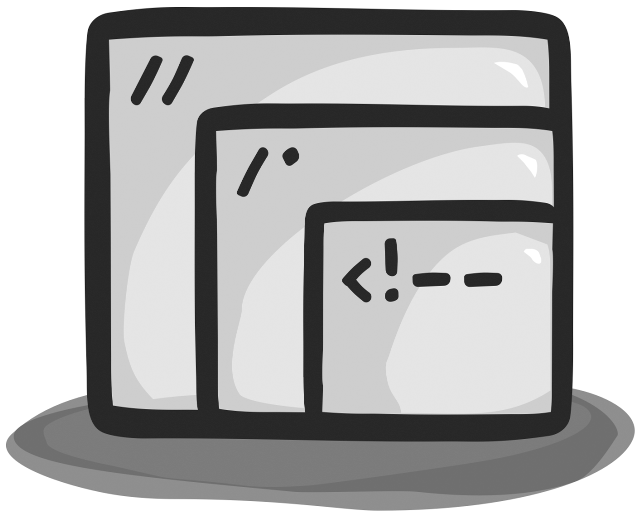


GYMNASIUM

INTRODUCTION TO MODERN WEB DESIGN

Lesson 3 Handout

Structuring HTML Content

ABOUT THIS HANDOUT

This handout includes the following:

- A list of the core concepts covered in this lesson
- The assignment(s) for this lesson
- A list of readings and resources for this lesson including books, articles, and websites mentioned in the videos by the instructor, plus bonus readings and resources hand-picked by the instructor
- A transcript of the lecture videos for this lesson

CORE CONCEPTS

1. Flow-level semantics in HTML are the elements that make up the body or main flow of a document, such as paragraphs <p> and lists <ul>,<ol>, <li>, etc.
2. Tabular data in HTML should be formatted with the <table> element and its various components such as rows, cells, headers, and footers. Properly structuring your tables will allow you to present data consistently and improve the accessibility of your documents.
3. Organizational semantics is the method of logically dividing and structuring your content with the appropriate tags including the <main>, <section>, and <article> elements, among others.
4. Other important organizational elements which you should use consistently are the <nav> element as well as the <header> and <footer> elements which help you divide your pages logically.
5. The role attribute is used to indicate an element's purpose within the page. Roles are part of the WAI-ARIA specification and help add additional context regarding the content on your page and help improve the accessibility of your pages.

ASSIGNMENTS

1. Quiz
2. Return to your HTML page that you created at the end of Lesson 2, and begin to organize your content using the knowledge gained in this lesson. For example, return to your main document and think about how you could break it up into multiple pages. Here's a typical workflow to help get you started (remember this is just an example; your assignment is to create your own pages from a Wikipedia article of your choosing):
    - Let's say you wanted 3 pages in your site. You would duplicate your original HTML document two times and then rename each copy appropriately for the content it contains. So for example, in addition to an index.html page, you would rename your two pages ecology.html and physiology. html. Next, you would go through each of these duplicates and remove any content that should live on the other pages.

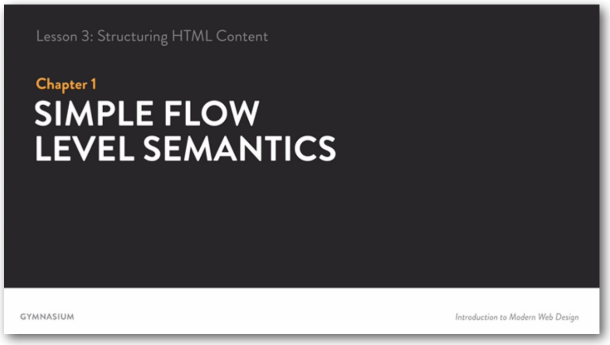
- Now you would add some navigation so that you can move between the different documents. (If necessary, you might want to review Lesson 2, Chapter 6 (URLs and Links) for a refresher on how to link from document to another.)
- Finally, go through all of your documents and add the logical flow and structural elements that were discussed in this lesson, such as <main> and<section> elements.
- When complete, publish your work up to GitHub and post a link in the forum.

RESOURCES

- References for the various list elements (<ul>,<ol>,<li>, <dl>, <dd>, and <dt>)
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/li
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dl
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dt
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dd
- The document heading elements
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements
- The table element and its supporting cast for displaying tabular data
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Table_content
- A reference for the content sectioning elements such as <article>, <header>, and <footer>
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content_sectioning
- A look at the details of the <nav> element
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav

INTRODUCTION

(Note: This is an edited transcript of the Modern Web Design lecture videos. Some students work better with written material than by watching videos alone, so we're offering this to you as an optional, helpful resource. Some elements of the instruction, like live coding, can't be recreated in a document like this one.)


Lesson 3: Structuring HTML Content

Chapter 1

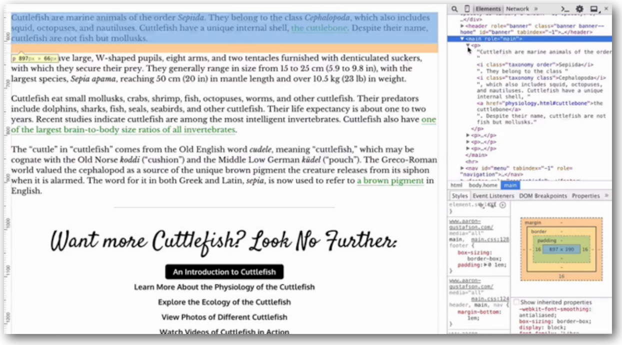
SIMPLE FLOW LEVEL SEMANTICS

GYMNASIUM

Introduction to Modern Web Design

In Lesson 3, we're going to talk a little bit more about HTML, moving out of the individual word and phrase elements to what we call flow-level semantics. In this first chapter, we're going to talk about simple flow-level semantics, starting with paragraphs.

Paragraphs, as you know, break up content and make it a little bit easier to scan. So if we take a look at a simple paragraph here—"Cuttlefish have large, W-shaped pupils," et cetera, et cetera—this is how we create line breaks between this piece of content and other pieces of content on the page.


Cuttlefish are marine animals of the order *Sepiida*. They belong to the class *Cephalopoda*, which also includes squid, octopuses, and nautiluses. Cuttlefish have a unique internal shell, the cuttlebone. Despite their name, cuttlefish are not fish but mollusks.

p 897px × 66px …ave large, W-shaped pupils, eight arms, and two tentacles furnished with denticulated suckers, with which they secure their prey. They generally range in size from 15 to 25 cm (5.9 to 9.8 in), with the largest species, *Sepia apama*, reaching 50 cm (20 in) in mantle length and over 10.5 kg (23 lb) in weight.

Cuttlefish eat small mollusks, crabs, shrimp, fish, octopuses, worms, and other cuttlefish. Their predators include dolphins, sharks, fish, seals, seabirds, and other cuttlefish. Their life expectancy is about one to two years. Recent studies indicate cuttlefish are among the most intelligent invertebrates. Cuttlefish also have one of the largest brain-to-body size ratios of all invertebrates.

