HTML5 VIDEO

What is the problem?

Generally speaking, HTML has a very limited set of interface controls and interactions.

As the demand for rich interactions has increased, JavaScript has become our saviour!

JavaScript provides us with many things including:

dynamic interactions:
such as drag and drop, resizing, hide and show, open and shut, switch views etc.

widgets and components:
such as modals, in-page tabs, date pickers, page loaders, sliders etc.

However, many of dynamic interactions and widgets are **problematic** for Assistive Technologies.

Assistive Technologies may not be aware of **content that has been updated** after the initial page has loaded.

Many widgets are not accessible for **keyboard-only users**.

WAI ARIA
to the rescue

WAI: Web Accessibility Initiative

ARIA: Accessible Rich Internet Applications

WAI-ARIA allows us to use HTML attributes to define the **role, states and properties** of specific HTML elements.

Role: what is it?

Is it a widget (menu, slider, progress bar etc.) or an important type of element (heading, region, table, form etc.)

State: What is the current state of the widget?

Is it checked, disabled etc.

Properties: What are the properties of the widget?

Does it have live regions, does it have relationships with other elements, etc?

Keyboard navigation:

ARIA also provides keyboard navigation methods for the web objects and events.

Things to avoid

Things to avoid

Don’t use ARIA unless you really need to.

“If you can use a native HTML element [HTML5] or attribute with the semantics and behaviour you require already built in, instead of re-purposing an element and adding an ARIA role, state or property to make it accessible, then do so.”

http://www.paciellogroup.com/blog/2014/10/aria-in-html-there-goes-the-neighborhood/

Where possible, use **correct semantic HTML elements**. Don’t use ARIA as a quick-fix.

```html
<!-- avoid this if possible -->
<span role="button">...</span>

<!-- this is preferred -->
<button type="button">...</button>
```

Avoid overuse of ARIA attributes. In many case, you do not need to describe an elements role or state.

Avoid misuse ARIA attributes. Make sure you don’t just copy and paste ARIA attributes and assume that they work.

