Web Accessibility Tutorials

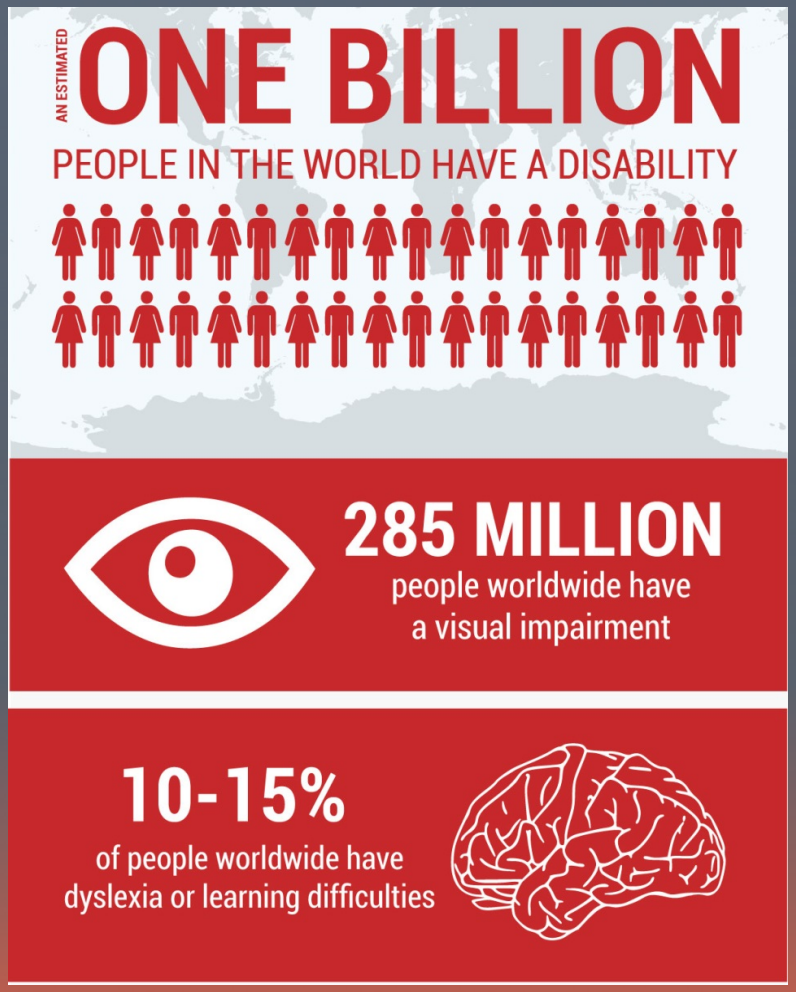
IMPLEMENTATION OF ACCESSIBLE WEBSITES

Agenda

- Introduction
- Assistive technologies
- Web accessibility
- Accessible design
- Accessible navigation
- Accessible forms
- Accessible images
- Accessible tables
- Testing for accessibility
- Exercises

Introduction

Facts and figures



World Health Organization, 2011

Assistive technologies

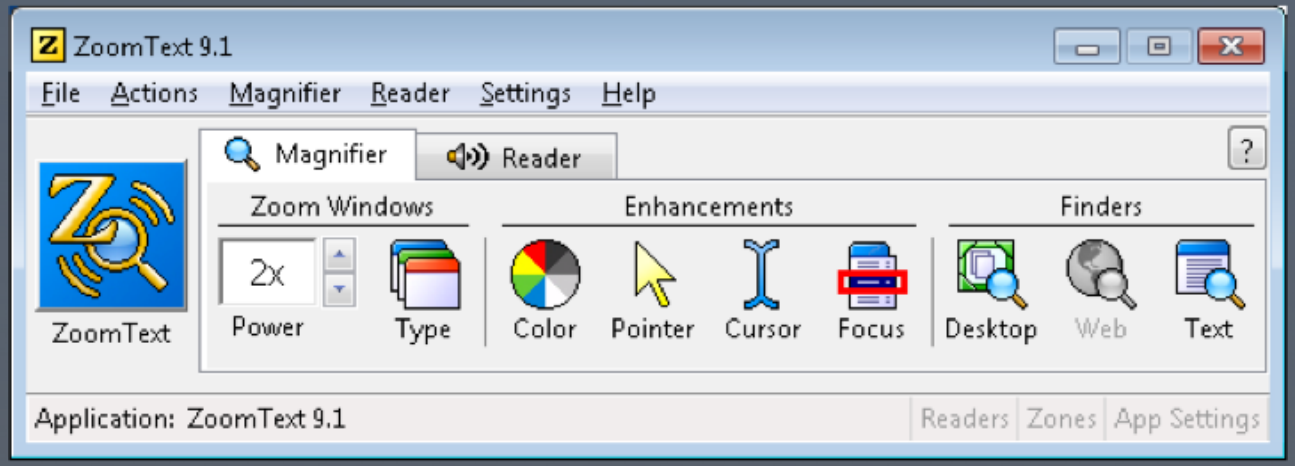
Screen readers

NVDA: Non-Visual Desktop Access
Open-source and free

Commercial software
Single license: USD $2,000

Assistive technologies

Screen magnifiers



Web accessibility
Basics

- Web accessibility refers to the inclusive practice of removing barriers that prevent interaction with, or access to websites, by people with visual/hearing/motoring/cognitive impairments
- When sites are correctly designed, developed and edited, all users have equal access to information and functionality

Web accessibility

WCAG 2.0

Web Content Accessibility Guidelines

(WCAG 2.0)

1. Perceivable
2. Operable
3. Understandable
4. Robust

https://www.w3.org/WAI/intro/wcag

Web accessibility
WCAG 2.0

	Live Content		Pre-Recorded Audio/Video Content					
WCAG 2.0 Level	Captions (audio/ video)	Captions (audio only content)	Captions	Transcript (audio-only content)	Media Alternative Transcript	Audio Description	Extended Audio Description	Sign Language Interpretation
A			✓	✓	✓ (or Audio Description)			
AA	✓		✓	✓		✓		
AAA	✓	✓	✓	✓	✓	✓	✓	✓

Web accessibility
UAAG 2.0

User Agent Accessibility Guidelines

(UAAG 2.0)

- Guidelines on how to make user agents accessible
- User agents include browsers, browser extensions, media players, screen readers, etc.

https://www.w3.org/WAI/intro/uaag

Web accessibility
ATAG 2.0

Authoring Tool Accessibility Guidelines

(ATAG 2.0)

- Guidelines on how to make authoring tools such as code editors accessible
- ATAG 2.0 lowers the barrier and provides support for people with disabilities to create more accessible Web content

https://www.w3.org/WAI/intro/atag

Web accessibility
ARIA

Accessible Rich Internet Applications (ARIA)

- A specification to make Web content more accessible to people with disabilities
- Especially helpful for dynamic content and advanced user interface controls developed using JavaScript

https://www.w3.org/WAI/intro/aria

Web accessibility
ARIA examples

- ARIA provides roles to
  - Describe the structure of the content such as header, main content, navigation and footer

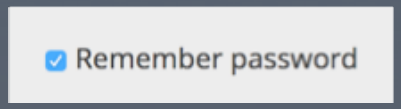
```
<div class="header" role="banner">…</div>
<div class="main" role="main">…</div>
<div class="nav" role="navigation">…</div>
<div class="footer" role="contentinfo">…</div>
```