The "cuttle" in "cuttlefish" comes from the Old English word *cudele*, meaning "cuttlefish," which may be cognate with the Old Norse *koddi* ("cushion") and the Middle Low German *küdel* ("pouch"). The Greco-Roman world valued the cephalopod as a source of the unique brown pigment the creature releases from its siphon when it is alarmed. The word for it in both Greek and Latin, *sepia*, is now used to refer to a brown pigment in English.

Want more Cuttlefish? Look No Further:

An Introduction to Cuttlefish

Learn More About the Physiology of the Cuttlefish

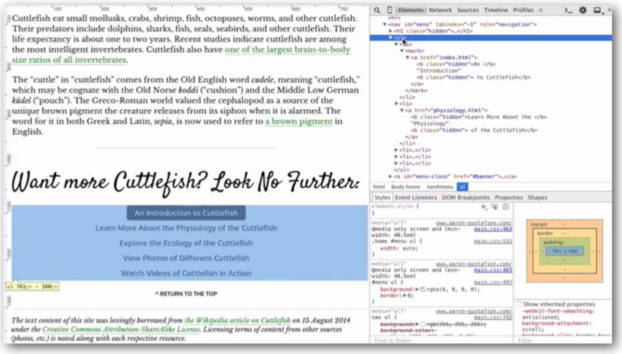
Explore the Ecology of the Cuttlefish

View Photos of Different Cuttlefish

Watch Videos of Cuttlefish in Action

So if we jump over to the browser really quickly, you can see here on the left-hand side that all of the paragraphs of content, even though I've put line breaks between them in the actual content, aren't separated. But once I put p elements around them, then all of a sudden we have these nice line breaks between each of those paragraphs. Paragraphs are probably the simplest of the flow-level semantics.

The next simplest are lists. There are a couple of different kinds of lists in HTML, the first two which make use of the li element for each of their list items. So here we have an example of two list items for two different video downloads, each contained inside of an li element.


Cuttlefish eat small mollusks, crabs, shrimp, fish, octopuses, worms, and other cuttlefish. Their predators include dolphins, sharks, fish, seals, seabirds, and other cuttlefish. Their life expectancy is about one to two years. Recent studies indicate cuttlefish are among the most intelligent invertebrates. Cuttlefish also have one of the largest brain-to-body size ratios of all invertebrates.

The "cuttle" in "cuttlefish" comes from the Old English word *cudele*, meaning "cuttlefish," which may be cognate with the Old Norse *koddi* ("cushion") and the Middle Low German *küdel* ("pouch"). The Greco-Roman world valued the cephalopod as a source of the unique brown pigment the creature releases from its siphon when it is alarmed. The word for it in both Greek and Latin, *sepia*, is now used to refer to a brown pigment in English.

Want more Cuttlefish? Look No Further:

An Introduction to Cuttlefish

Learn More About the Physiology of the Cuttlefish

Explore the Ecology of the Cuttlefish

View Photos of Different Cuttlefish

Watch Videos of Cuttlefish in Action

ul 761px × 160px

^ RETURN TO THE TOP

The text content of this site was lovingly borrowed from the Wikipedia article on Cuttlefish on 15 August 2014 under the Creative Commons Attribution-ShareAlike License. Licensing terms of content from other sources (photos, etc.) is noted along with each respective resource.

Now, if the order of these list items matters, then we want to mark them inside of an ol element, for ordered list. Here we see an example of that. But if the order doesn't matter, then we want to mark it up inside of a ul element for an unordered list. And this is what that looks like.

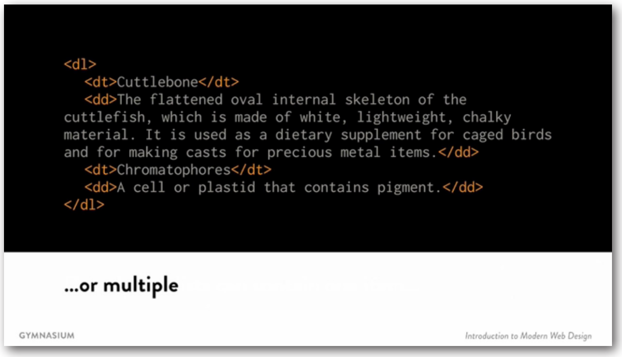
Now, the difference between these two semantically is that one, the order matters; the other, the order doesn't matter. From a visual display standpoint, browsers put numbers in front of ordered list items and bullets in front of unordered list items.

Now, in the example site I've prepared for you, I'm using list items for the navigation down here at the bottom. You can see the ul followed by a couple li's. It's all contained inside of a nav element, which we'll discuss a few chapters from now. And if I tweak the CSS just slightly, I can make the bullets appear.

Now, I've been using style sheets in order to make those bullets not appear, because I find them distracting. Now, using CSS, I had turned off the bullets. But if I disable the display inline block property on these list items, now all of a sudden, you see the bullets appearing next to them.

We have another list construct in HTML called description lists that are a little bit different than ordered and unordered lists. A description list is usually used for something like a glossary. So glossaries, as we know, have terms and definitions. So therefore, description lists have terms and definitions.

Terms are marked up in the dt element, and the definitions are marked up in a dd element. And then we wrap the whole thing in a dl element, and we get something that looks a little bit like this. Now, description lists can contain a single term and a single definition, or they can contain multiples. So here we see we've got Cuttleincremental and Chromatophores both defined within the same description list.



```
<dl>
  <dt>Cuttlebone</dt>
  <dd>The flattened oval internal skeleton of the
cuttlefish, which is made of white, lightweight, chalky
material. It is used as a dietary supplement for caged birds
and for making casts for precious metal items.</dd>
  <dt>Chromatophores</dt>
  <dd>A cell or plastid that contains pigment.</dd>
</dl>
```

…or multiple

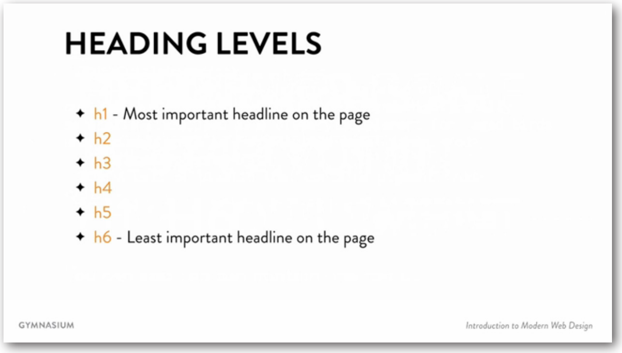
GYMNASIUM

Introduction to Modern Web Design

Now, there are instances where you might have multiple definitions for the same term, or multiple terms that share a definition. And description lists support this. You can have a dt followed by multiple dd's, or vice versa—multiple dt's followed by a dd.

