

## 1. WCAG definition

The **Web Content Accessibility Guidelines (WCAG)** are part of a series of web accessibility. They are a set of recommendations for making Web content more accessible, primarily for people with disabilities—but also for all user agents, including highly limited devices, such as mobile phones.

WCAG's accessibility standards are based on four principles (often referred to as POUR):

- Perceivable = Information and user interface components must be presented to users in ways that they can perceive. For example, it's important to present information that can be perceived in different ways, where a user can adjust color contrast or font size, or view captions for videos.
- Operable = User interface components and navigation must be functional for users in ways they can operate. For example, a user must be able to perform required interactions using a keyboard or voice commands, not just using a mouse.
- Understandable = Information and user interface operation must be understandable. For example, information and instructions should be clear and navigation methods should be easy to understand and use.
- Robust = Content must be robust enough so that it can be interpreted reliably by a wide variety of users and types of assistive technologies. As technologies evolve, code and content should remain accessible for users of common and current assistive devices and tools.

WCAG has three levels of conformance: A, AA, and AAA. Level A refers to the lowest level of conformance (minimum) and Level AAA is the highest (maximum).

Principles	Guidelines	Level A	Level AA	Level AAA
1. Perceivable	1.1 Text Alternatives	1.1.1		
	1.2 Time-based Media	1.2.1 – 1.2.3	1.2.4 – 1.2.5	1.2.6 – 1.2.9
	1.3 Adaptable	1.3.1 – 1.3.3		
	1.4 Distinguishable	1.4.1 – 1.4.2	1.4.3 – 1.4.5	1.4.6 – 1.4.9
2. Operable	2.1 Keyboard Accessible	2.1.1 – 2.1.2		2.1.3
	2.2 Enough Time	2.2.1 – 2.2.2		2.2.3 – 2.2.5
	2.3 Seizures	2.3.1		2.3.2
	2.4 Navigable	2.4.1 – 2.4.4	2.4.5 – 2.4.7	2.4.8 – 2.4.10
3. Understandable	3.1 Readable	3.1.1	3.1.2	3.1.3 – 3.1.6
	3.2 Predictable	3.2.1 – 3.2.2	3.2.3 – 3.2.4	3.2.5
	3.3 Input Assistance	3.3.1 – 3.3.2	3.3.3 – 3.3.4	3.3.5 – 3.3.6
4. Robust	4.1 Compatible	4.1.1 – 4.1.2		

## 2. WCAG points which should be covered in Hello World

Based on Hello world's requirements the following points are going to be covered in the Hello World project: 1.3.1 and 2.4.2. Their description is presented below:

## 2.1 Web Content Accessibility Guideline 1.3.1 - Info and Relationships

According to WCAG 1.3.1 Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)

All users benefit when your website structure is logical and each section of content has a clear relationship with the content around it. Visual cues like headings, bullet points, line breaks, tables, bolding, underlining links and other formatting choices help users understand the content.

Assistive technology often relies on correct formatting and logical structures to work. When a user experiences the site through a screen reader, other assistive technology or without CSS they should still understand the content.

The success criteria 1.3.1 info and relationships considers the problems encountered due to presentational cues available to assistive technology users programmatically determinable or described in text. When the content authors have a choice to provide a relationship between content programmatically determinable and also in text description, a programmatically determinable method is highly recommended.

### Specific Benefits of Success Criterion 1.3.1:



- This Success Criterion helps people with different disabilities by allowing user agents to adapt content according to the needs of individual users.
- Users who are blind (using a screen reader) benefit when information conveyed through color is also available in text (including text alternatives for images that use color to convey information).
- Users who are deaf-blind using braille (text) refreshable displays may be unable to access color-dependent information.

### 2.1.1 How do you do it?

- Use proper and complete native HTML code to define structure and relationships. Some examples include:
  - [Using semantic elements to mark up structure](#) AND [Using semantic markup to mark emphasized or special text](#)
  - [Using h1-h6 to identify headings](#)
  - [Using ol, ul and dl for lists or groups of links](#)

- [Using label elements to associate text labels with form controls](#)
  - [Using table markup to present tabular information](#)
- Use [WAI-ARIA](#) markup when appropriate to programmatically define cues that may be missed when accessing content via assistive technology. Some examples include:
  - [Using ARIA landmarks to identify regions of a page](#)
  - [Using role=heading to identify headings](#)
  - [Using aria-labelledby to provide a name for user interface controls](#)
  - [Using grouping roles to identify related form controls](#)

#### Additional resources to help you

- [Accessible Rich Internet Applications \(WAI-ARIA\) 1.1 - W3C Recommendation](#)
- [Use headings to convey meaning and structure - W3C Tips](#)
- [Semantic Structure: Regions, Headings, and Lists](#)
- [WCAG – 1.3.1 Info And Relationships \(Level A\)](#)
-  [WCAG 2.1 Article 1.3.1](#)
-  [WCAG 1.3.1 missing landmarks on Facebook login page](#)

#### 2.2 Web Content Accessibility Guideline 2.4.2 - Web pages have titles that describe topic or purpose. (Level A)

Having a descriptive title on each webpage or electronic document helps all users. Page titles are used to quickly identify where you are in the website, gain information regarding content on the page, and help differentiate web pages when multiple tabs are open. Having the same title

used on multiple pages in a website adds extra burden for screen reader users who rely on unique page titles to immediately understand where they are.