If you must use aria

1. Alert users to the widget or elements **role** (button, checkbox etc).

2. Alert users to the **properties and relationships** of the element (disabled, required, other labels etc).

3. Alert users to the **original state** of the element (checked, unchecked etc).

4. Alert users to **changes in state** of the element (now checked etc)

5. Make sure widgets are **keyboard accessible** (including predictable focus).

ARIA landmark roles

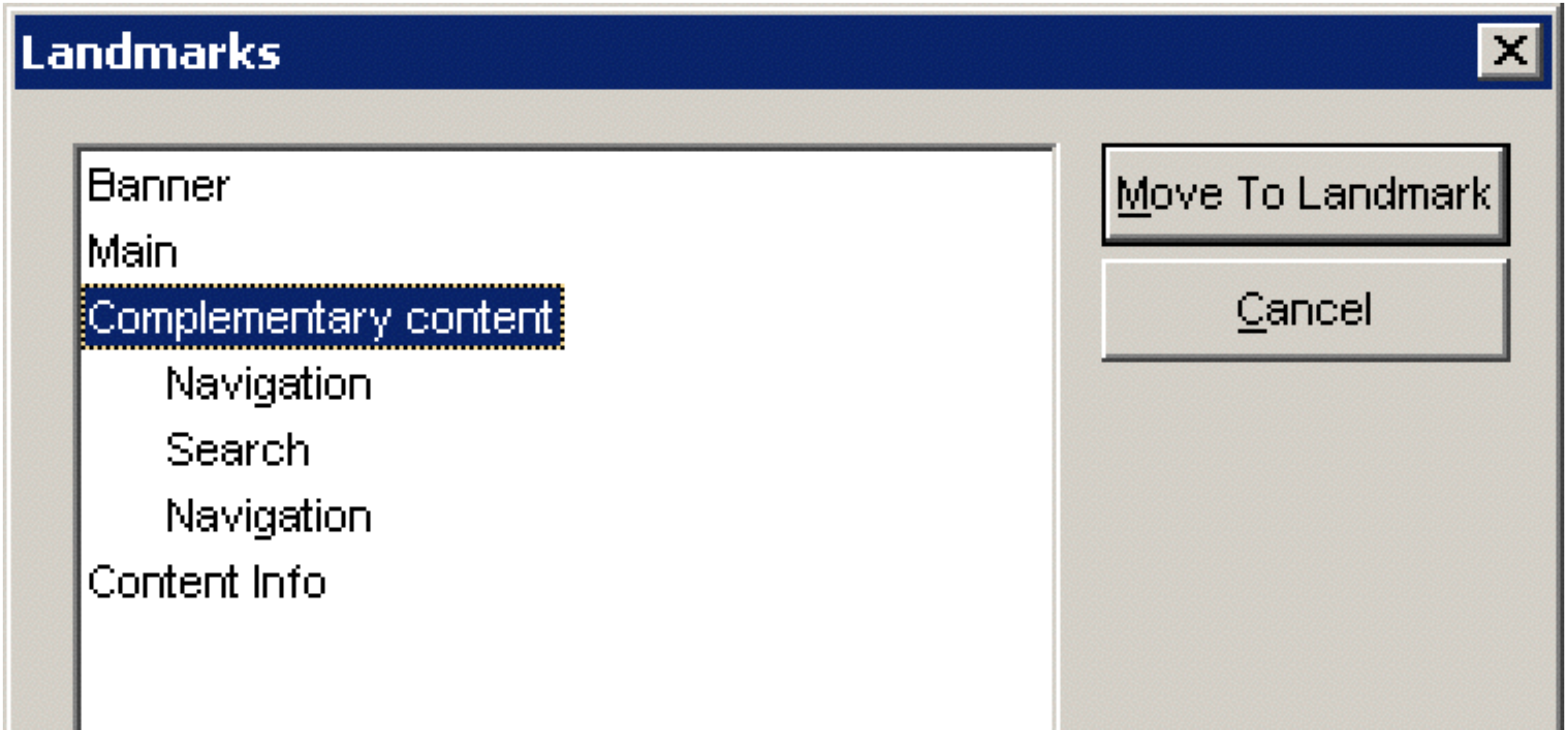
Landmark roles help us to **describe the overall document structure** to assistive devices.

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

Roles are **announced** to the Assistive Technology eg: “Navigation landmark”

```html
<nav role="navigation">
    <ul>
        <li><a href="#">home</a></li>
        <li><a href="#">about</a></li>
    </ul>
</nav>
```

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


Landmarks

Banner
Main
Complementary content
Navigation
Search
Navigation
Content Info

banner

The **banner role** is used to describe a region that is overall site orientated, such as a primary site header.

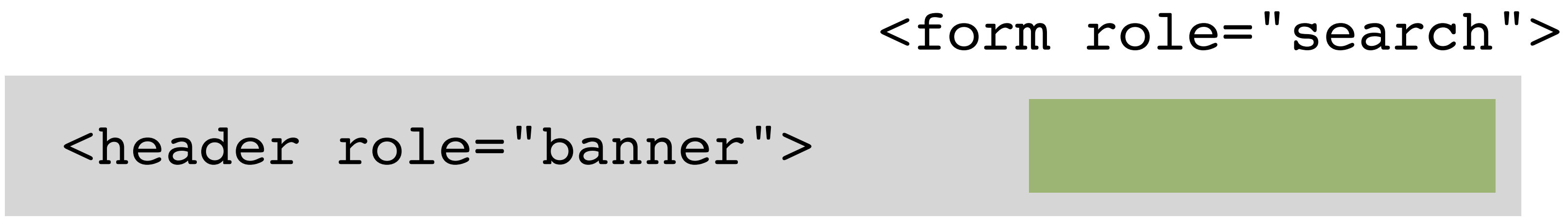
```html
<header role="banner">
</header>
```

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

search

The **search role** is used to describe a search function.