Moving on from lists, the next type of flow-level semantic I want to talk to you about are heading levels. Now, headings come in six flavors, from the h1 to the h6 element. And the way that we use these is the h1 is the most important headline on the page, and the h6 is the least important headline on the page.

Now, when we use these elements in our markup, we create a natural document outline. Outlines help us to show how organized our content is, and they're also exposed to assistive technology in order to allow users to more quickly navigate through a document. Here's a quick example of an h1 followed by two h2's. And you'll notice in the outline at the bottom that Cuttlefish Physiology is the overarching headline and Cuttlebone and Skin are nested underneath it. That's how the outlining algorithm works.



HEADING LEVELS

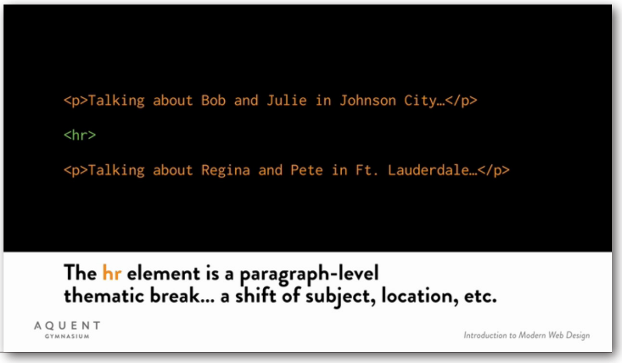
- h1 - Most important headline on the page
- h2
- h3
- h4
- h5
- h6 - Least important headline on the page

GYMNASIUM

Introduction to Modern Web Design

If you're interested in seeing the outlining algorithm, you can use the Web Developer toolbar, which is an extension for Chrome and Firefox. If you look in the upper right-hand corner of the browser here, there's a little cog. And when I click on that, it's going to show a little overlay, and I can go to the Information tab and click on View Document Outline. And that allows me to see how well-organized my page is. Looks pretty good to me.

Now, there are instances where you may not need a—heading when you're switching from one bit of content to another—and this is where the hr element comes in.



```
<p>Talking about Bob and Julie in Johnson City…</p>

<hr>

<p>Talking about Regina and Pete in Ft. Lauderdale…</p>
```

The hr element is a paragraph-level thematic break… a shift of subject, location, etc.

AQUENT GYMNASIUM

Introduction to Modern Web Design

Traditionally, the hr had been for a horizontal rule, but it's been recast in HTML as a paragraph-level thematic break. Now, what that means is that you're shifting the subject or the location, such as talking about Bob and Julie in Johnson City, and then switching to talking about Regina and Pete in Fort Lauderdale. It may not make sense to have a separate heading here, so an hr would be perfectly acceptable.

WHAT WE COVERED

- Paragraphs
- Ordered & unordered lists
- Description lists
- Headings
- Thematic Breaks

AQUENT GYMNASIUM

Introduction to Modern Web Design

And that brings us to the end of this chapter. We talked about paragraphs. We've discussed different types of lists, such as ordered and unordered lists, as well as description lists. We've discussed headings, and we've discussed thematic breaks. In the next chapter, we'll tuck into tables. I'll see you in a few minutes.

DEALING WITH DATA

Lesson 3: Structuring HTML Content

Chapter 2

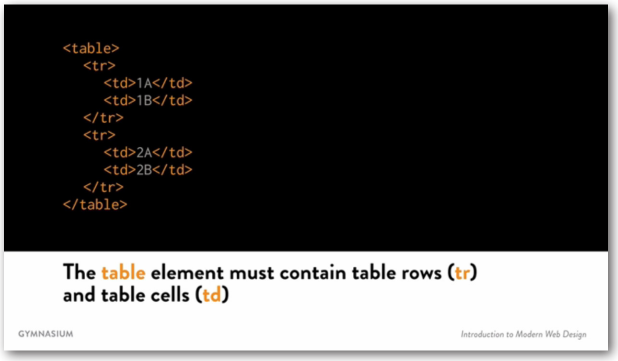
DEALING WITH DATA

GYMNASIUM

Introduction to Modern Web Design

Welcome back to Lesson 3, Structuring HTML Content. In this chapter, we'll be dealing with data—specifically data tables. When we're displaying tabular data, the table is absolutely the right choice in terms of a tag to use. Tables are a little bit more complicated in that there's a lot of elements that contribute to the overall table structure.

Here's a very simple table example containing two table rows, or tr elements, each containing two table cells, or td elements. So the tr, here in green, forms the row. And the individual td elements form the cells within that row. So this markup, all taken together, gives us something that looks like this. Using CSS, we can adjust how the lines look, the padding, all that sort of stuff. But we get a very basic structure like this from this basic table markup.



```
<table>
  <tr>
    <td>1A</td>
    <td>1B</td>
  </tr>
  <tr>
    <td>2A</td>
    <td>2B</td>
  </tr>
</table>
```

The table element must contain table rows (tr) and table cells (td)

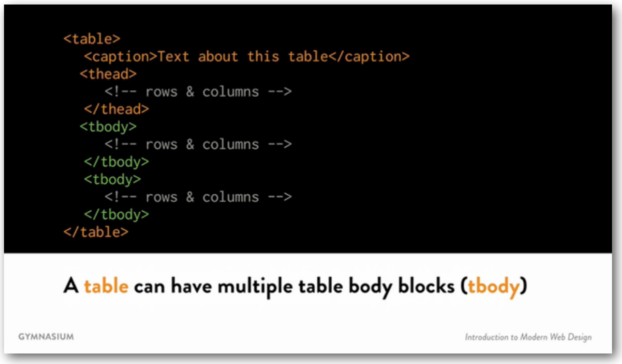
GYMNASIUM

Introduction to Modern Web Design

Now, in order to make our tables more accessible to users, it's suggested that we provide a caption, which is text about a table inside of a caption element. This should come immediately after the opening table tag, and should contain text, but it can contain HTML as well. We can also divide up the table into sections, like headers and body, using the thead and tbody elements to define the table head and the table body. Each of these can contain multiple rows and columns.

Tables can also have footers, which we use the tfoot element for. And this, as well, can contain multiple rows and columns. Interestingly, the tfoot element needs to come directly after the thead element. And the reason for this has to do with printed pages in any sort of paginated medium.

