ARIA
ROLES

We are now going to look in more detail at the **different** `role` **attributes**.

We'll start by reviewing this chart showing how the categories of roles have **changed over time**:

https://russmaxdesign.github.io/aria-roles/

It's not possible to cover all `role` attributes today, so I'm going to **highlight some attributes that are important to know about**.

Abstract Roles

ARIA Attributes

Roles (role)

- **Abstract**
- Widget
- Document Structure
- Landmark
- Live Region
- Window

States and Properties (aria-*)

- Widget
- Live Region
- Drag-and-Drop
- Relationship
- Global

Abstract roles are the foundation upon which all other WAI-ARIA roles are built. Content authors **must not use abstract roles** because they are not implemented in the API binding.

- command
- composite
- input
- landmark
- range
- roletype
- section
- sectionhead
- select
- structure
- widget
- window

Widget Roles

ARIA Attributes

Roles (role)

- Abstract
- **Widget**
- Document Structure
- Landmark
- Live Region
- Window

States and Properties (aria-*)

- Widget
- Live Region
- Drag-and-Drop
- Relationship
- Global

Widget roles act can be applied to **standalone user interface widgets** or to parts of larger, composite widgets.

- button
- checkbox
- gridcell
- link
- menuitem
- menuitemcheckbox
- menuitemradio
- option
- progressbar
- radio
- scrollbar
- searchbox
- separator
- slider
- spinbutton
- switch
- tab
- tabpanel
- textbox
- timer
- tooltip
- treeitem

Some developers mistakenly apply these roles to elements that **already have inbuilt semantics** such as:

```html
<button role="button"></button>
<input type="checkbox" role="checkbox">
<a href="#" role="link"></a>
<option value="one" role="option">One</option>
<input type="radio" role="radio">
```

Exercise 3:
Tab, tabpanel tablist

Open **exercise03-in-page-tabs/ start.html** in a browser and also in a text editor.

Review your work against **exercise03- in-page-tabs/finish.html**

```html
<ul role="tablist">
  <li> ... </li>
  <li> ... </li>
  <li> ... </li>
</ul>
```

```html
<ul role="tablist">
  <li>
    <a href="#panel1" id="tab1">Apple</a>
  </li>
  <li>
    <a href="#panel2" id="tab2">Pears</a>
  </li>
  <li">
    <a href="#panel3" id="tab3">Oranges</a>
  </li>
</ul>
```

```
<ul role="tablist">
  <li>
    <a href="#panel1" id="tab1" role="tab">Apple</a>
  </li>
  <li>
    <a href="#panel2" id="tab2" role="tab">Pears</a>
  </li>
  <li>
    <a href="#panel3" id="tab3" role="tab">Oranges</a>
  </li>
</ul>
```

```html
<ul role="tablist">
  <li>
    <a href="#panel1" id="tab1" role="tab"
    aria-controls="panel1">Apple</a>
   </li>
  <li>
    <a href="#panel2" id="tab2" role="tab"
    aria-controls="panel2">Pears</a>
   </li>
  <li>
    <a href="#panel3" id="tab3" role="tab"
    aria-controls="panel3">Oranges</a>
    </li>
</ul>
```

```html
<ul role="tablist">
  <li>
    <a href="#panel1" id="tab1" role="tab"
    aria-controls="panel1" aria-selected="true">Apple</a>
  </li>
  <li>
    <a href="#panel2" id="tab2" role="tab"
    aria-controls="panel2" aria-selected="false">Pears</a>
  </li>
  <li>
    <a href="#panel3" id="tab3" role="tab"
    aria-controls="panel3" aria-selected="false">Oranges</a>
  </li>
</ul>
```

```html
<div id="panel1" role="tabpanel">
  Panel 1
</div>
<div id="panel1" role="tabpanel">
  Panel 2
</div>
<div id="panel1" role="tabpanel">
  Panel 3
</div>
```

```html
<div id="panel1" role="tabpanel" aria-labelledby="tab1">
  Panel 1
</div>
<div id="panel1" role="tabpanel" aria-labelledby="tab2">
  Panel 2
</div>
<div id="panel1" role="tabpanel" aria-labelledby="tab3">
  Panel 3
</div>
```

Widget Container Roles

There are also composite user interface widgets. These roles typically **act as containers that manage other, contained widgets**.