In addition, having unique page titles also helps with internet searches. One method used by search engines is to look for relevance of page titles and display those titles in a prioritized list for users to choose from.

What should you do? First, be sure that each web page of a website has a title. Second, ensure that the page titles you create are unique, concise and reflect the content of the page.

#### Points to Ponder

- Provide a unique title.
- Make sure that title is between 50-75 characters.
- Make sure the title of the page is the heading level H1 on the page.
- Title should contain web page name, bit of description & site name.
- Make sure the title describes the purpose of the page.

#### 2.2.1 How do you do it?

Web Pages: Website titles should be coded in the HTML document using a <title> tag. Here is a code example:

```
<!DOCTYPE html>
<html>

<head>
<title>Understanding Page Titles</title>
</head>

<body>
The content of the document...
</body>

</html>
```

References:

[Understanding Success Criterion 2.4.2](#)

 [WCAG 2.1 Article 2.4.2](#)

### 3. The tools used for checking the semantic structure

#### **Bookmarklets**

Bookmarklets, also known as favelets, are little snippets of JavaScript dressed up as bookmarks and require no installation beyond dragging a link containing the script from a page into your browser's Favorites or Bookmarks Bar. Then, whenever you are on a page you want to test, you just click on the link to the bookmark and it'll run the script on the page you are viewing and display the results. The advantage of these is that they work in any browser.

#### **ANDI (Accessible Name & Description Inspector)**

A free tool used in the [Trusted Tester Process](#) that will:

- Provide automated detection of accessibility issues
- Reveal what a screen reader should say for interactive elements
- Give practical suggestions to improve accessibility and check 508 compliance

📺 ANDI Introduction

#### **Accessibility Bookmarklets**

A set of bookmarklets, that highlight [ARIA](#) Landmarks and HTML5 sectioning elements, headings, lists, images, and form-related elements.

## HTML\_CodeSniffer

A relatively comprehensive accessibility testing tool that reports unambiguous *Errors* for stuff it is confident is wrong, provides *Warnings* for content it thinks might be problematic but isn't sure about, and *Notifications*, which are reminders to check certain things based on the content in the pages

## JavaScript Bookmarklets for Accessibility Testing

A number of bookmarklets from J. Paul Adam that let you query attributes and visualize information related to [ARIA](#), [images](#), [forms](#), [headings](#), [tables](#), [landmarks](#), [language use](#), [tabindex](#), [title attributes](#), [iframes](#), [lists](#), and [focus](#).

## Tota11y

A simple accessibility testing tool that highlights problems on the page. It also includes a tool that lets you hover over elements on the page to view them as a screen reader would.

▶ Accessibility Tools - Tota11y

## Web Accessibility Favelets

A set of bookmarklets used in the Trusted Tester process used by the Department of Homeland Security

### Visual ARIA Bookmarklet

Visual ARIA allows engineers, testers, educators, and students to physically observe ARIA usage within web technologies, including ARIA 1.1 structural, live region, and widget roles, proper nesting and focus management, plus requisite and optional supporting attributes to aid in development.

### HTML5 Outliner( recommendation for use )

The outline contains links to every section and heading. Clicking one of these links will jump to that section or heading, and cause a flicker effect on the relevant heading to highlight it.

This tool, along with the [accessibility bookmarklets](#), is useful for evaluating the accessibility of a website. Organizing information in sectioning elements and under headings helps to organize information both for sighted users and screen reader users.

▶ How to view the structural outline of a website with the HTML Outliner

#### 4. Compare semantic structure

A [landmark](#) containing content that is relevant to a specific, author-specified purpose and sufficiently important that users will likely want to be able to navigate to the section easily and to have it listed in a summary of the page. Such a page summary could be generated dynamically by a user agent or assistive technology.

Authors **SHOULD** limit use of the region role to sections containing content with a purpose that is not accurately described by one of the other [landmark](#) roles, such as [main](#), [complementary](#), or [navigation](#).

