Applied Visual Design

Created By	Nate L
Last Edited	@Jan 12, 2019 8:41 AM

- Align text content via the text-align CSS property, ex: `text-align: justify;`, `text-align: center;`
  - Default is left-justified
- Specify the width of an element with the width property, ex: `width: 300px;`
- Specify height of an element with the height property, ex: `height: 40px;`
- To make text bold, use the HTML strong tag `<strong>text</strong>`
  - (recommended to use this for things of great seriousness or urgency such as warnings, denote labels of paragraphs which represent notes or warnings within text of a page)
  - Use `<b></b>` to draw the reader's attention to the element's contents
  - If you're just styling text, use the CSS font-weight property instead
- The u tag `<u></u>` represents a span of inline text which should be rendered in a way that indicates that it has a non-textual annotation.
  - Use cases would be: annotating spelling errors, applying a proper name mark to denote proper names in Chinese text, and other forms of annotation
  - Do not use `<u>` to simply underline text for presentation purposes or to denote titles of books
  - Such tags provide semantic meaning and can assist with SEO and accessibility
  - Other forms of emphasis:
    - To stress emphasis (italics): use the `<em></em>` tag
    - Mark key words or phrases (italics): `<mark></mark>`

    - Mark titles of books or other publications (italics): `<cite></cite>`
    - Denote technical terms, transliterations, thoughts of vessels in Western texts (italics): `<i></i>`
    - Provide textual annotations (italics): `<ruby></ruby>`
    - To provide an underlined appearance without semantic meaning use the CSS property text-decoration, ex: `text-decoration: underline;`, `text-decoration: red wavy underline;`
- For non-semantic meaning appearances, use the CSS property font-style, ex: `font-style: italic;`
    - The four values it has are: normal, italic, oblique, and oblique <angle>, ex: `font-style: oblique 40deg;`
- The s tag `<s></s>` renders text with a strikethrough, it is used to represent things that are no longer accurate or relevant
    - For indicating document edits use the `<del>` and `<ins>` elements
    - For non semantic meaning, use the text-decoration CSS property
- CSS property text-decoration is shorthand for text-decoration-line, text-decoration-color, and text-decoration-style
- Use `<hr>` (self-closing) to create a horizontal line that spans the width of the containing element which is useful for separating content visually.
    - Do note you can add a class or id and alter its width with the CSS width property
- Another method of changing a color value is the CSS property `rgba()`
    - It can take hexadecimal values or it can take the numeric values (0-255) for RGB and then take percentages or decimals for the alpha value (level of opacity), ex:

```
/* hexadecimal value */
background-color: #3A30; <= 0% opaque green
background-color: #3A3F; <= 100% opaque green;
background-color: #33AA3388; <= 50% opaque green (long form for example)

/* rgba w/decimal, functional syntax */
background-color: rgba(51, 170, 51, .5);
```

```
/* rgba w/percentage, whitespace syntax */
background-color: rgba(51 170 51 / .40%);

/* can also use floats for the alpha value ex: +.25e2% */
```

- Headers should always have larger fonts than paragraphs
- Box-shadows are often used for card-like elements
  - Documentation can be read here: https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow
  - You can layer multiple shadows for a better effect
  - Blur and spread values are optional
- Directly change the opacity of any element with the opacity property
  - 0 is fully transparent and 1 is fully opaque
- The text-transform property is used to ensure that certain text is always capitalized, without having to directly modify the text itself
  - Documentation can be read here: https://developer.mozilla.org/en-US/docs/Web/CSS/text-transform
- You can apply the font-size and font-weight (thickness) properties to headers
  - Probably best to re-balance all of them, if you're going to change one
  - You can't style all the header fonts at once
  - Can be done on other text elements as well
- The line-height property changes line spacing
  - Apply to paragraphs, since headers typically don't have multiple lines
- Pseudo-classes refer to various states of a certain element
  - For example, you can specify CSS for the hover state, where the user hovers their cursor over a page element
  - Selectors using pseudo-classes look like `h1:hover`
- CSS uses a box model, where every HTML element belongs to a box. This setup is called normal flow, and can be overridden using positional properties.

    - Setting an element's position property to "relative" will allow you to modify that element's position, relative to its normal flow position.
        - Top, bottom, left, and right offset properties will move the element around
        - Moving this element will not impact the elements surrounding it. They will treat the relative element as if it were still in its default normal flow location.
    - Setting an element's position property to "absolute" removes the element entirely from normal flow. This means that surrounding elements will not "see" the absolute element. It also allows you to modify that element's position, relative to the closest positioned ancestor.
        - A "positioned ancestor" literally means an ancestor that has a position of relative, absolute, or fixed.
    - Setting an element's position to "fixed" removes it from normal flow, just like in absolute positioning. However, in this case it's moved relative to the **browser window**.
    - Fixed elements won't move when the user scrolls the page, but absolute elements will move (since the browser window doesn't change position from scrolling)
- The float property is used to push an element to the left or right of its parent element
    - It's an alternative to using the position property
    - Example usage would be to have two columns under a parent element
- The z-index determines which elements appear on top. Larger the value, higher up on the stack.
- You can horizontally center block elements by setting their margin property to "auto"
    - Note that images are inline and not block elements by default. However, they can be changed to block elements by setting their display property to "block"