When browsers go to lay out a table, they typically want to repeat the head and the foot on every page. And the best way to do that is to be able to calculate that information ahead of time in order to be able to figure out how many of the tbody rows can be placed on a given page. For that reason, as counterintuitive as it sounds, we want to have the thead element, followed by the tfoot element, followed by the tbody element.



```
<table>
  <caption>Text about this table</caption>
  <thead>
    <!-- rows & columns -->
  </thead>
  <tbody>
    <!-- rows & columns -->
  </tbody>
  <tbody>
    <!-- rows & columns -->
  </tbody>
</table>
```

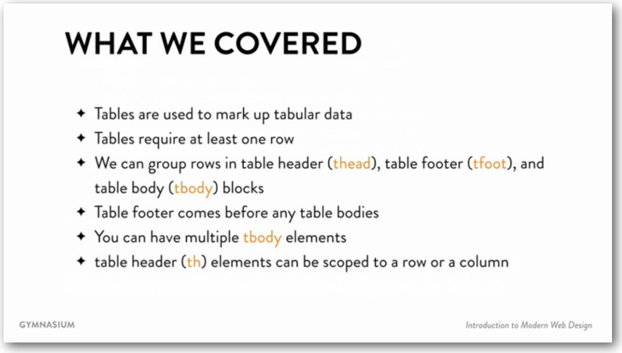
A table can have multiple table body blocks (tbody)

GYMNASIUM Introduction to Modern Web Design

Now, you can also have multiple bodies within a table. This would be where you want to group multiple rows together. Say they are all for a given year, and you want to break it up by year, and you have multiple years in the table. You could do a tbody element for each of those years so that those rows can be aggregated together. You could then take that further and use CSS to style each alternating tbody element's rows in a different color, or each row in different colors if you want to have nested zebra stripes.

For additional accessibility, we want to identify certain cells as being header cells. And for that, we have the th element, which operates very much like a td, with the addition of having the scope attribute. The scope attribute gives us the ability to define which direction the table header is scoped to.

In this example, we see four table header cells which are each scoped to the column. This makes sense, as they are contained within the thead element. Inside of a table body, we might have a thead that's operating for the row, in which case it would be scoped to that row within the tr element. In both of these instances, when the content for the individual cells are being read out, the table headers which are scoped to be associated with that particular table cell would also be read out by assistive technology.



WHAT WE COVERED

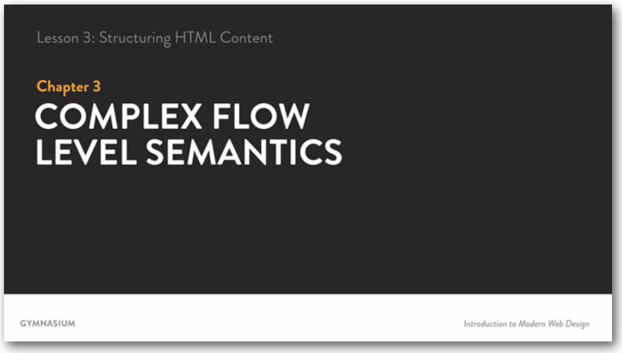
- Tables are used to mark up tabular data
- Tables require at least one row
- We can group rows in table header (thead), table footer (tfoot), and table body (tbody) blocks
- Table footer comes before any table bodies
- You can have multiple tbody elements
- table header (th) elements can be scoped to a row or a column

GYMNASIUM Introduction to Modern Web Design

In this brief chapter, we've covered how tables are used to markup tabular data; that tables require at least one row; that we can group rows together in the table header, table footer, and table body elements; that the table footer needs to come before any table bodies; that you can have multiple tbody elements; and that our table header elements can be scoped to a row or a column in order to improve the accessibility of our pages. In the next chapter, we're going to look at longer quotes, figures, and accordions. I'll see you in a few minutes.

COMPLEX FLOW LEVEL SYMANTICS

Welcome back to Lesson 3, Structuring HTML Content. In this chapter, we'll be talking about complex flow level semantics, starting with the blockquote element.



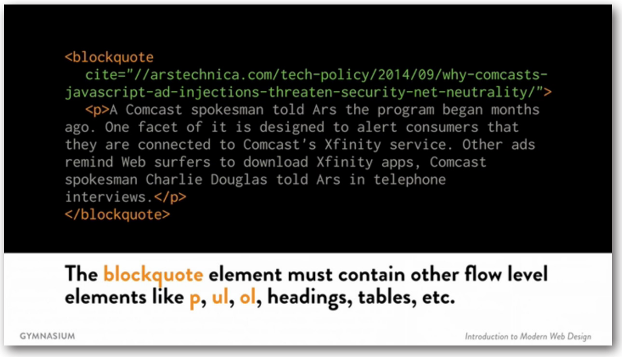
Lesson 3: Structuring HTML Content

Chapter 3

COMPLEX FLOW LEVEL SEMANTICS

GYMNASIUM Introduction to Modern Web Design

Blockquotes are used to indicate that a particular chunk of content is quoted from another source. Typically, this is used for quotes that are sizable in nature such as an entire paragraph, lists, or the like. Here we see a simple example that's been extracted from another web page. We have a paragraph contained within a blockquote element.



```
<blockquote
  cite="//arstechnica.com/tech-policy/2014/09/why-comcasts-
javascript-ad-injections-threaten-security-net-neutrality/">
  <p>A Comcast spokesman told Ars the program began months
ago. One facet of it is designed to alert consumers that
they are connected to Comcast's Xfinity service. Other ads
remind Web surfers to download Xfinity apps, Comcast
spokesman Charlie Douglas told Ars in telephone
interviews.</p>
</blockquote>
```

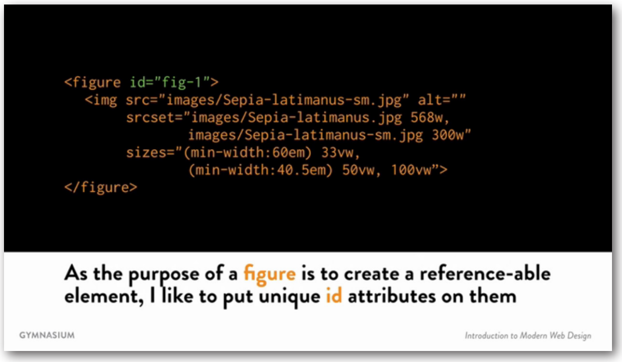
The blockquote element must contain other flow level elements like p, ul, ol, headings, tables, etc.

GYMNASIUM Introduction to Modern Web Design

Blockquotes can contain just about any other flow level semantic element including headings, paragraphs, lists, tables, other blockquotes, and so on. By default, browsers will actually indent blockquotes. If we take a look at this paragraph here, if all of a sudden I were to throw that inside of a blockquote, you see that it gets indented from the left and the right.

We can, of course, control that using CSS. But by default, browsers will provide that sort of indentation.