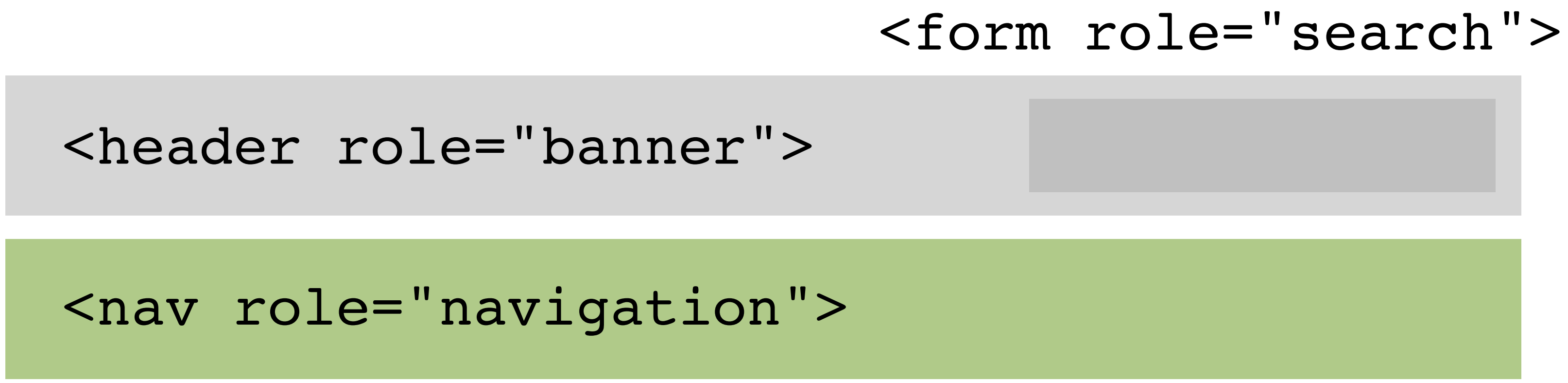
```html
<form role="search">
</form>
```



navigation

The **navigation role** is used to describe a collection of primary navigation elements.