- combobox
- grid
- listbox
- menu
- menubar
- radiogroup
- tablist
- tree
- treegrid

Exercise 4:
The radiogroup attribute

Open **exercise04-radio-group/ start.html** in a browser and also in a text editor.

Review your work against **exercise04- radio-group/finish.html**

Ideally, any set of radio buttons or checkboxes should have an **overall description associated with them** to provide context.

The **ideal solution** is to use the `<fieldset>` and `<legend>` elements.

The `<fieldset>` allows authors to **group thematically related controls** and labels. The `<legend>` element allows authors to **assign a caption** to a `<fieldset>`.

```html
<fieldset>
  <legend>Do you like apples?</legend>
  <div>
    <input type="radio" id="apples-y" name="apples">
    <label for="apples-y">Yes</label>
  </div>
  <div>
    <input type="radio" id="apples-n" name="apples">
    <label for="apples-n">No</label>
  </div>
</fieldset>
```

However, there may be times when you are **not able to use** a `<fieldset>` element. Luckily, we can use ARIA to help solve the problem

The `radiogroup` role **defines a group of radio buttons** to Assistive Technolgies.

```html
<div role="radiogroup" aria-labelledby="sub">
  <h3 id="sub">Pick a subscription</h3>
  <div>
    <input type="radio" id="daily" name="subscribe">
    <label for="daily">Daily</label>
  </div>
  <div>
    <input type="radio" id="weekly" name="subscribe">
    <label for="weekly">Weekly</label>
  </div>
</div>
```

The `aria-labelledby` attribute is used to **establish a relationship** between the group and it's label - in this case the `<h3>` element.

```html
<div role="radiogroup" aria-labelledby="sub">
  <h3 id="sub">Pick a subscription</h3>
  <div>
    <input type="radio" id="daily" name="subscribe">
    <label for="daily">Daily</label>
  </div>
  <div>
    <input type="radio" id="weekly" name="subscribe">
    <label for="weekly">Weekly</label>
  </div>
</div>
```

The `<h3>` will **now act in a similar way** to the `<legend>` element when inside a `<fieldset>` element.

Document Structure Roles

ARIA Attributes

Roles (role)

- Abstract
- Widget
- **Document Structure**
- Landmark
- Live Region
- Window

States and Properties (aria-*)

- Widget
- Live Region
- Drag-and-Drop
- Relationship
- Global

Document structure roles **describe structures that organise content in a page**. Document structures are not usually interactive.

- application
- article
- cell (1.1)
- columnheader
- definition
- directory
- document
- feed (1.1)
- figure (1.1)
- group
- heading
- img
- list

- listitem
- math
- none (1.1)
- note
- presentation
- row
- rowgroup
- rowheader
- separator
- table (1.1)
- term (1.1)
- toolbar
- tooltip

Some developers mistakenly apply roles to elements that **already have inbuilt semantics** such as:

```html
<h2 role="heading"></h2>
<img href="image2.png" role="img">
<ul role="list"></ul>
<table role="table"></table>
```

Issues with application and document

When the `role="application"` is applied, all content inside this element acts as if the screen reader is **locked in forms mode**. So users can interact with form controls but cannot read/navigate in read mode.

For this reason, `application` **should be used sparingly**!

Exercise 5:
Presentation and None

Open **exercise05-presentation-none/ start.html** in a browser and also in a text editor.

Review your work against **exercise05- presentation-none/finish.html**

The `presentation` role is used to **remove semantic meaning** from an element and any of its related child elements.

For example, a `<table>` used for layout purposes could have the role of `presentation` applied to the table element **to remove any semantic meaning from the table element**.

```
<table role="presentation">
  ...
</table>
```

The role of none was added in ARIA 1.1 as the concept of "presentation" **was confusing the developers**. The current thought is that `role="none"` would make more sense.

```html
<table role="none">
    ...
</table>
```

The values of `presentation` and `none` are **identical in purpose**.

The `presentation` value is **more backwards compatible** as it has been around for longer.

Add a role of presentation to the first table, and role of none to the second table.