    - For more on this: https://www.w3schools.com/css/css_display_visibility.asp

---

- Color design's basic idea involves finding and implementing complementary colors
- Color is a powerful tool, but we must consider that some users cannot see it
- The RGB color wheel consists of primary, secondary, and tertiary colors, for a total of 12
    - There are many color-choosing strategies
- One color is typically dominant, where its complementary colors are used to highlight certain information on the page. Equally using multiple colors throughout creates a jarring contrast that should be avoided.
- The setting a color to hsl() allows you to specify hue, saturation, and lightness
    - Hue is the location (angle) on the color wheel
    - Saturation is the amount of gray
    - Lightness is the amount of white or black
    - Saturation and lightness are commonly used to make multiple variations of a base hue, to create a color range of that base, with various levels of shading and tinting
- You can create a color gradient using the linear-gradient() function
    - The repeating-linear-gradient() function is similar, and used to make stripes
- You can import backgrounds with the url() function

- The transform property lets you manipulate size (scaling) and orientation (skewing or rotating)
- Using `::before` or `::after` pseudo-elements lets you add content before or after an element directly through CSS
    - You'll need to specify a content property, which can be an empty string if you're dealing with shapes, for example

- CSS allows for animations, in order to do so, you need to use the `@keyframes` CSS at-property
  - The keyframes rule controls the intermediate steps in a CSS animation sequence
  - This gives more control than using CSS transitions
  - To use keyframes, create a keyframe rule with a name that is to be used by the animation-name property
  - Each rule contains a style list of keyframe selectors, which specify percentages along the animation when the keyframe occurs, and blocks containing the styles for that keyframe
  - You can list the keyframe percentages in any order

```
/* Example keyframes usage */

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

- If a keyframe rule doesn't specify the start or end states the browser will use the element's existing styles for the start/end states.
  - Properties that can't be animated in keyframe rules are ignored
  - If multiple keyframe sets exist, the last one encountered is used
    - Keyframe rules don't cascade and won't derive rules from more than one rule set
    - However, if a given animation time offset is duplicated, all keyframes in the keyframes rule for that percentage are used for that frame.

        - There's cascading within a `@keyframes` rule if multiple keyframes specify the same percentage values
- As for animation, there a number of properties to choose from:
    - animation-name: sets one or more animations to apply to an element
    - animation-duration: sets the length of time that an animation takes to complete one cycle, ex: `animation-duration: 500ms`
    - animation-timing-function: sets how an animation progresses through the duration of each cycle, ex: `animation-timing-function: linear;`
    - animation-delay: sets when an animation starts, ex: `animation-delay: 2s;`
    - animation-iteration-count: sets the number of times an animation cycle should play before stopping, ex: `animation-iteration-count: 2;`
    - animation-direction: sets whether an animation should play forwards, backward, or alternating, ex: `animation-direction: reverse;`
    - animation-fill-mode: sets how a CSS animation applies styles to its target before and after its execution, ex: `animation-fill-mode: forwards; animation-delay: 2s;`
        - forwards - the target will retain the computed values set by the last keyframe encountered during execution
        - backwards - the target will retain the computed values set the first keyframe
        - both - target will follow rules for both forwards and backwards
    - animation-play-state: sets whether an animation is running or paused
- Refer to this: https://developer.mozilla.org/en-US/docs/Web/CSS/animation

```
/*
Animation example that creates a rectangle that repeats a color
change from blue to yellow and back again, infinitely repeating
*/

#rect {
    animation-name: rainbow;
    animation-duration: 4s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
```

```
}

@keyframes rainbow {
  0% {
    background-color: blue;
  }

  50% {
    background-color: green;
  }

  100% {
    background-color: yellow;
  }
}
```

- You may also use animations for pseudo-classes such as `:hover`
- If an element has `position: fixed` or `relative`, you can use the `right`, `left`, `top`, and `bottom` rules to create moving animation
- Change animation timing with the animation-timing-function property and use keywords such as `ease-out`, `ease-in`, or `linear` (ease is the default value)
  - You can also use the cubic-bezier curve
    - Syntax: `cubic-bezier(x1, y1, x2, y2)`
      - It consists of four points (p0, p1, p2, p3) that sit on a 1 x 1 grid consisting of the origin (p0 = 0,0) and the point (p3 = 1, 1)
      - You can set the values of the x and y coordinates of p1 and p2
      - x1 and x2 must be within the range of [0, 1], otherwise the curve is invalid
      - The y coordinates can be set to a value larger than 1, doing so will create a bouncing movement
      - Two points must be defined, there is no default value
      - If you define an invalid curve, the rule will be ignored
      - More detailed information: https://developer.mozilla.org/en-US/docs/Web/CSS/single-transition-timing-function