Session 2

Semantic markup

Slide instructions

`SPACEBAR` to **move forward** through slides.

`SHIFT` & `SPACEBAR` to **move backwards** through slides.

`LEFT ARROW` & `RIGHT ARROW` to **move through sections**.

`ESC` to **see overview** and `ESC` again to exit.

`F` to **enter presentation mode** and `ESC` to exit.

Introduction

A lot of developers work in environments where they **rarely interact with HTML markup**.

This can lead to developers thinking that HTML is irrelevant. However, **nothing could be further from the truth**.

Websites and web apps ultimately produce HTML markup **that is delivered to our users**.

Valid and semantic markup is crutial in order to **deliver content that is accessible to everyone**.

So, in this session we'll look at valid and semantic markup, as well as various HTML elements **from an accessibility perspective**.

Three important questions

We need to be able to answer **three questions for all users at all times**.

- Where am I?
- What is is thing?
- What will happen when I interact with this thing?

These questions can all be answered with **properly implemented semantic and valid markup**.

Where am I?

We can help users understand where they are in the UI with proper use of HTML **landmarks** and **headings**.

What is is thing?

We can help users know what things are with **valid** use of **semantic elements**. Element roles help communicate what things are.

What will happen when I interact with this thing?

When choosing the appropriate **semantic element** for the job, we also communicate expected behaviours.

What is valid markup?

Validation is a process of **checking your documents against a formal standard**.

In the case of HTML, the relevant standard is the **HTML Living Standard**.

Developers should always try to produce **valid, well formed markup** for:

- Browser consistency.
- Page speed.
- Searchability.
- Accessibility.

Why is valid markup important for accessibility?

There are **three specific validation issues** that can negatively affect assistive technologies:

1. Duplicate IDs

This can break the accessibility of labels for forms and table header cells.

```
<!-- Duplicate IDs in one document -->
<div id="one"></div>

<label for="one">Name</label>
<input id="one">
```

2. Missing end tags

This can cause issues for browser trying to determine when an element finishes and the next one starts.

```
<!-- Missing end tag -->
<p>some text.
<div>some other content.</div>
```

3. Improper nesting

This can cause issues for assistive technologies as it could impact on accessible names.

```
<!-- Improper nesting -->
<div><li>list item</div></li>
```

Any questions or comments?

Exercise: Validating some sites

Go to your main company website, find a complex page, and check the markup via the HTML validator.

Alternatively, you can use the Deque University validator bookmarklet.

What is semantic markup?

Semantic markup is about **using the appropriate HTML elements to convey meaning**.

All HTML elements have **a specific meaning and purpose**. For example:

Element	Semantic meaning
`<h1>`	Level 1 heading content
`<caption>`	A visible caption or title for a table
`<abbr>`	An abbreviation or acronym of a longer word or phrase
`<blockquote>`	A section that is quoted from another source

The `<div>` and `<span>` elements are the only elements that have no semantics and are **not represented in the accessibility tree**.

Element	Semantic meaning
`<div>`	Generic block-level container
`<span>`	Generic inline container

Who is responsible for semantics?

Designers need to understand the basics of semantic markup, **so they can communicate important information correctly to developers**.

Designers should specify the following **as part of handover**:

- Desired heading levels for all headings.
- Desired sections, possible landmarks and even how these sections should be labelled.

Developers need to know how to use semantic markup **as it vital for**:

- Accessibility.
- Searchability.
- Internationalisation.
- Interoperability.

Why is semantic markup important for accessibility?

Many users rely on specific elements, and their correct usage, in order to **understand the content and navigate around web pages effectively**.

Elements that are **particularly important** include:

- Headings.
- Landmarks.
- Links and buttons.
- Images.
- Tables.
- Lists.
- Forms and form elements.

Any questions or comments?

