Web accessibility

CONCEPT and GUIDELINES

POLITECNICO
DI TORINO

Laura Farinetti - DAUIN

e-Lite

Summary

- Inclusive design and accessibility principles
- Web accessibility
- Screen readers
- Standards
  - WCAG
  - WAI-ARIA
- Legge Stanca

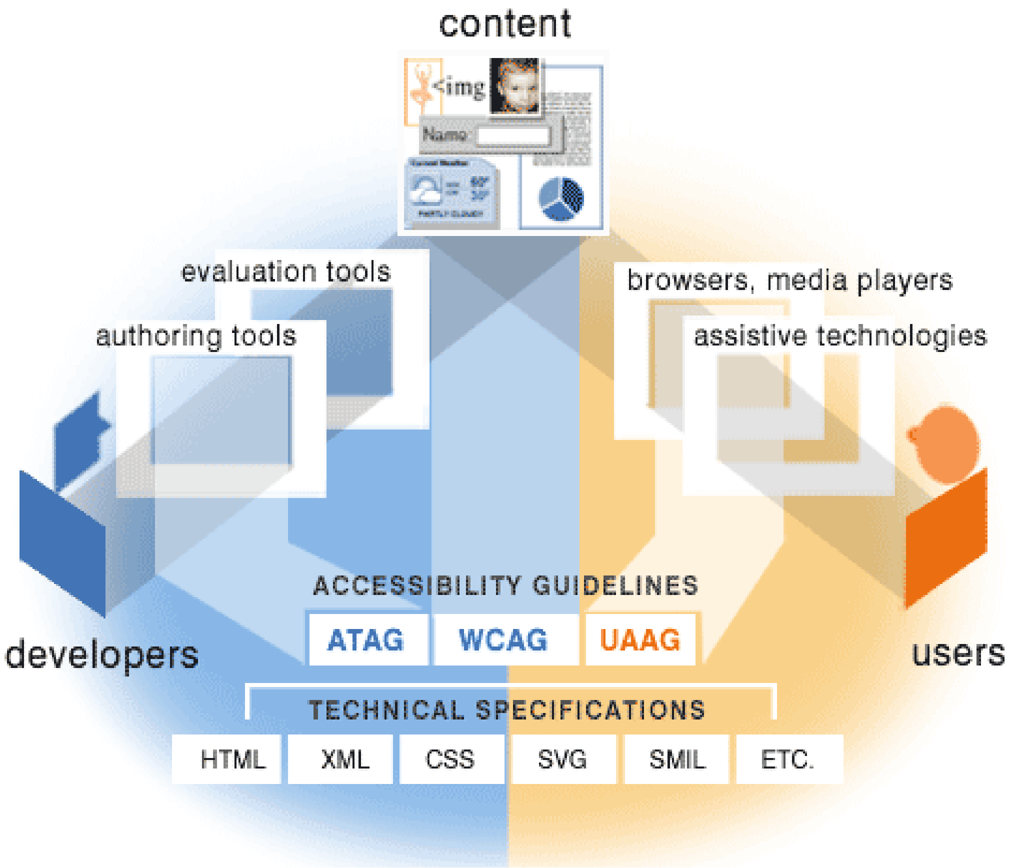
WEB ACCESSIBILITY STANDARDS

Web accessibility standards

- Web accessibility relies on several components that work together
  - Web content: refers to any part of a website, including text, images, forms, and multimedia, as well as any markup code, scripts, applications, and such
  - User agents: software that people use to access web content, including desktop graphical browsers, voice browsers, mobile phone browsers, multimedia players, plug-ins, and some assistive technologies
  - Authoring tools: software or services that people use to produce web content, including code editors, document conversion tools, content management systems, blogs, database scripts, and other tools

Web Accessibility Initiative (WAI)

- Develops strategies, guidelines, and resources to help make the Web accessible to people with disabilities



W3C® Web Accessibility initiative

Image by Michael Duffy, from: Essential Components of Web Accessibility. S.L. Henry, ed. Copyright W3C ® (MIT, ERCIM, Keio, Beihang). www.w3.org/WAI/intro/components.php

Web accessibility standards

- The W3C Web Accessibility Initiative (WAI) provides a set of guidelines that are internationally recognized as the standard for web accessibility
  - Web Content Accessibility Guidelines (WCAG)
  - User Agent Accessibility Guidelines (UAAG)
  - Authoring Tool Accessibility Guidelines (ATAG)

  - Accessible Rich Internet Applications (WAI-ARIA) includes dynamic content and advanced user interface controls developed with Ajax, JavaScript, and related web technologies

Web Content Accessibility Guidelines (WCAG)

- The WCAG documents explain how to make web content more accessible to people with disabilities
  - Web "content" generally refers to the information in a web page or web application, including: natural information such as text, images, and sounds code or markup that defines structure, presentation, etc.
- WCAG 1.0
  - W3C Rec May 5th, 1999
- WCAG 2.0
  - W3C Rec Dec 11th, 2008
- WCAG 2.1
  - W3C Rec Jun 5th, 2018

WCAG 2

- 4 principles

WCAG 2 guidelines
Perceivable ✓
Operable ✓
Abc
Understandable ✓
Robust ✓

- 12 guidelines
- Three levels: A, AA, and AAA

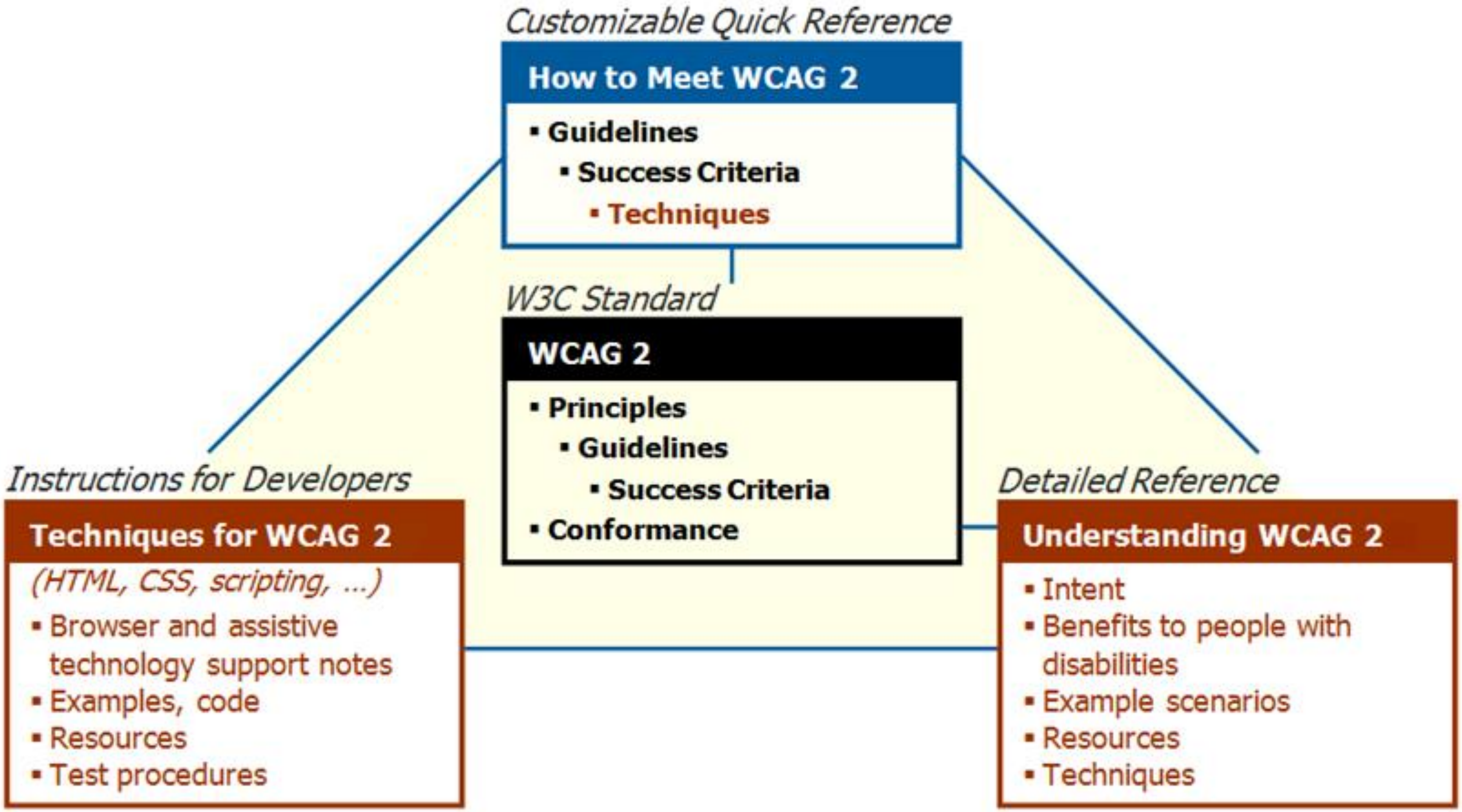
W3C WAI-A WCAG 2.1 W3C WAI-AA WCAG 2.1 W3C WAI-AAA WCAG 2.1

- Testable success criteria

WCAG 2 supporting technical material

- **How to Meet WCAG 2**: quick reference to Web Content Accessibility Guidelines 2.0 requirements (success criteria) and techniques, essentially the WCAG 2.0 checklist
  - https://www.w3.org/WAI/WCAG21/quickref/
- **Techniques for WCAG 2**: details on how to develop accessible Web content, such as HTML code examples. The techniques are "informative", that is, you do not have to use them. The basis for determining conformance to WCAG 2.0 is the success criteria from the WCAG 2.0 standard, not the techniques
  - https://www.w3.org/WAI/WCAG21/Techniques/
- **Understanding WCAG 2**: additional guidance on learning and implementing WCAG 2.0 for people who want to understand the guidelines and success criteria more thoroughly
  - https://www.w3.org/WAI/WCAG21/Understanding/

The WCAG 2 document family



Customizable Quick Reference

How to Meet WCAG 2

- **Guidelines**
  - **Success Criteria**
    - **Techniques**

W3C Standard

WCAG 2

- **Principles**
  - **Guidelines**
    - **Success Criteria**
- **Conformance**

Instructions for Developers

Techniques for WCAG 2

(HTML, CSS, scripting, ...)

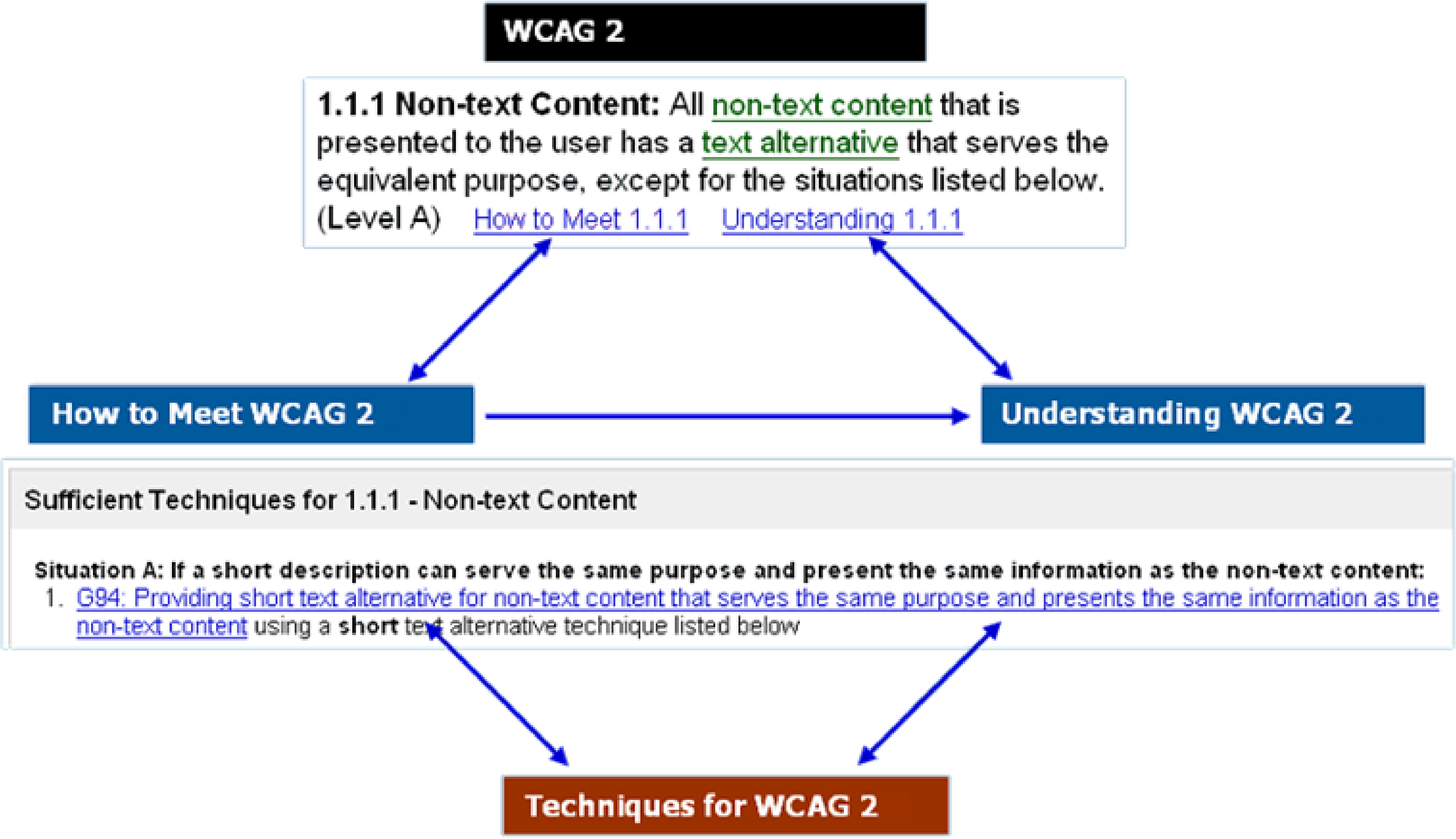
- Browser and assistive technology support notes
- Examples, code
- Resources
- Test procedures

Detailed Reference

Understanding WCAG 2

- Intent
- Benefits to people with disabilities
- Example scenarios
- Resources
- Techniques

Example



1.1.1 Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A) How to Meet 1.1.1 Understanding 1.1.1

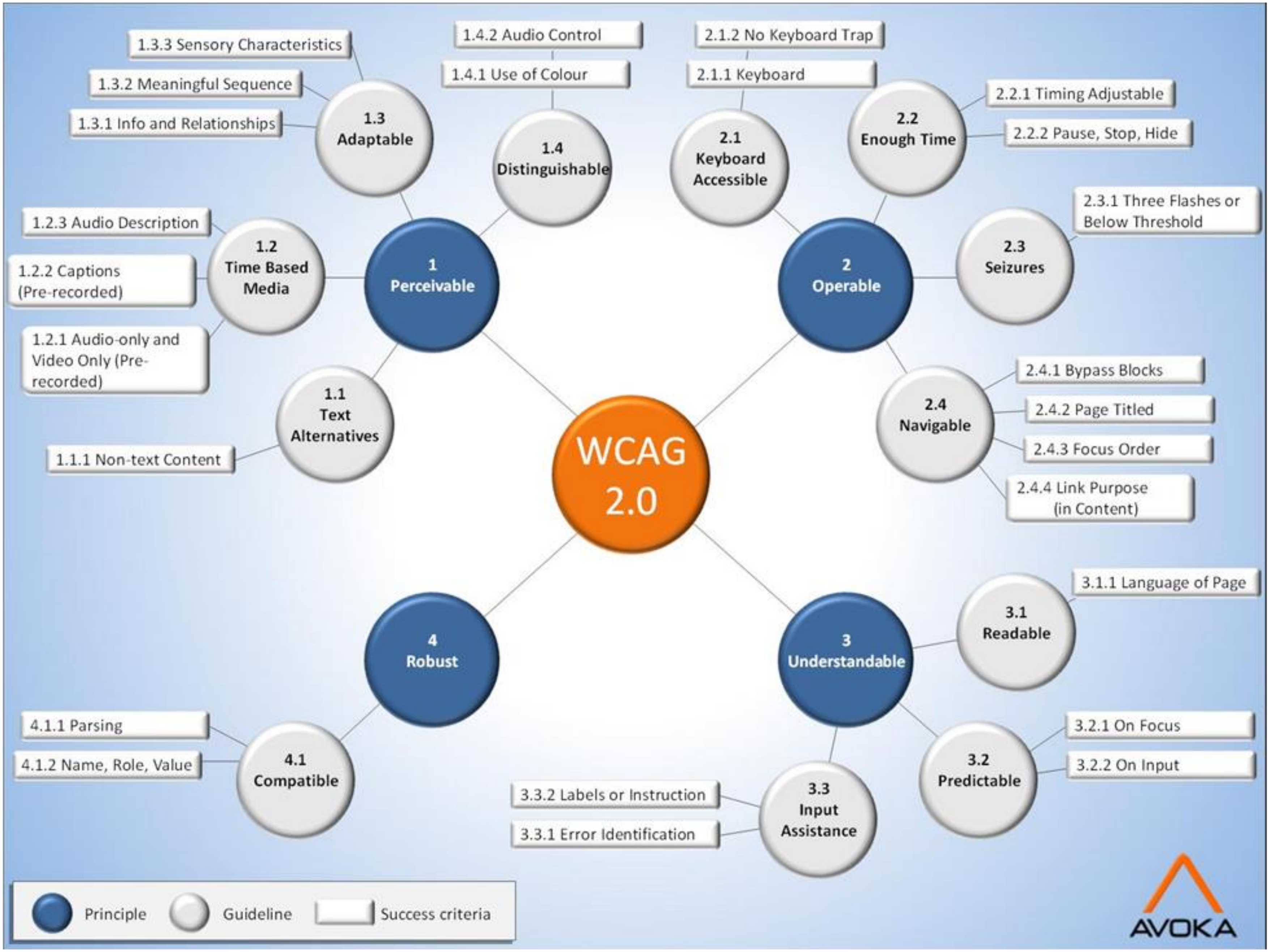
Sufficient Techniques for 1.1.1 - Non-text Content

Situation A: If a short description can serve the same purpose and present the same information as the non-text content:

1. G94: Providing short text alternative for non-text content that serves the same purpose and presents the same information as the non-text content using a **short** text alternative technique listed below

WCAG 2 principles and guidelines

- At the top are four principles (POUR) that provide the foundation for Web accessibility
- Guidelines are under the principles
- Perceivable
  - Provide text alternatives for non-text content
  - Provide captions and other alternatives for multimedia
  - Create content that can be presented in different ways, including by assistive technologies, without losing meaning
  - Make it easier for users to see and hear content
- Operable
  - Make all functionality available from a keyboard
  - Give users enough time to read and use content
  - Do not use content that causes seizures
  - Help users navigate and find content
- Understandable
  - Make text readable and understandable
  - Make content appear and operate in predictable ways
  - Help users avoid and correct mistakes
- Robust
  - Maximize compatibility with current and future user tools