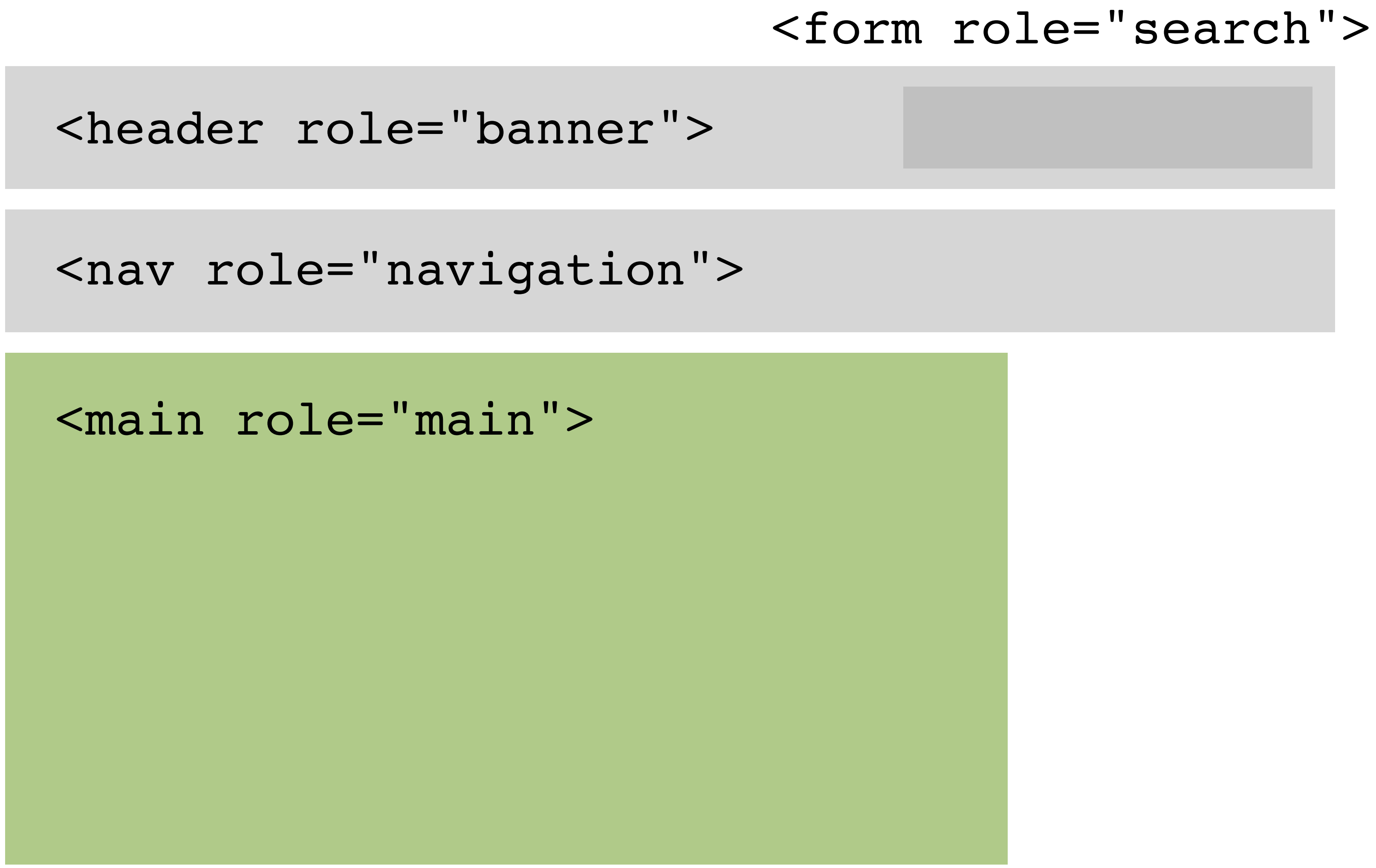
```
<nav role="navigation">
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Services</a></li>
    </ul>
</nav>
```



main

The **main role** is used to describe the main content of the document.