Exercise: Semantics quiz

On the next screen are **three different HTML elements**:

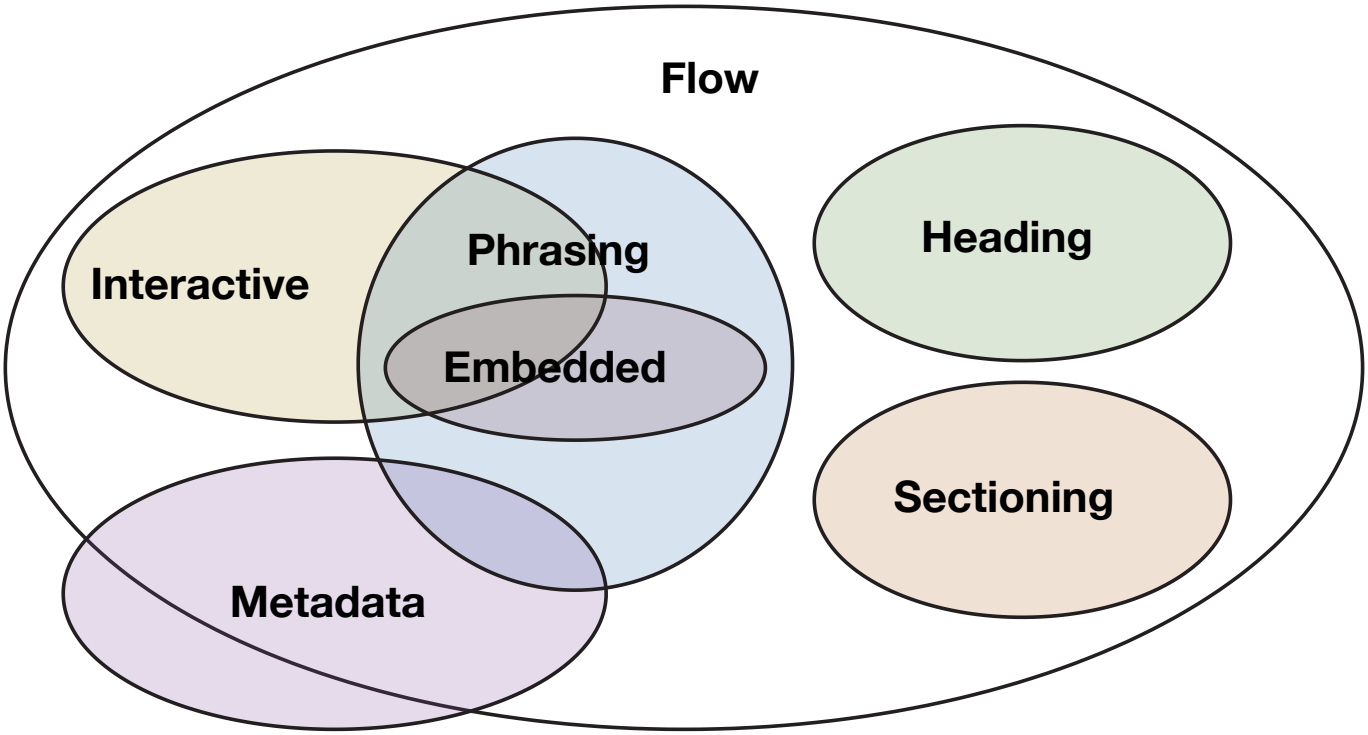
1. Guess the purpose of each element.
2. Add your answers in chat.

```
<datalist></datalist>
<dl></dl>
<optgroup></optgroup>
```

Element	Semantic meaning
`<datalist>`	Represents a set of option elements that represent predefined options for other controls.
`<dl>`	Represents a description list of zero or more name-value groups.
`<optgroup>`	Represents a group of option elements with a common label.

HTML Content Categories

All HTML elements can be grouped into broad semantic groups called “content categories”.



How are these categories relevant to accessibility?

Each of the categories **defines the overall semantics, and the way in which the elements can be used**.

The semantics and usage information **helps us create valid markup**, which can lead to improved accessibility.

We'll take a quick look at **two of these categories**.

Flow content

Most elements that are used in the body of documents are categorised as **flow content**.

```
<!-- Some Flow content examples -->
<a>
<article>
<blockquote>
<figure>
<h1>
<main>
<p>
<table>
```

Flow elements are generally **formatted visually as a block**, with white space above and below.

Most flow elements **can contain other flow elements**. In the following example, a `<div>` element is nested inside another `<div>`.

```
<!-- Nested flow elements -->
<div>
  <div></div>
</div>
```

Phrasing content

Phrasing content is any elements that mark up text within paragraphs.

```
<!-- Some Phrasing content examples -->
<a>
<abbr>
<b>
<bdi>
<bdo>
<del>
<i>
```

Most phrasing elements **do not form new blocks of content**. The content is distributed in lines.

Most phrasing elements **can contain other phrasing elements**. In the following example, an `<i>` element is nested inside another `<span>`.

```
<!-- Nested phrasing elements -->
<span>
  <i></i>
</span>
```

In almost all cases, phrasing elements **are not allowed to contain flow elements**. In the following example, a `<div>` is nested inside a `<span>`.

