Accessiblity Part 1

1. Who benefits from accessibility?

   keyboard only, color blind, and screen reader

2. How would you define inclusive and/or universal design?
   Inclusive design focuses on exploring ways of serving a full spectrum of people who make up a diverse market.
   In comparison, universal design serves the broadest range of users possible, rather than trying to address individual accessibility or inclusion objectives.

Techinical

1. What methods can you use to find an element’s accessible name

   The accessible name for a HTML element is exposed in the browser’s accessibility tree.

   - Element’s content
   - Atrribute, like `alt` in img
   - Associated element, The `for` attribute on the `<label>` and the `id` attribute on the `<input>` share the same value.
   - ARIA, `` `aria-label` `aria-labelledby ``

   Tools:

   - WAVE: expose accessibility information through the page.
   - Chrome dev to inspect under HTML
   - Screen Reader

2. What is the accessibility tree?

   The accessibility tree contains accessibility-related information for most HTML elements.
   The accessibility tree and the DOM tree are parallel structures.

   Can check it with Chrome browser.

3. Why are rems or ems preferable to pixels for setting type size?

   `rem` and `em` will be easy for making font size adjustments from browser settings or OS.

4. Why is it important to allow the viewport to be resized, and/or zoomed?

   Everyone can zoom the text to a readable size, and the content stays within the viewport.

5. How is the `title` attribute exposed to assistive technologies?

   Don't use them, except in special circumstances. It is problematic because it is not well supported in some crucial respects.

   Can be used by `iframe`

6. Provide an example of when you might need to add a description to an element.

   A label provides essential information about an object, while a description provides extended information that the user might need.

7. What is a focus trap, or focus trapping?

   Modal, tabbing while focused on the last form element will send us all the way back to the main document.

8. Describe a situation where the `tabindex` attribute would be useful.

   You can add the **tabindex** attribute to any element like this:

   ```
   <div tabindex="0">I'm focusable</div>
   ```

   Tabindex values

   1. a **negative** value means that the element should be focusable, but should not be reachable via sequential keyboard navigation;
   2. **0** means that the element should be focusable and reachable via sequential keyboard navigation, but its relative order is defined by the platform convention;
   3. a **positive** value means should be focusable and reachable via sequential keyboard navigation; its relative order is defined by the value of the attribute: the sequential follow the increasing number of the tabindex. If several elements share the same tabindex, their relative order follows their relative position in the document.

9. What are landmark regions and how can they be useful?

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

   Most common role:

   - banner
   - complementary
   - contentinfo
   - form
   - main
   - navigation
   - region

- search

Landmarks allow assistive techonologies to navigate and to find content quickly.

To create a landmark role, define the purpose of the content by using a semantic element such as <section>, <nav>, or <main>, or adding an ARIA role that is a subclass of the landmark role such as role="banner", role="complementary", or role="region". Do not use role="landmark".

It is not as good practice to use non-semantic elements, such as `<div>`, adding semantics with landmark roles.

10. In what situations might you use a toggle button, vs a switch control, vs a checkbox?

11. Describe methods to hide content:

    Screenreader only content:

    Sometimes you'll want to communicate with a screen reader directly! One cool example is announcing to screen reader users that you offer accessibility features! In that case you can make some HTML and wrap it in a visually hidden class.

```
.visuallyhidden {
    position: absolute;
    left: 0;
    top: -500px;
    width: 1px;
    height: 1px;
    overflow: hidden;
  }
```

    sighted user only content:

```
<p aria-hidden="true">Visible text screen readers should ignore</p>
```

Hide from all users:

To hide text from a screen reader and hide it visually use the hidden attribute.

```
<p hidden>Hidden text screen readers should also ignore</p>
```

Caveat:

If you use `aria-describedby` in the same element as `aria-hidden="true"` the screen reader may read it.

The `aria-hidden` attribute does not work on focusable elements such as form inputs, links, and buttons.

If you use `aria-hidden` on an element with child elements, the child elements will be hidden as well.

12. Provide examples of common **incorrect** usage of ARIA attributes.

    Fixing broken semantics:

```
<span class="link" onclick="..." role="link">
  Google
</span>
```

- it is not focusable
- browser history may be broken
- legacy screen readers may have no support

Adding redundant semantics:

```
<form role="form">
  ...
</form>
```

```
<a href="..." role="link">
  Google
</a>
```