WCAG 2.0

1 Perceivable

1.1 Text Alternatives

1.1.1 Non-text Content

1.2 Time Based Media

1.2.1 Audio-only and Video Only (Pre-recorded)

1.2.2 Captions (Pre-recorded)

1.2.3 Audio Description

1.3 Adaptable

1.3.1 Info and Relationships

1.3.2 Meaningful Sequence

1.3.3 Sensory Characteristics

1.4 Distinguishable

1.4.1 Use of Colour

1.4.2 Audio Control

2 Operable

2.1 Keyboard Accessible

2.1.1 Keyboard

2.1.2 No Keyboard Trap

2.2 Enough Time

2.2.1 Timing Adjustable

2.2.2 Pause, Stop, Hide

2.3 Seizures

2.3.1 Three Flashes or Below Threshold

2.4 Navigable

2.4.1 Bypass Blocks

2.4.2 Page Titled

2.4.3 Focus Order

2.4.4 Link Purpose (in Content)

3 Understandable

3.1 Readable

3.1.1 Language of Page

3.2 Predictable

3.2.1 On Focus

3.2.2 On Input

3.3 Input Assistance

3.3.1 Error Identification

3.3.2 Labels or Instruction

4 Robust

4.1 Compatible

4.1.1 Parsing

4.1.2 Name, Role, Value

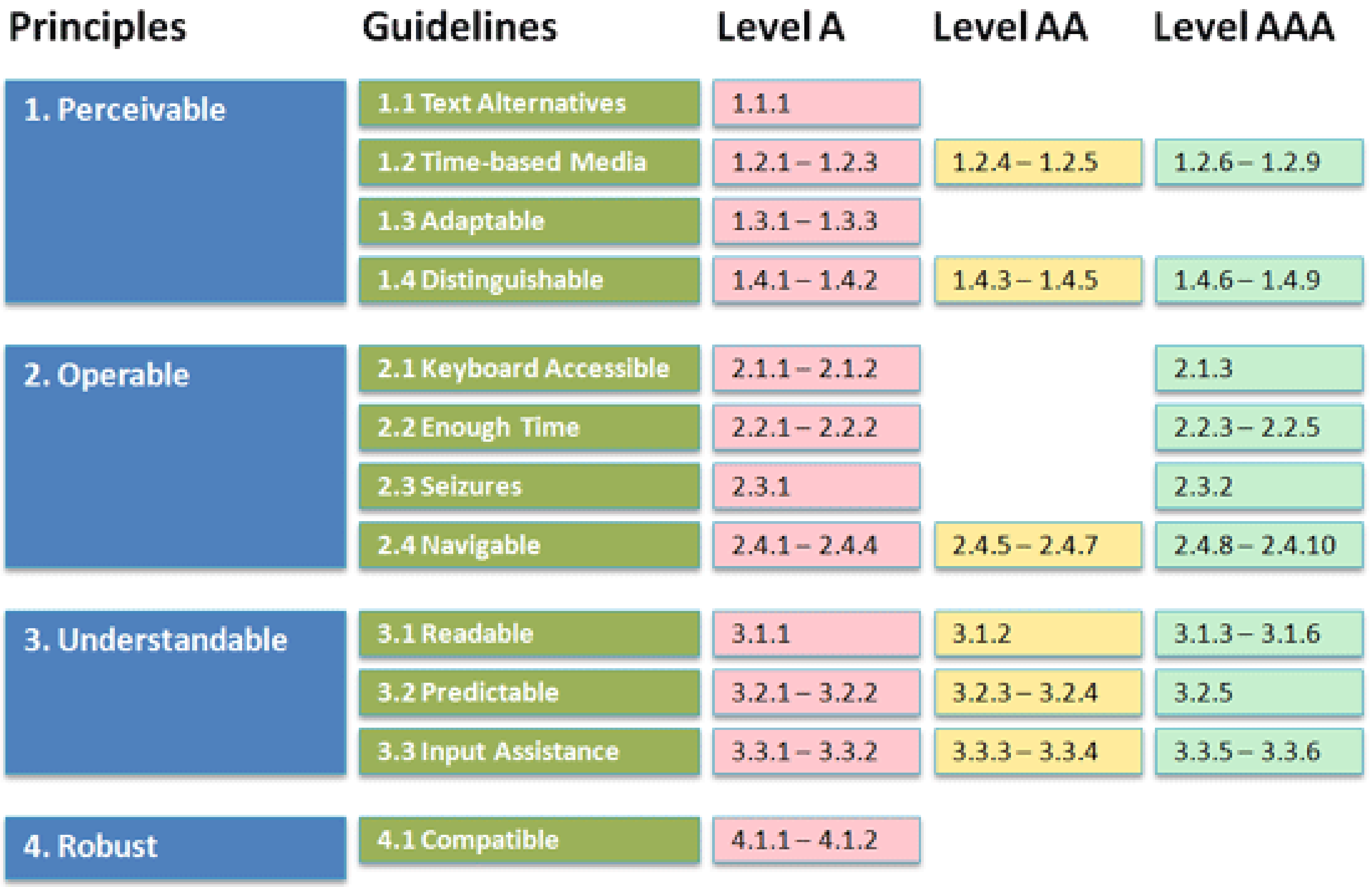
● Principle ○ Guideline □ Success criteria

AVOKA

WCAG 2 success criteria

- For each guideline, testable success criteria are provided to allow WCAG 2 to be used
- In order to meet the needs of different groups and different situations, three levels of conformance are defined
  - A (lowest)
  - AA
  - AAA (highest)

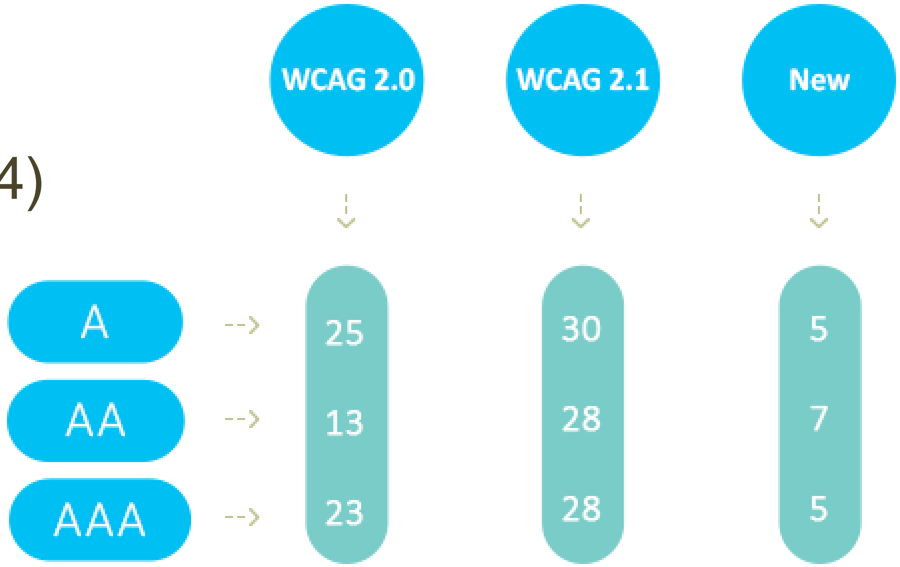
WCAG 2.0 success criteria



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		

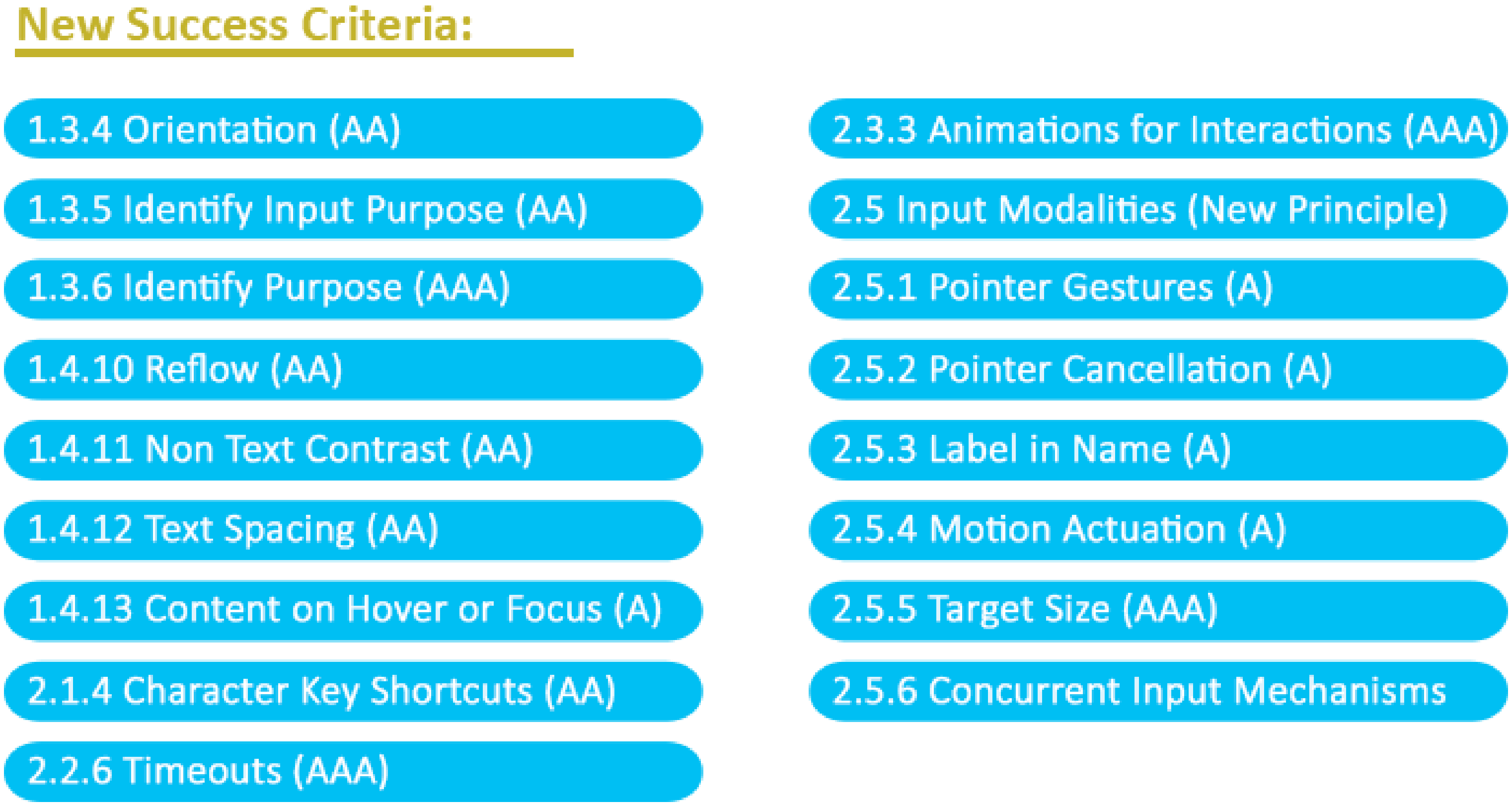
WCAG 2.1 success criteria

- All success criteria from 2.0 are included in 2.1
- The 2.0 success criteria are exactly the same (verbatim, word-for-word) in 2.1
- 17 additional success criteria
  - mobile accessibility (8)
  - people with low vision (5)
  - people with cognitive and learning disabilities (4)



	WCAG 2.0	WCAG 2.1	New
A	25	30	5
AA	13	28	7
AAA	23	28	5

WCAG 2.1 new success criteria



New Success Criteria:

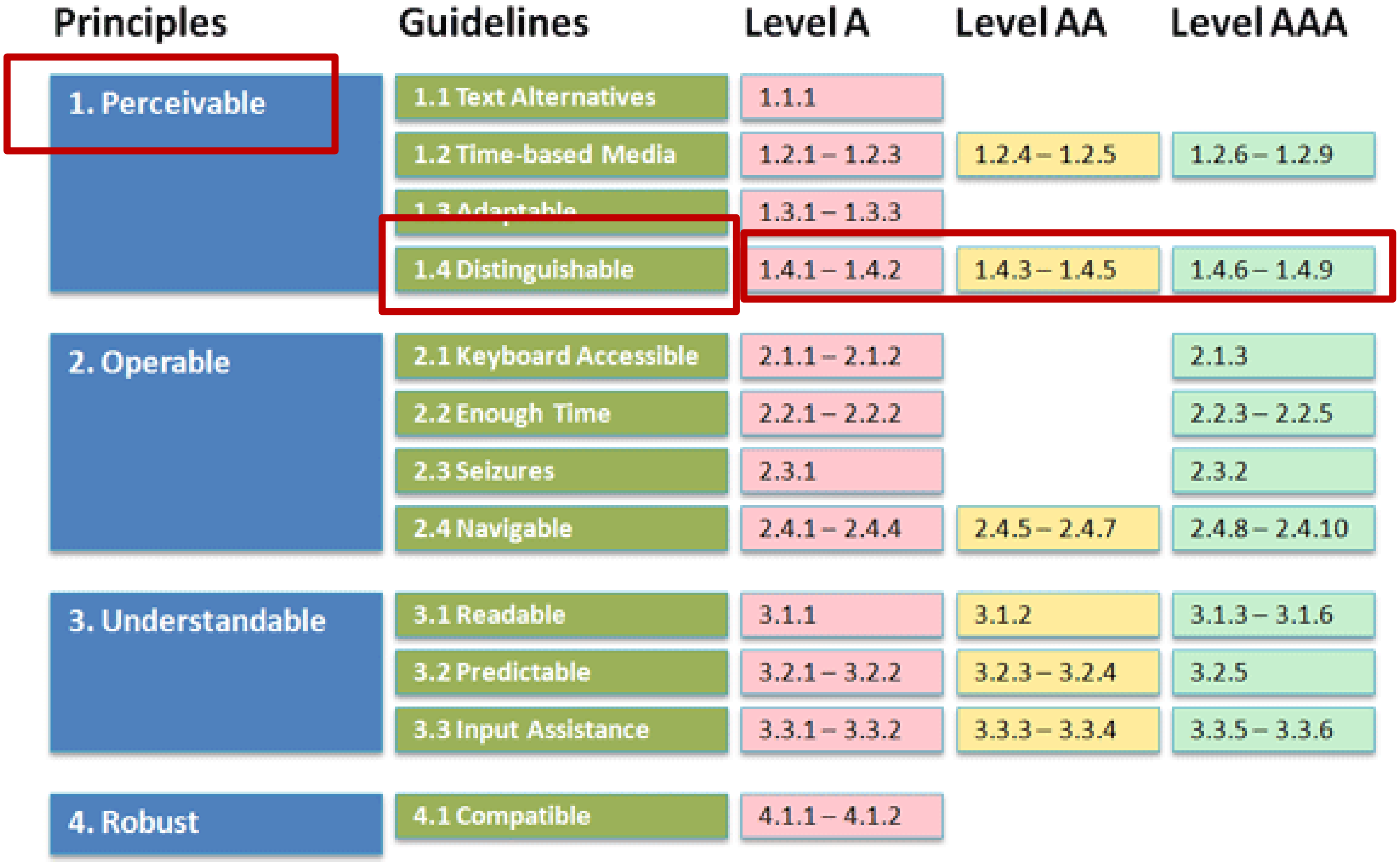
- 1.3.4 Orientation (AA)
- 1.3.5 Identify Input Purpose (AA)
- 1.3.6 Identify Purpose (AAA)
- 1.4.10 Reflow (AA)
- 1.4.11 Non Text Contrast (AA)
- 1.4.12 Text Spacing (AA)
- 1.4.13 Content on Hover or Focus (A)
- 2.1.4 Character Key Shortcuts (AA)
- 2.2.6 Timeouts (AAA)
- 2.3.3 Animations for Interactions (AAA)
- 2.5 Input Modalities (New Principle)
- 2.5.1 Pointer Gestures (A)
- 2.5.2 Pointer Cancellation (A)
- 2.5.3 Label in Name (A)
- 2.5.4 Motion Actuation (A)
- 2.5.5 Target Size (AAA)
- 2.5.6 Concurrent Input Mechanisms

Other WCAG 2 sections

- Sufficient and Advisory Techniques
  - An informative list of typical mistakes and good-practice techniques is provided
  - Techniques fall into two categories: those that are sufficient for meeting the success criteria and those that are advisory (optional)
- Common Failures
  - Describe authoring practices known to cause Web content not to conform to WCAG 2

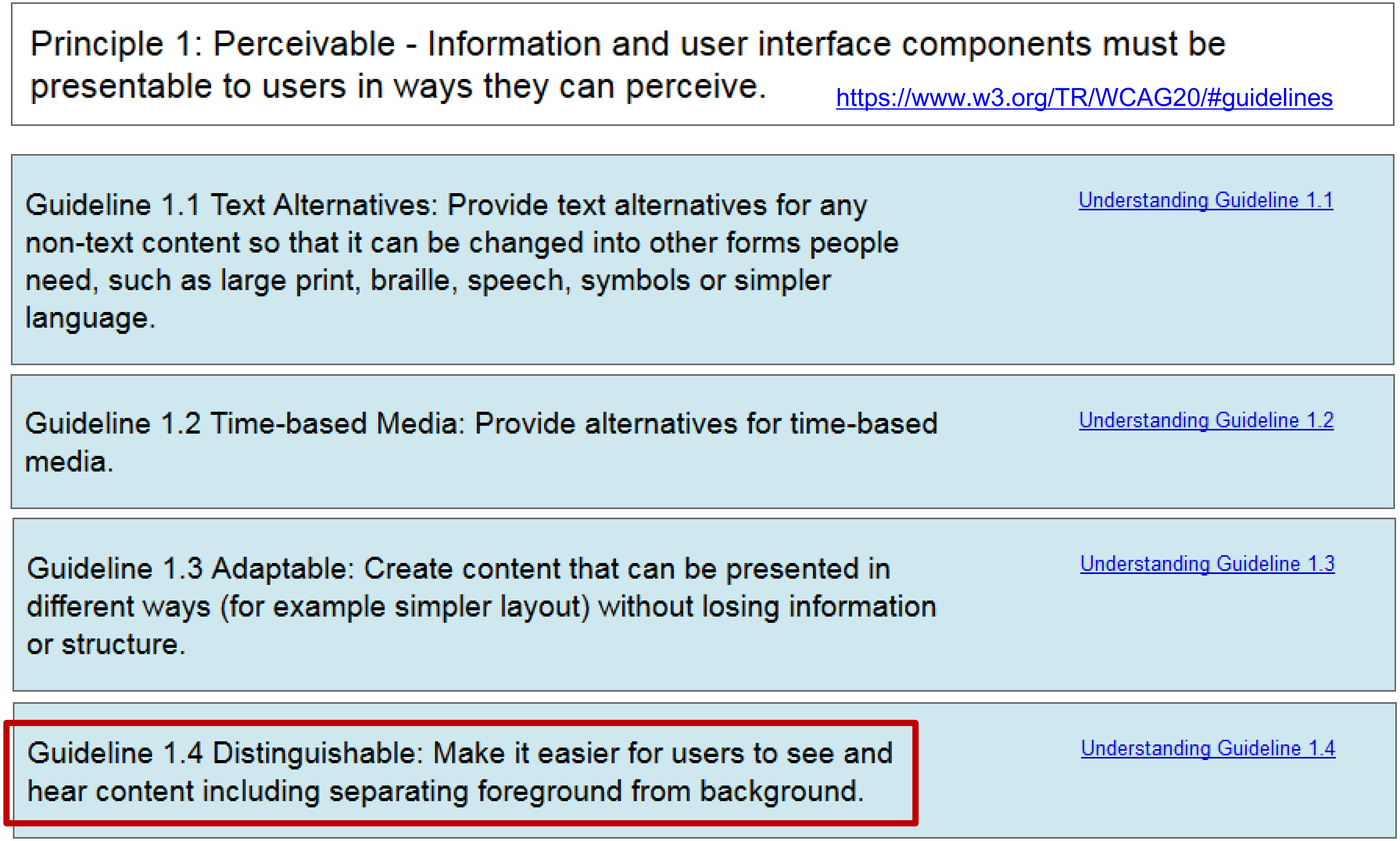
  - https://www.w3.org/WAI/WCAG21/Techniques/

Example



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		

Example: 1. Perceivable



Principle 1: Perceivable - Information and user interface components must be presentable to users in ways they can perceive. https://www.w3.org/TR/WCAG20/#guidelines

Guideline 1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

Understanding Guideline 1.1

Guideline 1.2 Time-based Media: Provide alternatives for time-based media.