This is considered invalid markup, and **could cause issues for some assistive technologies**.

```
<!-- Not allowed -->
<span>
  <div></div>
</span>
```

Any questions or comments?

Exercise: Content Categories quiz

We are going to use the HTML Living Standard to find out some **specific information about two elements**.

The information includes:

- Categories: The Content Categories it belong to.
- Contexts: How this element can be used.
- Content model: What content can be included as descendants.

Question 1: The `<legend>` element

- Categories:
- Contexts:
- Content model:

Question 1: The `<legend>` element

- Categories: None.
- Contexts: As the first child of a `<fieldset>`.
- Content model: Phrasing content, optionally heading content.

Question 2: The `<caption>` element

- Categories:
- Contexts:
- Content model:

Question 2: The `<caption>` element

- Categories: None.
- Contexts: As the first element child of a `<table>`.
- Content model: Flow content, no descendant `<table>` elements.

Headings

Why are headings important for accessibility?

Headings communicate **how the content on web pages is structured** for sighted users as well as assistive technology users.

Web browsers, plug-ins, and assistive technologies **can use them to provide in-page navigation**.

Heading ranking

Headings should be **nested by their rank**. The most important heading has the rank 1 (`<h1>`), the least important heading rank 6 (`<h6>`).

Headings with an equal or higher rank **should start a new section**.

```
<h1>Page title</h1>
  <h2>Section 1</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
  <h2>Section 2</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
```

Headings with a lower rank **should start new subsections** that are part of the higher ranked section.

```
<h1>Page title</h1>
  <h2>Section 1</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
  <h2>Section 2</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
```

Skipping heading ranks **can be confusing and should be avoided where possible**. For example, make sure that a `<h2>` is not followed directly by an `<h4>`.

```
<h1>Page title</h1>
  <h2>Section 1</h2>
    <h4>Subsection 1</h4>
```

It is acceptable to skip ranks **when closing subsections**. For example, a `<h2>` beginning a new section, can follow an `<h3>` as it closes the previous section.

```
<h1>Page title</h1>
  <h2>Section 1</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
  <h2>Section 2</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
```

Exception for shared page sections

In sections of the page which are shared between many pages, for example in sidebars, **the heading ranks should not change** depending on the ranks in the content area.

In those cases, **consistency across pages** is more important.

```
<h1>Page title</h1>
  <h2>Section 1</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>
  <h2>Section 2</h2>
    <h3>Subsection 1</h3>
    <h3>Subsection 2</h3>

<!-- Sidebar content across multiple pages -->
<h2>Sidebar heading</h2>
```

Any questions or comments?

Exercise: Identifying headings

- Go to the The Straits Times website
- Without looking at the markup, try to guess which heading levels would be used for each heading on the page.

Add Paul Adams Heading bookmarklet and see if you agree with the heading choices.

Landmarks

Logical page regions

Complex page layouts **can be overwhelming for users**. They may need to spend a lot of additional effort scanning the page to find the content they need.

Ideally, complex page layouts should be **broken into logical regions**:

- Visually.
- Within the markup.

Logical page regions allows users to:

- Skim the overall page layout and quickly find the content they need.
- Orientate themselves within the overall page.

Who is responsible?

Designers need to make sure page layouts:

- Are laid out in logical regions.
- All key regions are identified and labelled.
- Any labels are communicated to developers.

Developers need to make sure page layouts:

- Are marked up using semantic HTML elements.
- All key regions are labelled (accessible names).

How do developers mark up page regions semantically?
Using landmark elements.

HTML and ARIA landmarks

In the early days of the web, the only way to define page regions was **via implied heading hierarchy**.

ARIA introduced a range of landmark roles that were designed to **identify page regions for assistive technologies**.

```
<div role="navigation">
  <ul>
    <li>Home</li>
    <li>About</li>
  </ul>
</div>
```

HTML5 introduced a range of new elements to **define page regions**.

```
<nav>
  <ul>
    <li>Home</li>
    <li>About</li>
  </ul>
</nav>
```

This means that there are **now two ways** to identify landmarks, using HTML elements and ARIA roles.

