genus <em id="genus">Loxodonta</em>.</p>

<p class="physiology">African elephants are larger than
Asian elephants.</p>

<p class="physiology">Elephants' tusks are teeth; the second
set of incisors become the tusks. They are used for digging
for roots.</p>

<p id="conservation">Poaching significantly reduced the
population of African elephants in certain regions during
the twentieth century.</p>

<p id="editorial">In conclusion, I love elephants!</p>
```

EX. 1.4.6

**African elephants** are species of elephants of the genus *Loxodonta*.

African elephants are larger than Asian elephants.

Elephants' tusks are teeth; the second set of incisors become the tusks. They are used for digging for roots.

Poaching significantly reduced the population of African elephants in certain regions during the twentieth century.

In conclusion, I love elephants!

RES. 1.4.6

As you can see, the class *physiology* is utilized twice for two distinct paragraphs that discuss how the elephants look, but the ids *intro*, *genus*, *conservation*, and *editorial* are only used once. Thus, it is implied that there are multiple paragraphs that discuss elephant physiology but only one element of each trait intro, genus, etc. Using a class demonstrates your intention to reuse the class multiple times in the course of creating your page. Classes and ids comprise some of the most important applications of XHTML, as we shall see soon.

**Checkpoint, 1.4**

1. Define **attribute** and **value**.
2. What is the difference between **classes** and **ids**?
3. Give the code for a self-written paragraph containing a link to a website of your choice and an image from a source of your choice. Include a class and an id.

## 1.5 Lists and tables

**unordered list** a list lacking a specified sequence

**ordered list** a list assigning importance to the order of appearance

**definition list** a list consisting of definition-term pairs

With a solid structural understanding of how elements in XHTML function and are identified, we can proceed into the realm of applying different forms to different types of information. We have already treated phrase elements, inline elements that are contextual and inline. But it is imperative at this point that we delve into other organizational structures, namely lists and tables. We will discuss them in that order.

There are three types of lists currently in use: unordered, ordered, and definition. An **unordered list** is a list that has no specified sequence. An **ordered list** is a list that assigns importance to the order in which information appears. A **definition list** is a list consisting of definition-term pairs (such as a glossary).

Below are three examples that illustrate the different types of lists with their respective element names and structures:

```
<ul>
  <li>Burmese</li>
  <li>Arabic</li>
  <li>Czech</li>
  <li>Irish</li>
</ul>
```

EX. 1.5.1

- Burmese
- Arabic
- Czech
- Irish

RES. 1.5.1

```
<ol>
  <li>Mandarin</li>
  <li>Spanish</li>
  <li>English</li>
  <li>Hindi/Urdu</li>
  <li>Arabic</li>
</ol>
```

EX. 1.5.2

1. Mandarin
2. Spanish
3. English
4. Hindi/Urdu
5. Arabic

RES. 1.5.2

```
<dl>
  <dt>German</dt>
  <dd>a West Germanic language spoken in Germany</dd>
  <dt>Telugu</dt>
  <dd>a Dravidian language spoken in India</dd>
  <dt>Turkish</dt>
  <dd>a Turkic language spoken in Turkey</dd>
</dl>
```

EX. 1.5.3

German
: a West Germanic language spoken in Germany

Telugu
: a Dravidian language spoken in India

Turkish
: a Turkic language spoken in Turkey

RES. 1.5.3

The examples above demonstrate the wide variety of applications that lists maintain. Unordered lists are prevalent in many forms of content. Ordered lists have many potential uses, including rankings (RES. 1.5.2 shows the top five languages spoken by number of native speakers) and instructions (e.g. recipes). Definition lists are utilized most as a method to present terminology and definitions in a structurally relevant manner.

Another essential facet of content structures is that which comprises tabular data: tables. **Tables** are organized grids used to present statistics or other tabular information. Tables consist of several integral elements contained within a single umbrella *table*: *th* (table header), *tr* (table row), and *td* (table cell). There are several others, but they are not treated here. Tabular elements also have several important attributes, including *colspan*, *rowspan*, *summary*, *border*, *cellpadding*, and *cellspacing*.

**table** an organized grid used to present statistics or other tabular information

Consider the following example:

```
<table summary="Burrito preferences among age groups" border="1">
  <tr>
    <th>Age group</th>
    <th>Burrito preference</th>
  </tr>
  <tr>
    <td>10-14</td>
    <td>Alberto's Burritos</td>
  </tr>
  <tr>
    <td>15-17</td>
    <td>Monterey Mexican</td>
  </tr>
  <tr>
    <td>18-25</td>
    <td>El Burrito Blanco</td>
  </tr>
</table>
```

EX. 1.5.4

| Age group | Burrito preference |
|---|---|
| 10-14 | Alberto's Burritos |
| 15-17 | Monterey Mexican |
| 18-25 | El Burrito Blanco |

RES. 1.5.4

EX. 1.5.4 demonstrates how a simple table is structured. The *table* element, with its descriptive *summary* attribute, contains various *tr* elements, table rows, that in turn contain either *th* elements, table headers, or *td* elements, table cells. But we can expand the structures of tables to cater to different sets of data using *colspan* and *rowspan*. Also, we can control how tables look to some extent using *border*, *cellpadding*, and *cellspacing*.

The following example illustrates this.

```
<table summary="Burrito preferences among age groups" border="1"
cellpadding="5" cellspacing="5">
  <tr>
    <th>Candidate</th>
    <th>Affiliation</th>
    <th>Supported</th>
  </tr>
  <tr>
    <td rowspan="2">Dan Edmundson (Republican)</td>
    <td>Democratic</td>
    <td>26%</td>
  </tr>
  <tr>
    <td>Republican</td>
    <td>72%</td>
  </tr>
  <tr>
    <td rowspan="2">Nora Gomez (Democratic)</td>
    <td>Democratic</td>
    <td>66%</td>
  </tr>
  <tr>
    <td>Republican</td>
    <td>21%</td>
  </tr>
  <tr>
    <td colspan="2">Joel Yang (Independent)</td>
```

```
    <td>17%</td>
  </tr>
</table>
```

EX. 1.5.5

| Candidate | Affiliation | Supported |
|---|---|---|
| Dan Edmundson (Republican) | Democratic | 26% |
| | Republican | 72% |
| Nora Gomez (Democratic) | Democratic | 66% |
| | Republican | 21% |
| Joel Yang (Independent) | | 17% |

RES. 1.5.5

As seen in the example above, the *colspan* attribute determines how many columns a cell should occupy (in the case of the "Joel Yang" cell, two). The *rowspan* attribute, meanwhile, determines how many rows a cell should occupy (in the case of the "Dan Edmundson" cell, two). *Cellpadding* describes the amount of space that should separate cell content from its border, and *cellspacing* describes the amount of space that should separate each cell from another.

HTML was created with an effective system for organizing and representing data that is manifested in the remarkable simplicity of the list and table elements described. Many types of data can be described using HTML's capabilities. We have only touched on the aesthetic flexibility of lists and tables, for many of the extensive attributes utilized are beyond the scope of this book.

So far, we have discussed solely the structural aspects of HTML in representing content, whether this content take the form of text or statistical data. We have also developed the ability to establish emphasis and other inline modifications to contextual elements. In part 2, we will delve into how to extend this structural basis into a more aesthetic, presentational realm. ■

**Checkpoint, 1.5**

1. What is the difference between an (un)ordered list and a definition list?
2. What are the basic elements that comprise a simple table?
3. Find a data table on a website of your choice and replicate its contents precisely without viewing the source code.

## Summary

Web design is the application of principles of design to provide an interactive online experience that is informative or entertaining. Those who practice web design are responsible for the creative integrity of the projects in which they participate. XHTML, the language of web design, is made up of set standards that aid accessibility.

Elements, their attributes, and attributes' values comprise the structure of XHTML. Inline elements are elements that are contextually situated and cannot contain block-level elements, which occupy a block of space. Unordered lists, ordered lists, definition lists, and tables are useful in an unending variety of real-world applications.

**A LOOK AHEAD**

The term *padding* will be used on a very frequent basis in discussions regarding layout, part 3. Familiarize yourself with a structural understanding now.

**VOCABULARY**

web design, *p. 6*
web development, *p. 6*
user experience, *p. 6*
web standards, *p. 8*
validation, *p. 8*
web accessibility, *p. 8*
XHTML, *p. 10*
element (tag), *p. 10*
block-level element, *p. 10*
inline element, *p. 10*
attribute, *p. 11*
value, *p. 11*
phrase element, *p. 12*
class, *p. 13*
id, *p. 13*
unordered list, *p. 14*
ordered list, *p. 14*
definition list, *p. 14*
table, *p. 15*

**ELEMENTS**

strong, *p. 10*
em, *p. 10*
p, *p. 11*
a, *p. 12*
abbr, *p. 12*
acronym, *p. 12*
sup, *p. 12*
sub, *p. 12*
q, *p. 12*
ul, *p. 15*
li, *p. 15*
ol, *p. 15*
dl, *p. 15*
dt, *p. 15*
dd, *p. 15*
table, *p. 16*
tr, *p. 16*
th, *p. 16*
td, *p. 16*

# 1 Exercises

**1–2**
Respond to the following. *(1.1–2)*

1. Describe an example of good user experience.
2. Why is web accessibility important?

**3–8**
Correct the errors in each of the code examples below. *(1.3–4)*

3. `<p>I think there is no better activity than <strong>mountain biking</p></strong>`
4. `<a href=http://news.bbc.co.uk title=BBC World News>News from the United Kingdom</a>`
5. `<img src="african_elephant.jpg" alt=""></img>`
6. `<p>The <abbr title="UN">United Nations</abbr> does a lot of good work.</p>`
7. `<strong id="heavy">This</strong> is bold just like <strong id="heavy">this</strong>.`
8. `<a href=http://www.amazon.com title="Amazon"><p class="highlight">Amazon.com</a></p>`

**9–11**
Provide code for each of the descriptions below. *(1.3–4)*

9. A paragraph stating "Hello, world!" with "Hello" as a link to Google and "world" as a boldfaced word.
10. Two images of your choice, the first with id "image-a" and class "image", and the second with class "image".
11. The chemical equation $6CO_2 + 6H_2O = C_6H_{12}O_6 + 6O_2$ in one paragraph, followed by another paragraph stating, "This equation is important in the AP Biology course administered by the CEEB." AP stands for Advanced Placement, and CEEB stands for College Entrance Examination Board.

**12–14**
Provide code for each of the descriptions below. *(1.5)*

12. Create a definition list with all of the vocabulary words in part 1. You can print it and use it as a study aid.
13. Find the graduation rate for your institution and compare it in a table with others around the region.
14. A table of your choice that contains at least one example of *colspan* and *rowspan* each.

**15–24**
Name the elements used that have the described characteristics. *(1.3–5)*

15. a paragraph
16. an unordered list with three items
17. a table cell
18. an inline superscript
19. an inline quotation
20. a definition
21. a numbered list item
22. an image
23. a link
24. italicization

**25–30**
Answer the following questions. *(1.3–5)*

25. Why can inline elements not surround block-level elements?
26. How does *cellpadding* differ from *cellspacing*?
27. Why does *img* lack a terminal tag?
28. What are the types of values attributes can have?
29. What are phrase elements?
30. Marty wants to rank his favorite football teams on his website. Which type of list would he use, and why?

**31–35**
Give the code that generates the following results. *(1.3–5)*

31. Result where "Adieu" has id "goodbye" and "s'il vous plait" has id "please":

*Adieu* and *s'il vous plait* are common expressions in **French**.

32. Result where image source is "zeldman.jpg" and the alt text is "Jeffrey Zeldman":

33. Result where cell padding and spacing are both default:

| Preference | Percentage |
|---|---|
| Blue | 40% |
| Red | 24% |
| Yellow | 36% |

34. Result where cell padding and spacing are both default:

| State Bill | Yea | Nay | Abstain |
|---|---|---|---|
| *419* | 52 | 40 | 8 |
| *420* | 24 | 51 | 25 |
| *421* | Removed from floor | | |
| *422* | Vetoed by governor | | |
| | 78 | 12 | 10 |

35. Result:

- Welsh
  1. Northern
  2. Southern
- Norwegian
  1. Nynorsk
  2. Bokmål
- Cherokee