  - Change the default behavior of element such as making an anchor text to be treated as a button

```
<a href="…" role="button">…</div>
```

Web accessibility
ARIA examples

- ARIA provides properties to describe the state widgets are in, for example
  - Use aria-checked to indicate the state of a checkbox




```
<!-- Set aria-checked to true (false) when the
     checkbox is (not) selected -->
<input type="checkbox" value="check" id="checkbox"
       aria-checked="true">
<label for="checkbox">Remember password</label>
```

  - Set aria-haspopup to true when submenu items exist in a menu item

Web accessibility
ARIA examples

- ARIA provides properties to define live regions that receive updates and the notification policy
  - When live regions get updated, screen reader users will be notified
  - The interruption priority from high to low is assertive and polite

```
<div>
  <label for="username">Username</label>
  <input type="text" name="username" id="username">
  <span id="username-error" aria-live="polite"></span>
</div>
```

Web accessibility
ARIA examples

- ARIA also provide the aria-hidden property to display or hide element from screen readers
  - When aria-hidden is set to true, the element is hidden from screen readers but visually displayed
  - Otherwise, the element is visible to both screen readers and normal users

```
<p aria-hidden="true">
  I am not visible to screen readers
</p>
<p aria-hidden=”false">
  I am not visible visually but I am visible for screen readers
</p>
```

Web accessibility
CSS accessibility support

- Some CSS properties provide accessibility support to display or hide element visually
  - When the display of an element is set to none, the element is hidden both visually and for screen readers
  - However, when the visibility of an element is set to hidden, the element is hidden only visually but it is pronounced by screen readers

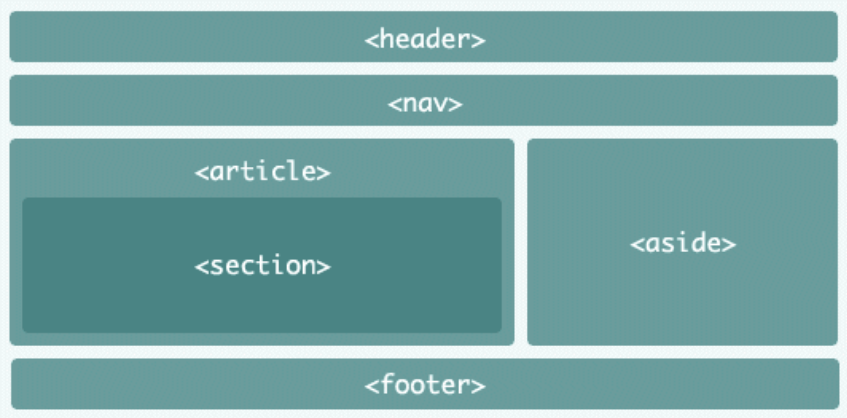
```
<p style="display: none">
  I am not visible both visually and for screen readers
</p>
<p style="visibility: hidden">
  I am not visible visually but I am visible for screen readers
</p>
```

Accessible design
Importance

- Well-structured content allows more efficient navigation and processing, which is essential for
  - People with cognitive disabilities
  - People using screen reader and keyboard
  - People using software to show only the main content
  - Improving search engine indexing result
- Support various devices such as laptops, tablets, smart phones, screen readers, etc.

Page structure
Semantic tags

- Mark up different regions with proper HTML5 semantic tags when possible



Region	HTML5 tag
Header	<header>…</header>
Navigation	<nav>…</nav>
Main content	<main>…</main>
Section	<section>…</section>
Article	<article>…</article>
Complementary content	<aside>…</aside>
Footer	<footer>…</footer>

Page structure
ARIA roles

- To maximize compatibility with old browsers, it is recommended to add ARIA roles to the corresponding elements

```
<!-- Use semantic tags when allowed -->
<header role="banner">…</header>
<main role="main">…</main>
<nav role="navigation">…</nav>
<footer role="contentinfo">…</footer>

<!-- Use this when semantic tags are not supported -->
<div class="header" role="banner">…</div>
<div class="main" role="main">…</div>
<div class="nav" role="navigation">…</div>
<div class="footer" role="contentinfo">…</div>
```

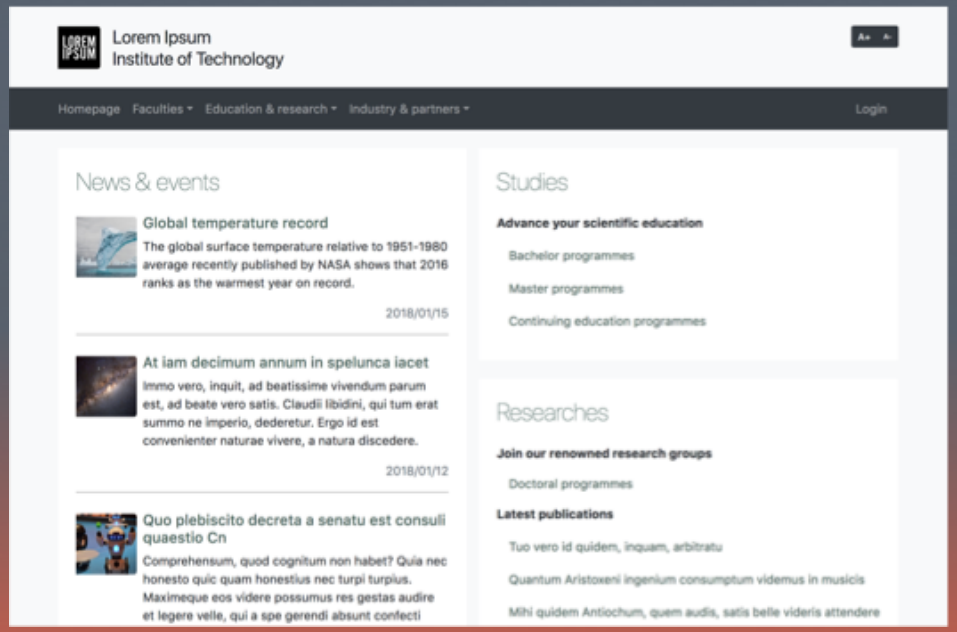
Page structure
ARIA labels

- Use aria-labelledby or aria-label to
  - Distinguish two regions of the same type
  - Change the default identification of page regions