In addition to semantically indicating and visually indicating that something is a quote, we can also use the site attribute on a blockquote to indicate the source. Most browsers do not directly expose this information to the user. But this is the sort of thing that you could expose using JavaScript. And we'll show an example of this in the final lesson.



```
<figure id="fig-1">
  <img src="images/Sepia-latimanus-sm.jpg" alt=""
       srcset="images/Sepia-latimanus.jpg 568w,
               images/Sepia-latimanus-sm.jpg 300w"
       sizes="(min-width:60em) 33vw,
              (min-width:40.5em) 50vw, 100vw">
</figure>
```

As the purpose of a figure is to create a reference-able element, I like to put unique id attributes on them

GYMNASIUM Introduction to Modern Web Design

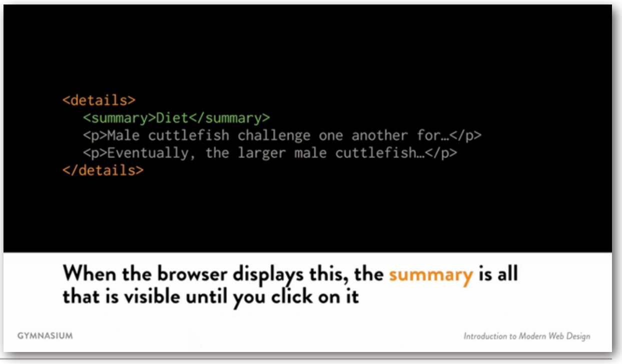
Just as we use blockquotes to reference content from other sites, we sometimes want to reference content within our own pages. And for that, we have the figure element. Here's a simple example from the cuttlefish site that I put together.

Obviously this is an example of a figure that is an image with a figure caption below it. So we might see something like this—a figure element with an image inside of it.

But figures don't just have to contain images. They can contain tables. They can contain charts. Or pretty much anything that you might want to reference within your page.

Because it is something we would want to reference from within our document, I recommend putting an ID on your figures. That way, it's something that can be referenced and anchored to using a link within the page. If you remember back to our discussion of links, an anchor is that bit in the URL that comes after the octothorpe or # symbol, otherwise known as the fragment identifier.

In many cases, you want to provide some context around why you've chosen to include a particular figure. In this case, you want to use a caption. Here's an example of a figure caption using the figcaption element that contains a paragraph with information that describes this particular image.



```
<details>
  <summary>Diet</summary>
  <p>Male cuttlefish challenge one another for…</p>
  <p>Eventually, the larger male cuttlefish…</p>
</details>
```

When the browser displays this, the summary is all that is visible until you click on it

GYMNASIUM Introduction to Modern Web Design

The final complex flow level semantic structure that I want to discuss with you is the accordion. Now, you may be familiar with the accordion as a bit of content that is expandable and collapsible, as you see here in this example.

We create accordions natively in HTML using the details and summary elements. As with the figure element, the details and summary elements were introduced in HTML5. The way that it works is that the details element is collapsed so that only the summary element is visible. And when you click on the summary, it expands to show the entire contents of the details element.

WHAT WE COVERED

- Long Quotes
- Figures
- Accordions

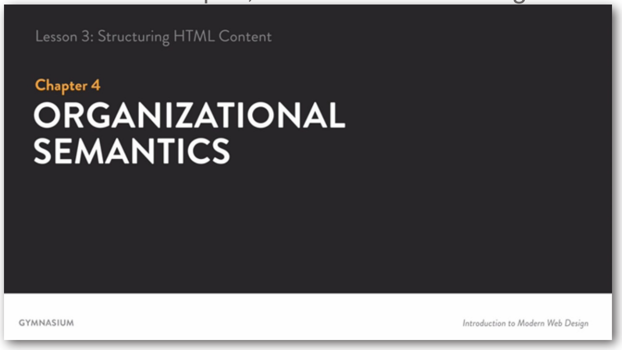
GYMNASIUM

Introduction to Modern Web Design

As this is a new piece of HTML5, older browsers don't know what to do with it. But because HTML is fault tolerant, meaning that browsers ignore what they don't understand, we still end up with something that's usable. So instead of what you see here, what we would end up with in an older browser is something akin to this.

As you can see, the content is simply exposed. There's no interaction to it at all. But that's OK. You can still access the content, and you can still read it.

In this chapter, we've covered blockquotes, figures, and accordions. In the next chapter, we'll talk about dividing up our content, identifying primary content on the page, sections, and articles, as well as tangential content. In other words, organizational semantics.



Lesson 3: Structuring HTML Content

Chapter 4

ORGANIZATIONAL SEMANTICS

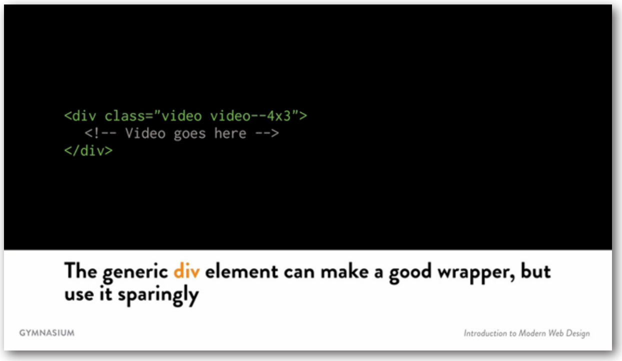
GYMNASIUM

Introduction to Modern Web Design

I'll see you in a few minutes.

ORGANIZATIONAL SYMANTICS

Welcome back to Lesson 3, Structuring HTML Content. Now, in this chapter, I'm going to talk to you about organizational semantics or how we organize our HTML pages. I'm not talking about organizing them in folders. I'm talking about organizing the actual elements we use within our HTML files.



```
<div class="video video--4x3">
  <!-- Video goes here -->
</div>
```

The generic div element can make a good wrapper, but use it sparingly

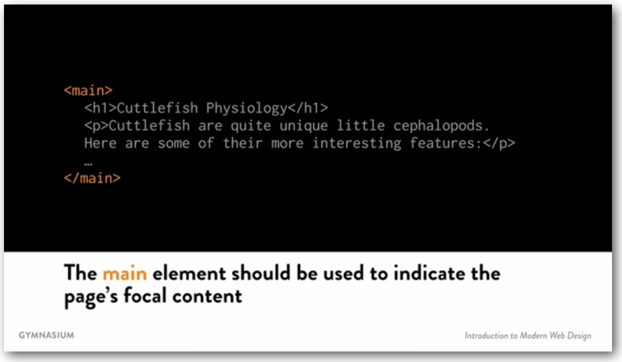
GYMNASIUM

Introduction to Modern Web Design

Now, when you're creating an HTML document, you often have times where you need to group certain elements together because they're related. The most generic way that we can group them together is using the div element. All a div does is imply that these elements are related and that they are placed inside the same generic division.

There's no semantic value to a div element. As such, the div element often has a class associated with it or an id associated with it to give it more meaning to the web author. Divs can make a good generic wrapper. But because we have more options available to us in HTML5, it's a good idea to use those instead and use the div sparingly.

The first of the new HTML5 organizational elements that I want to talk to you about is the main element. The main element is used to indicate the primary content of a given page. It simply is main, as you'd expect. So here we see the main element wrapped around the entire content of the cuttlefish physiology page. In the future, it wouldn't be surprising to find search engines using this as an indicator of where the main content of the page is so that they can ignore things like navigation, branding headers, footers, and so on, in order to concentrate their attention on the focal content of the page.

