Introduction

This set of tutorials complements the lecture slides to equip you with some good practices for building modern accessible Web content. The instructions on this page will show you how to setup your local environment so that you can start implementing accessibility features and loading the page in your browser for testing.

Note that there is not unique solution to implement an accessible feature, so when solving the exercise problem, try to be creative in coming up alternative solutions.

Setup

- *Decompress the zip file **student_without_solutions.zip** (_MacOS or _Windows, depending on what you are using) into one of your hard-drives.*
- *Inside the **student** folder, you will find the following sub-folders:*
  - *The folder **installable** contains the software packages to be used during development and testing phases. Specifically, the software we use include:*
    - ***Chrome** to load and render the sample Web pages.*
    - ***JAWS** to experiment with screen readers (Note: for MacOS users, use VoiceOver instead of JAWS; VoiceOver is free and is usually pre-installed on Macs)*
    - ***Zoomtext** (for Windows only)*
    - ***Python** (version 3) to setup a localhost that serves the sample Web pages.*
  - *The folder **src** contains the source code of the sample Web page.*
  - *The folder **handout** contains the slides and exercises (currently empty)*
- *Please make sure you either have the software mentioned before already available on your computer or install them before proceeding to the next steps.*
  - *To install the WAVE Accessibility Plugin, you need to first open your Chrome browser. Then visit https://chrome.google.com/webstore/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh, click on "Add to Chrome" and follow the instruction to complete.*
  - *For the other software packages, please use the files in the **installable/** folder or inside the **installable/win32 folder** if you have a 32-bit Windows system, or the files inside the **installable/win64** folder if you have a 64-bit Windows system.*

Start the localhost

- *Open your command line tool and change the current directory to the folder **student/src***
- *To start the local server, enter the following command, we assume **python** is on your execution path*

```
python -m http.server 8000
```

- *Now you can open Chrome browser, and the sample Web page should be available at the following url: localhost:8000/index.html*

Exercise 1 – Explore the sample Website

Now that you have your environment set up. Please take a look at the sample Website and answer the following questions:

- *Interact with different elements and navigate to different pages on the sample Website,*
  - *Try to find features that do not work as expected.*
  - *Try to find features that are implemented but not very accessible.*
  - *Discuss with your colleagues about your findings and try to propose some solutions or improvements.*
- *Now turn of the CSS feature from the browser. Navigate to different pages and try to find structures that have changed compared to when CSS is enabled. Do you think those changes are good or bad? Why?*

Exercise 2 – Accessible design

Accessible design is essential not only for people with disabilities, but also for normal people to efficiently gather and process information on a Web page.

Contrast ratio

Can you change the contrast ratio of the content so that they become more legible?

Delete the body's attribute color #dfdfdf. So the constrast ratio will be bigger to make it more legible.

Page regions

This task practices your understanding of using HTML5 semantic tags or ARIA roles to define different regions of a Web page.

*For the homepage, the HTML code is inside the **index.html** file. After opening it in your favorite code editor, try to identify different regions in this page and use appropriate landmarks to mark them. How many regions, e.g. header, navigation, etc. are there in the Web page?*

*You can repeat the same process for the **login.html**, **article.html** and **empty.html** files.*

Add HTML5 semantic tags: header/nav/main/footer; and their ARIA roles: banner/navigation/main/contentinfo. In article.html part, we specify the main part into article, section and some other small semantic tags and corresponding ARIA roles.

Accessible font size

Have you already tested the two buttons on the top of the of Web page where you can increase or decrease the font sizes of the whole page? Does it work as expected? If not, can you implement the JavaScript that will dynamically scale all the font sizes?

Hint: You need to first create an event listener that watches the click events on the buttons. Since the font size of the whole page is defined relative to font size of the root html element, you can achieve the result by updating the font size property of the root html element.

We add the watched events in the index.js, we add two click events seperately for A+ and A- buttons

What screen reader will announce for these two buttons? Is this good accessibility practice? If not, can you propose an improvement so that the purposes of these two buttons become clearer using ARIA support?

We use the ARIA labels to make the purposes more clearly for these two buttons. If you use the voiceOver, you could hear the description we add for them.

Reading order (screen reader)

*Open the **login.html** file in your code editor and navigate to the HTML code for the login form. Also load this page in your browser. Compare the HTML code carefully against the actual layout, can you find something*

counter intuitive? If you turn off the CSS, do you see any difference? If you use a screen reader, what would the reading order be?

Can you propose a solution to improve the HTML code so that the HTML code structure better reflects what the user sees?

There are some thing hidden, not showing all the text content at first render. The invalid-feedback shows at the first line of these form, and it doesn't show the detail which error leading this invalid submit
After turning off the CSS, all the contents show now.
The hidden texts are not read. After submit error occurring, the error message will be read at first and give no instructions why the error appears. And the label will be read after the textareas. It's not proper.

I change the position of label HTML dom, and seperately add the two textareas' error messages under its textarea. Thus, the voiceOver can read the HTML structure in order. And the error mention can be more clear.

Exercise 3 – Accessible navigation

This exercise focuses on building headings and navigation menus that are accessible. If you need a review on the relevant topics, please refer to the accessible navigation section of the slides.

Headings

*Open the **index.html** file in your code editor, can you find any headings in the source code? Do you think if it is good accessibility practice or not? Why?*

If you view the same page in your browser, can you visually identify contents that are emphasized with enlarged font size or bolded font face? Find the content in the source code, can you propose and implement changes so that those emphasized contents are identified using appropriate heading hierarchy?

We replace some apparent title's HTML elements to <h1>,<h2> elmemts. Thus, the heading hierarchy will be more clear. Because the default style of h1 and h2 including the biger fontsize or bold font style. So it makes the browser show structural outlook.

Article and their titles

Look at the index page in your browse. You will find three articles in the news & events section. In the corresponding HTML code, each article is enclosed inside the generic div element. Can you make improvement to this by switching the div elements to HTML5 sematic tags?

We change the three div into article tags.

Imagine if you are a screen reader user who is looking at the three articles, what would the screen reader announce to the user? Can you make the user experience better so that the screen reader announces the artilce title already when the user is on the article, instead of requiring the user to navigate to the title before the announcement is made?

We add the title attribute in the article tags, and the value is the article's title. Thus, screen reader will announces the title first when user is on the article.

Menu structure

*Open the **index.html** file in your code editor, pay attention to the HTML code for the navigation menus. Do you think the generic div elements are good enough to convey the menu structure? Are there other HTML elements that are better at indicating sub-menu hierarchy?*

It is not a good way to use the generic div elements, because there is lack of some structure and not clear for some disabled people. So we could use <nav>,<ul>, tags to replace the single type.

Drop-down menu

When you click on a menu item, if it has a sub-menu and the sub-menu is not displayed, the sub-menu would be shown visually. If you click the same menu item again, the sub-menu would be collapsed. Use the developer tool in Chrome to inspect the updates to the HTML code while the sub-menu displays or collapses. Do you think a screen reader user will be updated about the change of status, i.e. whether the sub-menu is displayed or not?

We use the aria-expanded and display these two attributes to make the screenreader to read the state of drop-down menus. JS code we write is in the index.js.

If not, try to fix the menu s.t. the screenreader will read "menu", "expanded/collapsed" correctly.

Menu keyboard interaction

It is common practice for keyboard users to use the ESC key to cancel an operation. Therefore, it would be a user experience enhancement if an opened sub-menu gets closed automatically when the user presses the ESC key. After closing, the menu header should be focused. Can you implement this feature by using JavaScript to listen to keyboard events on sub-menu and closes it if it is open?

Code in index.js

Furthermore, the menu items are currently implemented using anchor texts, which for keyboard user are only activated by pressing the ENTER key. To improve the user experience, it would be ideal if the menu items can also be activated by pressing the SPACE key, which is the default for button types.

Code in index.js

When navigating outside an open menu with TAB, the now inactive menu should be closed as well. Can you also implement this feature by using JavaScript to listen to the TAB event and determine when a menu should be closed?

Code in index.js

Skip links

Skip links provides shortcuts for screen reader users to jump to a section of the page quickly. Can you implement this feature?

We program for it in index.html and index.css, you could check it with VoiceOver

Exercise 4 – Accessible forms

Forms are critical to enable interactions between users and the Website. In this section, we will move onto the login page, where it contains to two forms, and try to enhance its accessibility features.

Form control labelling

When rendered in the browser, all input controls have their corresponding descriptions displayed visually. Now open the ***login.html*** *file in your code editor and exam the source code carefully. Do you think the form labels are implemented and associated with the corresponding input controls correctly? Would a screen reader user experience any difficulty when interacting with these forms? Can you improve the situation?*

We think the labels should be label tag instead of p tag for inputs. Thus, the structure and meaning will be more clear. Before we changes the tags, screen readers will feel challenge when the forms be read. But after we use label tags, the labels associate with the corresponding input correctly. Except the label tags, we also could choose aria-label or aria-labelledby to describe the purpose to screen reader users

Related control grouping

Look at the create new account form where the input controls are divided into two groups, i.e. basic information and additional information. Can you update the HTML source code so that the two groups are grouped using the proper element instead of the generic div elements?

We could choose fieldset and legend to implememt the related control grouping.

Form input validation

If you leave the login form empty and click the login button, do you see any indication of errors? Based on your knowledge, are those features sufficient to notify the user about his/her mistake? Can you introduce some improvement to the form input validation?

Similarly, examine the account creation form and introduce changes or improvements whenever possible.

We use a heading element to notify the existence of errors. Set the aria-live attribute to assertive so that the screen reader notifies the user about the error. Then, we add detailed error message to each invalid input. To achieve different validation regular, we use js' condition statement to decide the different error situations.

code in login.js

Exercise 5 – Accessible images

Images are very important to modern Web applications, no matter they are for aesthetic reasons to just decorate the pages, or for functional or even informative purposes to present information in a concise way. Therefore, accessible images are of critical importance in order to make the Web application more inclusive. In this exercise, we will take a look at the images of the sample Website and you will have the opportunity to make them more accessible.

Informative images

Informative images are used to convey a simple concept. Common example are logos of a brand, a scene picturing the context of an article and so on. Please identify all the informative images in the sample Website. After inspecting the corresponding HTML source code, please answer the following questions. Are these images accessible to people using screen readers? If not, what changes would you like to make to resolve the issue?

We could use img tag's alt attribute to describe the purpose the image. And we add the aria-label to the img, thus the screen reader could read the brief content of this img.

Complex images

Compared to informative images, complex images usually contain substantial amount of information. Examples of informative images are line chart depicting the average price of the real estate market over time, a pie chart showing the percentage of male and female students enrolled in a university and so on. Because of the complexity of such image, adding a detailed description about them is essential of screen readers since technically screen readers cannot just look at an image and describe its content.

*In the **article.html** file, we have introduced an image showing the evolution of global average temperature over the last two centuries. Can you add accessibility features to make this part more accessible? There are several ways to implement the solution, which one do you use? What are the pros and cons?*

1. Use aria-describedby to associate the description with the image
pros:
a. brief to explain
b. easy to code
cons:
a. structure are not very clear
2. Use figcaption to associate the description with the image
pros:
a. good structure to explain this picture
b. fit for more complex figure
con:
a. a little bit difficult to code
b. if the figure is not complex enough, this way will cost more than it gets.
Compare with above 2 methods and consider our figure, we choose use the first one.

Exercise 6 – Accessible tables

Tables are excellent choices to represent relational data in grids. In order to make them also accessible to visually impaired people, we need to add extra attribute to the table markups so that they become legible to screen readers.

Header cells vs data cells

Open the article page in your browser and scroll down to the table containing samples of average global temperature for the last two centuries. Can you identify which rows (columns) belong the table header while which rows (columns) belong to the table data? If you inspect the HTML code of ***article.html*** *in your code editor, are the header cells and data cells marked up correctly? If not, how can you fix it?*

In my code editor, the table header and table data cannot identify differently. So we should change the table head's and some table body's <td> tags to <th> tags. And change the table head <tbody> to <thead>. And add th's attribute "scope".

Column and row groups

In the table about average global temperature, there are some headers that span multiple rows (columns) but the corresponding scopes are not correctly established. Based on the slides containing column and row groups, can you change the HTML code?

We change the rows'(columns') which span multiply scope='col(row)' to scope='col(row)group'. Thus, the corresponding scopes can be built correctly. The concrete code can be seen in article.html.

Exercise 7 – Accessibility test

Congratulations on being creative and completing all the exercises! As we have discussed, testing is an important aspect for all kinds of software development. So now is time for us to test the feature you have just implemented

Use the WAVE accessibility tool to evaluate the accessibility features of the improved Website. Comparing your finding carefully against the evaluation result, have you overlooked some issues? Or have you found something that is not reported by the plugin?

Close the WAVE accessibility tool, now try to reload the page with CSS disabled. Is the page layout behaving the same as you expected?

Lastly, try to use a screen reader (JAWS or VoiceOver) to access the Website. Do you encounter any difficulty?

Take some time to summarize your findings, if there is still space for improvement, go back and make further changes.