Understanding Guideline 1.2

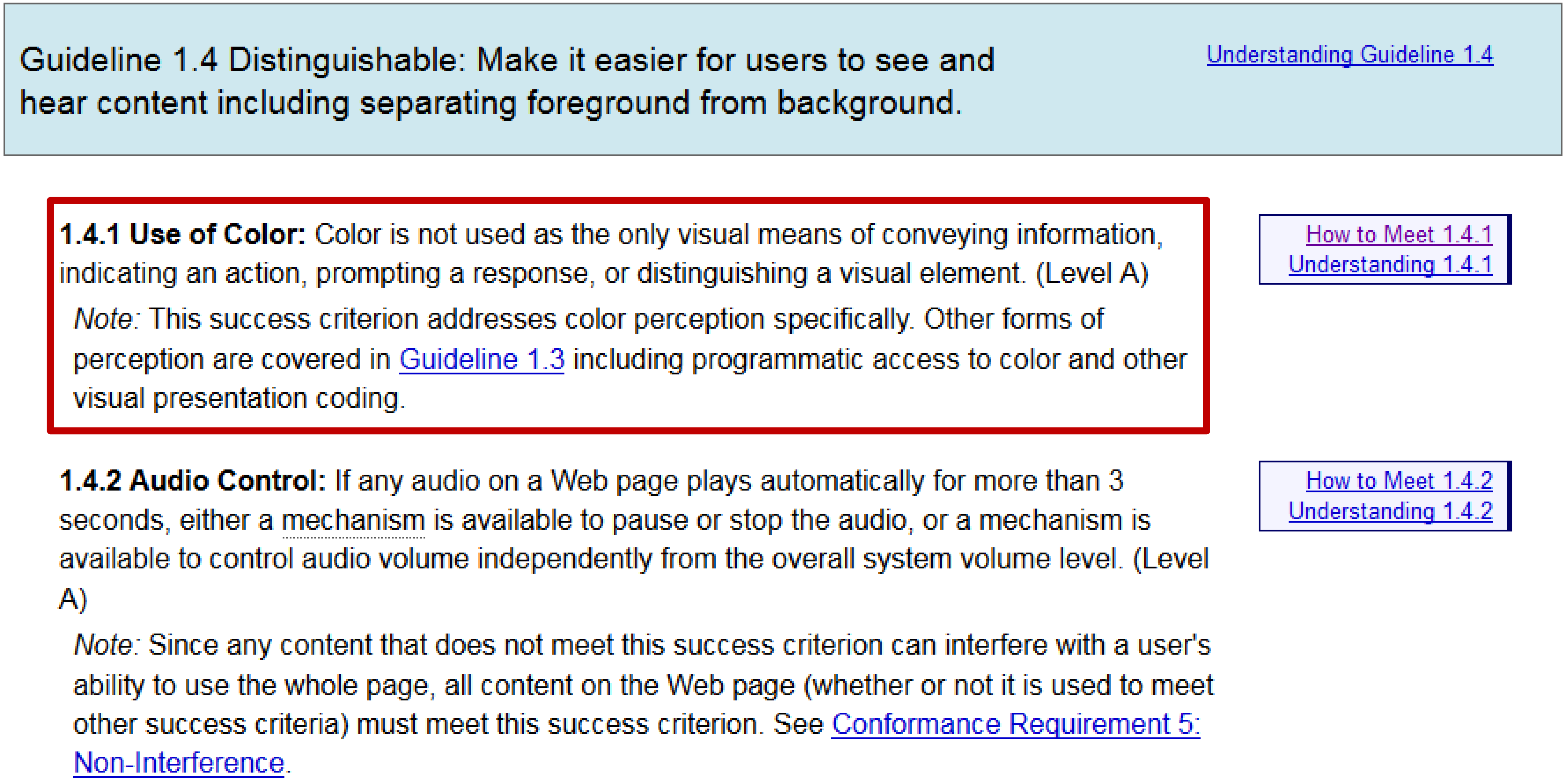
Guideline 1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

Understanding Guideline 1.3

Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Understanding Guideline 1.4

Example: 1.4 Distinguishable



Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

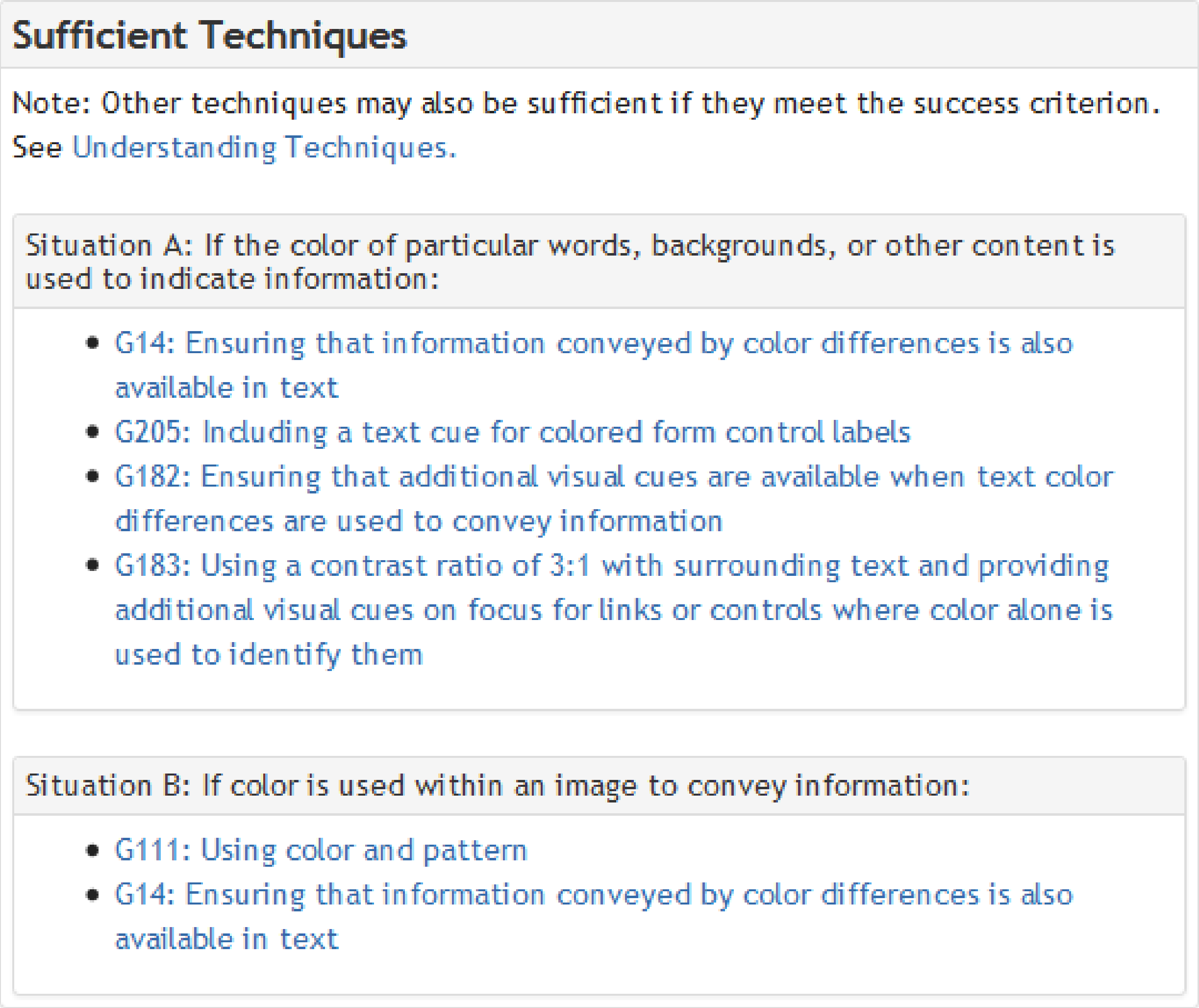
1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)

Note: This success criterion addresses color perception specifically. Other forms of perception are covered in Guideline 1.3 including programmatic access to color and other visual presentation coding.

1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A)

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether or not it is used to meet other success criteria) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

Example: 1.4.1 Use of Colors



Sufficient Techniques

Note: Other techniques may also be sufficient if they meet the success criterion. See Understanding Techniques.

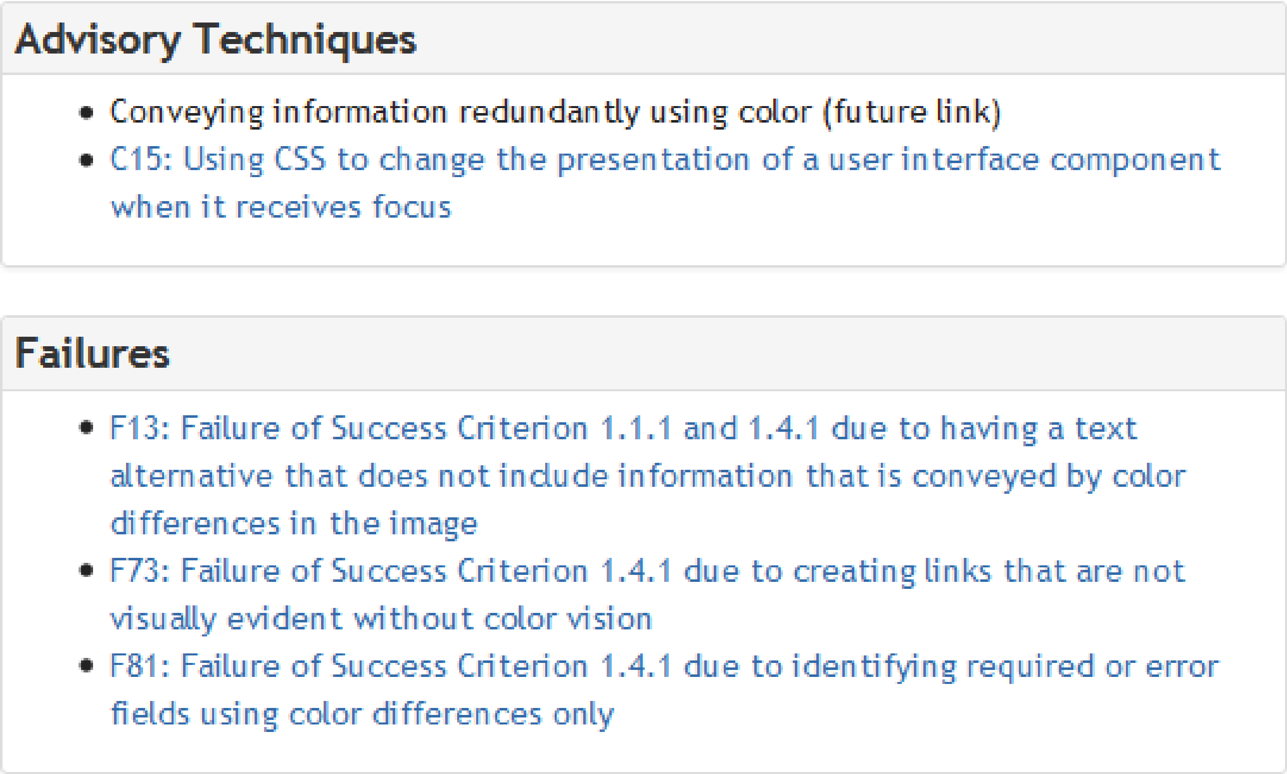
Situation A: If the color of particular words, backgrounds, or other content is used to indicate information:

- G14: Ensuring that information conveyed by color differences is also available in text
- G205: Including a text cue for colored form control labels
- G182: Ensuring that additional visual cues are available when text color differences are used to convey information
- G183: Using a contrast ratio of 3:1 with surrounding text and providing additional visual cues on focus for links or controls where color alone is used to identify them

Situation B: If color is used within an image to convey information:

- G111: Using color and pattern
- G14: Ensuring that information conveyed by color differences is also available in text

Example: 1.4.1 Use of Colors



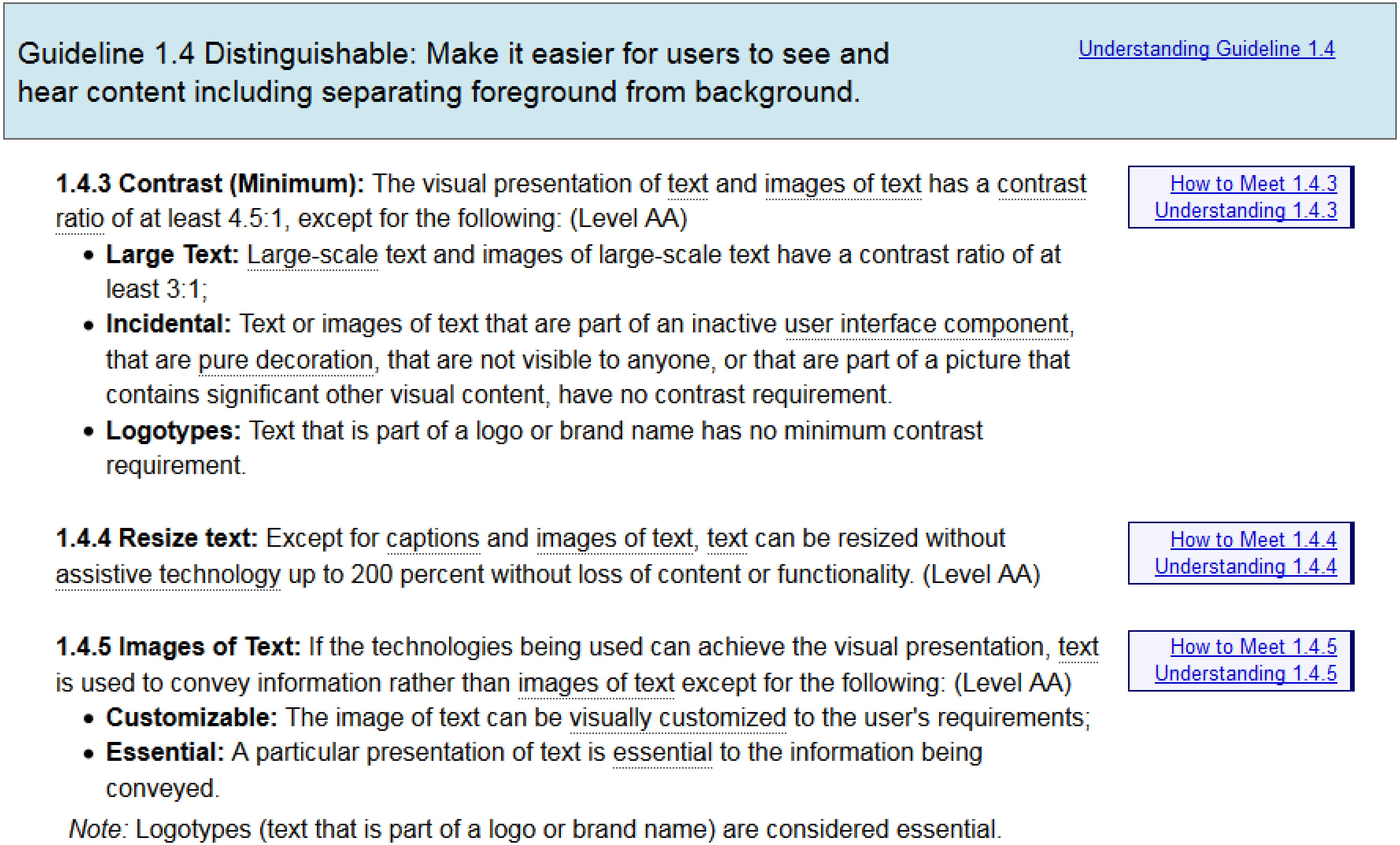
Advisory Techniques

- Conveying information redundantly using color (future link)
- C15: Using CSS to change the presentation of a user interface component when it receives focus

Failures

- F13: Failure of Success Criterion 1.1.1 and 1.4.1 due to having a text alternative that does not indude information that is conveyed by color differences in the image
- F73: Failure of Success Criterion 1.4.1 due to creating links that are not visually evident without color vision
- F81: Failure of Success Criterion 1.4.1 due to identifying required or error fields using color differences only

Example



Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Understanding Guideline 1.4

1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA)

- **Large Text:** Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;
- **Incidental:** Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
- **Logotypes:** Text that is part of a logo or brand name has no minimum contrast requirement.

How to Meet 1.4.3
Understanding 1.4.3

1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA)

How to Meet 1.4.4
Understanding 1.4.4

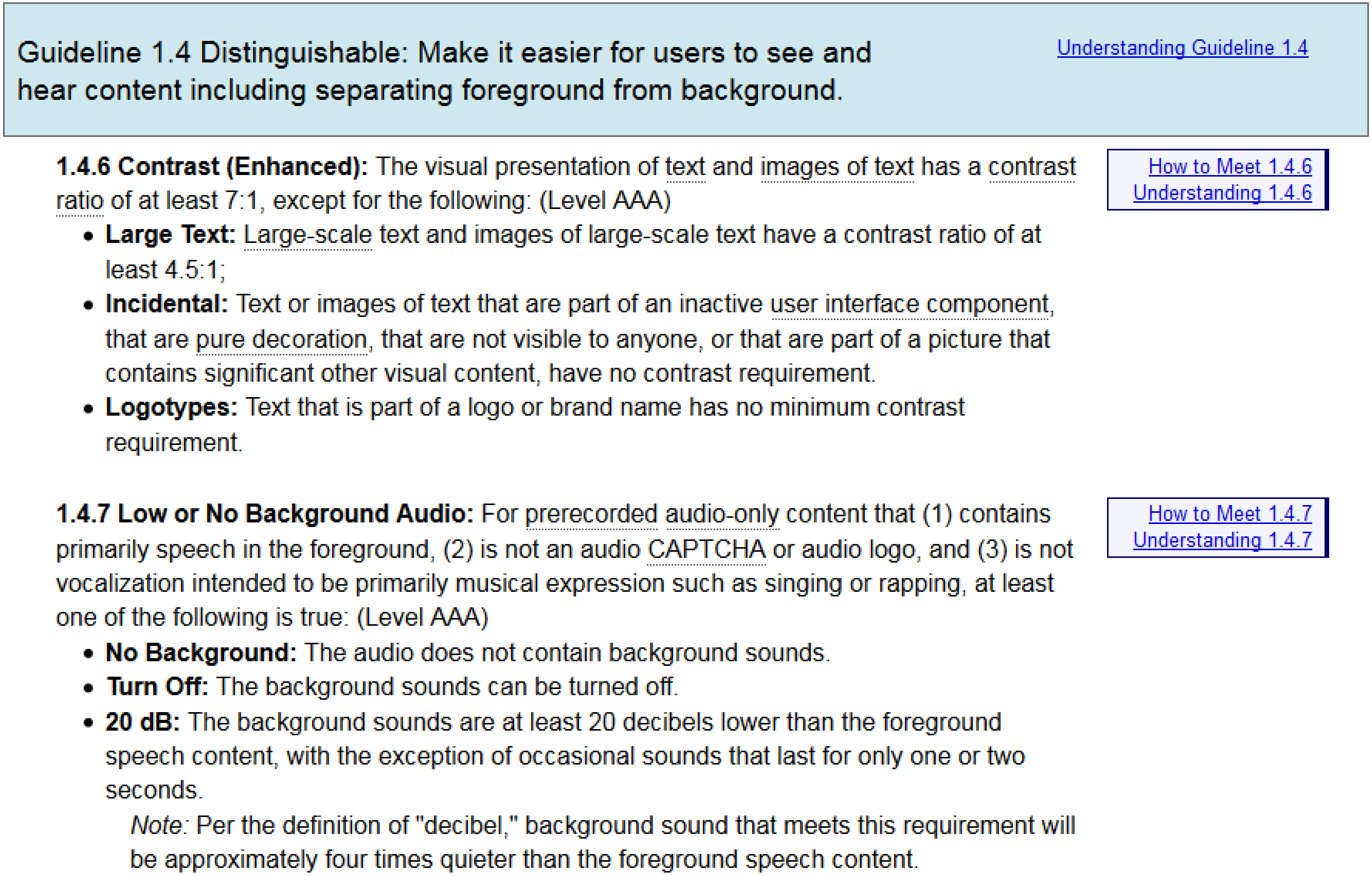
1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA)

- **Customizable:** The image of text can be visually customized to the user's requirements;
- **Essential:** A particular presentation of text is essential to the information being conveyed.

Note: Logotypes (text that is part of a logo or brand name) are considered essential.

How to Meet 1.4.5
Understanding 1.4.5

Example



Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Understanding Guideline 1.4

1.4.6 Contrast (Enhanced): The visual presentation of text and images of text has a contrast ratio of at least 7:1, except for the following: (Level AAA)

- **Large Text:** Large-scale text and images of large-scale text have a contrast ratio of at least 4.5:1;
- **Incidental:** Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
- **Logotypes:** Text that is part of a logo or brand name has no minimum contrast requirement.

How to Meet 1.4.6
Understanding 1.4.6

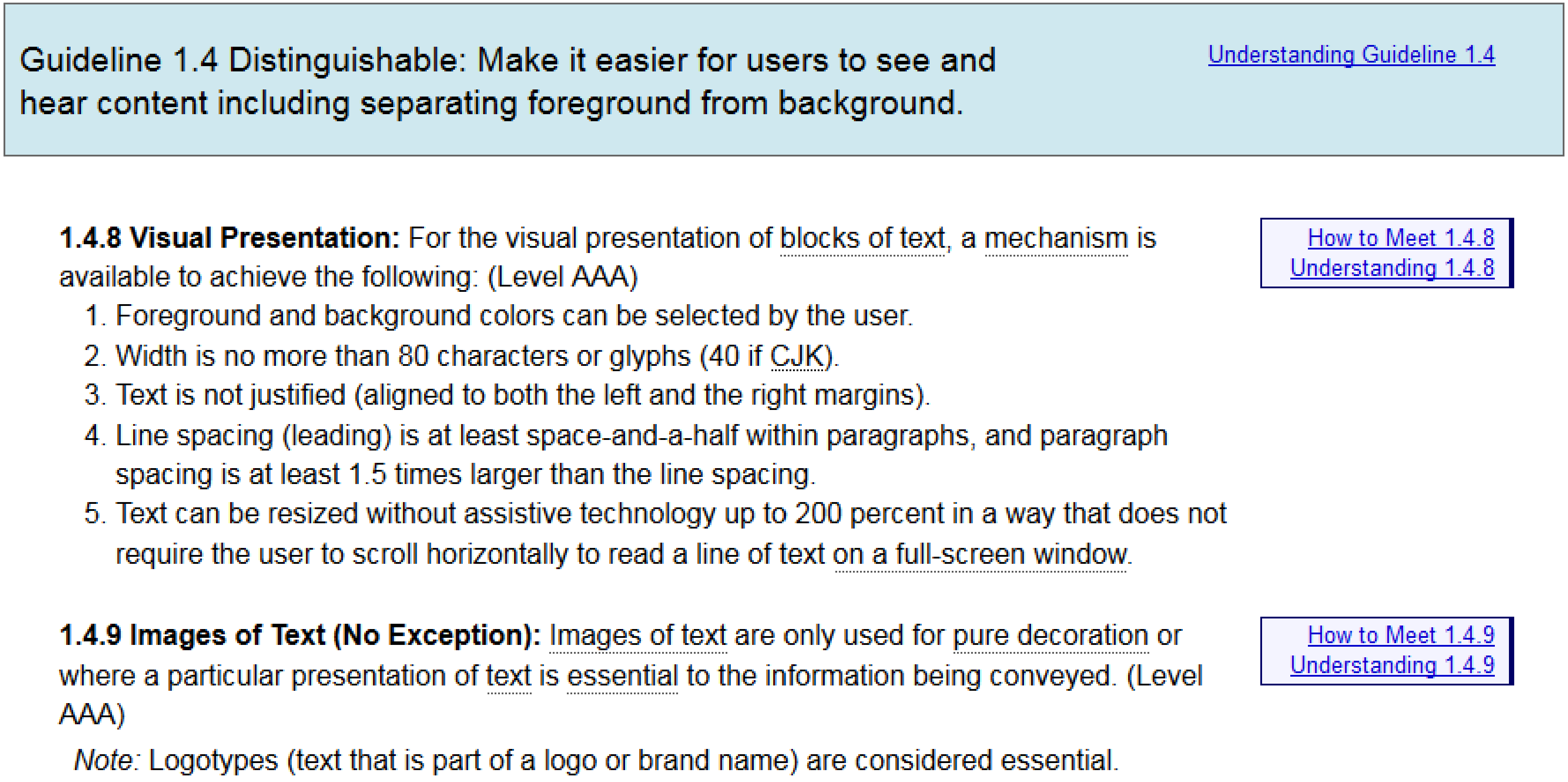
1.4.7 Low or No Background Audio: For prerecorded audio-only content that (1) contains primarily speech in the foreground, (2) is not an audio CAPTCHA or audio logo, and (3) is not vocalization intended to be primarily musical expression such as singing or rapping, at least one of the following is true: (Level AAA)

- **No Background:** The audio does not contain background sounds.
- **Turn Off:** The background sounds can be turned off.
- **20 dB:** The background sounds are at least 20 decibels lower than the foreground speech content, with the exception of occasional sounds that last for only one or two seconds.

  Note: Per the definition of "decibel," background sound that meets this requirement will be approximately four times quieter than the foreground speech content.

How to Meet 1.4.7
Understanding 1.4.7

Example



Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Understanding Guideline 1.4

1.4.8 Visual Presentation: For the visual presentation of blocks of text, a mechanism is available to achieve the following: (Level AAA)

1. Foreground and background colors can be selected by the user.
2. Width is no more than 80 characters or glyphs (40 if CJK).
3. Text is not justified (aligned to both the left and the right margins).
4. Line spacing (leading) is at least space-and-a-half within paragraphs, and paragraph spacing is at least 1.5 times larger than the line spacing.
5. Text can be resized without assistive technology up to 200 percent in a way that does not require the user to scroll horizontally to read a line of text on a full-screen window.

How to Meet 1.4.8
Understanding 1.4.8

1.4.9 Images of Text (No Exception): Images of text are only used for pure decoration or where a particular presentation of text is essential to the information being conveyed. (Level AAA)

Note: Logotypes (text that is part of a logo or brand name) are considered essential.

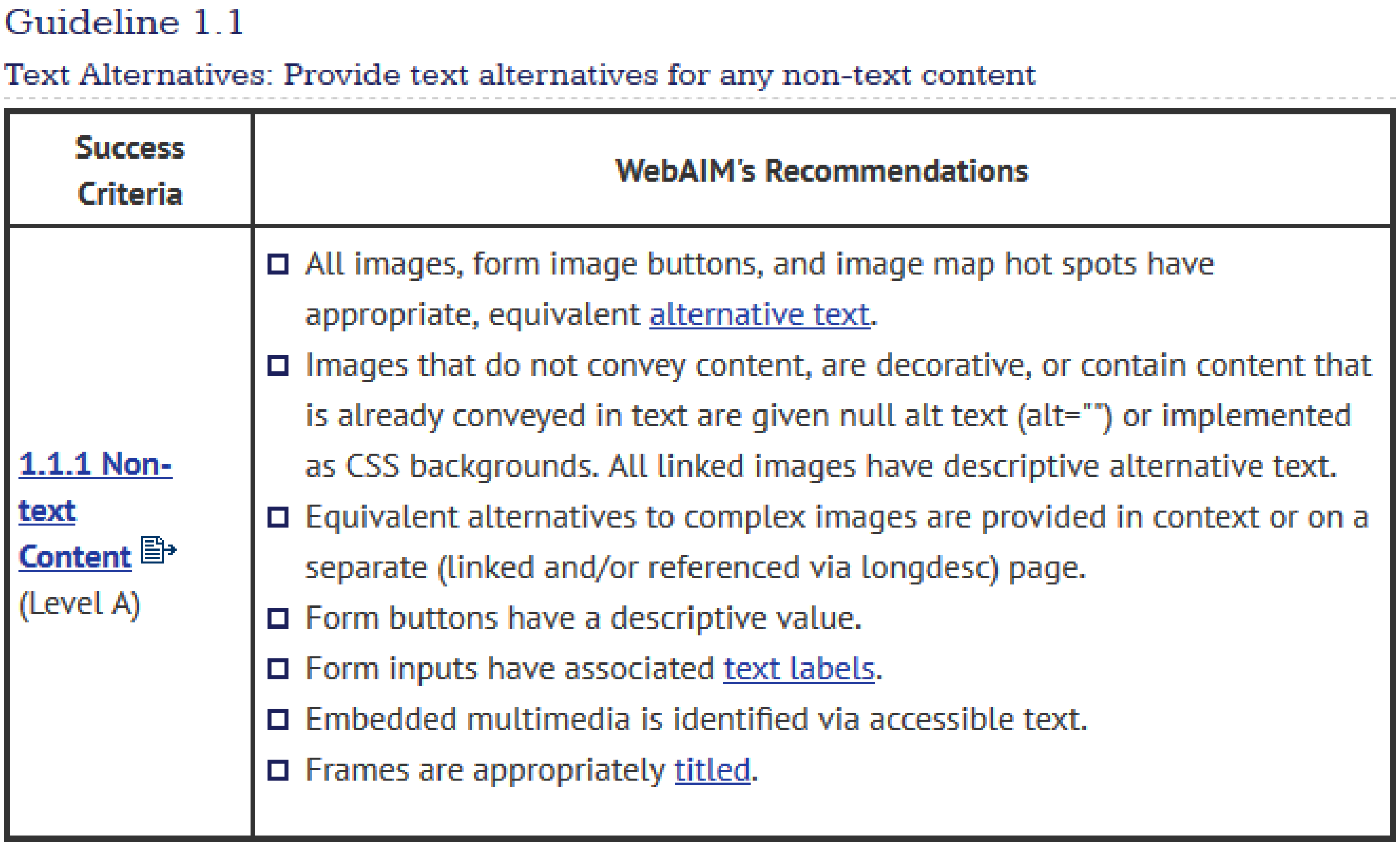
How to Meet 1.4.9
Understanding 1.4.9

WebAIM's WCAG 2 checklist

- A simple checklist that presents WebAIM recommendations for implementing HTML-related principles and techniques for those seeking WCAG 2.0 conformance
  - WCAG 2.0 covers accessibility of all web content and is not technology specific
  - The checklist has been targeted primarily for evaluation of HTML content
  - contains WebAIM's interpretation of WCAG guidelines and success criteria and our own recommended techniques for satisfying those success criteria
  - https://webaim.org/standards/wcag/checklist

WebAIM's WCAG 2 checklist

- Example: perceivable



Guideline 1.1

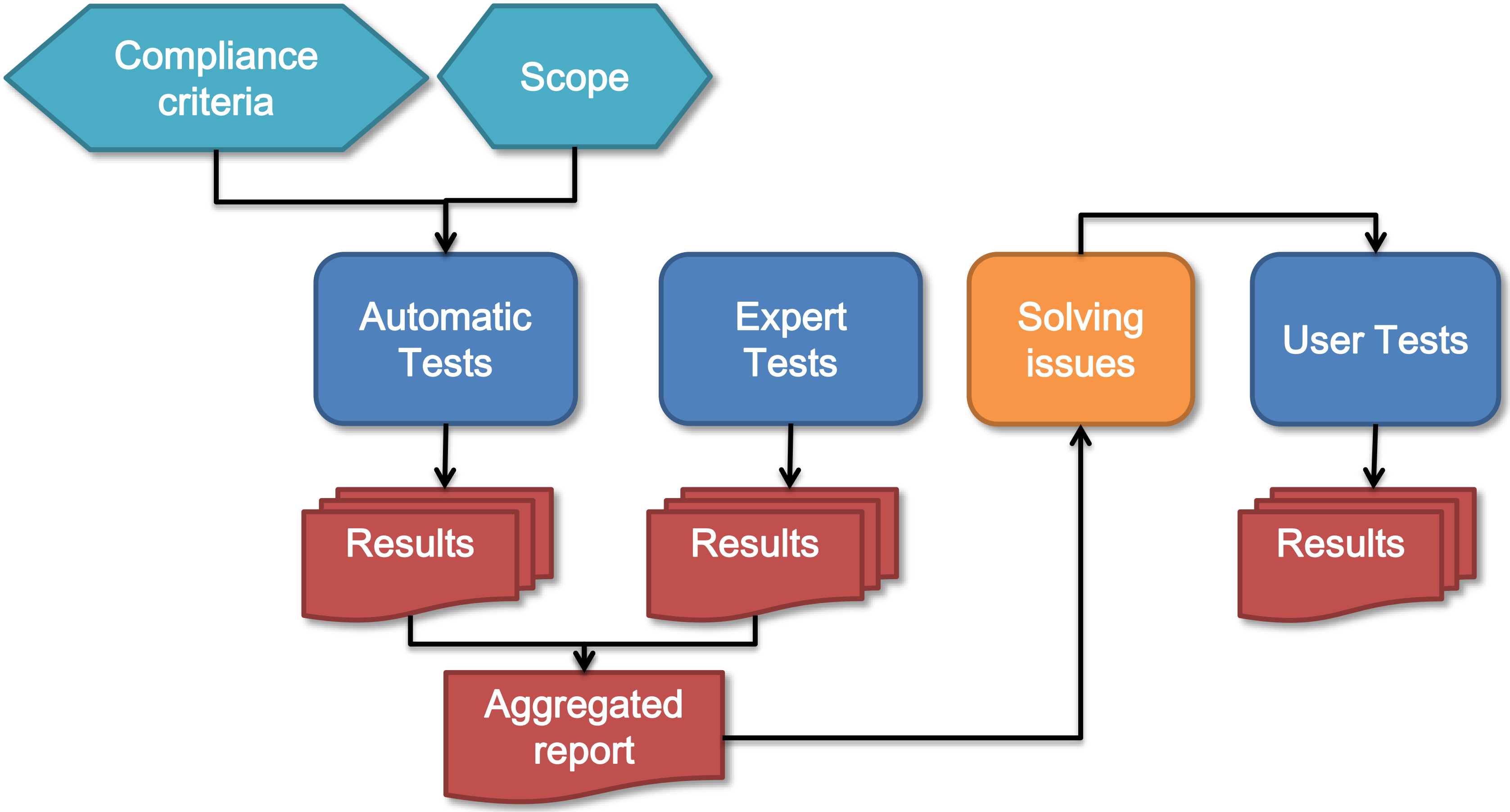
Text Alternatives: Provide text alternatives for any non-text content

Success Criteria	WebAIM's Recommendations
1.1.1 Non-text Content (Level A)	□ All images, form image buttons, and image map hot spots have appropriate, equivalent alternative text. □ Images that do not convey content, are decorative, or contain content that is already conveyed in text are given null alt text (alt="") or implemented as CSS backgrounds. All linked images have descriptive alternative text. □ Equivalent alternatives to complex images are provided in context or on a separate (linked and/or referenced via longdesc) page. □ Form buttons have a descriptive value. □ Form inputs have associated text labels. □ Embedded multimedia is identified via accessible text. □ Frames are appropriately titled.

Evaluation tools

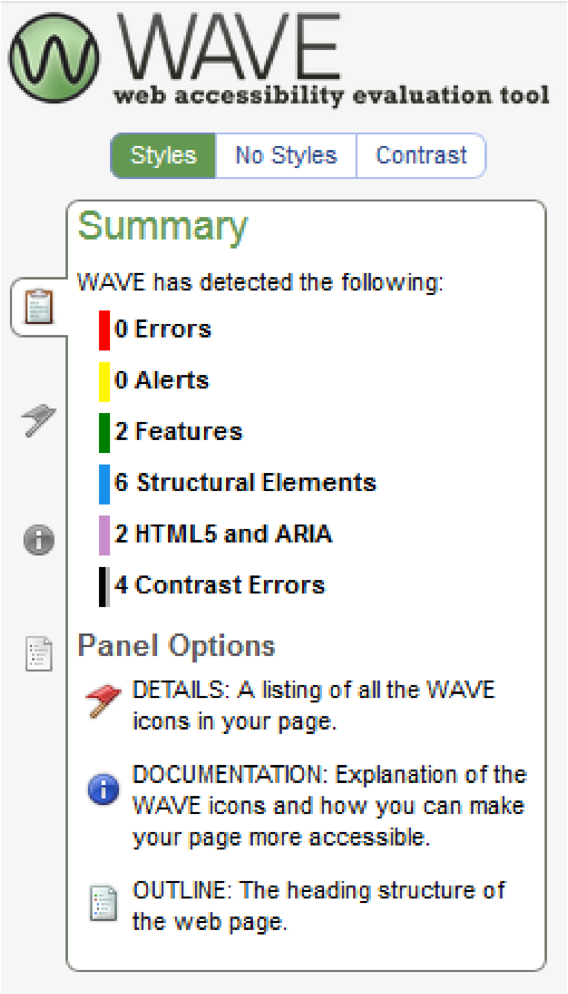
- While Web accessibility evaluation tools can significantly reduce the time and effort to evaluate Web sites, no tool can automatically determine the accessibility of Web sites
- W3C does not endorse specific vendor products
- Overview of Web accessibility evaluation tools
  - https://www.w3.org/WAI/ER/tools/index.html

Typical evaluation process



WAVE evaluation tool

- The WAVE Chrome and Firefox extensions allow to evaluate web content for accessibility issues directly within Chrome and Firefox browsers
  - https://wave.webaim.org/help
- Developed by WebAIM
  - https://addons.mozilla.org/en-US/firefox/addon/wave-accessibility-tool/
  - https://chrome.google.com/webstore/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh



Summary

WAVE has detected the following:

- 0 Errors
- 0 Alerts
- 2 Features
- 6 Structural Elements
- 2 HTML5 and ARIA
- 4 Contrast Errors

Panel Options

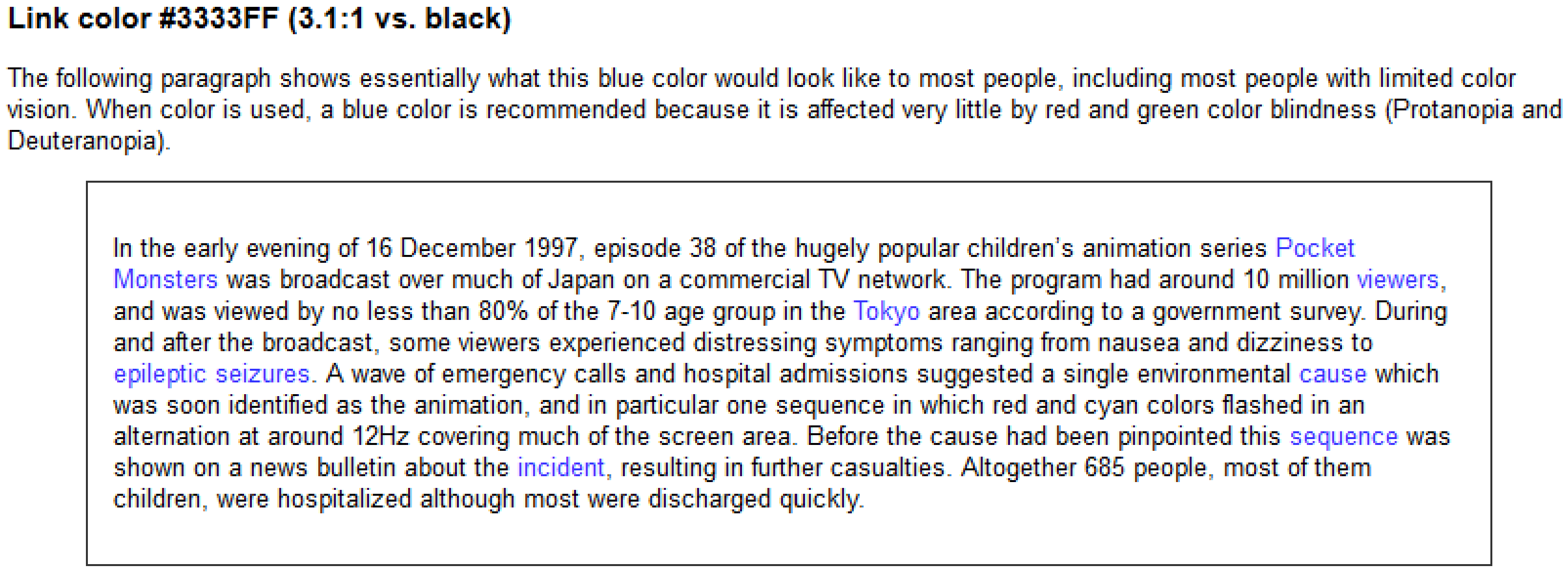
DETAILS: A listing of all the WAVE icons in your page.

DOCUMENTATION: Explanation of the WAVE icons and how you can make your page more accessible.

OUTLINE: The heading structure of the web page.

Color contrast

- A contrast ratio of 3:1 is the minimum level recommended by ISO and ANSI standards (ISO-9241-3, ANSI-HFES-100-1988) for standard text and vision
- The 4.5:1 ratio is used to account for the loss in contrast that results from moderately low visual acuity, congenital or acquired color deficiencies, or the loss of contrast sensitivity that typically accompanies aging
  - See https://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/working-examples/G183/link-contrast.html



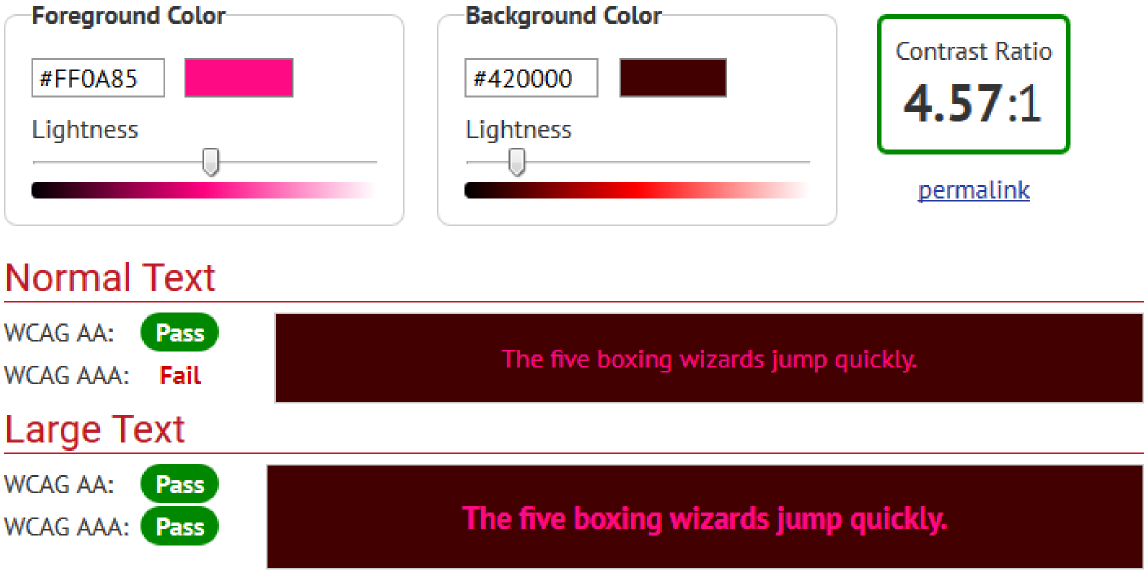
Link color #3333FF (3.1:1 vs. black)

The following paragraph shows essentially what this blue color would look like to most people, including most people with limited color vision. When color is used, a blue color is recommended because it is affected very little by red and green color blindness (Protanopia and Deuteranopia).

In the early evening of 16 December 1997, episode 38 of the hugely popular children's animation series Pocket Monsters was broadcast over much of Japan on a commercial TV network. The program had around 10 million viewers, and was viewed by no less than 80% of the 7-10 age group in the Tokyo area according to a government survey. During and after the broadcast, some viewers experienced distressing symptoms ranging from nausea and dizziness to epileptic seizures. A wave of emergency calls and hospital admissions suggested a single environmental cause which was soon identified as the animation, and in particular one sequence in which red and cyan colors flashed in an alternation at around 12Hz covering much of the screen area. Before the cause had been pinpointed this sequence was shown on a news bulletin about the incident, resulting in further casualties. Altogether 685 people, most of them children, were hospitalized although most were discharged quickly.

Color contrast checker

- WCAG 2.0 level AA requires a contrast ratio of at least 4.5:1 for normal text and 3:1 for large text
- Level AAA requires a contrast ratio of at least 7:1 for normal text and 4.5:1 for large text.
- Large text is defined as 14 point (typically 18.66px) and bold or larger, or 18 point (typically 24px) or larger



Foreground Color

#FF0A85

Lightness

Background Color

#420000

Lightness

Contrast Ratio

4.57:1

permalink

Normal Text

WCAG AA: Pass

WCAG AAA: Fail

The five boxing wizards jump quickly.

Large Text

WCAG AA: Pass

WCAG AAA: Pass

The five boxing wizards jump quickly.

https://webaim.org/resources/contrastchecker/

WAI-ARIA

WAI-ARIA

- The last ten years have seen the rise of Ajax, JavaScript, HTML5, and countless front-end frameworks
  - The internet is no longer a place of static HTML pages, but it is has become a playground for complex, almost desktop-like web applications, each with their own widgets, controls, and behavior
- Sometimes web development is pushed to the limit… and people with disabilities struggle with these new techniques
- This is not due to disabled JavaScript or insufficient capabilities of current assistive technology (AT)
  - On the contrary, in 2012 WebAIM found that over 98 percent of screen reader users had JavaScript enabled
  - Additionally, ATs like screen readers or refreshable Braille displays are getting better every year

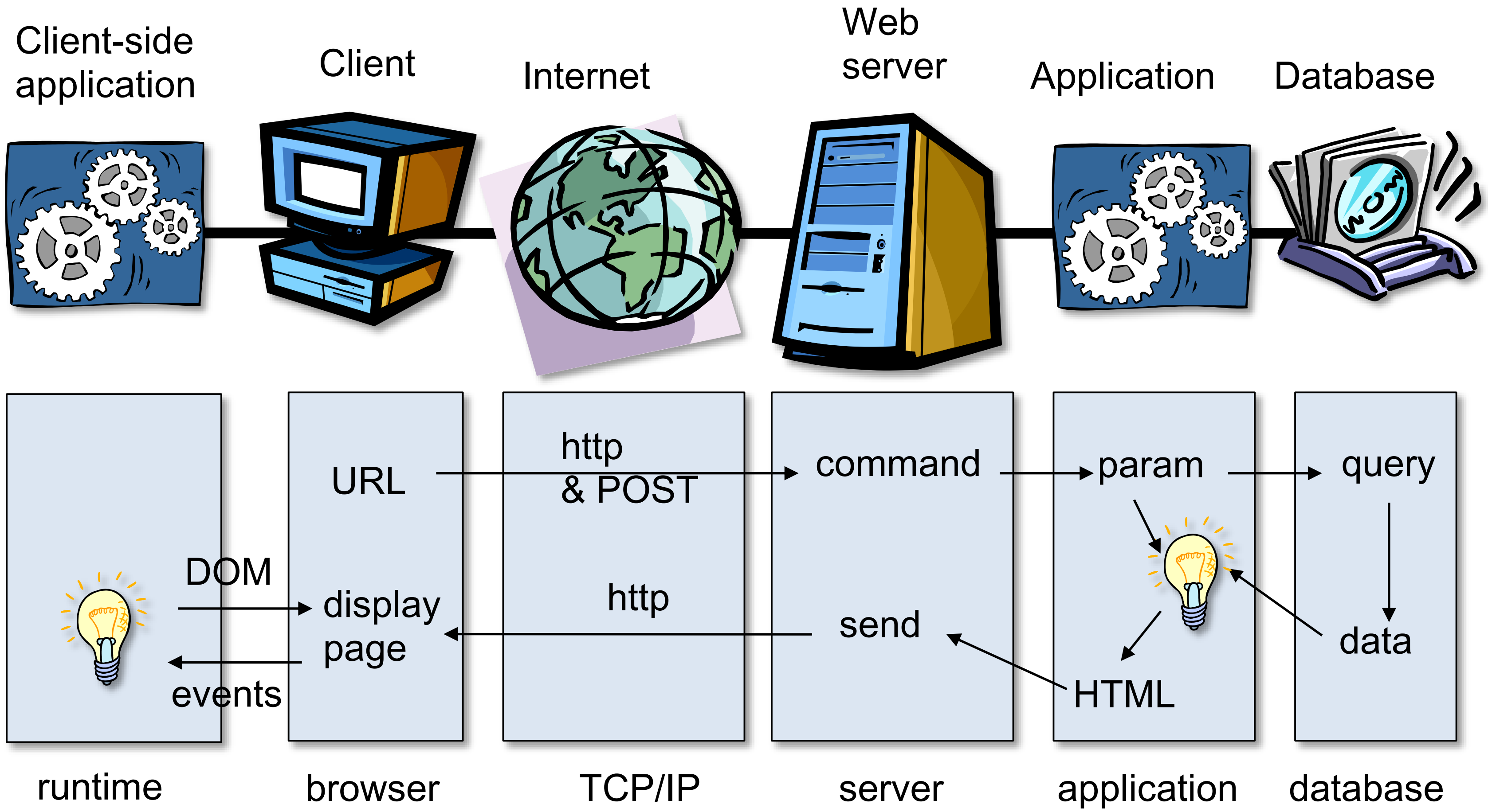
WAI-ARIA

- The problem lies with HTML limited ability to mark up web applications that make heavy use of JavaScript and produce a huge amount of dynamic content
- Four key obstacles can be identified when assistive technologies deal with JavaScript applications
  - Unknown functionality of components
  - Unknown states and properties of components
  - Unreported change of dynamic content
  - Bad keyboard accessibility

WAI-ARIA

- In general, accessibility issues with rich internet applications can be characterized as
  - Inability to provide the semantic structure of page areas and functionality (e.g., navigation, main content, search, etc.)
  - Inaccessibility of content that is dynamic and may change within the page (e.g., AJAX content updates)
  - Inability to change keyboard focus to page elements (e.g., setting focus to an error message within the page)
  - Difficulty with keyboard and screen reader accessibility with complex widgets and navigation elements (e.g., sliders, menu trees, etc.)
- ARIA can help address many of these issues

Rich-client transactions



Client-side application

Client

Internet

Web server

Application

Database

runtime

browser

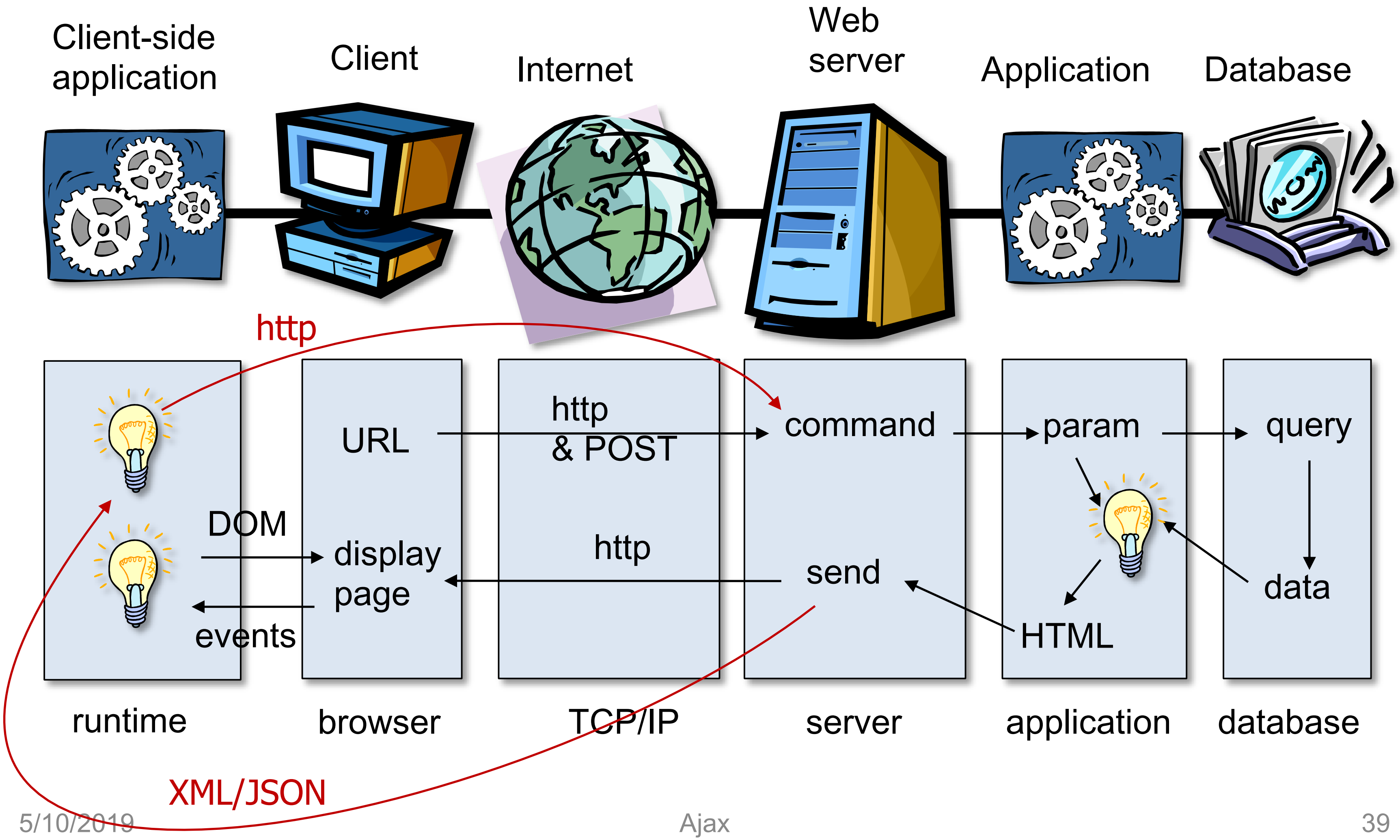
TCP/IP

server

application

database

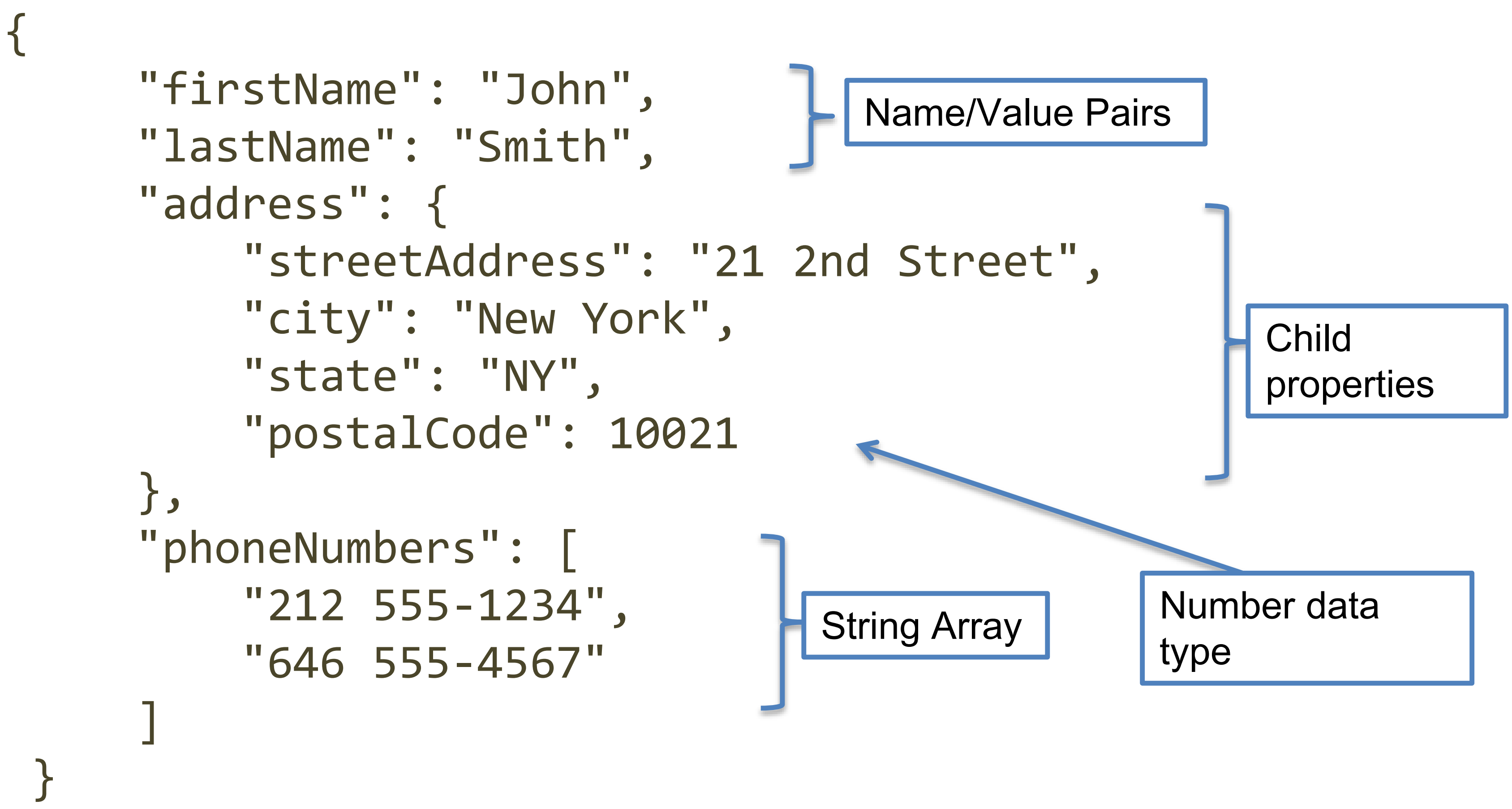
Rich-client asynchronous transactions



JSON

- “JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate” – JSON.org
- Important: JSON is a subset of JavaScript
- JSON is built on two structures
  - A collection of name/value pairs: in various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array. { … }
  - An ordered list of values: in most languages, this is realized as an array, vector, list, or sequence. […]

JSON example



```
{
     "firstName": "John",
     "lastName": "Smith",
     "address": {
         "streetAddress": "21 2nd Street",
         "city": "New York",
         "state": "NY",
         "postalCode": 10021
     },
     "phoneNumbers": [
         "212 555-1234",
         "646 555-4567"
     ]
 }
```

Name/Value Pairs

Child properties

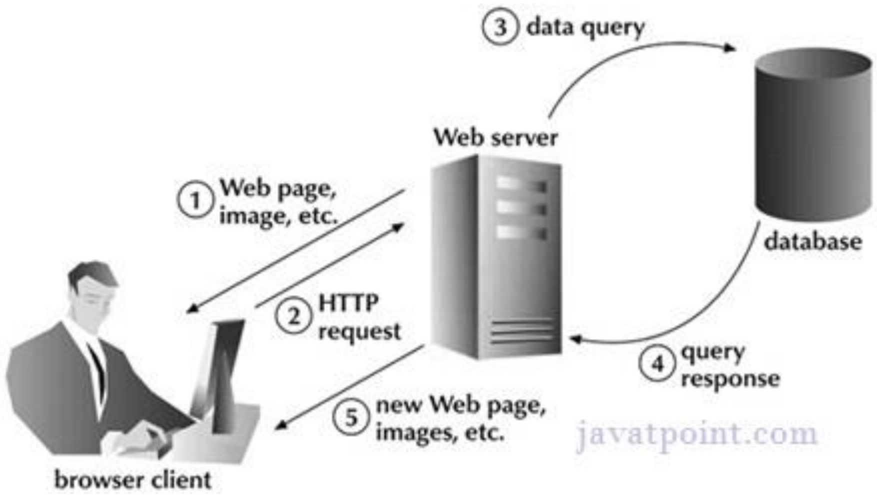
String Array

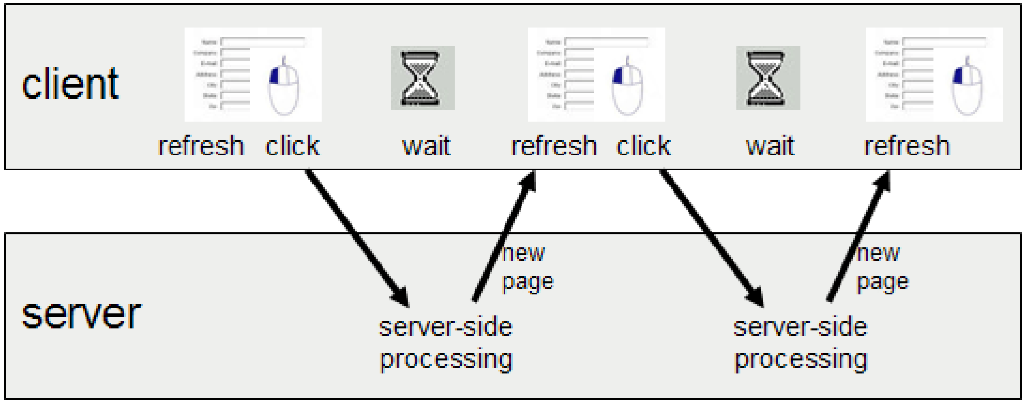
Number data type

Rich-client asynchronous transactions

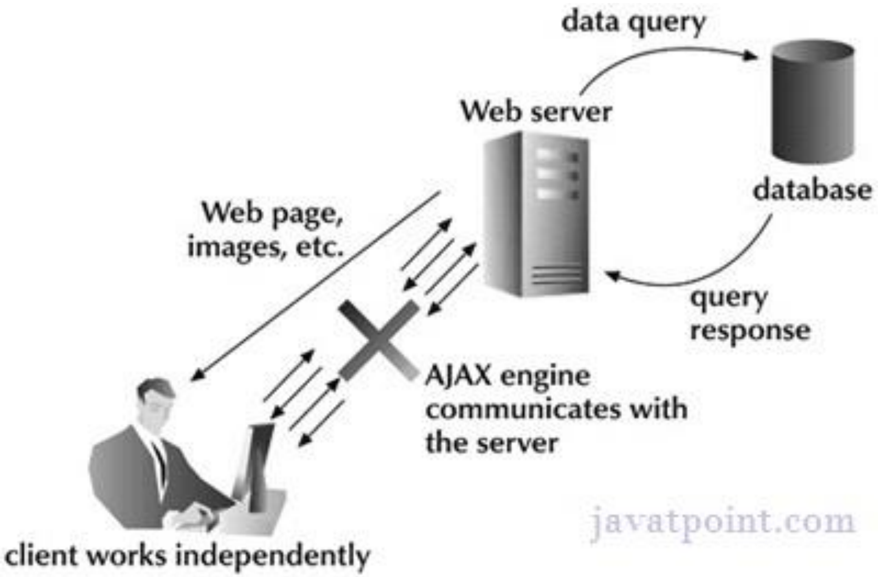
- In 2005, Jesse James Garrett wrote an online article titled “Ajax: A New Approach to Web Applications” (https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf)
- This article outlined a technique that he referred to as Ajax, short for Asynchronous JavaScript+XML, consisting in making server requests for additional data without unloading the web page, for a better user experience
- Garrett explained how this technique could be used to change the traditional click-and-wait paradigm that the Web had been stuck in since its start

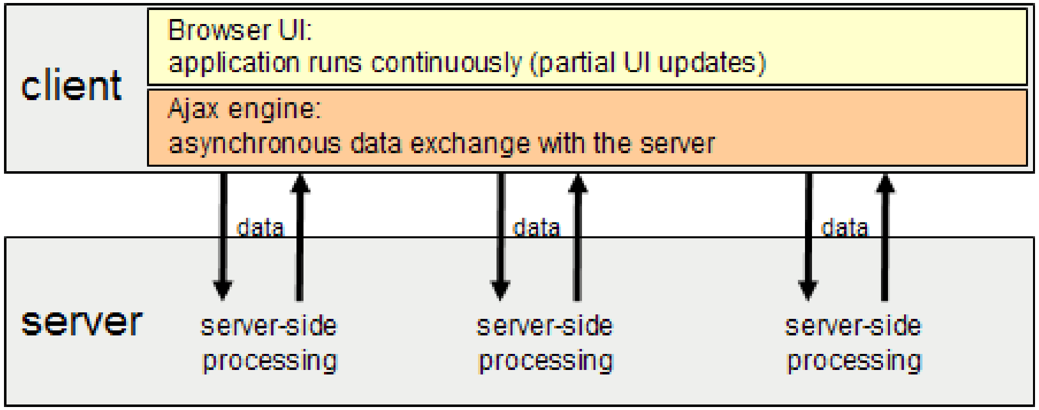
Synchronous (classic) web application model





Asynchronous web application model




client

Browser UI:
application runs continuously (partial UI updates)

Ajax engine:
asynchronous data exchange with the server

data

data

data

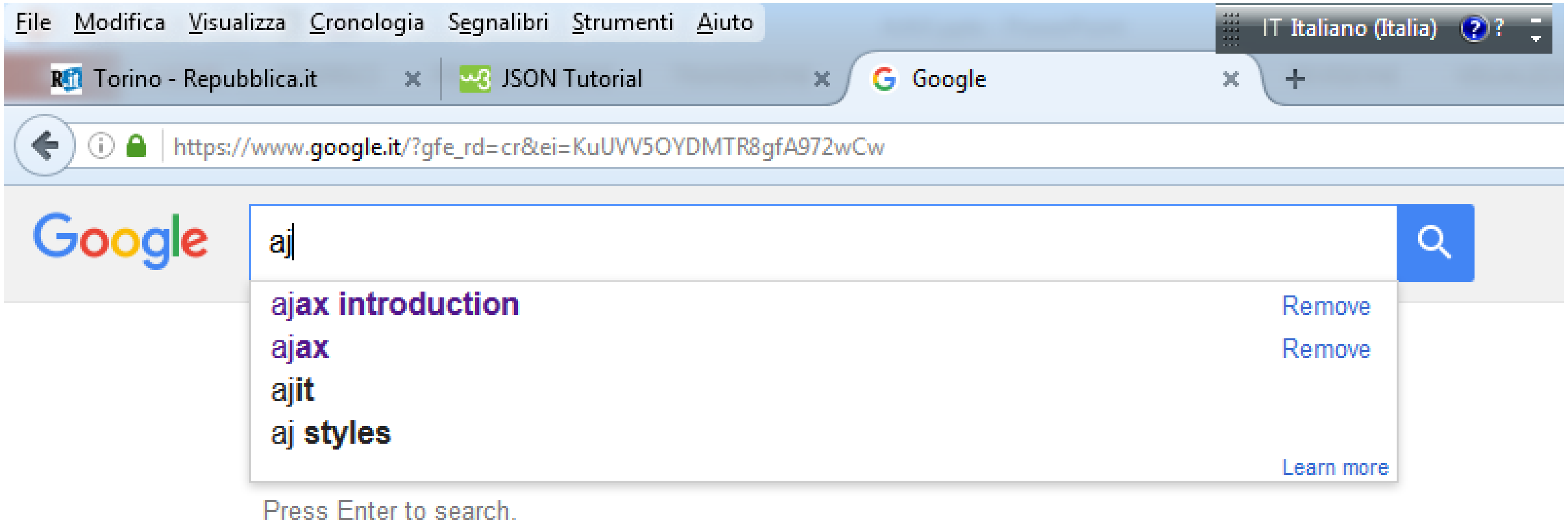
server

server-side processing

server-side processing

server-side processing

Example



File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
IT Italiano (Italia)
Torino - Repubblica.it
JSON Tutorial
Google
https://www.google.it/?gfe_rd=cr&ei=KuUVV5OYDMTR8gfA972wCw
Google
aj
ajax introduction
Remove
ajax
Remove
ajit
aj styles
Learn more
Press Enter to search.

WAI-ARIA

- “Web Accessibility Initiative Accessible Rich Internet Applications (WAI-ARIA)
- A technical specification that defines a way to make Web content and Web applications more accessible to people with disabilities
  - W3C Recommendation WAI-ARIA 1.1 (14/12/2017)
- Provides attributes for extending HTML markup with roles, states and properties to expose Web applications to Assistive Technologies
- ARIA was designed to be recognized only by assistive technology and does not affect the DOM or the style in any way

WAI-ARIA

- WAI-ARIA provides web authors with
  - Roles to describe the structure of the Web page, such as headings, regions, and tables (grids)
  - Roles to describe the type of widget presented, such as “menu”, “treeitem”, “slider”, “progressmeter”, …
  - Properties to describe the state widgets are in, such as “checked” for a check box, or “haspopup” for a menu
  - Properties to define live regions of a page that are likely to get updates (such as stock quotes), as well as an interruption policy for those updates—for example, critical updates may be presented in an alert dialog box, and incidental updates occur within the page
  - Properties for drag-and-drop that describe drag sources and drop targets
  - A way to provide keyboard navigation for the Web objects and events

WAI-ARIA core components

- Roles
  - ARIA roles define what an element is or does
  - Most HTML elements have a default role that is presented to assistive technology
  - E.g., a button has a default role of "button" and a form has a default role of "form“
  - With ARIA, you can define roles that are not available in HTML, or you can override HTML default roles
  - Example: <form role="search">: in HTML, all forms have the same semantics but with ARIA, you can add to the semantics of a particular form to define it as the search form
- Types of ARIA roles
  - Landmark roles (the most important)
  - Widget roles
  - Document structure roles
  - Abstract roles (ontology, not for developers)
  - https://www.w3.org/WAI/PF/aria/roles

WAI-ARIA core components

- Properties
  - ARIA properties define properties or meanings of elements
  - You can extend HTML native semantics by defining properties for elements that are not allowed in standard HTML
  - Example: <input aria-required="true">: this property will cause a screen reader to read this input as being required (meaning the user must complete it)
- States
  - ARIA states are properties that define the current condition of an element
  - They generally change based on user interaction or some dynamic variable
  - Example: <input aria-disabled="true">: this property will cause a screen reader to read this input as being disabled

WAI-ARIA core components

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

- ARIA roles, states, and properties can be defined in markup or they can be defined and dynamically set and changed using scripting
- ARIA states and property attributes always start with "aria-" (e.g., aria-required="true")

HTML5 and ARIA

- Use ARIA only if necessary
  - 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
  - Example:

```
<!------ avoid these if possible ------>
<span role="button">...</span>
<div role="link">...</div>

<!------ these are preferred ------>
<button type="button">...</button>
<a href="link">...</a>
```

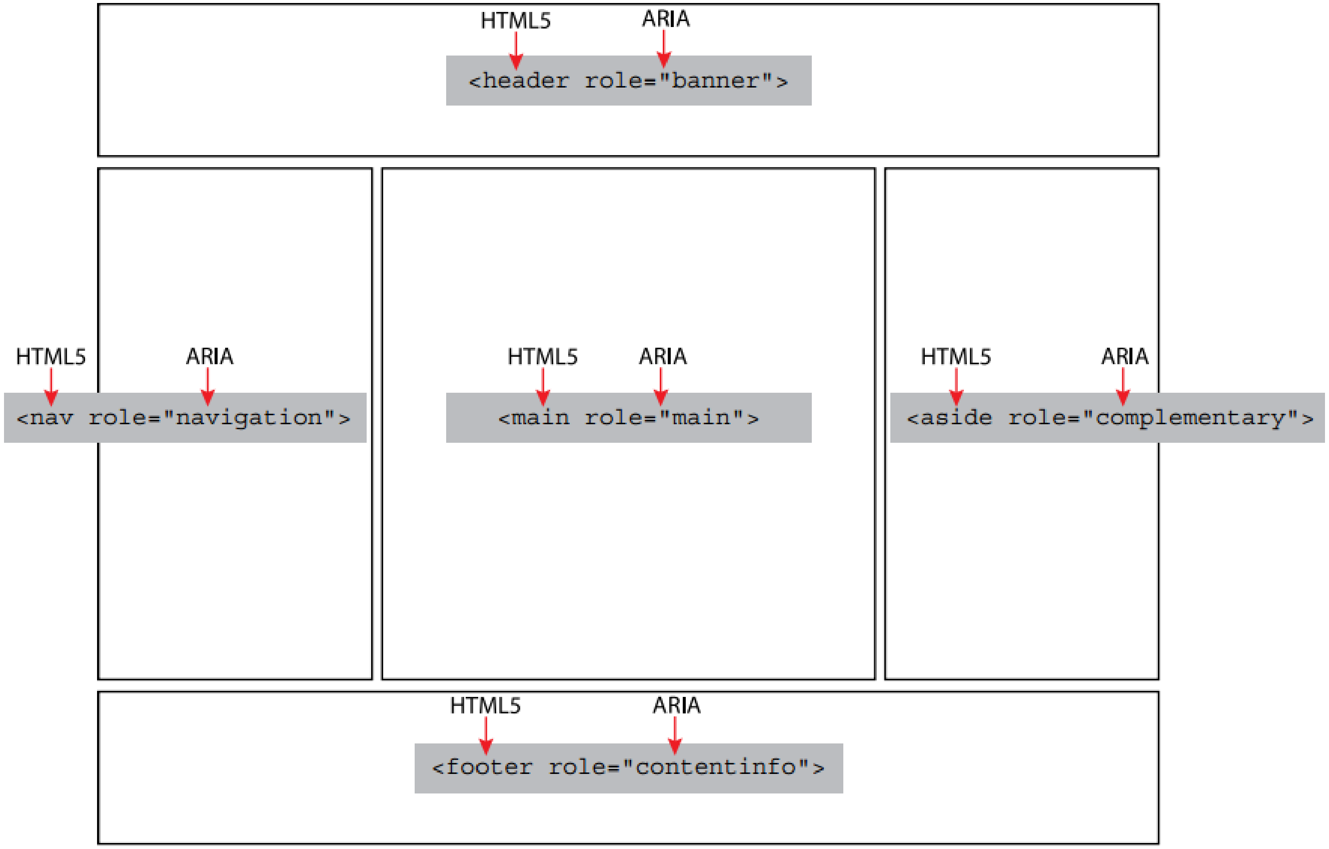
ARIA roles

- ARIA provides a rich role taxonomy that enables developers to classify otherwise meaningless tags
- This prepares the tags for assistive technologies by revealing the functionality or the part they play in the overall web document

```
<ul id="myTab" class="nav nav-tabs" role="tablist">
  <li class="active"> <a href="#home" role="tab"
      data-toggle="tab">Home</a> </li>
  <li> <a href="#profile" role="tab"
      data-toggle="tab">Profile</a> </li>
  <li> <a href="#articles" role="tab"
      data-toggle="tab">Articles</a> </li>
</ul>
```

WAI-ARIA for landmark roles

- HTML5 elements and ARIA roles are complementary
  - Including both of them in your site provides a solid code structure and good navigation around the page



HTML5 ARIA

`<header role="banner">`

HTML5 ARIA

`<nav role="navigation">`

HTML5 ARIA

`<main role="main">`

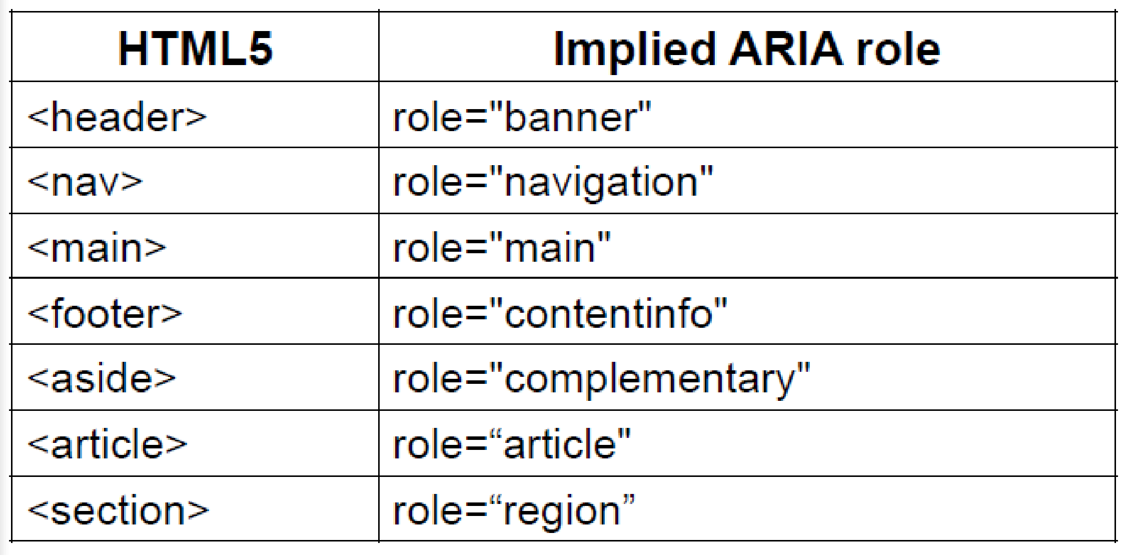
HTML5 ARIA

`<aside role="complementary">`

HTML5 ARIA

`<footer role="contentinfo">`

HTML5 semantic tags and ARIA landmark roles

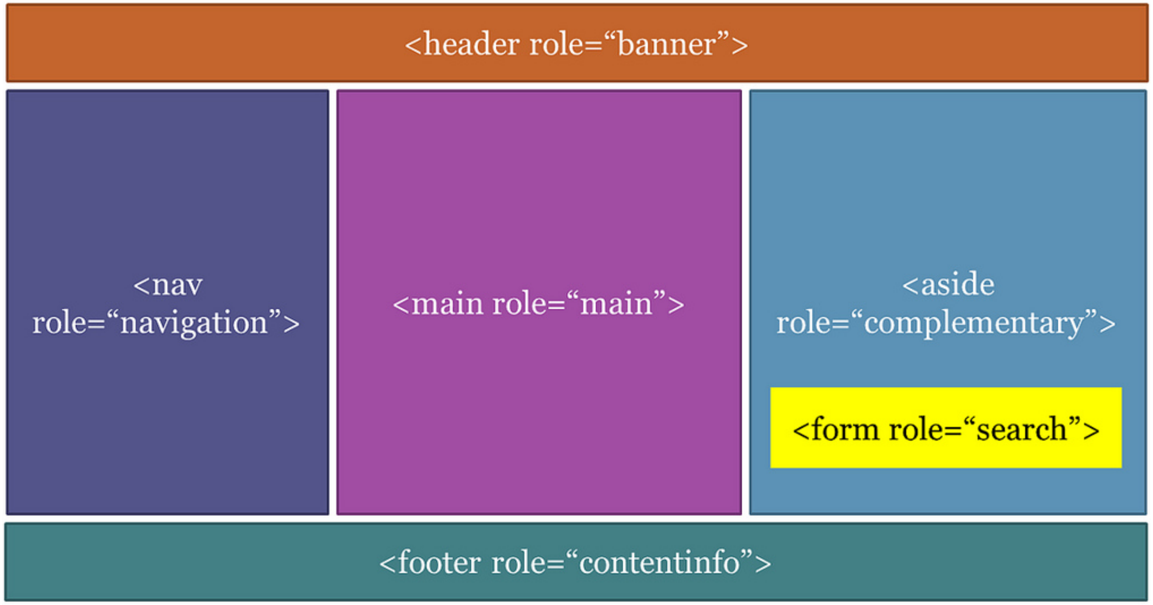


HTML5	Implied ARIA role
<header>	role="banner"
<nav>	role="navigation"
<main>	role="main"
<footer>	role="contentinfo"
<aside>	role="complementary"
<article>	role=“article"
<section>	role=“region”

- If HTML5 sectioning elements are used without understanding the associated landmark structure, assistive technology users will most likely be confused and less efficient in accessing content and interacting with web pages

Landmark roles

- Landmarks provide a powerful way to identify the organization and structure of a web page
  - Support keyboard navigation to the structure of a web page for screen reader users
- All content should reside in a semantically meaningful landmark in order that content is not missed by the user



<header role=“banner”>

<nav role=“navigation”>

<main role=“main”>

<aside role=“complementary”>

<form role=“search”>

<footer role=“contentinfo”>

Landmark roles

- Eight roles, each representing a block of content that occurs commonly on web pages
  - role=”banner”
  - role=”navigation” (e.g., a menu)
  - role=”main” (the main content of the page)
  - role=”complementary” (e.g., a sidebar)
  - role=”contentinfo” (meta data about the page, e.g., a copyright statement)
  - role=”search”
  - role=”form”
  - role=”application” (a web application with its own keyboard interface)
- If a role is used more than once on a page, the aria-label attribute should also be used in order to distinguish between the two regions
  - <div role=”navigation” aria-label=”Main menu”>
  - <div role=”navigation” aria-label=”User menu”>

Landmark roles usage

- Step 1: Identify the logical structure
  - Break the page into perceivable areas called "areas"
  - Typically, designers indicate areas visually using alignment and spacing of content
  - Regions can be further defined into logical sub-areas as needed
- Step 2: Assign landmark roles to each area
  - Assign landmark roles based on the type of content in the area
  - banner, main, complementary and contentinfo landmarks should be top level landmarks
  - Landmark roles can be nested to identify parent/child relationships of the information being presented

Landmark roles usage

- Step 3: Label each area
  - If a specific landmark role is used more than once on a web page, it should have a unique label
  - If a area begins with a heading element (e.g. h1-h6), it can be used as the label for the area using aria-labelledby attribute
  - If a area does not have a heading element, provide a label using the aria-label attribute
  - Avoid using the landmark role as part of the label: a navigation landmark with a label “Site Navigation” will be announced by a screen reader as “Site Navigation Navigation”, the label should simply be “Site”

Widget roles

- 25 widget roles
  - alert, alertdialog, button, checkbox, dialog, gridcell, link, log, marquee, menuitem, menuitemcheckbox, menuitemradio, option, progressbar, radio, scrollbar, slider, spinbutton, status, tab, tabpanel, textbox, timer, tooltip, treeitem
- 9 composite roles
  - Typically act as containers that manage other contained widgets
  - combobox, grid, listbox, menu, menubar, radiogroup, tablist, tree, treegrid

Document structure roles

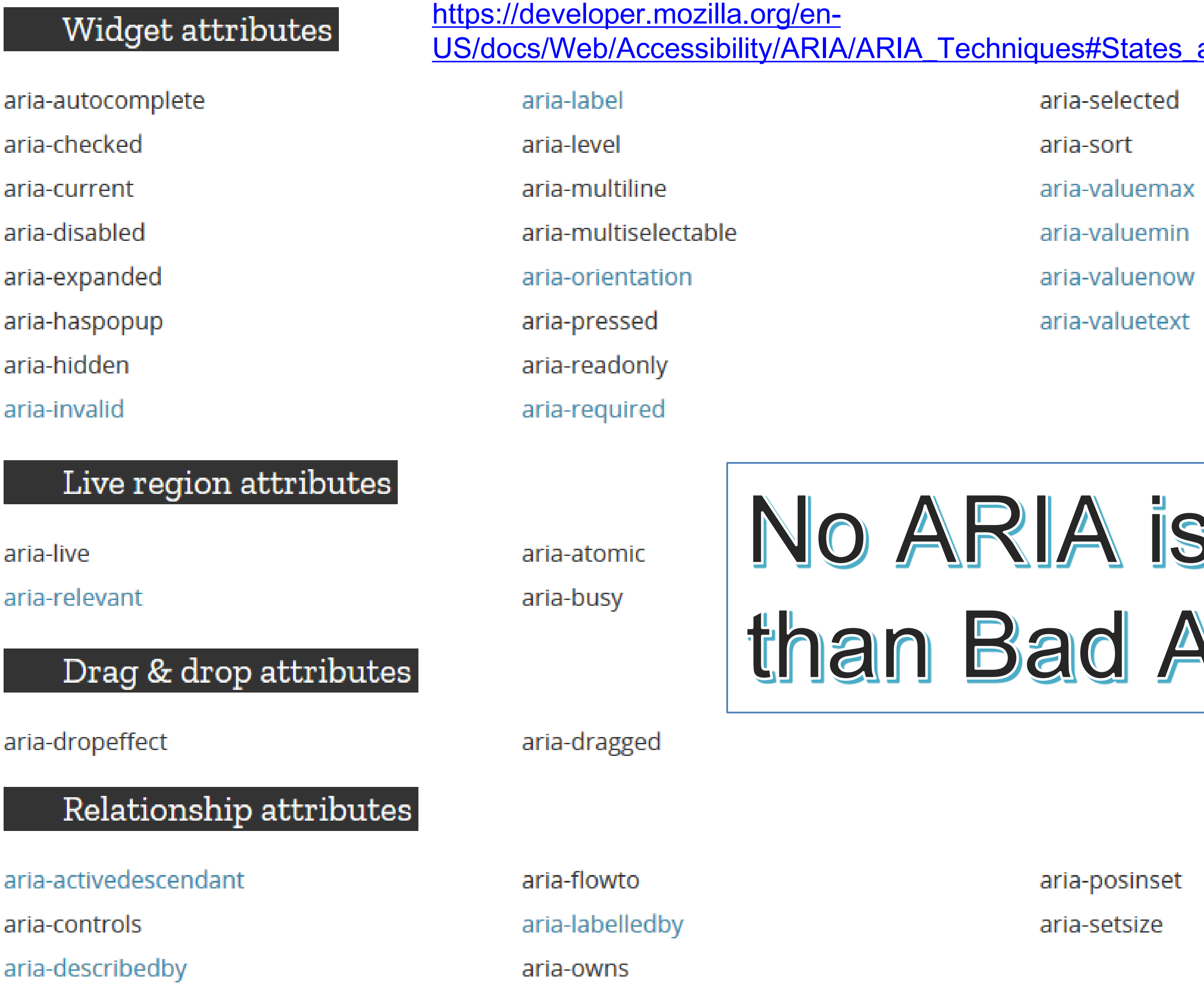
- Describe structures that organize content in a page
  - Usually not interactive content
  - article, columnheader, definition, directory, document, group, heading, img, list, listitem, math, note, presentation, region, row, rowgroup, rowheader, separator, toolbar

States and attributes

- Similar features
  - Both provide specific information about an object, and contribute to the nature of roles
- Major difference
  - the values of properties (e.g., aria-labelledby) are often less likely to change throughout the application life-cycle than the values of states (e.g, aria-checked) which may change frequently due to user interaction
- aria-prefixed markup attributes
- Categorized as
  - Widget attributes
  - Live Region attributes
  - Drag-and-Drop attributes
  - Relationship attributes

States and attributes

https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/ARIA_Techniques#States_and_properties



Widget attributes

- aria-autocomplete
- aria-checked
- aria-current
- aria-disabled
- aria-expanded
- aria-haspopup
- aria-hidden
- aria-invalid
- aria-label
- aria-level
- aria-multiline
- aria-multiselectable
- aria-orientation
- aria-pressed
- aria-readonly
- aria-required
- aria-selected
- aria-sort
- aria-valuemax
- aria-valuemin
- aria-valuenow
- aria-valuetext

Live region attributes

- aria-live
- aria-relevant
- aria-atomic
- aria-busy

Drag & drop attributes

- aria-dropeffect
- aria-dragged

Relationship attributes

- aria-activedescendant
- aria-controls
- aria-describedby
- aria-flowto
- aria-labelledby
- aria-owns
- aria-posinset
- aria-setsize

No ARIA is better than Bad ARIA

WAI-ARIA in 5 steps

1. Alert users to what each element or widget is: the element's role (such as a menu or a tree)
2. Alert the user to each element properties and important relationships (such as "aria-haspopup", "aria-describedby" and other labels)
3. Alert the user to what each element is doing: the element's state (such as "aria-expanded" or "aria-disabled")
4. Alert users to any changes in the element's state
5. Make sure the widget is keyboard accessible and the focus is predictable
   - Interactive controls should receive focus through the keyboard
   - Events can be triggered through the keyboard
   - How to trigger events should be intuitive to the user

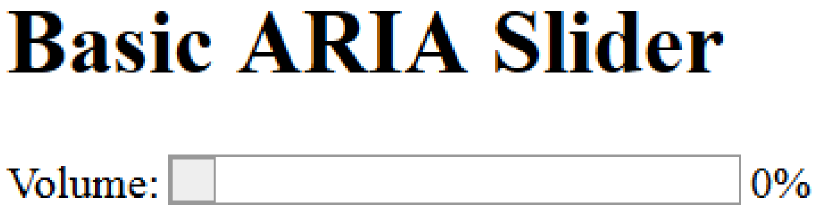
States and Properties

- The W3C provides some detailed WAI-ARIA best practices to follow
  - https://www.w3.org/TR/wai-aria-practices/

No ARIA is better than Bad ARIA

```
<li class="active">
  <a id="tab1" href="#home" role="tab" aria-controls="home"
     data-toggle="tab">Home</a>
</li>
```

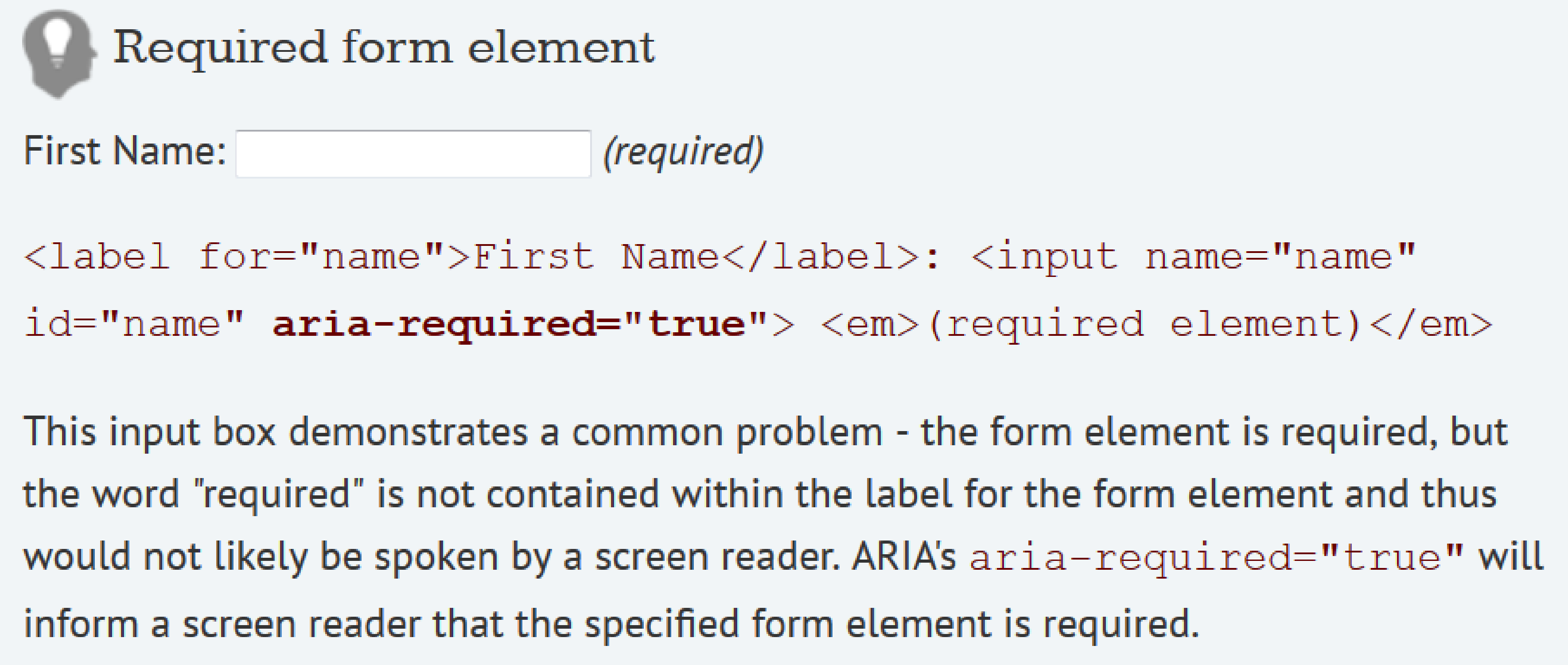
Example: slider



Basic ARIA Slider

```
<div class="clearfix">
  <span id="sliderLabel" class="floatLeft">Volume:</span>
  <div id="sliderRail1" class="sliderRailfloatLeft">
    <button class="sliderThumb" id="sliderThumb1" role="slider"
     aria-labelledby="sliderLabel" aria-valuemin="0" aria-valuemax="100"
     aria-valuenow="0" aria-valuetext="0%"></button>
  </div>
  <span id="sliderValue1"class="floatLeft">0%</span>
</div>
```

Example



Required form element

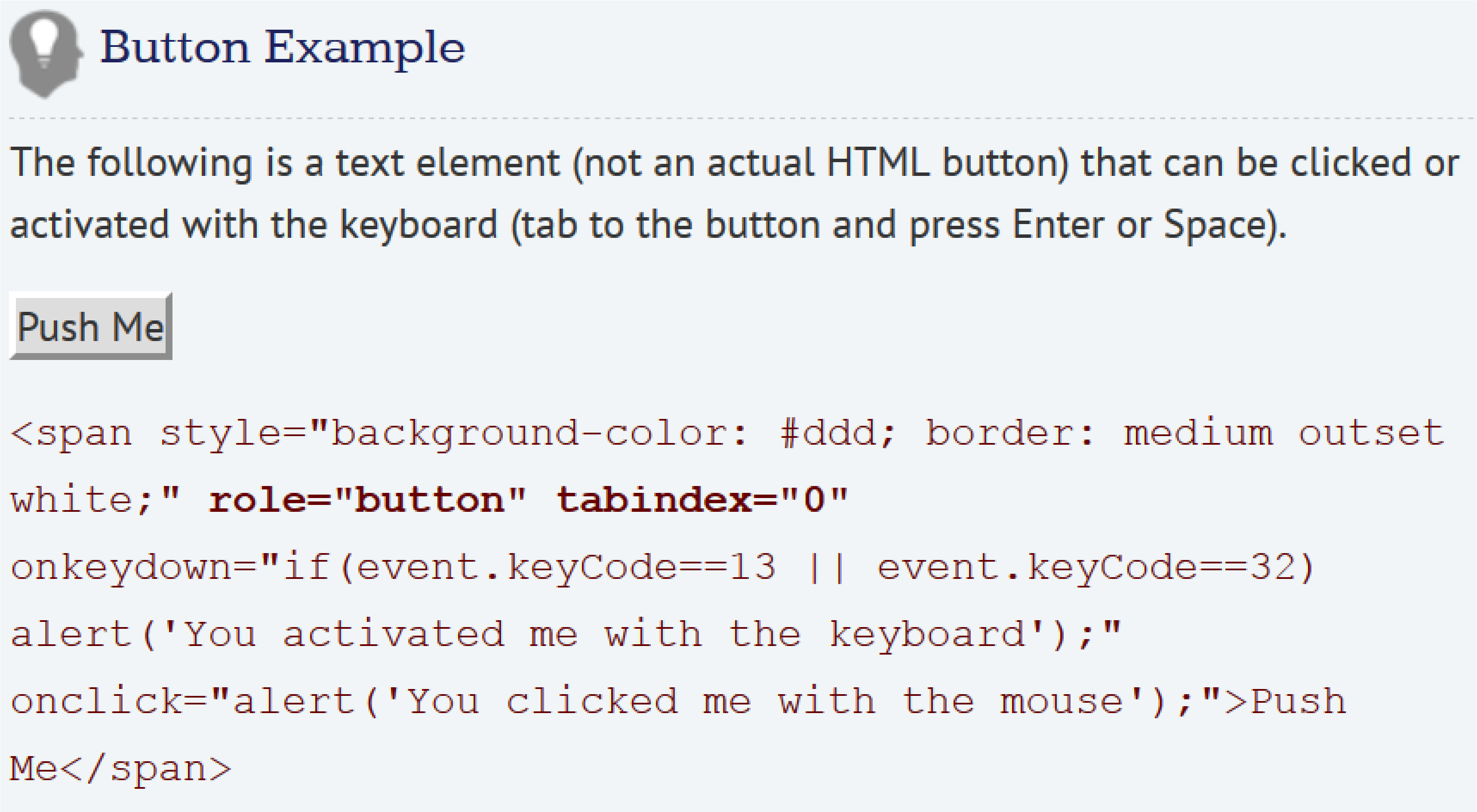
First Name: ▭ *(required)*

```
<label for="name">First Name</label>: <input name="name"
id="name" aria-required="true"> <em>(required element)</em>
```

This input box demonstrates a common problem - the form element is required, but the word "required" is not contained within the label for the form element and thus would not likely be spoken by a screen reader. ARIA's `aria-required="true"` will inform a screen reader that the specified form element is required.

Example

https://webaim.org/techniques/aria/



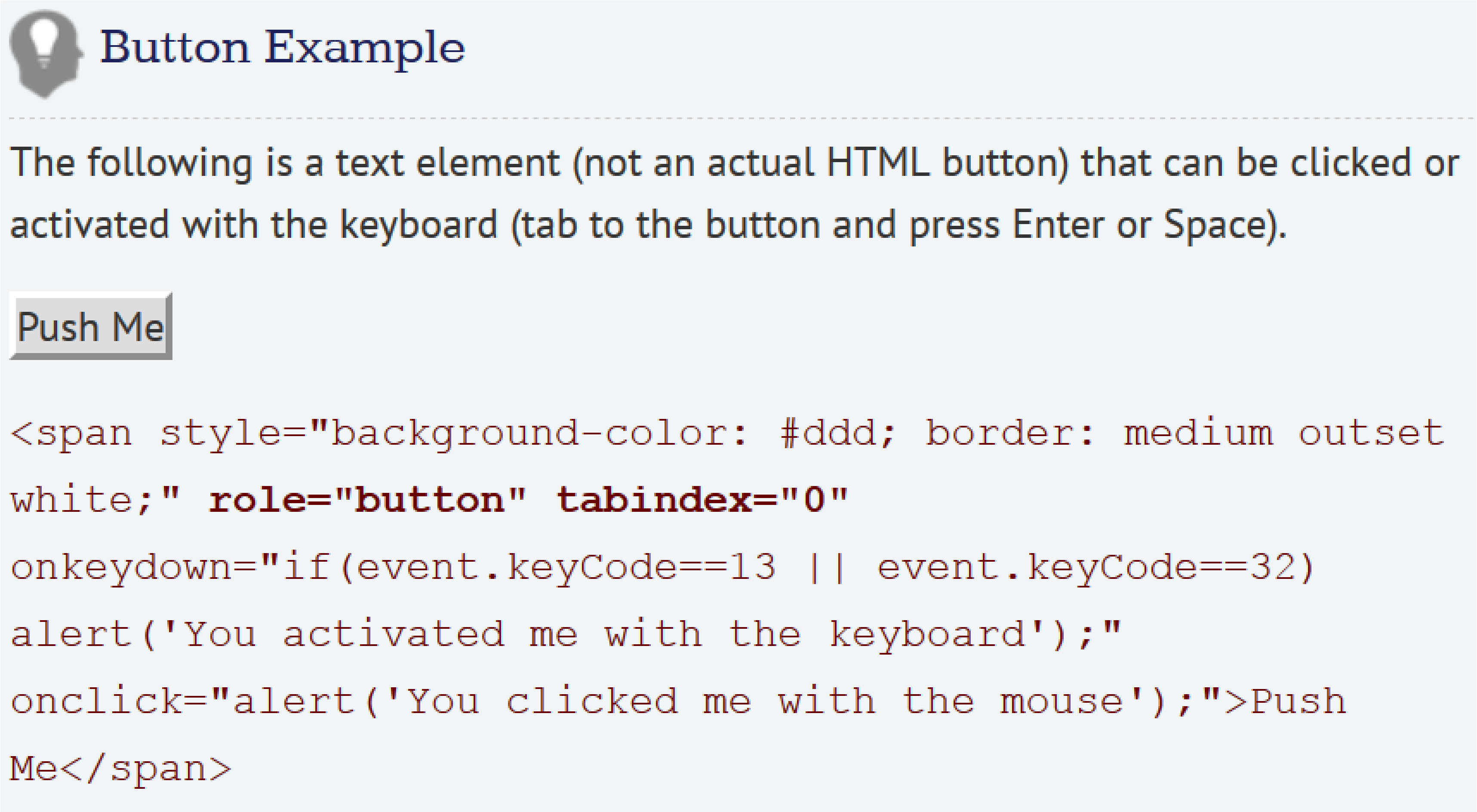
Button Example

The following is a text element (not an actual HTML button) that can be clicked or activated with the keyboard (tab to the button and press Enter or Space).

```
<span style="background-color: #ddd; border: medium outset
white;" role="button" tabindex="0"
onkeydown="if(event.keyCode==13 || event.keyCode==32)
alert('You activated me with the keyboard');"
onclick="alert('You clicked me with the mouse');">Push
Me</span>
```

Example

https://webaim.org/techniques/aria/



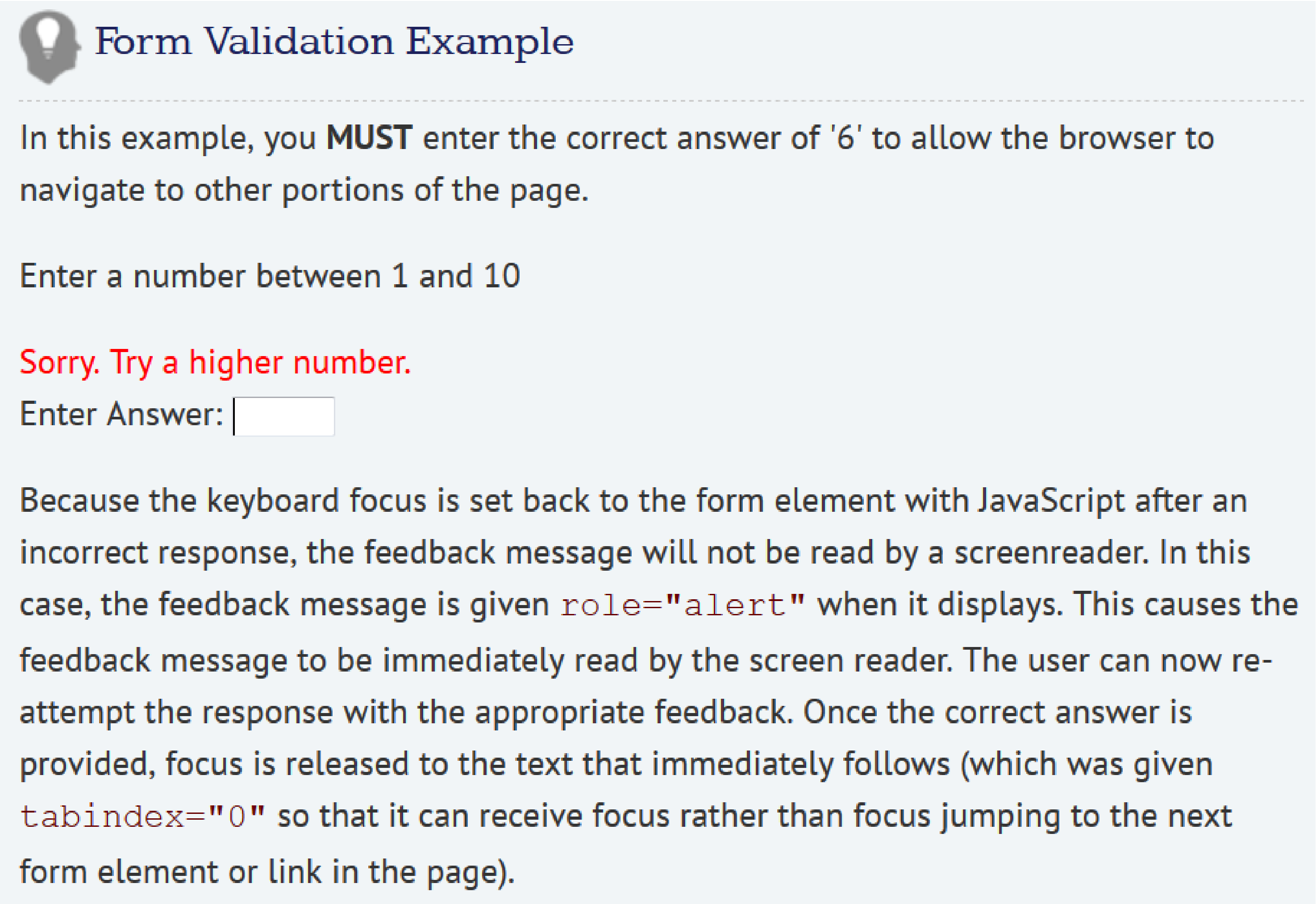
Button Example

The following is a text element (not an actual HTML button) that can be clicked or activated with the keyboard (tab to the button and press Enter or Space).

```
<span style="background-color: #ddd; border: medium outset
white;" role="button" tabindex="0"
onkeydown="if(event.keyCode==13 || event.keyCode==32)
alert('You activated me with the keyboard');"
onclick="alert('You clicked me with the mouse');">Push
Me</span>
```

Example

```
<p class="incorrect" id="feedback" role="alert">
   Sorry. Try a higher number.
</p>
```



Form Validation Example

In this example, you **MUST** enter the correct answer of '6' to allow the browser to navigate to other portions of the page.

Enter a number between 1 and 10

Sorry. Try a higher number.

Enter Answer:

Because the keyboard focus is set back to the form element with JavaScript after an incorrect response, the feedback message will not be read by a screenreader. In this case, the feedback message is given `role="alert"` when it displays. This causes the feedback message to be immediately read by the screen reader. The user can now re-attempt the response with the appropriate feedback. Once the correct answer is provided, focus is released to the text that immediately follows (which was given `tabindex="0"` so that it can receive focus rather than focus jumping to the next form element or link in the page).

Dynamic content updates

- When content changes dynamically within a web page, it may cause accessibility problems
  - What happens if a screen reader is currently reading an element that is updated?
  - If the updated content is important, should you interrupt the user and set focus immediately to the new content, do you simply inform the user of the update, or do you do nothing?
  - How do you set focus or allow the user to jump to the updated content?
- With WAI-ARIA the developer can identify regions that dynamically change as a live region
  - A live region allows content updates in a way that a screen reader understands
  - It also allows the developer to add additional functionality to alert the user, provide controls for the live region, determine the amount of new content that would be read, …

Live regions

- To create a live region, the developer adds the aria-live property to the element
  - The value, or politeness level (or alternatively the intrusiveness level) specifies what a screen reader should do when the element is updated
- aria-live="off“ tells the screen reader not to announce the update
  - Should be used for non-important or irrelevant content updates
- aria-live=“polite“ notifies the user of the content change as soon as he/she is done with the current task
  - This might take the form of a beep or an audio indication of the update, and the user can then choose to directly jump to the updated content
  - Should be the most common for content updates, especially for things like status notification, weather or stock updates, chat messages, etc.

Live regions

- aria-live=“assertive” will result in the user being alerted to the content change immediately or as soon as possible
  - Assertive would be used for important updates, such as error messages

- Example

```
<div id="myTabContent" class="tab-content"
     aria-live="polite">
    ...
</div>
```

Enhanced Keyboard Navigation

- In HTML, the only elements that could receive keyboard focus with the TAB key are links and form elements
- With scripting, however, you can add mouse interactivity to nearly any element (e.g. spans, paragraphs, …)
- The functionality of these non-focusable elements cannot be made accessible to screen reader and keyboard-only users
- ARIA enables every HTML element to receive keyboard focus by extending the “tabindex” attribute so that it can be applied to any element

```
<li class="tab active">
   <a id="tab1" href="#home" role="tab" aria-controls="home"
      aria-selected="true" data-toggle="tab" tabindex="0">Home</a>
</li>
```

Enhanced Keyboard Navigation

- The tabindex attribute has three distinct uses:
  - tabindex="1" (or any number greater than 1) defines an explicit tab order; this is almost always a bad idea
  - tabindex="0" allows elements besides links and form elements to receive keyboard focus; it does not change the tab order, but places the element in the logical navigation flow, as if it were a link on the page
  - tabindex="-1" removes the element from the default navigation flow but allows it to receive "programmatic" focus, i.e. focus can be set to the element through scripting, links, etc.
    - Example: a modal dialog window

LEGGE STANCA

Legge stanca

- D.M. 9 luglio 2004 che regola l’accessibilità dei siti web in Italia
  - «Disposizioni per favorire l’accesso dei soggetti disabili agli strumenti informatici»
  - Definita come la capacità dei sistemi informatici di erogare informazioni fruibili, senza discriminazioni, anche da parte di coloro che a causa di disabilità necessitano di tecnologie assistive o di configurazioni particolari

Soggetti destinatari della legge

https://www.webaccessibile.org/articoli/legge-stanca-guida-ai-22-requisiti-tecnici/

- Pubbliche amministrazioni
- Enti pubblici economici, aziende private concessionarie di servizi pubblici
- Aziende municipalizzate regionali
- Enti di assistenza e di riabilitazione pubblici
- Aziende di trasporto e di telecomunicazione a prevalente partecipazione di capitale pubblico
- Aziende appaltatrici di servizi informatici

Requisiti tecnici

- 22 requisiti tecnici, ridotti a 12 nel 2013
    - Riferimento: WCAG 2.0, livello AA
- Requisito 1 – Alternative testuali: fornire alternative testuali per qualsiasi contenuto di natura non testuale in modo che il testo predisposto come alternativa possa essere fruito e trasformato secondo le necessità degli utenti, come per esempio convertito in stampa a caratteri ingranditi, in stampa Braille, letto da una sintesi vocale, simboli o altra modalità di rappresentazione del contenuto.
- Requisito 2 – Contenuti audio, contenuti video, animazioni: fornire alternative testuali equivalenti per le informazioni veicolate da formati audio, formati video, formati contenenti immagini animate (animazioni), formati multisensoriali in genere.

Requisiti tecnici

- Requisito 3 – Adattabile: creare contenuti che possano essere presentati in modalità differenti (ad esempio, con layout più semplici), senza perdita di informazioni o struttura.
- Requisito 4 – Distinguibile: rendere più semplice agli utenti la visione e l’ascolto dei contenuti, separando i contenuti in primo piano dallo sfondo.
- Requisito 5- Accessibile da tastiera: rendere disponibili tutte le funzionalità anche tramite tastiera.
- Requisito 6- Adeguata disponibilità di tempo: fornire all’utente tempo sufficiente per leggere ed utilizzare i contenuti.
- Requisito 7- Crisi epilettiche: non sviluppare contenuti che possano causare crisi epilettiche.

Requisiti tecnici

- Requisito 8- Navigabile: fornire all’utente funzionalità di supporto per navigare, trovare contenuti e determinare la propria posizione nel sito e nelle pagine.
- Requisito 9- Leggibile: rendere leggibile e comprensibile il contenuto testuale.
- Requisito 10- Prevedibile: creare pagine web che appaiano e che si comportino in maniera prevedibile.
- Requisito 11- Assistenza nell’inserimento di dati e informazioni: aiutare l’utente ad evitare gli errori ed agevolarlo nella loro correzione.
- Requisito 12- Compatibile: garantire la massima compatibilità con i programmi utente e con le tecnologie assistive.

https://www.webaccessibile.org/categorie/legge-stanca/guida-ai-22-requisiti-tecnici/

References

- WCAG 2.1 guidelines
  - https://www.w3.org/TR/WCAG21/
- WebAIM: WCAG 2 checklist
  - https://webaim.org/standards/wcag/checklist
- WAI-ARIA
  - https://www.w3.org/WAI/intro/aria.php
- Accessible Rich Internet Applications (WAI-ARIA) 1.1
  - https://www.w3.org/TR/wai-aria/
- ARIA landmarks example
  - https://www.w3.org/TR/2017/NOTE-wai-aria-practices-1.1-20171214/examples/landmarks/index.html
- WebAIM: accessibility to Rich Internet Applications
  - https://webaim.org/techniques/aria/
- Legge Stanca
  - http://www.camera.it/parlam/leggi/04004l.htm

License