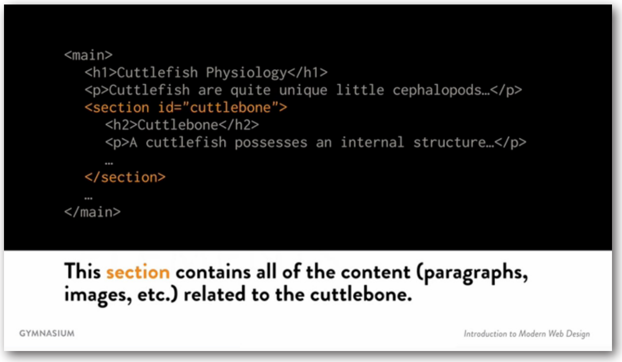


```
<main>
  <h1>Cuttlefish Physiology</h1>
  <p>Cuttlefish are quite unique little cephalopods.
  Here are some of their more interesting features:</p>
  …
</main>
```

The main element should be used to indicate the page's focal content

GYMNASIUM *Introduction to Modern Web Design*

Now, within the main content, we sometimes need to subdivide content into particular chunks. For that, we have the section element. Now, here we have an example of a section element being used within the main content of this particular HTML page. The section's been given an id of cuttlebone so that it becomes an anchor point that we can direct users to within the page if they want information specifically on the cuttlebone and not on cuttlefish physiology on the whole. Section elements can contain any other flow level semantics you might want to use as well. So you see here, heading levels, paragraphs, lists, figures, and the like are all usable within a section.

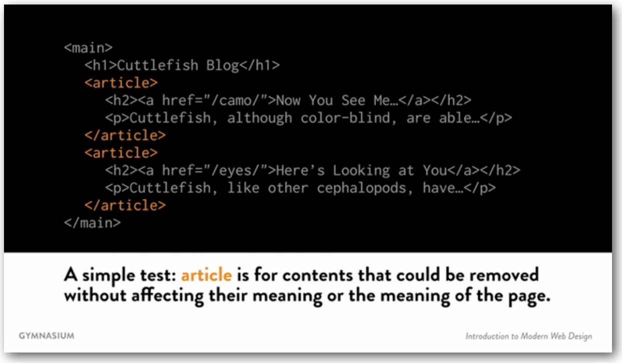


```
<main>
  <h1>Cuttlefish Physiology</h1>
  <p>Cuttlefish are quite unique little cephalopods…</p>
  <section id="cuttlebone">
    <h2>Cuttlebone</h2>
    <p>A cuttlefish possesses an internal structure…</p>
    …
  </section>
  …
</main>
```

This section contains all of the content (paragraphs, images, etc.) related to the cuttlebone.

GYMNASIUM *Introduction to Modern Web Design*

Now, similar to a section, we have the article element. An article element is like an autonomous section. So here we have an example of the cuttlefish blog. We have the main content and the heading level of the cuttlefish blog right below that.

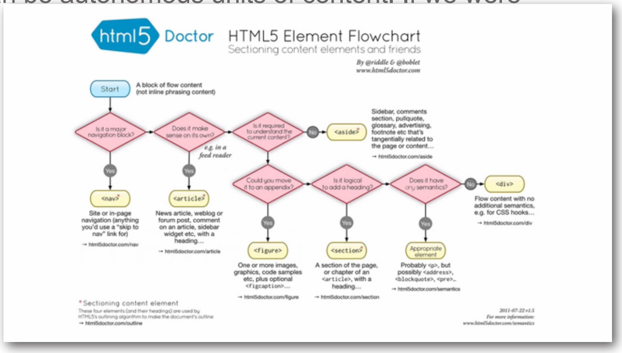


```
<main>
  <h1>Cuttlefish Blog</h1>
  <article>
    <h2><a href="/camo/">Now You See Me…</a></h2>
    <p>Cuttlefish, although color-blind, are able…</p>
  </article>
  <article>
    <h2><a href="/eyes/">Here's Looking at You</a></h2>
    <p>Cuttlefish, like other cephalopods, have…</p>
  </article>
</main>
```

A simple test: article is for contents that could be removed without affecting their meaning or the meaning of the page.

GYMNASIUM *Introduction to Modern Web Design*

And then within the cuttlefish blog, we have individual teasers for a particular blog post. Each of these teasers is marked up in an article as opposed to a section because these can be autonomous units of content. If we were to remove the first article, it would not change the meaning of the page overall.

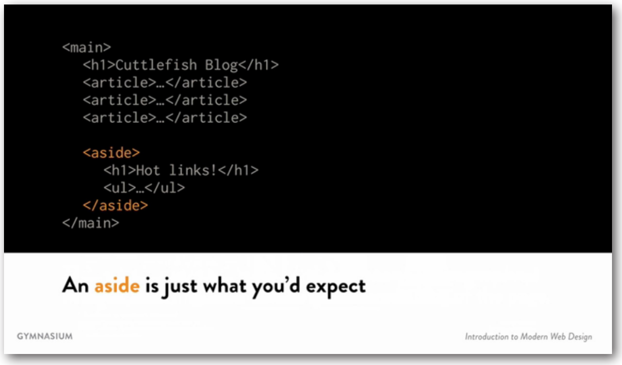

html5 Doctor
HTML5 Element Flowchart
Sectioning content elements and friends
By @riddle & @boblet
www.html5doctor.com
Start
A block of flow content (not inline phrasing content)
Is it a major navigation block?
Does it make sense on its own?
e.g. in a feed reader
Is it required to understand the current content?
No
<aside>*
Sidebar, comments section, pullquote, glossary, advertising, footnote etc that's tangentially related to the page or content…
→ html5doctor.com/aside
Yes
<nav>*
Site or in-page navigation (anything you'd use a "skip to nav" link for)
→ html5doctor.com/nav
Yes
<article>*
News article, weblog or forum post, comment on an article, sidebar widget etc, with a heading…
→ html5doctor.com/article
Could you move it to an appendix?
Is it logical to add a heading?
Does it have any semantics?
No
<div>
Flow content with no additional semantics, e.g. for CSS hooks…
→ html5doctor.com/div
Yes
<figure>
One or more images, graphics, code samples etc, plus optional <figcaption>…
→ html5doctor.com/figure
Yes
<section>*
A section of the page, or chapter of an <article>, with a heading…
→ html5doctor.com/section
Yes
Appropriate element
Probably <p>, but possibly <address>, <blockquote>, <pre>…
→ html5doctor.com/semantics
*Sectioning content element
These four elements (and their headings) are used by HTML5's outlining algorithm to make the document's outline
→ html5doctor.com/outline
2011-07-22 v1.5
For more information:
www.html5doctor.com/semantics

The same thing would happen with the second article. They don't affect the meaning of the page, and the page doesn't affect the meaning of the articles. You can think of them kind of like an article of clothing. You take an article of clothing off a person, and it's still an article of clothing. You put it onto the person, it's still an article of clothing. Neither defines the other.