Authors **MUST** give each element with role region a brief label that describes the purpose of the content in the region. Authors **SHOULD** reference a visible label with [aria-labelledby](#) if a

visible label is present. Authors **SHOULD** include the label inside of a heading whenever possible. The heading **MAY** be an instance of the standard host language heading element or an instance of an element with role [heading](#).

[Assistive technologies](#) **SHOULD** enable users to quickly navigate to elements with role region. Mainstream [user agents](#) **MAY** enable users to quickly navigate to elements with role region.

[HTML Techniques](#) [ARIA Techniques](#)

Use the HTML `section` element to define a `region` landmark.

**HTML Example: Using `section[aria-labelledby]` attribute**  
Defines a label for each region using existing content on the page.

```
<main>

  <h1>title for main content area</h1>

  <section aria-labelledby="region1">

    <h2 id="region1">title for region area 1</h2>

    ... content for region area 1 ...

  </section>

  <section aria-labelledby="region2">

    <h2 id="region2">title for region area 2</h2>

    ... content for region area 2 ...

  </section>

</main>
```

[HTML Techniques](#) [ARIA Techniques](#)

A `role="region"` attribute is used to define a `region` landmark.

**ARIA Example: Using `role="region"` attribute**

```
<div role="main">

  <h1>title for main content area</h1>

  <div role="region" aria-labelledby="region1">
    <h2 id="region1">title for region area 1</h2>
    ... content for region area 1 ...
  </div>

  <div role="region" aria-labelledby="region2">
    <h2 id="region2">title for region area 2</h2>
    ... content for region area 2 ...
  </div>

</div>
```

For instance :



<https://www.w3.org/WAI/ARIA/apg/patterns/landmarks/examples/main.html>

The image shows a browser window displaying the W3C ARIA Landmarks Example page. The page is titled "ARIA Landmarks Example" and includes a sidebar with navigation links: Principles, HTML, Banner, Complementary, Contentinfo, Form, Main, Navigation, Region, Search, Asst. Tech., and Resources. The main content area is divided into sections: "Main Landmark", "Design Patterns", "HTML Example: One Main Landmark", and "HTML Example: Multiple Main Landmarks". The "Main Landmark" section explains that a main landmark identifies the primary content of the page and provides a list of design patterns. The "HTML Example: One Main Landmark" section shows a code snippet for a single main landmark. The "HTML Example: Multiple Main Landmarks" section shows a code snippet for multiple main landmarks. The browser's developer tools are open, showing the source code of the page. The code includes a header with a navigation menu and a main content area. The main content area is divided into sections: "Design Patterns", "HTML Example: One Main Landmark", and "HTML Example: Multiple Main Landmarks". The code for the "HTML Example: One Main Landmark" section is highlighted, showing the use of the `main` element to define a main landmark. The code for the "HTML Example: Multiple Main Landmarks" section is also highlighted, showing the use of the `main` element with a unique `aria-labelledby` attribute to define multiple main landmarks.

1. ARIA Landmarks Example

1. Untitled NAV

2. Main Landmark

1. Design Patterns

2. HTML Example: One Main Landmark

3. HTML Example: Multiple Main Landmarks

4. ARIA Example: One Main Landmark

5. ARIA Example: Multiple Main Landmarks

6. Landmarks

7. Related Documents

```
<h1>ARIA Landmarks Example</h1>
<div id="inst" class="inst"></div>
<button type="button" onclick="showLandmarks(event)" aria-describedby="inst">Hide Landmarks</button>
<button type="button" onclick="showHeadings(event)" aria-describedby="inst">Hide Headings</button>
</div>
</header>
<div class="row">
  <div class="col-xs-12 col-sm-3 col-md-3 col-lg-2">
    <nav id="h5o-2"></nav>
  </div>
  <div class="col-xs-12 col-sm-9 col-md-6 col-lg-7">
    <main></main>
  </div>
  <div class="col-xs-12 col-sm-12 col-md-3 col-lg-3">
    <aside aria-labelledby="id4"></aside>
    <aside aria-labelledby="id3"></aside>
  </div>
</div>
<div class="well well-sm"></div>
</div>
<script type="text/javascript" src=".../content-assets/wai-aria-practices/pat
terns/landmarks/examples/js/jquery-2.1.1.min.js"></script>
<script type="text/javascript" src=".../content-assets/wai-aria-practices/pat
terns/landmarks/examples/js/bootstrap.min.js"></script>
<script type="text/javascript" src=".../content-assets/wai-aria-practices/pat
terns/landmarks/examples/js/bootstrap-accessibility.min.js"></script>
<script type="text/javascript" src=".../content-assets/wai-aria-practices/pat
terns/landmarks/examples/js/bootstrap-accessibility-2.js"></script>
```

Comparison of the semantic structure that we have at the moment...