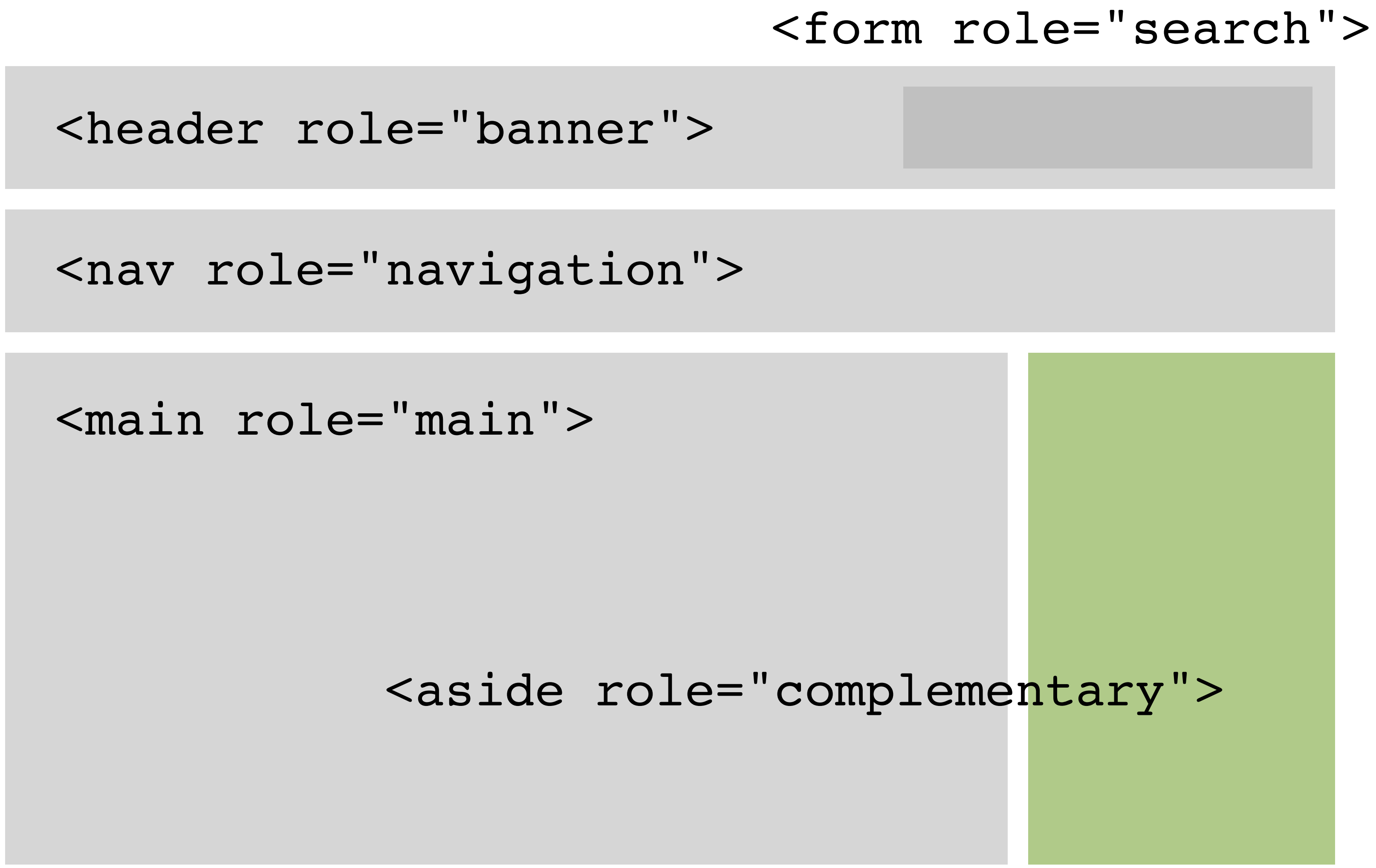
```html
<main role="main">
</main>
```



complementary

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

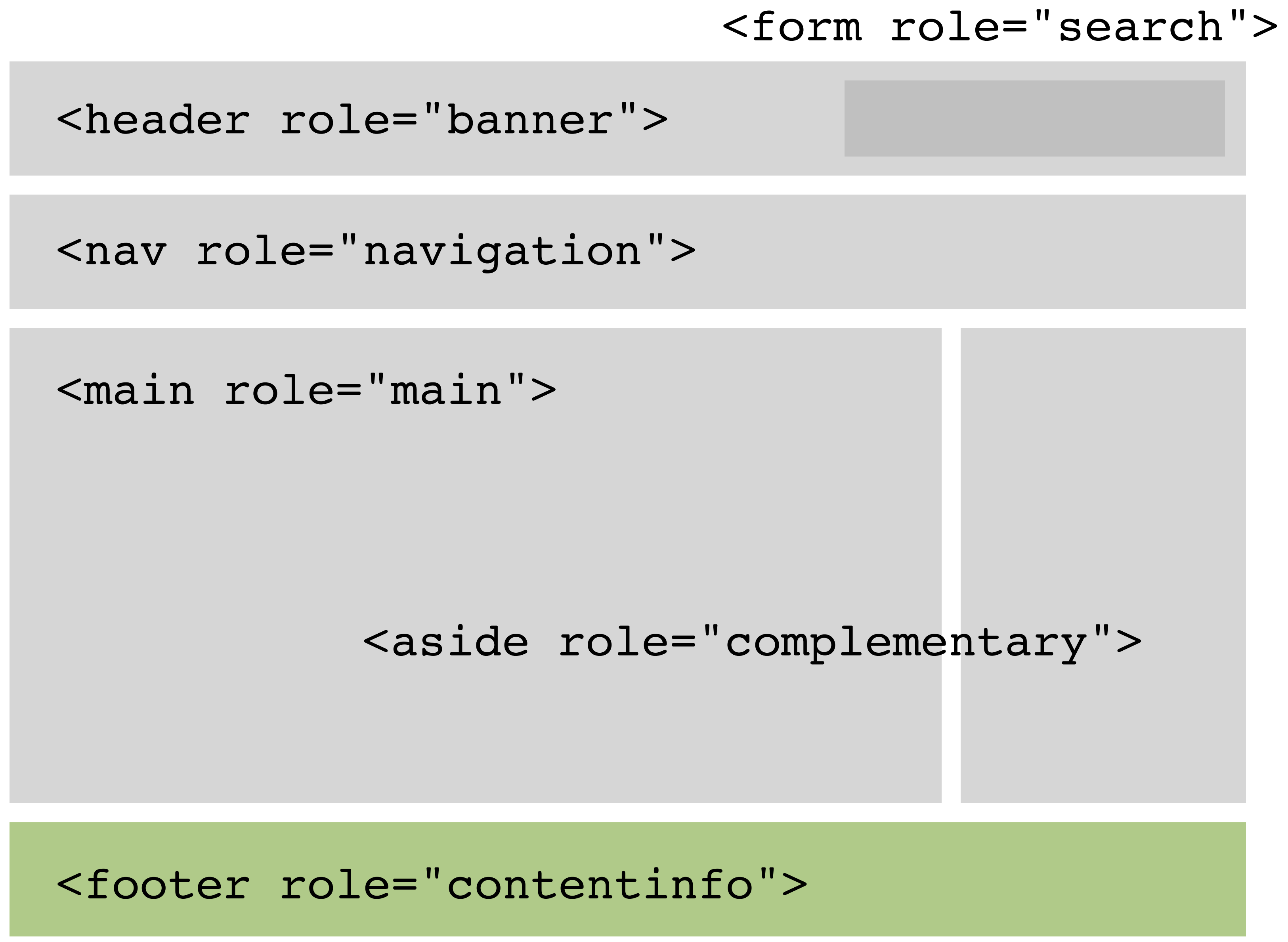
```html
<aside role="complementary">
</aside>
```



contentinfo

The **contentinfo role** is used to describe a regions that define the overall document, like a footer.

```html
<footer role="contentinfo">
</footer>
```



ARIA in forms

ARIA roles are particularly useful **when used with form controls**, allowing us to describe functions in a variety of powerful ways.

For example, many instructions associated with form controls use **presentational strategies alone**.

We can enhance these strategies for ATs using ARIA to **describe aspects of the form and/or its functionality**.

aria-required

The **aria-required attribute** allows one of two possible values - “true” or “false”.

```html
<input aria-required="true" type="text">
<input aria-required="false" type="text">
```

The HTML5 **“required” attribute** defines the element as “required”.

```html
<input required aria-required="true"
name="name" id="name" type="text">
```

The **“aria-required” attribute** describes this to Assistive Technologies.

```html
<input required aria-required="true"
name="name" id="name" type="text">
```

aria-disabled

The aria-disabled attribute allows one of two possible values - “true” or “false”.

```html
<input aria-disabled="true" type="text">
<input aria-disabled="false" type="text">
```

The HTML5 **“disabled” attribute** defines the element as “disabled”.

```
<input disabled aria-disabled="true"
name="name" id="name" type="text">
```

```html
<input disabled aria-disabled="true"
name="name" id="name" type="text">
```