Landmark Roles

ARIA Attributes

Roles (role)

- Abstract
- Widget
- Document Structure
- **Landmark**
- Live Region
- Window

States and Properties (aria-*)

- Widget
- Live Region
- Drag-and-Drop
- Relationship
- Global

Landmark roles, sometimes just referred to as "landmarks", help to **programmatically identify** sections of a web page.

- banner
- complementary
- contentinfo
- form
- main
- navigation
- region (1.1)
- search

```html
<div role="banner"></div>
<div role="complementary"></div>
<div role="contentinfo"></div>
<div role="form"></div>
<div role="main"></div>
<div role="navigation"></div>
<div role="region"></div>
<div role="search"></div>
```

Landmarks help assistive technology users orient themselves to a page and **navigate easily to various sections** of a page.

Landmarks also provide an easy way for assistive technology users to **skip over blocks of content** that are repeated on multiple pages.

For example, if there is a common navigation menu found on every page, landmark roles **can be used to skip over it and navigate** from section to section.
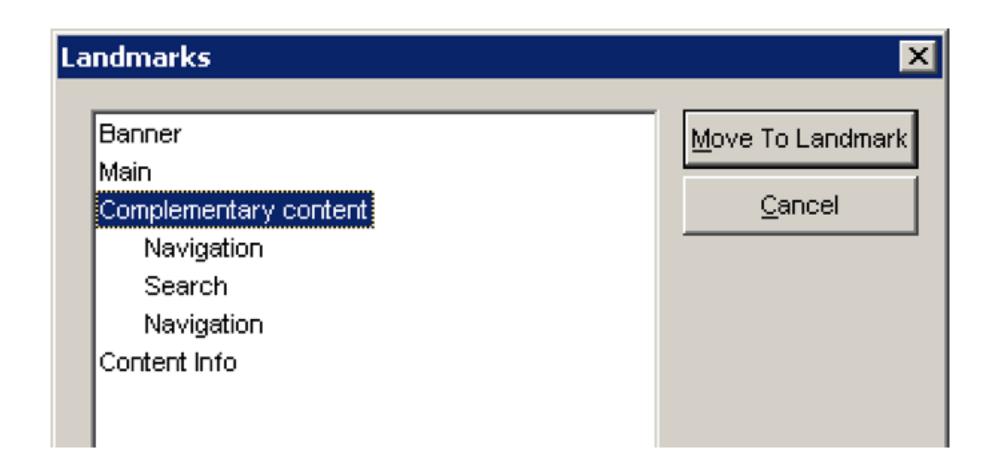
```
<nav role="navigation">
  <ul>
    <li><a href="#">About</a></li>
    <li><a href="#">Services</a></li>
    ...
  </ul>
</nav>
```

This saves assistive technology users and keyboard users from **having to tab through a large amount of content** to find what they are really after, much like a traditional "skip links" mechanism.

Landmark roles are generally **well supported** by JAWS, NVDA and Mac OSX Voiceover.

Roles are **announced to Assistive Technologies** as something like: "Navigation landmark"

Assistive technology users can use **keyboard shortcuts** or (in the case of JAWS and VoiceOver) a **dialog box** to navigate around web pages via Landmark roles.

Landmarks

- Banner
- Main
- **Complementary content**
  - Navigation
  - Search
  - Navigation
- Content Info

[Move To Landmark]

[Cancel]

For most modern browsers and Assistive Technologies, there is no need to include the **native HTML5 element as well as the role attribute** as the HTML element is announced as "navigation".

```html
<nav role="navigation">
  <ul>
    <li><a href="#">About</a></li>
    <li><a href="#">Services</a></li>
    ...
  </ul>
</nav>
```

However, to support IE11 and previous versions of IE, **the native element and the role attribute should be used**.

Exercise 6: Landmark roles

Open **exercise06-landmark-roles/ start.html** in a browser and also in a text editor.

Review your work against **exercise06- landmark-roles/finish.html**

Open **exercise04/start.html** in a browser and also in a text editor.

banner

A `banner` landmark identifies **site-oriented content** at the beginning of each page within a website.

```html
<header role="banner">
</header>
```

Site-oriented content **typically includes** things such as the logo or identity of the site sponsor, and site-specific search tool.