Now, as if that's not confusing enough, articles can have sections, and sections can contain articles as well. Now, the fine folks at HTML Doctor have put together a beautiful flow chart to help us to understand which elements are the best to pick out. You start at the start, a block of flow level content. In other words, it's not phrasing content like we saw in the last lesson.

The first question is, is it a major navigation block? We'll discuss the nav element in the next chapter. But if that's not the case, then we ask, does it make sense on its own? If it does, then we use the article element. If it doesn't, we move on.

Is it required to understand the current content? If not, maybe aside, which we'll talk about in a second. If not aside, could you move it to an appendix? In other words, is it referenced content? Then the figure might make sense. If not, is it logical to add a heading at this point? If that's the case, then the section element makes sense.



```
<main>
  <h1>Cuttlefish Blog</h1>
  <article>…</article>
  <article>…</article>
  <article>…</article>

  <aside>
    <h1>Hot links!</h1>
    <ul>…</ul>
  </aside>
</main>
```

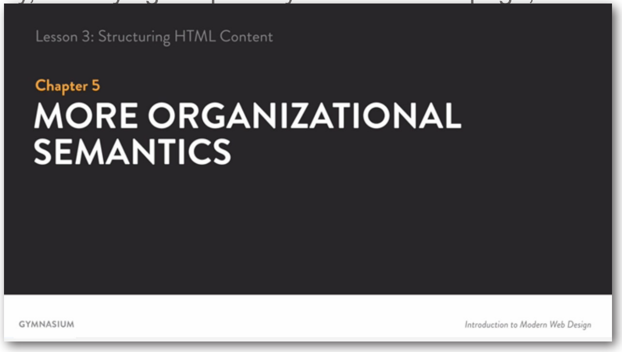
An aside is just what you'd expect

GYMNASIUM Introduction to Modern Web Design

If the section element doesn't make sense, does it have any semantic value whatsoever? If it does, then use the appropriate semantic element, like a paragraph element or a blockquote. If not, this is when it makes sense to use the generic div element.

Now, I mentioned that we were going to talk about aside. Aside is for tangentially-related content. So if we were on the main blog page, here we have the main content of the blog page. And we have Cuttlefish Blog in an h1, followed by several articles. We might also have an aside within that main content for hot links that somebody may want to look at about cuttlefish. So aside is somewhat like a sidebar. It's a tangent. It's something else additional that's not necessary, but is beneficial and related to the content.

So in this chapter, we've talked about dividing up content generically, identifying the primary content of the page, how to use section and article elements, as well as how to identify tangential content. In the next section, we'll wrap up the remainder of the new HTML5 elements by talking about navigation, headers and footers, and then using ARIA to define an element's purpose. I'll see you in a minute.



Lesson 3: Structuring HTML Content

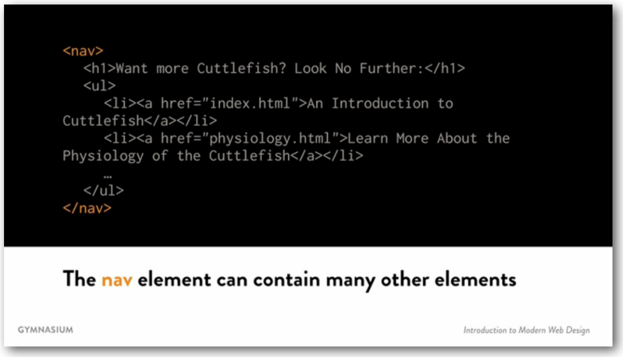
Chapter 5

MORE ORGANIZATIONAL SEMANTICS

GYMNASIUM Introduction to Modern Web Design

MORE ORGANIZATIONAL SYMANTICS

Welcome back to Lesson 3, Structuring HTML Content. And in this chapter, the final chapter of this lesson, we'll talk about more organizational semantics, starting with navigation. The nav element is used to contain any elements that are being used as major navigation within the page or the site.



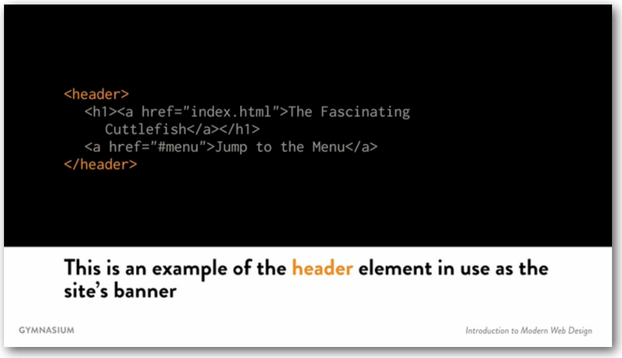
```
<nav>
  <h1>Want more Cuttlefish? Look No Further:</h1>
  <ul>
    <li><a href="index.html">An Introduction to
Cuttlefish</a></li>
    <li><a href="physiology.html">Learn More About the
Physiology of the Cuttlefish</a></li>
    …
  </ul>
</nav>
```

The nav element can contain many other elements

GYMNASIUM Introduction to Modern Web Design

Here's an example from my cuttlefish site where I've got the navigation with a heading "Want More Cuttlefish? Look No Further," and then links to all of the pages within the site starting with the intro page, the index page, an Introduction to Cuttlefish, a page about their physiology, a page about their ecology, a page of videos, a page of photos, and so on and so forth.

As this shows, the navigation element can contain many other elements. And in most cases, we want to mark up our navigation in a list, because that's what navigation is—a list of links. Another important organizational element we got in HTML5 is the header element. And the header is for intro content.

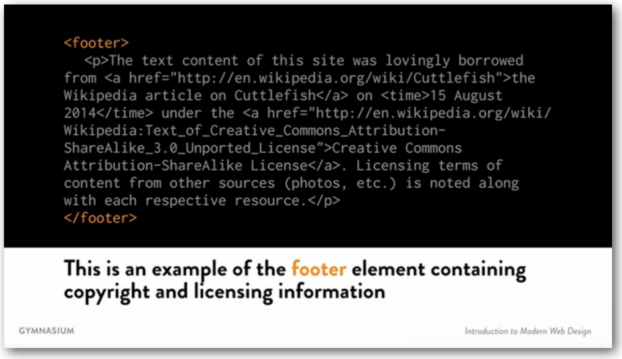


```
<header>
  <h1><a href="index.html">The Fascinating
    Cuttlefish</a></h1>
  <a href="#menu">Jump to the Menu</a>
</header>
```

This is an example of the header element in use as the site's banner

GYMNASIUM Introduction to Modern Web Design

So in the case of my cuttlefish site, here I have a header element with the h1, the title of the site, "The Fascinating Cuttlefish," and it also contains my link to Jump to the Menu. This jumps down to the navigation menu for the site. This is acting as the site's banner.



```
<footer>
  <p>The text content of this site was lovingly borrowed
from <a href="http://en.wikipedia.org/wiki/Cuttlefish">the
Wikipedia article on Cuttlefish</a> on <time>15 August
2014</time> under the <a href="http://en.wikipedia.org/wiki/
Wikipedia:Text_of_Creative_Commons_Attribution-
ShareAlike_3.0_Unported_License">Creative Commons
Attribution-ShareAlike License</a>. Licensing terms of
content from other sources (photos, etc.) is noted along
with each respective resource.</p>
</footer>
```

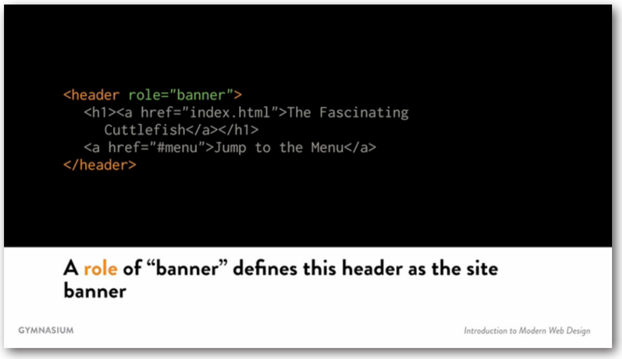
This is an example of the footer element containing copyright and licensing information

GYMNASIUM Introduction to Modern Web Design

Now, as you'd probably expect, as there is a header, there is also a footer. And the footer is used to contain meta information concerning the content. So in the case of my site about cuttlefish, the footer contains a paragraph saying that this content was lovingly borrowed from the Wikipedia page on cuttlefish. I also have additional information in there regarding copyright and licensing of the content.

Now, it's worth noting that articles and sections can also contain their very own headers and footers. Headers and footers are not only intended for use on the page as a whole. Here's a quick example of an article element with a header containing the h1, or the title of the article, and some additional information about the article, such as the author, me, and the time at which it was published, the 12th of December in 2014. Similarly, this article could contain a footer that would have things like tags that were used, or individual licensing information for the article, or perhaps it contains a mini bio about me, or some additional supplementary information about the article specifically and not about the page as a whole.

Now, the final edition of HTML5 that I want to go through is the role attribute. The role attribute is used to indicate an element's purpose within the page. Roles are part of the WAI-ARIA specification. W-A-I A-R-I-A, for the Web Accessibility Initiatives Accessible Rich Internet Applications Spec. It's a bit of a mouthful which is why we say WAI-ARIA, or simply ARIA.



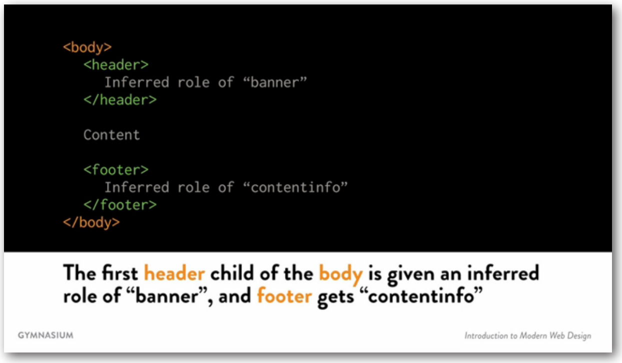
```
<header role="banner">
  <h1><a href="index.html">The Fascinating
    Cuttlefish</a></h1>
  <a href="#menu">Jump to the Menu</a>
</header>
```

A role of "banner" defines this header as the site banner

GYMNASIUM Introduction to Modern Web Design

ARIA defines a number of roles for us, such as the role of banner which is used to indicate that this particular header element is operating as the banner for the site. Similarly, we can also indicate that a particular footer element is acting as the content info for the page. In other words, it contains the meta information for the page as a whole.

We also have roles of navigation, roles of search, and various other ones. The role of navigation looks pretty redundant when we look at it on a nav element. But there is a reason for this redundancy. Both the HTML5 spec and the ARIA spec were evolving at the same time and were evolving to solve some of the same problems.



```
<body>
  <header>
    Inferred role of "banner"
  </header>

  Content

  <footer>
    Inferred role of "contentinfo"
  </footer>
</body>
```

The first header child of the body is given an inferred role of "banner", and footer gets "contentinfo"

GYMNASIUM Introduction to Modern Web Design

Most assistive technology manufacturers were tracking the ARIA spec a lot more closely than they were tracking the HTML5 spec. And for that reason, they had implementations in place to handle the role attribute before they had the ability to handle the native HTML5 semantics. That said, most of them are catching up now. And over time, this redundancy will become less necessary.

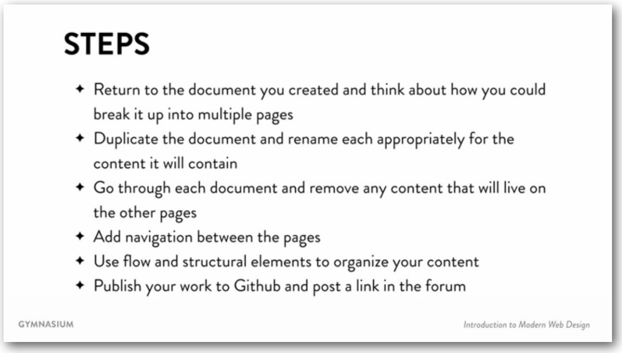
Lesson 3: Structuring HTML Content

Assignment 6:

ORGANIZE YOUR CONTENT

GYMNASIUM Introduction to Modern Web Design

It's worth noting that some roles can actually be inferred from the position of the element within the page. For example, within the body, the first header element that's encountered will be given a role of banner automatically. Similarly, the first footer element that is a direct child of the body will be given the role of content info.



STEPS

- Return to the document you created and think about how you could break it up into multiple pages
- Duplicate the document and rename each appropriately for the content it will contain
- Go through each document and remove any content that will live on the other pages
- Add navigation between the pages
- Use flow and structural elements to organize your content
- Publish your work to Github and post a link in the forum

GYMNASIUM Introduction to Modern Web Design

And that wraps it up for this chapter in which we discussed navigation, headers and footers, and defining an element's purpose using the role attribute.

Now it's time for a quiz. I want you to take Quiz #2 and post your answers on the forum. And finally, for assignment number 6, I want you to go back to your HTML page and organize your own content using the knowledge that you've gained in this lesson. Return to your main document and think about how you could break it up into multiple pages.

I'd like you to duplicate the document and rename each copy appropriately for the content it will contain. So in other words, you'll go from something like index.html to having perhaps ecology and physiology as well. Now, go through each of these duplicates and remove any content that should live on the other pages. And then add some navigation so that you can move between the different documents. Remember back to Lesson 2 and the way we talked about being able to link from document to document.

Finally, go through each of these documents and use the flow and structural elements we discussed in this lesson to organize your content on those pages. Then publish your work up to GitHub and post a link in the forum. I look forward to seeing you in Lesson 4.