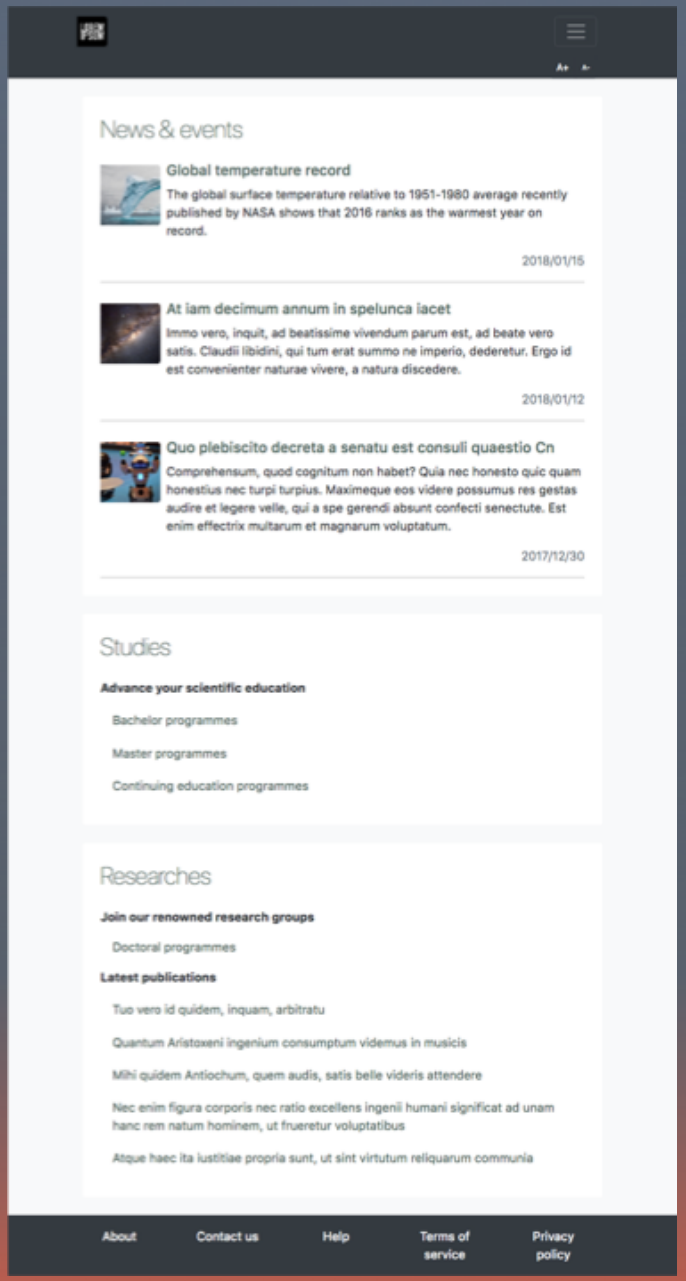
```
<!– Distinguish the two sections -->
<section aria-label="Advertisement">…</section>
<section aria-label="Side bar">…</section>

<!– Overwrite the default identification -->
<article aria-labelledby="article-title">
  <h1 id=”article-title">Introduction to Web accessibility</h1>
</article>
```

Page layout
Responsive design

- Provide responsive design for different types of devices



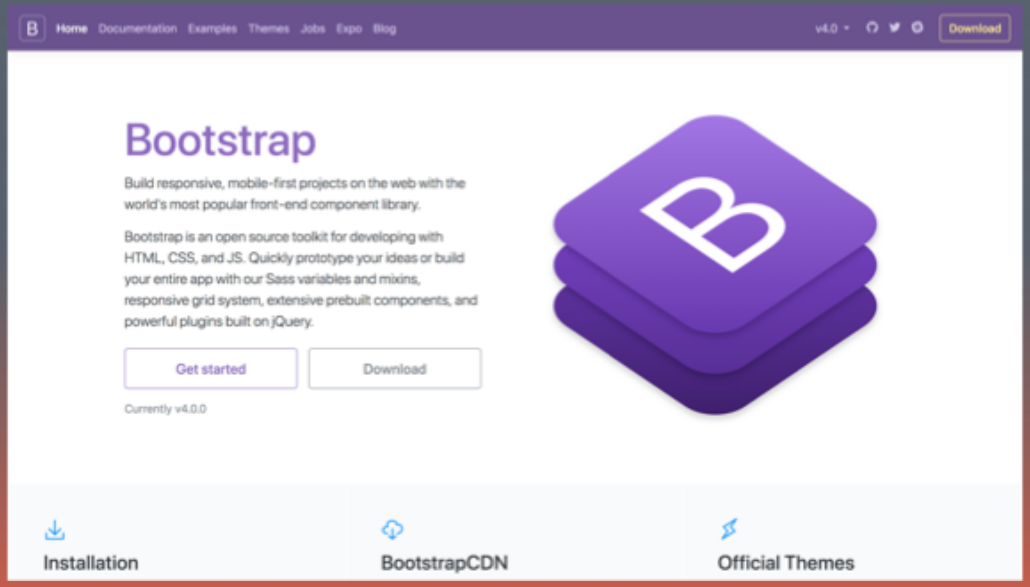
On wide screens



On narrow screens

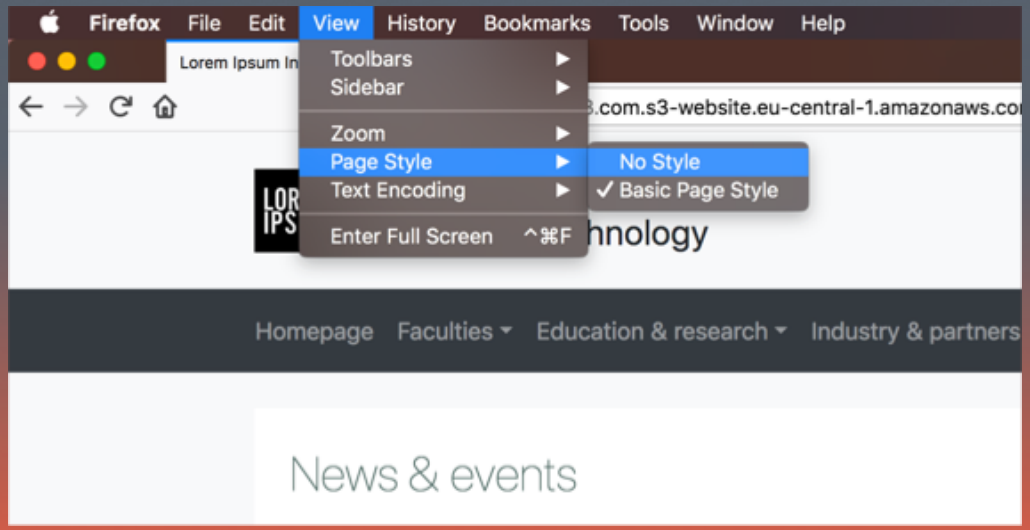
Page layout
Responsive design

- To maximize browser compatibility, consider using popular and well-tested responsive libraries such as Bootstrap
- https://getbootstrap.com/



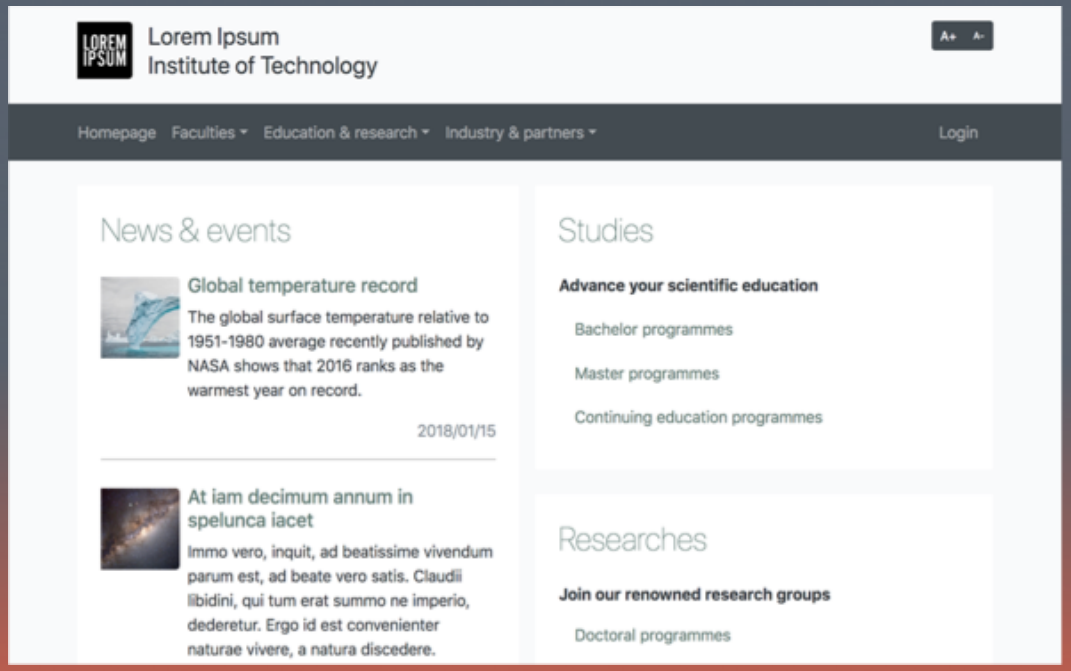
Page layout
No style

- The page should also function well when the user turns of the CSS, which is essentially how a screen reader sees the page
- To turn off CSS in Firefox, go to Menu > View > Page Style > No Style

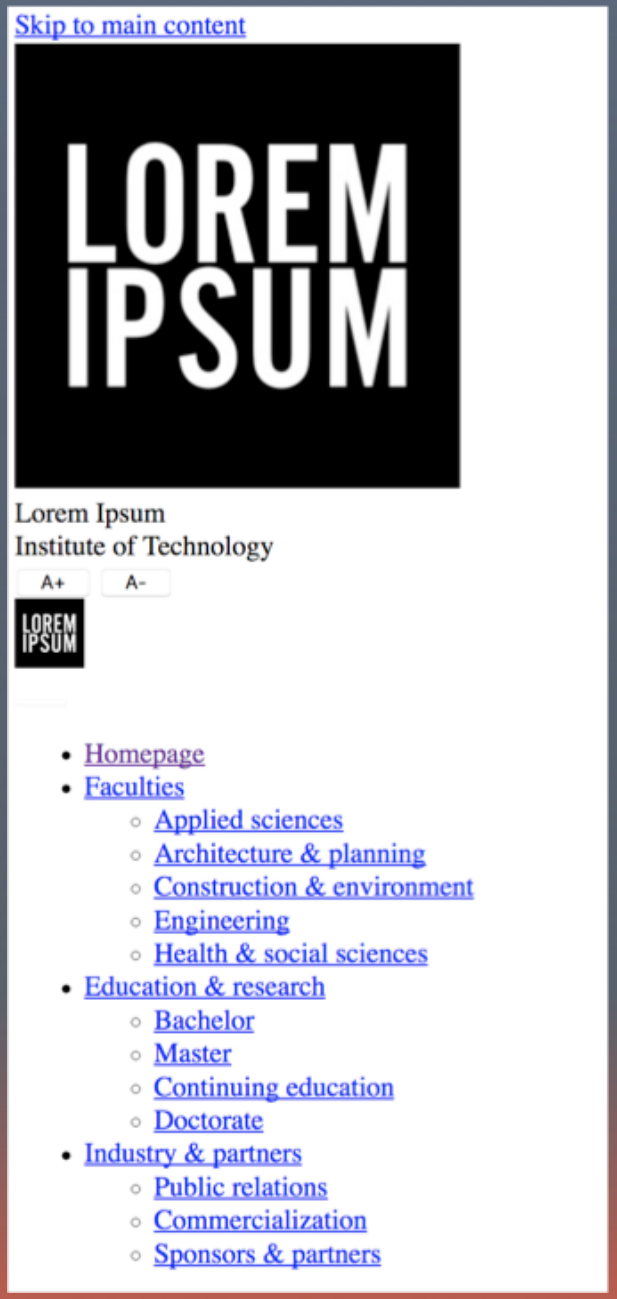


Page layout
No style

- Example of a page showing with/without CSS



CSS enabled

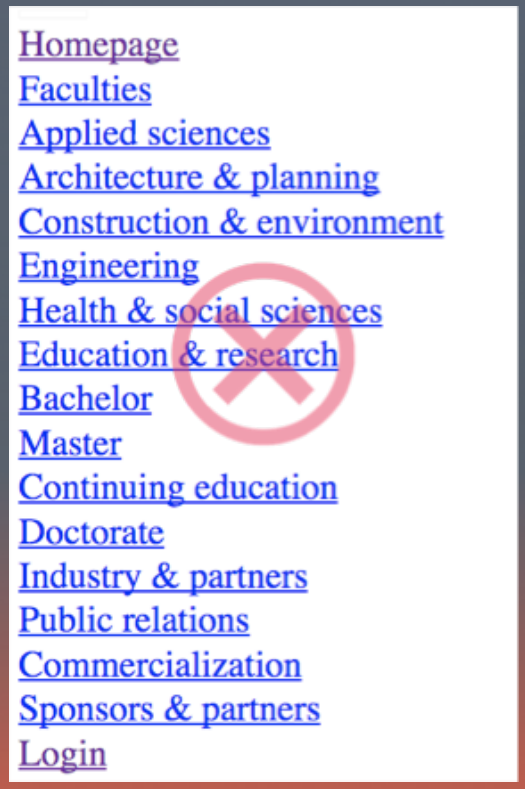


CSS disabled

Page layout
No style

- The examples below show the different between accessible and inaccessible design for menus

Accessible design



Inaccessible design

Page layout
Caveat on layout table

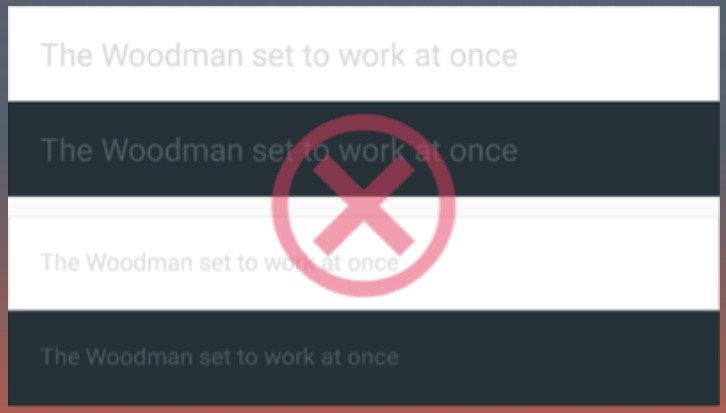
- Try to avoid using tables to layout a page since
  - Tables are not intended to be a layout tool, instead they are used to display relational data
  - Layout table source code is difficult to understand
  - Difficulty to incorporate modern responsive design techniques

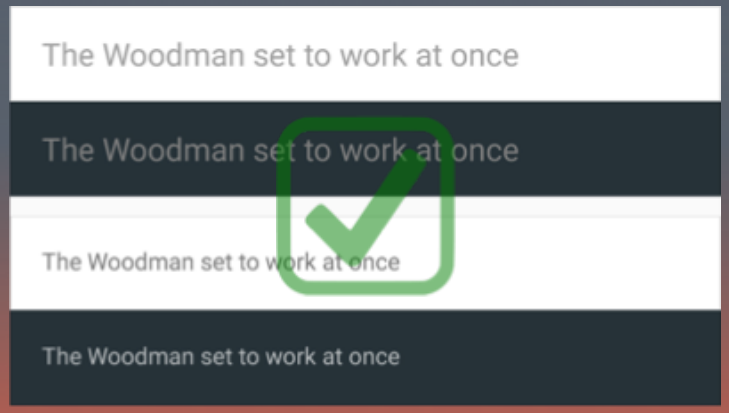
Color
Contrast

- Contrast ratio, ranging from 1 – 21, defines the difference in relative luminance between two colors
- The higher the ratio, the greater the difference
- Maintain minimum contrast ratio so that people with reduced vision can still read the content

Color
Contrast

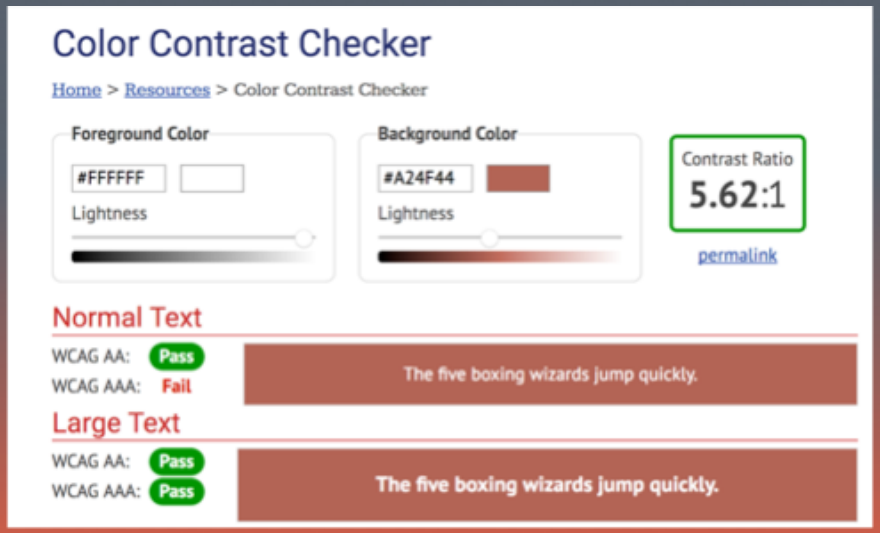
- Minimum 4.5:1 ratio against its background for small text
- Minimum 3:1 ratio against its background for large text (14+pt bold or 18+pt regular)





Color
Contrast

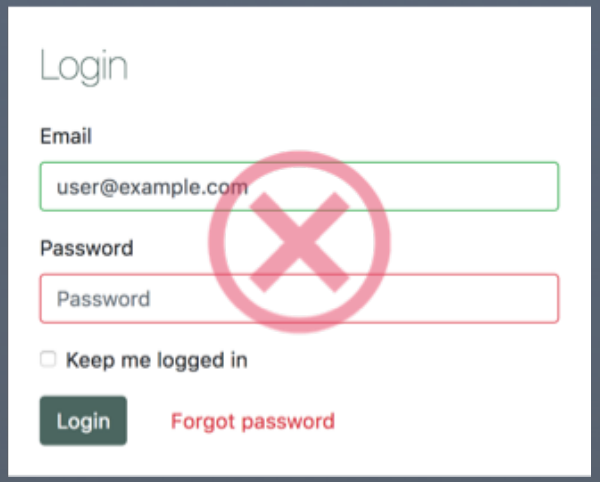
- Color contrast can be measured using the WebAIM tool by simply entering a foreground and background color in RGB hexadecimal format
- https://webaim.org/resources/contrastchecker/

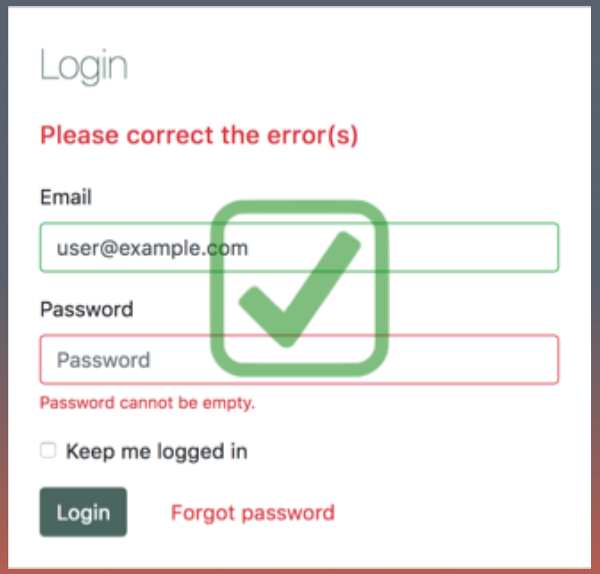


Color

Color palette

- Color blinded people may have difficulty to distinguish nearby elements when they are poorly colored
- Avoid relying only on colors to convey important information
- Include extra textual description, patterns, strokes or other indicators





Audio/video contents

- People with hearing disabilities may have difficulty to perceive audio content
- Provide captions, transcript or another visual alternatives to important audio/video elements and sound alerts

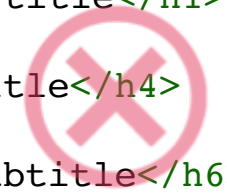
Accessible navigation
Importance

- Headings and navigation menus
  - Reflect the underlying structure of websites
  - Provide access to the essential functionality of an application
- Keyboard interoperability is essential for screen reader and keyboard users
- People with cognitive disabilities benefits from clear heading structure and navigation menus

Accessible navigation
Headings

- Use proper headings to communicate the organization of the content
  - Nest headings by their rank from h1 to h6
    - Heading with an equal or higher rank start a new section
    - Heading with lower rank start a new subsection
  - Avoid skipping heading ranks which causes confusion

```
<!-- Avoid skipping heading
levels like this -->
<article>
  <h1>Main title</h1>
  <p>…</p>
  <h4>Subtitle</h4>
  <p>…</p>
  <h6>Subsubtitle</h6>
  <p>…</p>
</article>
```



```
<!-- Use sequential heading
levels like this -->
<article>
  <h1>Main title</h1>
  <p>…</p>
  <h2>Subtitle</h2>
  <p>…</p>
  <h3>Subsubtitle</h3>
  <p>…</p>
</article>
```

Accessible navigation

Menu structure

- Use relevant semantic tags and list elements to convey the menu structure to users

```
<nav>
  <ul>
    <li><a href="…">Home</a></li>
    <li><a href="…">Departments</a></li>
    <li><a href="…">Studies</a></li>
    <li><a href="…">Researches</a></li>
  </ul>
</nav>
```

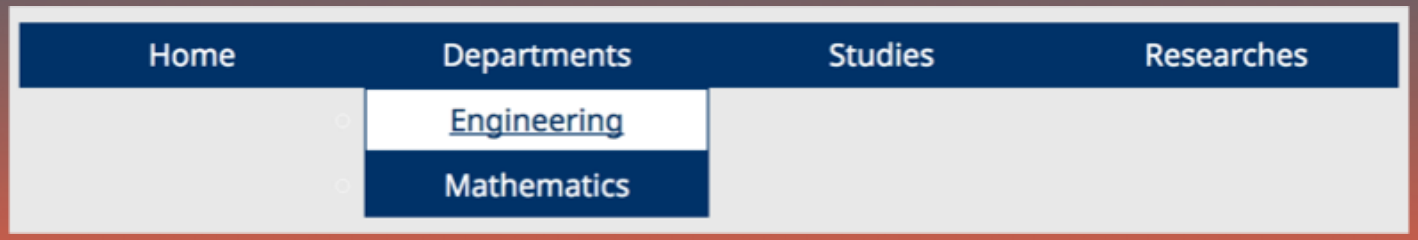
Accessible navigation
Menu styling

- Display menu at commonly expected locations such as
  - Vertically on the left (right)
  - Horizontally on the top
- Convey menu states using color and other styling options, e.g. invert the color and add an underscore when hovering or clicking on an item

Accessible navigation

Drop-down menu

- Consider drop-down design for sub-menus
- Indicate sub-menu items visually and update the following ARIA attributes dynamically for screen reader users
  - Set aria-haspopup to true on menu items with submenus
  - Set aria-expanded to true (false) on menu items when their submenus are expanded (collapsed)



Accessible navigation

Drop-down menu

```
<!– Sample HTML code of a drop-down menu -->
<nav aria-label="Main Navigation">
  <ul>
    <li><a href="…">Home</a></li>
    <li>
      <!– Set aria-haspopup to true and aria-expanded to false -->
      <a id="departments" href="…" role="button"
         aria-haspopup="true" aria-expanded="false">Departments</a>
      <!– Set display to none to hide menu items initially -->
      <ul style="display: none;">
        <li><a href="…">Engineering</a></li>
        <li><a href="…">Mathematics</a></li>
      </ul>
    </li>
    <li><a href="…">Studies</a></li>
    <li><a href="…">Researches</a></li>
  </ul>
</nav>
```

Accessible navigation

Drop-down menu

```
// Sample JavaScript code to dynamically show/hide drop-down menu
document
  .querySelector('#deparments a') // Select the menu element
  .addEventListener('click', function(event) { // Add event listener
    if (this.parentNode.style.display === 'none') {
      // Open the menu if it is closed
      this.parentNode.style.display = 'block’;
      this.setAttribute('aria-expanded', "true");
    } else {
      // Close the menu it if is open
      this.parentNode.style.display = 'closed';
      this.setAttribute('aria-expanded', 'false');
    }

    event.preventDefault(); // Prevent default event handling
}, false);
```

Accessible navigation
Skip links

- Skip links are useful for screen reader users to jump directly to certain regions in the same page without going through all the content manually

```
<body>
  <!-- Set visibility to hidden to hide it from normal users -->
  <div style="visibility: hidden">
    <!-- The anchor points to the id of the element -->
    <a href="#main" >Skip to main content</a>
    <a href="#footer">Skip to footer</a>
  </div>
  <nav>…</nav>
  <main id="main">…</main>
  <aside id="advertisement">...</aside>
  <footer id="footer">...</footer>
</body>
```

Accessible forms

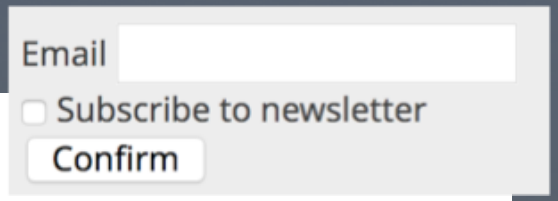
Importance

- Forms are used to perform important actions, e.g. login, registration, purchasing, etc.
- Forms should be easy to understand even for people with cognitive disabilities
- Form controls should be properly labelled to provide access to screen reader users

Accessible forms

Form control labelling

- Use label to explicitly describe the purposes of all form controls including inputs, check boxes whenever possible

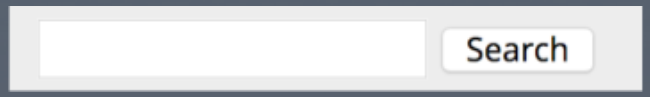



```
<form>
  <div>
    <label for="email">Email</label>
    <input type="email" name="email" id="email">
  </div>
  <div>
    <input type="checkbox" name="subscribe" id="subscribe">
    <label for="subscribe">Subscribe to newsletter</label>
  </div>
  <div>
    <!-- The label of a button is set inside the element directly -->
    <button>Confirm</button>
  </div>
</form>
```

Accessible forms

Form control labelling

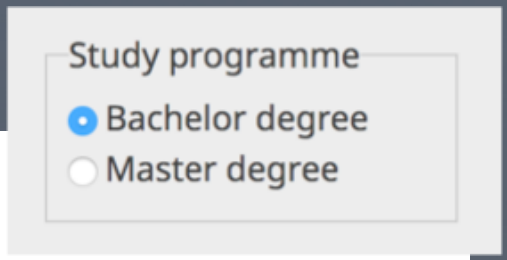
- When the purpose is clear from the context, use hidden label, aria-label or aria-labelledby to describe the purpose to screen reader users




```
<!-- Visibility hidden elements are still visible to screen readers -->
<label for="search" style="visibility: hidden">Search</label>
<input type="text" name="search" id="search">
<button type="submit">Search</button>

<!– Describe the  purpose using the content of aria-label -->
<input type="text" name="search" aria-label="Search">
<button type="submit">Search</button>

<!– Describe the purpose using the element pointed to by aria-labelledby -->
<input type="text" name="search" aria-labelledby="search-button">
<button id="search-button" type="submit">Search</button>
```

Accessible forms
Related control grouping

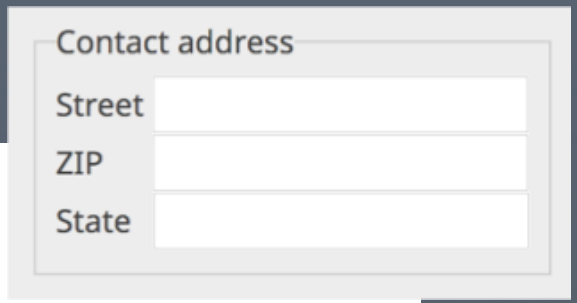
- Use fieldset to group related form controls and legend to describe the purpose of the group




```
<fieldset>
  <legend>Study programme</legend>
  <div>
    <input type="radio" name="degree" id="bachelor" value=”BA">
    <label for="bachelor">Bachelor degree</label>
  </div>
  <div>
    <input type="radio" name="degree" id="master" value=”MA">
    <label for="master">Master degree</label>
  </div>
</fieldset>
```

Accessible forms
Related control grouping

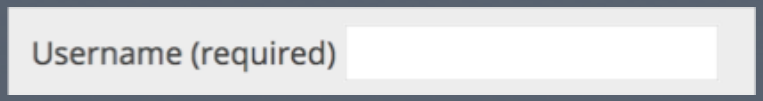
- Another example of using fieldset and legend




```
<fieldset>
  <legend>Contact address</legend>
  <div>
    <label for=”street">Name</label>
    <input type="text" name=”street" id=”street">
  </div>
  <div>
    <label for=”zip">ZIP</label>
    <input type=”text" name=”zip" id=”zip">
  </div>
  <div>
    <label for=”state">State</label>
    <input type="text" name=”state" id=”state">
  </div>
</fieldset>
```

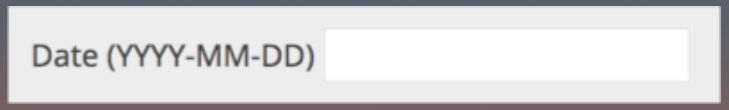
Accessible forms
Form control instructions

- Provide instructions to help users understand how to complete the form correctly such as
  - Whether the input is required or not




```
<label for="name">Username (required)</label>
<input type="text" name="username"
       id="username" required aria-required="true">
```

  - Expected format of the input




```
<label for="date">Date (YYYY-MM-DD)</label>
<input type="text" name="date" id="date">
```

Accessible forms
Form input validation

- Validate user input to avoid mistakes
- Use a heading element to notify the existence of errors
  - Set the aria-live attribute to polite or assertive so that the screen reader notifies the user about the error
- Add detailed error message to each invalid input
  - Put the error message inside an anchor element, which points to the corresponding input control so that the user can be redirected to the control with simply a click

Accessible forms

Form input validation

```
<!– Sample HTML code of a form -->
<div>
  <h1>Login</h1>
  <!– The error heading is updated using JavaScript -->
  <h2 id="login-error" aria-live="assertive"></h2>
  <form id="login-form" novalidate>
    <div>
      <label for="login-email-control">Email</label>
      <input id="login-email-control" type="email"
             placeholder="Email" required
             aria-required="true">
      <!– The error message is updated using JavaScript -->
      <small id="login-email-error"></small>
    </div>
    ...
  </form>
</div>
```

Accessible forms

Form input validation

```
// Sample JavaScript code to validate a form
function validate() {
  var hasError = false;
  var emailControl = document.getElementById('login-email-control’);
  var emialError = document.getElementById('login-email-error');
  if (!emailControl.validity.valid) {
    // Add detailed error message
    emialError.innerHTML = 'Please correct the errors';
  } else {
    // Remove detailed error message
    emialError.innerHTML = '';
  }
  // More validation logic omitted here
  …

  var errorHeading = document.getElementById('login-error');
  if (hasError) {
    // Add error notification
    errorHeading.innerHTML = 'Please correct the errors';
  } else {
    errorHeading.innerHTML = ''; // Remove error notification
  }
}
```

Accessible forms

Tab ordering

- Tab ordering follows the native HTML structure
- CSS only changes the visual display but not the tab ordering
- In the right example,
  - The tab ordering is day, year and month based on HTML
  - The elements are however displayed as year, month and day based on CSS


```
<fieldset>
 <legend>Birthday</legend>
 <div style="float: right;" >
   <label for="day">Day</label>
   <input type="number" id="day">
 </div>
 <div style="float: left">
   <label for="year">Year</label>
   <input type="number" id="year">
 </div>
 <div style="float: left;" >
   <label for="month">Month</label>
   <input type="number" id="month">
 </div>
</fieldset>
```

Accessible images
Importance

- Images and graphics
  - Make content more pleasant and appealing
  - Convey complicated information such as stock price trend visually and concisely
- Accessible images are useful for
  - People using screen readers
  - People browsing speech-enabled websites
  - Mobile web users who turn off image loading
  - Search engine optimization

Accessible images
Decorative images

- Use a null alternative text for decorative images, e.g. borders, spacers, and corners, since they add no information to the content
- Do not remove the alt attribute because some screen readers will announce the image filename

```
<div>
  <!-- Decorative image -->
  <img src="./top-border.png" alt="">
  <p>…</p>
  <!-- Decorative image -->
  <img src="./bottom-border.png" alt="">
</div>
```

Accessible images

Informative images

- Informative images convey a simple concept that should be expressed in a short sentence using the alternative text


```
<div>
  <span>
    <img src=”./phone.png" alt="Telephone"> 0123 456 7890
  </span>
  <span>
    <img src=”./fax.png" alt="Fax"> 0123 456 7891
  </span>
</div>
```

Accessible images

Informative images

```
<div>
  <img src="./mountain.jpg"
       alt="Photo of the Alps in Winter">
</div>
```

Accessible images
Functional images

- Functional images are used to initiate actions
- Mostly used in buttons and links
- The alternative text should describe the purpose the image, i.e. the action that will be initiated

```
<div>
  <a href="#" role="button" onclick="...">
    <img src="./print.png" alt="Print this page">
  </a>
</div>
```

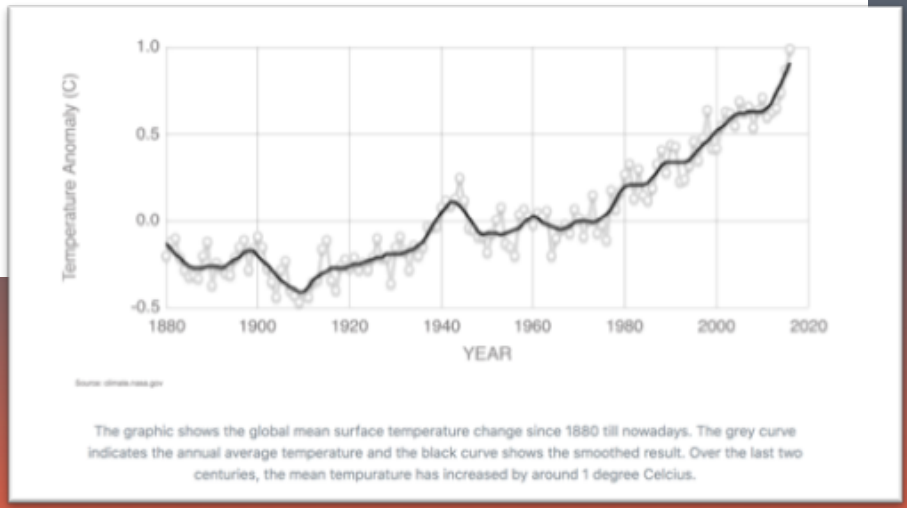
Accessible images
Complex images

- Complex images contain substantial information
  - Graphs and charts such as organizational charts
  - Diagrams and illustrations showing map locations and weather information
- Use a two-part alternative text to improve accessibility
  - A short title to summarize the image
  - A long description of the image using a paragraph, a table or the new figcaption option from HTML5

Accessible images

Complex images

```
<!-- Use aria-describedby to associate the description with the image -->
<img src="./temperature.jpg"
     alt=”Line chart of global average annual temperature"
     aria-describedby="description">
<p id="description">The graphic shows…</p>

<!-- Use figcaption to associate the description with the image -->
<figure role="group">
  <img src="./temperature.jpg"
       alt="Line chart of global average annual temperature">
  <figcaption>
    <h2>Overview</h2>
    <p>The graphic shows…</p>
    <h2>Values</h2>
    <table>…</table>
  </figcaption>
</figure>
```



Accessible tables
Importance

- Tables are excellent choice to organize relational data in grids
- Accessible tables need specific HTML markups to differentiate header and data cells, and to define their relationship, which is essential for
  - People using screen readers
  - People using alternative ways such as custom stylesheet to render the data

Accessible tables
Basics

- Header cells must be marked up using the th tag
- Data cells must be marked up using td tag
- For complex tables, explicit associations of header and data cells may be needed using scope, id, and headers attributes
- The caption functions as a heading for a table, which helps user to identify the most important information

Accessible tables

Only row header

```
<table>
  <caption>Events</caption>
  <tbody>
    <tr>
      <!-- Use th for header cells -->
      <th>Date</th>
      <th>Event</th>
      <th>Location</th>
    </tr>
    <tr>
      <!-- Use td for data cells -->
      <td>12 February</td>
      <td>Web accessibility workshop</td>
      <td>Main Hall</td>
    </tr>
    <tr>
      <td>24 March</td>
      <td>Mobile web development workshop</td>
      <td>West Wing</td>
    </tr>
  </tbody>
</table>
```

Events

Date	Event	Location
12 February	Web accessibility workshop	Main Hall
24 March	Mobile web development workshop	West Wing

Accessible tables

Only column header

```
<table>
  <caption>Events</caption>
  <tbody>
    <tr>
      <th>Date</th>
      <td>12 February</td>
      <td>24 March</td>
    </tr>
    <tr>
      <th>Event</th>
      <td>Web accessibility workshop</td>
      <td>Mobile web development workshop</td>
    </tr>
    <tr>
      <th>Location</th>
      <td>Main Hall</td>
      <td>West Wing</td>
    </tr>
  </tbody>
</table>
```

Events

Date	12 February	24 March
Event	Web accessibility workshop	Mobile web development workshop
Location	Main Hall	West Wing

Accessible tables

Row and column headers

- When both row and column headers exist in a table, the relationship between the header and the data cells become ambiguous
- Use the scope attribute to denote whether a header applies to the entire row or column

Accessible tables

Row and column headers

```
<table>
  <caption>Opening hours</caption>
  <tr>
    <td></td>
    <th scope="col">Monday - Friday</th>
    <th scope="col">Saturday - Sunday </th>
  </tr>
  <tr>
    <th scope="row">09:00 - 11:00</th>
    <td>Open</td>
    <td>Open</td>
  </tr>
  <tr>
    <th scope="row">13:00 - 17:00</th>
    <td>Open</td>
    <td>Closed</td>
  </tr>
</table>
```

Opening hours

	Monday - Friday	Saturday - Sunday
09:00 - 11:00	Open	Open
13:00 - 17:00	Open	Closed

Accessible tables

Column and row groups

- Header that spans multiple columns should have the scope attribute set to colgroup
- The span of the column group can be set using the attribute colspan
- A column group is defined using the colgroup element

Accessible tables

Column and row groups

- Header that spans multiple row should have the scope attribute set to rowgroup
- The span of the column group can be set using the attribute rowspan
- A row group is defined by the thead, tfoot or tbody element
  - The thead and tfoot can be used once in a table
  - The tbody elements can be used multiple times with each one defining a row group

Accessible tables

Column and row groups

```
<table>
  <caption>Opening hours</caption>
  <thead>
    <tr>
      <th scope="col">Days</th>
      <th scope="col">Time</th>
      <th scope="col">Status</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th rowspan="2" scope="rowgroup">Weekday</th>
      <th scope="row">09:00 – 11:00</th>
      <td>Open</td>
    </tr>
    …
  </tbody>
  <tbody>
    <tr>
      <th rowspan="2" scope="rowgroup">Weekend</th>
      …
</table>
```

Opening hours

Days	Time	Status
Weekday	09:00 - 11:00	Open
	13:00 - 17:00	Open
Weekend	09:00 - 11:00	Open
	13:00 - 17:00	Closed

Testing for accessibility
Importance

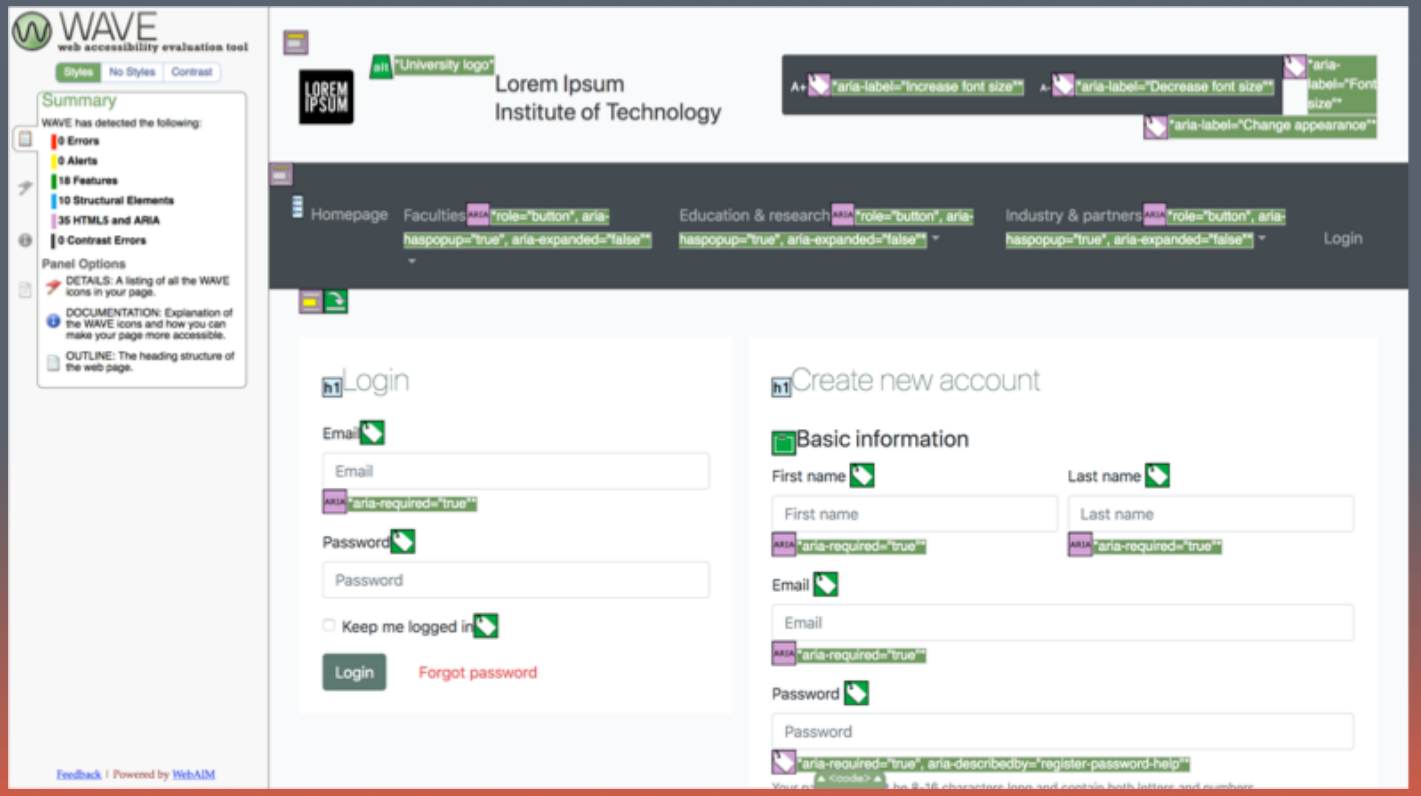
- Test-driven development is fundamental to all kinds of software development
- Testing for accessibility can be achieved in many ways:
  - Software such as the Web Accessibility Evaluation Tool (WAVE) can help to identify some error and non-standard practice
  - Testing with real users with disabilities can help to reveal special usage patterns and wrong interaction assumptions

Testing for accessibility

WAVE extension

- WAVE is available as a Firefox browser extension

https://wave.webaim.org/extension/



Testing for accessibility

WAVE extension

- Bad accessibility practices are shown as errors and alerts