Landmark	ARIA role	HTML element
Banner	`role="banner"`	`<header>`
ContentInfo	`role="contentinfo"`	`<footer>`
Navigation	`role="navigation"`	`<nav>`
Main	`role="main"`	`<main>`
Complementary	`role="complementary"`	`<aside>`
Region	`role="region"`	`<section>`
Article	`role="article"`	`<article>`

Where possible, you should **always use HTML elements for landmarks** rather than ARIA roles as these elements have semantics and behaviour already built-in.

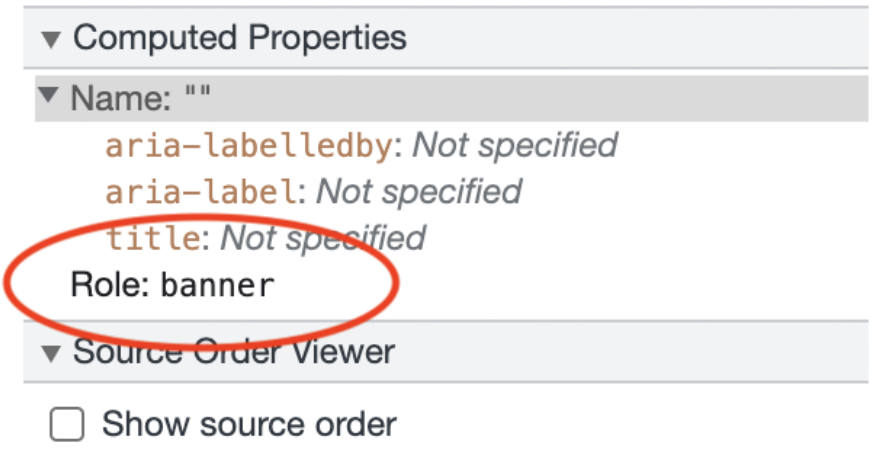
The <header> element

The `<header>` element can be used **as a group for introductory or navigational aids**. It can be used multiple times within a layout.

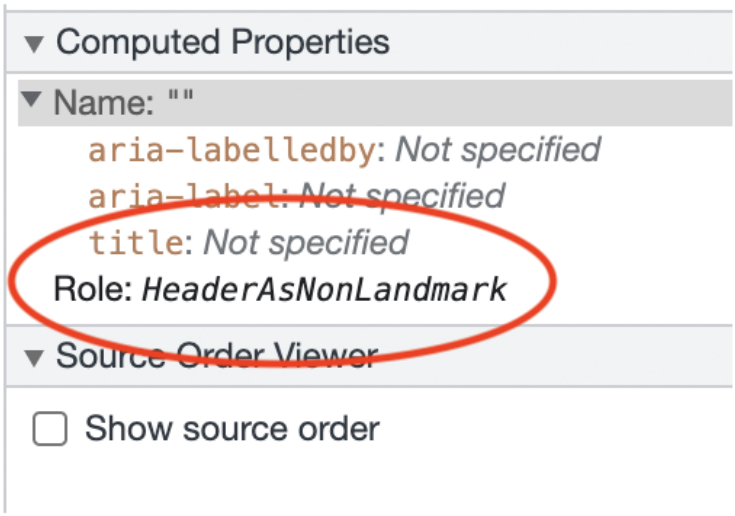
```
<body>
  <header></header>
  <main>
    <header></header>
  </main>
  <section>
    <header></header>
  </section>
  <aside>
    <header></header>
  </aside>
</body>
```

In order for a `<header>` to be defined as a `banner` landmark it **must not be placed inside any other landmark elements**.

```
<body>
  <header>Banner landmark</header>
  <main>
    <header></header>
  </main>
  <section>
    <header></header>
  </section>
  <aside>
    <header></header>
  </aside>
</body>
```



If the `<header>` is placed inside any other landmarks it **will not be defined as a `banner` landmark**.



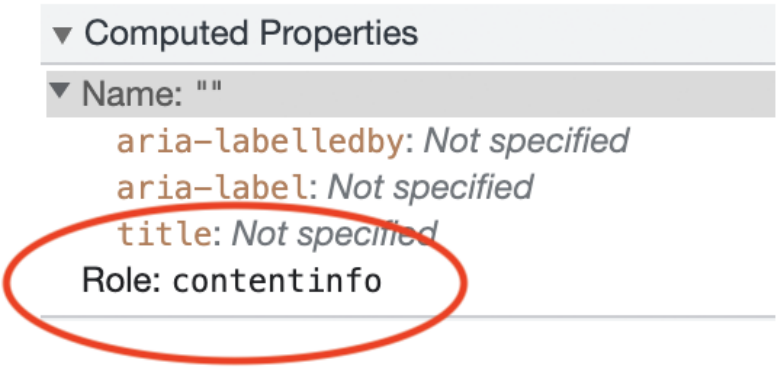
The <footer> element

The `<footer>` element can be used **for information about its section**. It can be used multiple times within a layout.

```
<body>
  <main>
    <footer></footer>
  </main>
  <section>
    <footer></footer>
  </section>
  <aside>
    <footer></footer>
  </aside>
  <footer></footer>
</body>
```

In order for a `<footer>` to be defined as a `contentinfo` landmark it **must not be placed inside any other landmark elements**.

```
<body>
  <main>
    <footer></footer>
  </main>
  <section>
    <footer></footer>
  </section>
  <aside>
    <footer></footer>
  </aside>
  <footer>Contentinfo landmark</footer>
</body>
```



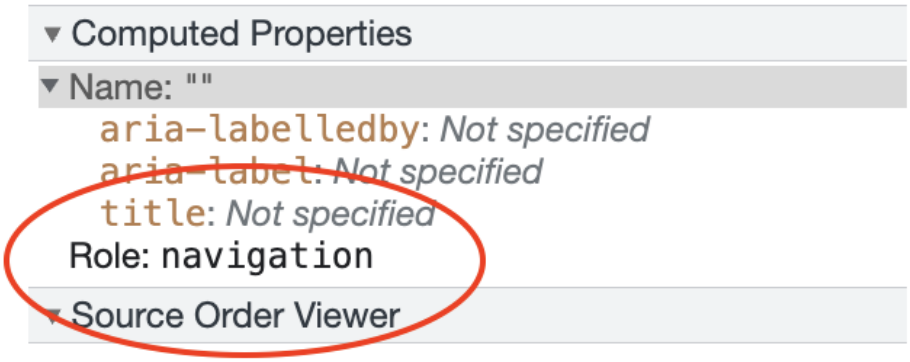
If the `<footer>` is placed inside any other landmarks it **will not be defined as a `contentinfo` landmark**.

The <nav> element

The <nav> element is used to **identify the navigation landmark**, which contain groups of links used for site or page navigation.

```
<nav>
  <ul>
    <li><a href="#">Link1</a></li>
    <li><a href="#">Link2</a></li>
    <li><a href="#">Link3</a></li>
  </ul>
</nav>
```

It can be used **multiple times within a single layout** and each instance will generate a `navigation` landmark.



If a page includes more than one navigation landmark, **each should have a unique label** using either `aria-label` or `aria-labelledby`.

```
<nav aria-label="Primary">
  <ul>
    <li><a href="#">Link1</a></li>
    <li><a href="#">Link2</a></li>
    <li><a href="#">Link3</a></li>
  </ul>
</nav>
```

```
<nav aria-labelledby="aaa">
  <h3 id="aaa">Sidebar heading</h3>
  <ul>
    <li><a href="#">Link1</a></li>
    <li><a href="#">Link2</a></li>
    <li><a href="#">Link3</a></li>
  </ul>
</nav>
```

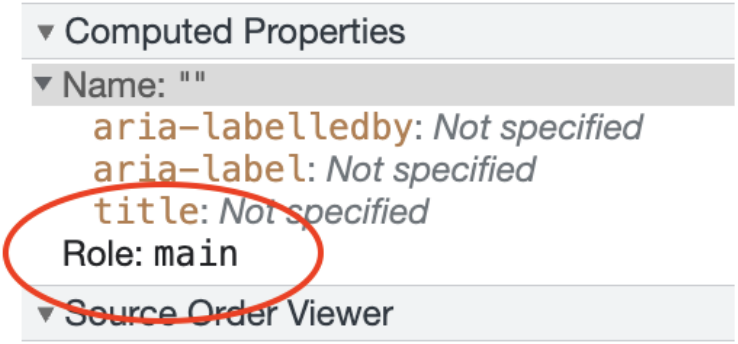
The <main> element

The `<main>` element is used to **identify the main landmark**, which is the primary content for the page.

```
<main>
</main>
```

Each page must have **only one** `<main>` element.

In order for a `<main>` to be defined as a `main` landmark it **must not be placed inside any other landmark elements**.

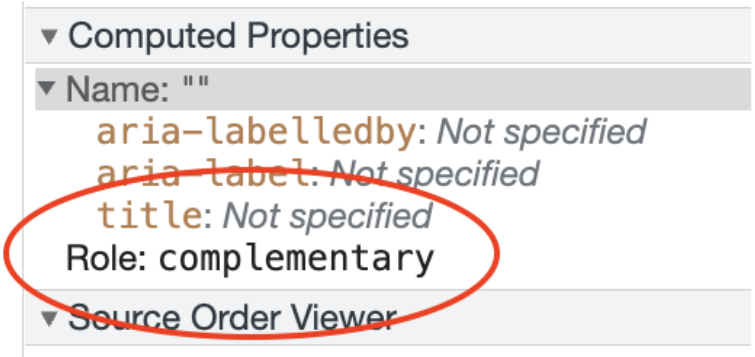


The <aside> element

The `<aside>` element is used to **identify page regions that are complementary to the main content**.

```
<aside>
</aside>
```

In order for a `<aside>` to be defined as a `complementary` landmark it **must not be placed inside any other landmark elements**.



If a page includes more than one `<aside>` element, **each should have a unique label** using either `aria-label` or `aria-labelledby`.

```
<aside aria-labelledby="aaa">
  <h3 id="aaa">Sidebar</h3>
</aside>
```

```
<aside aria-label="Sidebar">
  ...
</aside>
```

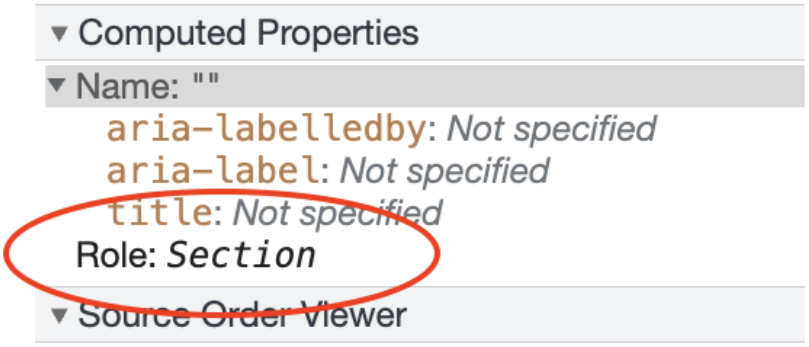
The <section> element

The `<section>` element is used to **identify a page regions**.

This element should only be used if the page region is **sufficiently important that users may want to navigate to it**.

```
<section>
</section>
```

The `<section>` element can be used **multiple times within a single layout** and each instance will generate a `section` landmark.



Each `<section>` element should be identified, typically by **including a heading that is a direct child element**.

```
<section>
  <h3>Section heading</h3>
</section>
```

These headings could be **programmatically associated with their parent elements** using the `aria-labelledby` attribute.

```
<section aria-labelledby="section01">
  <h3 id="section01">Section heading</h3>
</section>
```

The W3C Validator will **present a warning message** for any section that does not contain a heading as a direct child element.

Warning: *Section lacks heading. Consider using h2-h6 elements to add identifying headings to all sections.*

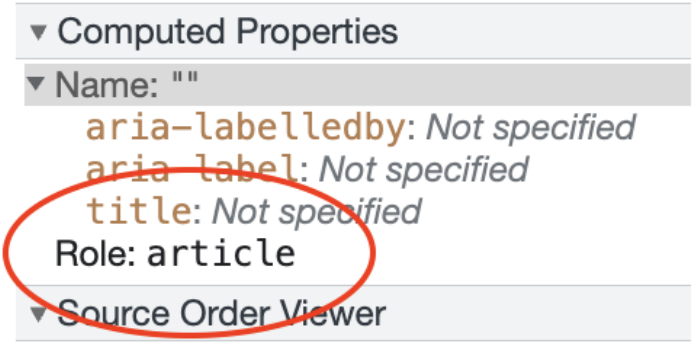
The <article> element

The `<article>` element for **self-contained compositions in a document**, which are intended to be reusable (e.g. syndication).

Examples include: blog posts, articles, product cards, user-submitted comments, or any other independent item of content.

```
<article>
</article>
```

The `<article>` element can be used **multiple times within a single layout** and each instance will generate an `article` landmark.



If a page includes more than one `<article>` element, **each should have a unique label** using either `aria-label` or `aria-labelledby`.

```
<article aria-labelledby="aaa">
  <h3 id="aaa">Article heading</h3>
</article>
```

```
<article aria-label="Comment 302">
  ...
</article>
```

Any questions or comments?

Exercise: Creating landmarks

Accessing the exercise:

DEVELOPER EXERCISE: Creating landmarks.

Look at the layout and see where you could add the following landmarks:

1. `<header>`.
2. `<nav>`.
3. `<main>`.
4. `<aside>`.
5. `<footer>`.
6. `<section>`.

Links and buttons

Semantic meaning

The purpose of links is to **send users to a new location**, i.e. a new page or a different location on the same page.

```
<a href="#">Find out more</a>
```

The purpose of buttons is to **trigger some sort of action** - like submitting a form or opening and closing toggles.

```
<button>Submit</button>
```

Avoiding confusion

From a visual perspective:

- Links **can be** visually styled to look like buttons.
- Buttons **can be** visually styled to look like links.

From a semantic/markup perspective:

- Buttons **should not be** used to operate like links.
- Links **should not be** used to operate like buttons.

Links and assistive technologies

- Screen readers announce links as: *“Link [Link text]”*.
- Screen readers also have a shortcut to list all of the links on a page.

Buttons and assistive technologies

- Screen readers announce buttons as: *“Button [Button text]”*.
- Most screen readers have shortcuts to jump to the next button on the page.
- When a button comes into focus, most screen readers will switch into forms mode.

All of these things **set expectations in the minds of assistive technology users**. If the behaviour does not match their expectations, this can lead to confusion.

So, buttons and links should **always be used for their correct semantic purpose**.

Any questions or comments?

Exercise: Links vs button quiz

When I give you a description of a site function, **answer either “link” or “button” in chat**.

1. Functionality that triggers an accordion to **open and shut**?

2. Functionality that allows users to **download a PDF file**?

3. Functionality that **sends users to a different location further down on the same screen**?

Images and SVGs

A quick overview of image types

Image type	Purpose	Accessible name
Decorative	An image which has no content-related purpose	Use an empty `alt`, or apply as a background image via CSS
Meaningful	An image that contains valuable information and adds to the content	Requires an `alt`
Functional	An image which is part of a UI control - such as a button icon	Requires an `alt`

Accessibility concerns with images

In order for **“meaningful” and “functional” images** to be accessible they require:

- An accessible name.
- In the case of complex images, a description.

The `<img>` element

Images that are classified as “meaninful” or “functional” must have an `alt` attribute as **this provides the accessible name of the image**.

```
<img
  src="path/filename.png"
  alt="Dog"
>
```

In the case of complex images, they can be **given a description** via the `aria-describedby` attribute.

```
<img
  src="path/filename.png"
  alt="Dog"
  aria-describedby="d1"
>
<p id="d1">Longer description.</p>
```

SVG images

SVG images that are classified as “meaninful” or “functional” can be given an accessible name **using three different methods**:

Method 1:

The `<title>` element can be used to provide an accessible name.

```
<svg>
  <title>SVG title</title>
</svg>
```

Method 2

The `aria-label` can be applied to the SVG element.

```
<svg aria-label="SVG name">
</svg>
```

Method 3

The `aria-labelledby` can be applied to the SVG element.

```
<svg aria-labelledby="a1">
</svg>
<p id="a1">Accessible name</p>
```

For complex SVG images, they can be **given a description** via two different methods:

Method 1:

The `<desc>` element can be used to provide an accessible description.

```
<svg>
  <desc>SVG description</desc>
</svg>
```

Method 2

The `aria-describedby` can be applied to the SVG element.

```
<svg aria-describedby="d1">
</svg>
<p id="d1">Accessible description</p>
```

Using `<title>` and `<desc>`

When using `<title>` and `<desc>` elements, the `aria-labelledby` and `aria-describedby` attributes **should also be be used**.

```
<svg aria-labelledby="n1" aria-describedby="d1">
  <title id="n1">SVG title</title>
  <desc id="d1">SVG description</desc>
</svg>
```

Other attributes

To make VoiceOver on the desktop **announce SVGs as images**, you will also need to add `role="image"`.

```
<svg role="img">
</svg>
```

Any questions or comments?

Exercise: Accessible names for images and SVGs

Accessing the exercise:

DEVELOPER EXERCISE: Accessible names for images and SVGs.

Tables

A quick overview of basic table markup

List	Purpose
`<table>`	Presents tabular data in a two-dimensional table
`<thead>`	Defines a set of rows as the head of the table
`<tbody>`	Defines a set of rows as the body of the table
`<tr>`	Defines a row of cells in a table
`<td>`	Defines a cell of a table that contains data

```
<table>
  <thead>
    <tr>
      <td></td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td></td>
    </tr>
  </tbody>
</table>
```

But how do we make our tables more accessible?

The `<caption>` element **specifies a visible label for a table** and must come directly after the `<table>` element's start tag.

This element is important for screen readers as it **provides the table with an accessible name**.

```
<table>
  <caption>Table caption</caption>
</table>
```

The `<th>` element defines **a cell as header of a group of table cells** (table header).

This element is important for screen readers as it **allows users to orientate header cells to table cells within the table**.

```
<table>
  <thead>
    <tr>
      <th></th>
      <th></th>
      <th></th>
    </tr>
  </thead>
...
```

```
...
  </thead>
  <tbody>
    <tr>
      <th></th>
      <td></td>
      <td></td>
    </tr>
  </tbody>
</table>
```

The `scope` attribute defines **whether a header cell relates a column or row of cells within a table**.

This attribute is important for screen readers as it **improves the orientatation of header cells to table cells**.

The `scope="col"` value defines whether the header relates to **all cells of the column it belongs to**.

```
<table>
  <thead>
    <tr>
      <th scope="col"></th>
      <th scope="col"></th>
      <th scope="col"></th>
    </tr>
  </thead>
...
```

	Header cell	scope="col"	Header cell
Header cell	Table cell	Table cell	Table cell
Header cell	Table cell	Table cell	Table cell
Header cell	Table cell	Table cell	Table cell

The `scope="row"` value defines whether the header cell relates to **all cells of the row it belongs to**.

```
...
  </thead>
  <tbody>
    <tr>
      <th scope="row"></th>
      <td></td>
      <td></td>
    </tr>
  </tbody>
</table>
```

	Header cell	Header cell	Header cell
Header cell	Table cell	Table cell	Table cell
scope="row"	Table cell	Table cell	Table cell
Header cell	Table cell	Table cell	Table cell

The `rowgroup` and `colgroup` keywords are also **helpful for irregular tables**.

In extreme cases, such as complex nested tables, **you may need to use the** `headers` attribute.

Unique `id` values are applied to each header cell, and matching `headers` values **are applied to each cell within the table**.

```
<table>
  <thead>
    <tr>
      <th></th>
      <th id="h1"></th>
      <th id="h2"></th>
    </tr>
  </thead>
...
```

```
...
  </thead>
  <tbody>
    <tr>
      <th id="r1"></th>
      <td headers="h1 r1"></td>
      <td headers="h2 r1"></td>
    </tr>
  </tbody>
</table>
```

	id="a"	id="b"	id="c"
id="d"	headers="a d"	headers="b d"	headers="c d"
	id="aa"	id="bb"	id="cc"
id="dd"	headers="aa dd"	headers="bb dd"	headers="cc dd"

Any questions or comments?

Exercise: Marking up tables

Accessing the exercise:

DEVELOPER EXERCISE: Marking up tables.

Lists

Different types of lists

List	Purpose	Child elements allowed
`<ul>`	Unordered list	`<li>` elements only
`<ol>`	Ordered list	`<li>` elements only
`<dl>`	Description list	`<dt>` and `<dd>` elements only

```
<!-- Unordered list -->
<ul>
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
</ul>
```

```
<!-- Ordered list -->
<ol>
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
</ol>
```

```
<!-- Description list -->
<dl>
  <dt>Description term 1</dt>
    <dd>Description definition 1</dd>
  <dt>Description term 2</dt>
    <dd>Description definition 2</dd>
</dl>
```

Why are lists important for accessibility?

From an accessibility point of view, **lists are important for the following reasons**:

1. List can help break up long paragraphs, and **make it easier for anyone to scan groups of similar information**.

2. Lists provide useful visual cues for **people with reading or comprehension difficulties**.

3. When marked up correctly, lists help screen reader users **understand and navigate different types of information**.

Most screen readers provide quick keystroke commands that allow users to **jump the next or previous list**.

When screen readers encounter lists, they will **announce the list and the number of items within the list**.

Any questions or comments?

Exercise: Marking up lists

Accessing the exercise:

DEVELOPER EXERCISE: Accessible names for images and SVGs.

Recap

The way we mark up our layouts and content in HTML can **make an enormous difference to the accessibility of our site**.

Create structure with headings and landmarks, and **choose your content elements carefully**. Keep including those names and roles!