Each page **may have one** banner landmark.

The **banner** landmark **should be a top-level landmark** (e.g. not contained within any other landmarks).

The banner landmark **can be used in conjunction with the** `<header>` element.

```html
<header role="banner">
</header>
```

complementary

The `complementary` role is used to describe a region of content that is **complementary to the main content** - such as an `<aside>`.

```html
<aside role="complementary">
  <h2>Title of complementary area</h2>
</aside>
```

This content should still be **meaningful** when separated from the main content.

The `complementary` landmark **should be a top level landmark** (e.g. not contained within any other landmarks).

The `complementary` landmark **can be used in conjunction with the** `<aside>` element.

```html
<aside role="complementary">
  <h2>Title of complementary area</h2>
</aside>
```

If a page includes more than one `complementary` landmark, **each should have a unique label**.

```html
<aside role="complementary" aria-labelledby="one">
  <h2 id="one">Title of complementary area</h2>
</aside>
```

contentinfo

The `contentinfo` landmark can be used to **identify common information at the bottom of each page** within a website, typically called the "footer" of the page.

```html
<div class="contentinfo">
  <h2>Footer heading</h2>
</div>
```

Footer information **could include** copyright information and links to privacy and accessibility statements.

Each page **may have one** `contentinfo` landmark.

The `contentinfo` landmark **should be a top-level landmark** (e.g. not contained within any other landmarks).

The `contentinfo` landmark **can be used in conjunction with the** `<footer>` element.

```html
<footer role="contentinfo">
  <h2>Footer heading</h2>
</footer>
```

form

The `form` landmark is **used to describe a region that is a** `<form>`, but only when no other named landmark is appropriate (such as main or search).

```html
<div role="form">
  <form action="#">
    ...
  </form>
</div>
```

If the `<form>` is used for search functionality, the `search` landmark **should be used instead**.

If the `form` landmark is present, the element should also have a **label to help users understand the purpose** of the `<form>`.

A label for the `form` landmark **should be identified using** `aria-labelledby` to a visible heading element.

```html
<div role="form" aria-labelledby="contact">
  <form action="#">
    <h2 id="contact">Add contacts</h2>
  </form>
</div>
```

The `form` landmark **should not be used directly on the** `<form>` element, as it can override the native form semantics.

Instead, **the landmark can be placed on a** `<div>` element which is then wrapped around the outside of the `<form>` element.

```html
<div role="form">
  <form action="#">
    ...
  </form>
</div>
```

If a page includes more than one `form` landmark, **each should have a unique label**.

```
<div role="form" aria-labelledby="one">
  <form action="#">
    <h2 id="one"></h2>
  </form>
</div>
<div role="form" aria-labelledby="two">
  <form action="#">
    <h2 id="two"></h2>
  </form>
</div>
```

main

The `main` landmark is used to **describe the primary content of the page**.

```html
<div role="main">
  <h1>Title for main content</h1>
</div>
```

Each page **should have only one** `main` landmark.

The `main` landmark **can be used in conjunction with the** `<main>` element.

```html
<main role="main">
    <h1>Title for main content</h1>
</main>
```

navigation

The `navigation` landmark provide a way to **identify groups of links** that are intended to be used for website or page content navigation.

```html
<div role="navigation">
  <h2>Title for navigation</h2>
  <ul>
    <li><a href="#">Link1</a></li>
    <li><a href="#">Link2</a></li>
    <li><a href="#">Link3</a></li>
  </ul>
</div>
```

The `navigation` landmark **can be used in conjunction with the** `<nav>` element.

```html
<nav role="navigation">
  <h2>Title for navigation</h2>
  <ul>
    <li><a href="#">Link1</a></li>
    <li><a href="#">Link2</a></li>
    <li><a href="#">Link3</a></li>
  </ul>
</nav>
```

Developers **may want to include more than one** `navigation` **landmark**, such as the primary and secondary navigation menus.

If a page includes more than one `navigation` landmark, **each should have a unique label**.