The **aria-disabled attribute** describes this to Assistive Technologies.

```html
<input disabled aria-disabled="true"
name="name" id="name" type="text">
```

aria-describedby

The **aria-describedby attribute** allows developers to associate an element with some additional descriptive information.

```html
<div>
    <label for="a">Date:</label>
    <input aria-describedby="format"
        type="text" name="a" id="a">

    <span id="format">
        (must be mm/dd/yyyy)
    </span>
</div>
```

aria-label

The **aria-label attribute** allows developers to provide different content to Assistive Technologies - such as the word “Download” being replaced with “Download full movie”.

```
<button aria-label="Download full movie">
    Download
</button>
```

It can also be used to **replace content that is purely presentational** - with descriptive information - such as a “x” close button.

```
<button type="button" aria-label="Close and
return to account details">
    x
</button>
```

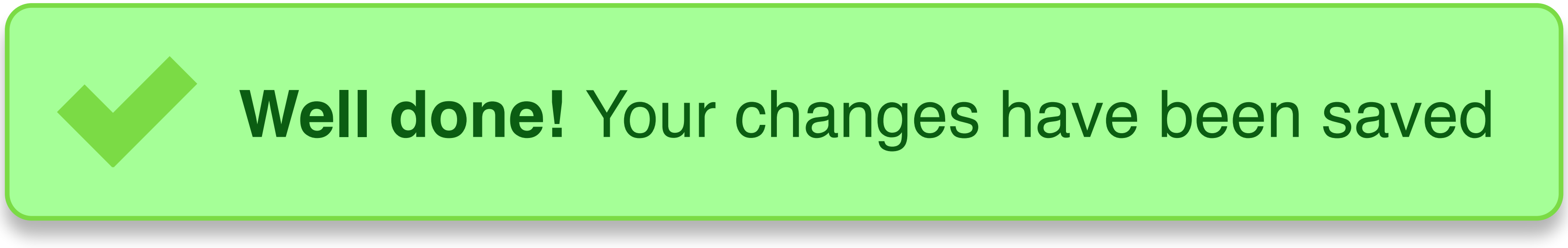
The **aria-label attribute** can also be used to provide meaning to Assistive Technologies that is not shown to others.

```
<a href="#" aria-label="Falkland Islands
Malvinas Pound">
    <span aria-hidden="true">FKP</span>
</a>
```

ARIA live

What is aria-live?

In a web application, you want a simple notification to **appear at the top of the page** when a user completes a task.



✓ **Well done!** Your changes have been saved

This dynamically inserted notification can cause **two problems for screen readers**.

Problem 1:

Screen readers “buffer” pages as they are loaded. Any content that is added after this time many not be picked up by the screen reader.

Problem 2:

Screen readers can only focus on one part of the page at a time. If something changes on another area of the page, screen readers may not pick this up.

The aria-live attribute allows us to **notify screen readers when content is updated** in specific areas of a page.

How is aria-live applied?

We can apply the aria-live attribute to **any HTML element**.

```html
<div aria-live="polite">
</div>
```

If we then use JavaScript to inject/hide/show content within this element screen readers will be **made aware of any DOM changes within that element**.

```html
<div aria-live="polite">

    <!-- Dynamic content -->

</div>
```

There are **three possible values** for aria-live:

```html
<div aria-live="off">
</div>
```

aria-life: “off”

Assistive technologies should not announce updates unless the assistive technology is currently focused on that region.

aria-life: “off” can be used for information that is **not critical** for users to know about immediately.

```html
<div aria-live="polite">
</div>
```

aria-life: “polite”

Assistive technologies should announce updates at the next graceful opportunity (eg end of current sentence).

aria-life: “polite” can be used for **warning notifications** that users may need to know.

```
<div aria-live="assertive">
</div>
```

aria-life: “assertive”

Assistive technologies should announce updates immediately.

aria-life: “assertive” should only be used if the interruption is **imperative for users to know immediately** such as error alerts.

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley