Learning CSS

Edward Sharick - Week 4 and 5 (9/18/2023 and 9/25/2023)

Learning CSS (Cascading Style Sheets)

- CSS is the language used to style a web page
- It formats the layout of a webpage. It can control the color, font, size of text, spacing, position, background images, background color, etc. of HTML elements
- It can be applied to entire webpages which can save developers time and energy

3 Use cases

- In-line (using attribute style):

```
<h1 style="color:blue;">A Blue Heading</h1>
```

- Internal (using <style> tag to define styles for elements):

```
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
```

- External (using an external .css file)

```
<head>
 <link rel="stylesheet" href="styles.css">
</head>
```

CSS syntax (like a python dictionary)

- selector {property1:value1; property2:value2; property3:value3;}
- For example: h1 {color:blue; font-size:12px;}
- Comments:
    - Single-line or multi-line: /* comment */

CSS colors

- By name: gray, violet, yellow, etc.
    - https://www.w3schools.com/colors/colors_names.asp
- By rgb: rgb(255, 0, 0);
- By hex: #ffffff;
- By hsl: hsl(0, 100%, 100%);
- You can use ‘currentcolor’ to use the most recent color
- You can use ‘inherit’ to use the parent element’s color

CSS Selectors

- The ‘selector’ can be:
    - a single element name, id (use #), or class (use .)
    - a combination of element and class (p.classname)
    - a group of elements (separate by ‘,’ ex: h1, h2, p)
    - universal – all elements on page (use *)

CSS Combinators

- A combinator explains the relationship between selectors
- Four combinators
    - Descendant selector (space) – child, grandchildren, great grand children, etc.

        div p
    - Child selector (>) – child only

        div > p
    - Adjacent sibling selector (+) – same parent and sibling element immediately follow

        div + p
    - General sibling selector (~) – same parent

        div ~ p

CSS Pseudo-class

- A pseudo-class is used to define a special state of an element
    - Mouse over, visited vs unvisited, focus element, etc.
- Examples:
    - a:link
    - a:visited
    - a:hover
    - a:active
    - a:highlight:hover /*differs from hover, only applies to <a class=”highlight” */
    - div:hover p { display: block; } /* <p> in displayed as blocks when hovered */
    - p:first-child /*any element that is the first child of any element */
    - :nth-child(n)
- And many more…

CSS Pseudo-element

- A pseudo-element is used to style specified parts of an element
    - Style the first letter, insert content before or after the content of an element, etc.
- Examples:
    - p::first-line
    - ::first-letter
    - ::before and ::after
    - ::marker /* formats the marker for a list */
    - ::selection /* selection of cursor */

CSS Attribute Selectors

- Styling elements that have specific attributes
    - Ex: a[target] will select <a> elements with a target attribute.
- You can check the attribute has:
    - an exact value: a[target="_blank"]
    - has the exact value or value followed by '-': [class|="top"]
    - starts with a value: [class^="top"]
    - ends with a value: [class$="top"]
    - contains a certain space-separated word somewhere in the attribute: [title~="flower"]
    - contains a value somewhere in the attribute: [class*="te"]
- Can be used to define elements that don't have id or class, such as in a form.

CSS Specificity

- If two or more CSS rules point to the same element, the selector with the highest specificity will be applied.
- Elements and pseudo-elements < class, pseudo-classes, attribute selectors < id < inline styles
- If two rules have equal specificity, the latest rule wins.
    - Rules in the HTML file will win over external CSS files

CSS Properties

Background

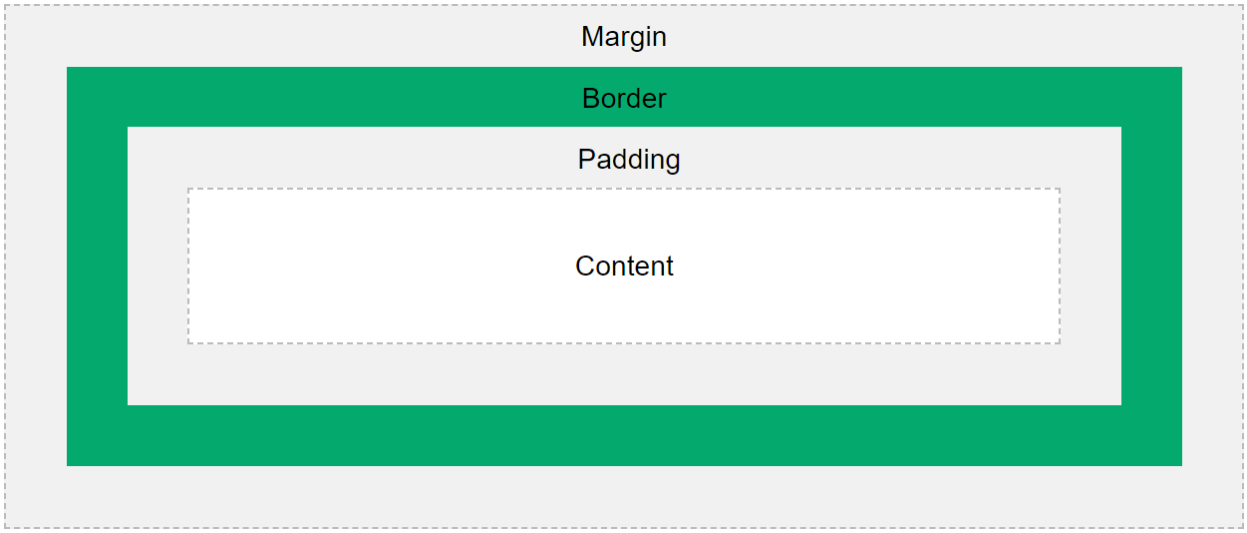
- background-color: green;
- opacity: 0.3; /*between 0.0 and 1.0*/
- background-image: url(image.png);
    - Can add multiple images, separate urls with commas
    - Have to add values for each image's properties
- background-repeat; /*both x and y */
- background-repeat: repeat-x;
- background-repeat: repeat-y;
- background-repeat: no-repeat;
- background-position: right top; /* can use left, bottom; default is center */
- background-origin: content-box; /* padding-box is default, border-box is another option */
- background-attachment: fixed; /*fixed - move with scrolling; scroll - scrolls with page*/
- can use all together with 'background' property

Gradients

- Linear Gradients (down/up/left/right/diagonally)
- Radial Gradients (outward from center)
- Conic Gradients (rotated around a center point)
- Use the 'background-image' property

    - background-image: linear-gradient(to bottom right, red, yellow);
    - background-image: radial-gradient(red 5%, yellow 15%, green 60%);
    - background-image: conic-gradient(red 45deg, yellow 90deg, green 210deg);
- Add border-radius: 50% to a conic gradient to make pie chart

Spacing elements: CSS uses the box model for border, margin, and padding:



- width, height, and max-width can be used to set the width, height, and max-width of the content, not including the border, margin, padding.
- Can be defined as a length (in px, cm, pt, etc.) a % of containing block, or inherit from parent container

Border

- border-style: dotted /* dashed, solid, double, groove, ridge, inset, outset, hidden */
- border-width: 5px /*px, pt, cm, em, thin, medium or thick */
    - Use multiple values to specify differences between top, right, bottom, and left
- border-color: green
    - Also use multiple colors to specify sides
- border-top-style: dotted /* can set bottom, left, and right too */
    - Or feed multiple values to border-style
- Can use all together with 'border' property
- border-radius: 5px; /*rounded corners – can be applied to any element */
- border-image: url(img.png) 30 round; /* 30 is the slice value; round/stretch */

Margin

- Space around an element outside its border
- margin-right: 200px; /* set a left, right, top, or bottom margin */
- can be set to auto (browser calculates margin), px or other length amount, % of container's width, or inherit (get margin from parent)
- margin collapse – happens when a bottom and top margin collide, the margin becomes the larger of the two

Padding

- Space around an element inside its border
- padding-top: 50px; /* can set bottom, right or left as well */
- can be set px, pt, cm or other length amount, % of container's width, or inherit from parent
- can shorthand with 4 (t, r, b, l), 3 (t, r/l, b), 2 (t/b, r/l), or 1(all 4) values
- Since the padding is added to the content 'width' (or 'height'), the actual rendered 'width' will be larger if the padding is not 0. The 'box-sizing: border-box;" property can be used to keep the width at the set amount, regardless of the padding.

Outline

- Outline shows up outside the border, may intersect with other elements, and does not effect the width or height of the element.
- outline-style, outline-color, outline-width, and outline all work similarly to border
- outline-offset adds space between the outline and the edge of an element

Text Formatting

CSS Text

- color: green; /*or other hex, rgb, hsl values */
- text-align: center; /* left, right or justified */
- direction: rtl; unicode-bidi: bidi-override; /* used to change text direction */
- vertical-align: baseline; /*or text-top, text-bottom, sub, super */
- text-decoration-line: overline; /* or underline, line-through, overline underline */
- text-decoration-color: blue;
- text-decoration-style: solid; /*double, dotted, dashed, wavy */
- text-decoration-thickness: 5px;
- text-decoration: /* shorthand order: line, color, style, thickness */
- text-decoration: none; /* remove underline from links */
- text-shadow: 2px 2px 5px color;
- text-transform: uppercase; /* lowercase, capitalize */
- text-indent: 50px;
- letter-spacing: 5px;
- line-height: 0.8; /*scalar value */
- word-spacing: 5px;
- white-space: nowrap; /*prevents text wrapping */
- word-wrap: break-word; /*
- word-break: keep-all; /*break-all; */
- text-overflow: clip; /* ellipsis does summary text … */
- writing-mode: horizontal-tb; /* vertical-rl */

CSS Fonts

- Fonts belong to one of 5 families: serif, sans-serif, monospace, cursive, fantasy
- font-family: “Times New Roman”, Times, serif; /*contain fallback fonts */
    - always end the list with the generic family name
- font-style: normal; /*italic, oblique */
- font-weight: normal; /*bold */
- font-variant: normal; /*small-caps */
- font-size: 40px; /* can be absolute */
- font-size: 100%; /* or relative */
- You can use the ‘font’ shorthand property:
    - font: style variant, weight, size/line-height, font-family;
    - font: italic small-caps bold 12px/30px Georgia, serif;
- style=”font-size:10vw” can be used to set responsive font size to the viewport width

Google Fonts

- Can add Google Fonts by referencing a google css file:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>
```

- And font effects:

```
<head>
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Sofia&effect=fire">
<style>
body {
  font-family: "Sofia", sans-serif;
  font-size: 30px;
}
</style>
</head>
<body>

<h1 class="font-effect-fire">Sofia on Fire</h1>

</body>
```

Web Fonts

- You can use the ‘@font-face’ rule to define a custom font. Set the ‘font-family: customName’ and ‘src: url(fontfile.ttf)’. Then reference your font-family from any element.

- You may also need to define another rule for ‘bold’ characters.

CSS Icons

- Use an icon library:
    - Bootstrap Icons

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    - Google Icons (link stylesheet)

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

    - Font Awesome Icons (requires login and script file)
- Then place icon in an inline element like

CSS Links

- Style link <a> with any CSS property
    - a { color: hotpink; }
- Links can also be styled differently based on the state they are in
- 4 states:
    - a: link – unvisited link
    - a: visited – visited by user
    - a: hover – user is mousing over link
    - a: active – the link when it is clicked
- You can use many different properties to style links in many different ways, including making them look like buttons. You can style the cursor as well over links to help draw attention to the link.

CSS Collections (Lists and Tables)

CSS Lists

- Set different list item markers for ordered lists
    - list-style-type: upper-roman; /* use ‘none’ to remove markers */
- Set different list item markers for unordered lists
    - list-style-type: circle;
- Set an image as the list item marker
    - list-style-image: url(‘img.gif’);
- Set the position of the bullet or marker
    - list-style-position: inside; /* or outside */
- Add background colors to lists and list items

CSS Tables

- Style the elements of an HTML table: <table>, <th>, <tr>, and <td>
- You can specify the table borders, and whether double borders should collapse
    - table { border-collapse: collapse; }

- You can style the table size and the position of the text with the properties width, height, padding, text-align, etc.
- Using the tr:nth-child(even) selector, you can style odd or even rows
- To make a responsive table that will add scroll bars if the table is too large for the display, use a container element like and set the overflow-x:auto property. Wrap the table in the

CSS Forms

- Style similar elements using the attribute selectors
- Use :hover, :focus, etc. to add appearance changes on forms

CSS Display and Visibility

- The 'opacity' property can be used to make elements appear transparent
- RGBA can be used to make background's transparent without affecting the text inside the element.
- The 'display' property specifies if/how an element is displayed (none, block, or inline)
    - Block-level elements always start on new line; take up full width available
        - But you can set the width or max-width (recommended for small devices) to limit the space it takes
    - In-line elements take space necessary
    - display: none; and visibility: hidden; are commonly used with JS to hide and show elements without deleting or recreating them
- display: none; hides the element and page appears as if the element doesn't exist
- display: inline-block;
    - allows setting of width and height, respects top and bottom margins, and doesn't add a line-break
- visibility: hidden; hides the element but leaves space for the element still
- The 'position' property specifies the type of positioning method used for an element
    - 'static' – default; nothing special; not affected by top, bottom, left and right
    - 'relative' – relative to its normal position; left, right, top, and bottom adjust the element from its normal position
    - 'fixed' – positioned relative to the viewport (stays in the same place even if the page is scrolled); can then be set with left, right, top and bottom
    - 'absolute' – positioned relative to the nearest positioned ancestor
    - 'sticky' – positioned based on user's scroll position; relative until a given offset position is met, then sticks in place like fixed
- The 'z-index' property controls the stack order of elements; 0 is the default, negative values are behind, and positive values are in front
    - If no z-index is specified, the element defined last will show up on top (elements are drawn on top of each other)
- The 'overflow' property specifies what to do when content is to big to fit the area
    - 'visible' – default; content renders outside the element's box

    - 'hidden' – overflow is clipped and not visible
    - 'scroll' – overflow is clipped and scrollbars appear
    - 'auto' – similar to scroll, but adds scrollbars only when necessary
- You can specify overflow-x, and overflow-y separately
- The 'float' property positions content right or left of another element
    - Useful with elements so that they don't display next to each other.
- The 'clear' property makes the next element go below (it undoes the 'float')
    - Match the 'float' left or right
- If a floated element is taller than the containing element, set 'overflow: auto;' for the containing
- You can use the float property and width % to create grids of side-by-side boxes
    - Useful in creating a navigation menu or a sidebar menu
- Setting the 'margin: auto;' for block elements (like) will center the block element. Setting the 'width:50%;' will give 25% to each right and left margin
- Centering the text inside the can be done with 'text-align: center;'
- 'text-shadow' and 'box-shadow' can be used to add shadows to elements

CSS Flexbox

- Can be used to design flexible responsive layout structures without using float or positioning.
    - display: flex;
- Then set the 'flex-direction', 'flex-wrap', 'flex-flow', 'justify-content', 'align-items', and 'align-content' properties.
- CSS Flexbox (Flexible Box) (w3schools.com)

Images

- For displaying images, you can apply a filter with the 'filter' property.
    - https://www.w3schools.com/cssref/css3_pr_filter.asp
- Use :hover and transitions to do hover over effects
- You can use JS to create 'modal' pages for viewing image
    - W3Schools Tryit Editor
- You can reflect images using the '-webkit-box-reflect: right' property.
- 'object-fit' property can be used to scale images and keep aspect ratio if desired
    - fill – default behavior; fills container exactly and aspect ratio not maintained.
    - contain – keeps aspect ratio and fits in container with white space on top/bottom or left/right
    - cover – keeps aspect ratio and fits in container exactly, but crops the image if necessary
    - none – image is not resized
    - scale-down – image is called down to the smallest version of none or contain
- 'object-position' property can tell an <img> or <video> how it should be positioned in a container. This is for when the entire content cannot fit in the container, we can move the image around and decide what part will be in view.

    - