CSS PART 3

AGENDA

- Browser prefixes and utilization during styling
- Notes on accessibility
- Pseudo classes and elements
- Transitions
- Transforms

BROWSER CAPABILITIES

Ensuring same experience irrespective of how users access your site

DIFFERENT BROWSERS HAVE DIFFERENT CAPABILITY

- Different browsers may behave differently when it comes to rendering html elements and css styling rules

- As devs, we have make sure we develop a website/webpages that keeps in mind a wider audience base.

ELIMINATE BROWSER DIFFERENCES - 1

- CSS allows to do remove the differences between different browser displays
- Include what is called a default style sheet as the simplest solution (there are other ways, of course)
  - Remove all browser defaults in this default style sheet i.e.
    - Take all elements and enforce a standardized rule for all of them like setting margins, borders, paddings etc and set it to a fixed value.
    - Downside –
      - pages might not look aesthetically appealing, you will have to keep testing styling rules to make it look better
      - You will have to make sure you have included the style rules for all elements used

ELIMINATE BROWSER DIFFERENCES – 11

- What if some browsers don't support certain HTML tags?
- What if some browsers don't support certain CSS properties?

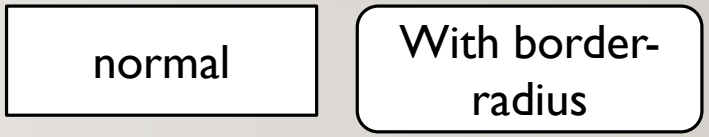
- Solution – use of *browser prefixes*
  - Fixes/hacks for unsupported CSS properties

BROWSER PREFIXES

- -webkit-
  - Android, chrome, iOS, safari
- -moz-
  - Firefox
- -ms-
  - internet explorer
- -o-
  - Opera
- Check out css3 generator site and see how they generate the style rules with browser prefixes

SOME UNSUPPORTED PROPERTIES FOR WHICH YOU CAN USE BROWSER PREFIXES

- column-count and column-gap
  - Specify as a part of display, how many columns you want and what is the gap between them
- border-radius
  - Rounding off the edges in your containers instead of square edges



- gradient
  - Adding a gradient to your colors
- use the following site to know more about unsupported properties and when to use browser prefixes
  - http://caniuse.com

REMEMBER~

- Its not about memorization
- You cannot memorize all properties and all its values
- However, practicing is important
  - You'll need to experiment with different properties – know what it looks like
- Remember to use inspect element tool from browsers and see what looks best for a property, then use that value in your code.
- Do not add browser prefixes to every single style rule. Its only for unsupported rules. Go to sites like caniuse.com to learn more

NOTES ON ACCESSIBILITY

STYLING AND ACCESSIBILITY

- Remember the accessibility guidelines
- Follow POUR
  - Perceivable <applicable in case of styling>
  - Operable
  - Understandable <applicable in case of styling>
  - Robust

NOTES ON PERCEIVABLE

- Specify alt text for your images
- Provide closed captions on videos. Provide transcripts for audios
- HTML content must use semantic tags
- Use good color contrast – WAVE your pages to double check on color contrast

NOTES ON OPERABLE

- All your webpage content should also be accessible by using keyboard
- For multi-media, users should be able to control how the media is being delivered to them
  - Media controls
- Do not make your flashy that may cause seizures
- Make page easy to navigate

NOTES ON UNDERSTANDABLE

- Use plain and simple to understand language
  - You page should not use language that is too complex or sound like a riddle.
- Make sure your content is to the point
  - Don’t write essays for page instructions or introductions
- Nav bars and page structure should be easily accessible and evident on the page
  - Make sure navigating your site is predictable and intuitive

NOTES ON ROBUST

- Your website should be functional across different devices
  - Cellphones, tablets, desktops, laptops, screen readers, etc.
- Later, we will discuss about what is known as "responsive designs" which accounts for screen-width etc.
- There should not be any syntax error in your HTML markup – this will cause issues for people using screen readers
- WAVE your page to adhere to standards

REMEMBER~~

- In the quest of styling your page and making it look *cool*, don't sacrifice accessibility. Web is an enabling tool!!

PSEUDO CLASSES AND ELEMENTS

WHAT ARE PSEUDO-CLASSES/ELEMENTS

- Elements that are either
  - Dynamically populated
  - Dependent on the DOM tree structure

- Example of this:
  - a:hover {…} (a link state)
  - Rules activate only when your mouse hovers over the link – dynamic in nature

DIFFERENT TYPES OF PSEUDO CLASSES 1

- Link
  - a:link, a:visited
- Based on user action (on links or paragraphs or page heading and sub-headings)
  - :hover, :active, :focus
- HTML forms and interfaces(buttons, checkboxes, etc)
  - :enabled, :checked, :disabled

DIFFERENT TYPES OF PSEUDO CLASSES II

- Defining pseudo classes based on the DOM structure or DOM position of elements
  - :first-child, :last-child, :nth-child(), :only-child
  - :first-of-type, :last-of-type, :only-of-type
- Usage examples
  - li:first-child{ … } //select the first child of the li element and apply the style rules to it
  - li:nth-child(4) {…}
  - p:empty{}
  - img:only-of-type{}
  - p:last-of-type{ }

DIFFERENT TYPES OF PSEUDO ELEMENTS

- Pseudo elements are not part of the actual DOM but can be used to style parts of the contents. Examples shown below.
- Textual pseudo elements
  - :first-letter, :first-line
- Positional or generated pseudo elements
  - :before, :after
- Fragments
  - :selection
- More on pseudo classes and elements when you discuss styling HTML tables.

TRANSITIONS

HOW TO ADD TRANSITIONS/ANIMATIONS

- Recall the following (from link states)
  - Using a:hover, you can change the color of the link over which you have your mouse
  - Using img:focus, you can change the height and width of an image when being focused

ADDING TRANSITIONS – CSS PROPERTIES

- transition-property
  - Specify thing to change – color, size, position, etc?
- transition-duration
  - duration of your transition animations
- transition-timing
  - smooth transitions i.e. linear or different like ease-in, ease-out, etc.
- transition-delay
  - should there be wait time before starting the animation?
  - should a mouse hover instantly cause the animation, or should you wait a bit?

STEPS TO ADD TRANSITIONS

- Define the basic styling rules for your elements
- Select the elements you want to add transitions to
- Once selected, define new values for styling rules of these elements
  - You must combine this process with a pseudo-class {another way of saying is, what will trigger the transitions? How will the browser know?}

EXAMPLE FOR TRANSITION – BASIC RULES

```
div{
    color: …;
    background:…;
    text-align:…;
    width:…;
    height: …;
    border-radius: 5px;
}
```

EXAMPLE FOR TRANSITION – ADD TRANSITION RULES

```
div{
    color: …;
    background:…;
    text-align:…;
    width:…;
    height: …;
    border-radius:5px;
    transition-property: color, width, background, broder-radius;
    transition-duration: .7s;
    transition-timing-function: linear;
    transition-delay: .5s;
}
```

EXAMPLE FOR TRANSITION – ADD PSEUDO CLASS A.K.A WHAT WILL CAUSE THE TRANSITION

```
div{
    color: …;
    background:…;
    text-align:…;
    width:…;
    height: …;
    border-radius:5px;
    transition-property: color, width, background, border-radius;
    transition-duration: .7s;
    transition-timing-function: linear;
    transition-delay: .5s;
}

div:hover{
    color: …
    width: …;
    background: …;
    border-radius: 50%;
}
```

SHORTHANDS FOR TRANSITION PROPERTY VALUES

- transition: background .2s linear, border-radius 1s ease-in 1s;
  - i.e property duration timing-function delay, property duration timing-function delay,…;

- Note – don't make use of pseudo classes and transitions in such a way that they are a pre-requisite to access content. This will hamper accessibility in many ways.

TRANSFORMS

Transitions to change appearance of existing elements on page

WHAT ARE TRANSFORMS

- Transforms are like transitions, but instead of an animation, they can change the way your existing elements look on the page.
- This can be done in both 2-D fashion and 3-D fashion

TRANSFORM – 2D OPTIONS

- 2D transforms options
  - translate // move elements based on (x,y) input. Works according to Cartesian co-od system
  - rotate // rotate(45deg)
  - scale // make elements big or small in size. scale(width, height)
  - skew // skew(20deg, 10deg)
  - matrix // combine all other

TRANSFORM – 3D OPTIONS

- Same as 2D, but adds the Z axis or Z degree

NOTE

- Transforms will be triggered by some state changes
- Transitions and transforms will require browser prefixes
- Accessibility concerns