These labels are created using an `aria-labelledby` attribute on the container, and a matching ID value on the heading inside.

```
<nav role="navigation" aria-labelledy="one">
  <h2 id="one">Main navigation</h2>
</nav>


<nav role="navigation" aria-labelledy="two">
  <h2 id="two">Sub navigation</h2>
</nav>
```

region

The `region` landmark is **used to identify a section of content** that is important enough to stand on its own.

```html
<div role="region">
  <h2>Region heading</h2>
</div>
```

The `region` landmark **can be used in conjunction with the** `<section>` element.

```
<section role="region">
  <h2>Region heading</h2>
</section>
```

A `region` landmark **must have a label**. If a page includes more than one `region` landmark, each should have a unique label.

```
<section role="region" aria-labelledby="one">
  <h2 id="one">Region heading one</h2>
</section>

<section role="region" aria-labelledby="two">
  <h2 id="two">Region heading two</h2>
</section>
```

search

The `search` landmark is **used to describe a region that is a** `<form>`, but only that with search functionality for content on the website.

```
<div role="search">
  <form action="#">
    ...
  </form>
</div>
```

The `search` landmark **should not be used directly on the** `<form>` element, as it can override the native form semantics.

The landmark can be placed on a `<div>` element **which is then wrapped around the outside of the** `<form>` element.

```html
<div role="search">
  <form action="#">
    ...
  </form>
</div>
```

If a page includes more than one `search` landmark, **each should have a unique label**.

```
<div role="search" aria-labelledby="one">
  <form action="#">
    <h2 id="two">Form heading one</h2>
  </form>
</div>

<div role="search" aria-labelledby="two">
  <form action="#">
    <h2 id="two">Form heading two</h2>
  </form>
</div>
```

Live Region Roles

ARIA Attributes

Roles (role)

- Abstract
- Widget
- Document Structure
- Landmark
- **Live Region**
- Window

States and Properties (aria-*)

- Widget
- Live Region
- Drag-and-Drop
- Relationship
- Global

Live Region roles define live regions of a document and may be modified by live region attributes.

- alert
- log
- marquee
- status
- timer

Exercise 7:
Alert and Status

Open **exercise07-alert-status/ start.html** in a browser and also in a text editor.

Review your work against **exercise07- alert-status/finish.html**

role=alert

Alerts are used to **convey messages to alert the user** - usually time-sensitive, information.

The alert role goes on the **element containing the alert message**.

```html
<div role="alert">
  ...
</div>
```

Alerts are a specialised type of the `status` **role**.

Authors are **not required to set focus on alerts** in order for them to be processed.

Elements with the `role` of `alert` have an **implicit** `aria-live` value of `assertive`, and an **implicit** `aria-atomic` value of `true`.

role=status

A type of live region whose content is **advisory information** for the user but is not important enough to justify an alert.

```html
<div role="status">
  ...
</div>
```

Authors should **not set focus on status messages** as a result of change in status.

Elements with the role status have an **implicit** `aria-live` value of `polite` and an **implicit** `aria-atomic` value of `true`.

Add a `role` of `alert` to the **first message**, and `role` of `status` to the **second message**.

Window Roles

ARIA Attributes

Roles (role)

Abstract

Widget

Document Structure

Landmark

Live Region

Window

States and Properties (aria-*)

Widget

Live Region

Drag-and-Drop

Relationship

Global

Window roles act as windows within the browser or application.

- alertdialog
- dialog

Exercise 8: Alertdialog

Open **exercise08-alertdialog/ start.html** in a browser and also in a text editor.

Review your work against **exercise08- alertdialog/finish.html**

They are a type of dialog that contains a message to alert users, **where initial focus goes to an element within the dialog**.

The `alertdialog` role goes on the element **containing both the alert message and the rest of the dialog**.

When the `alert` dialog is displayed, authors **SHOULD set focus to an active element** within the alert dialog, such as a form edit field or an OK button.

Authors should make sure that while the `alertdialog` is shown, keyboard and mouse interactions **only operate within the dialog**.

```html
<div role="alertdialog">
  <p>Alert message</p>
</div>
```

Authors should use `aria-describedby` on an `alertdialog` to reference the alert message element in the dialog.

```
<div role="alertdialog" aria-describedby="a1">
  <p id="a1">Alert message</p>
</div>
```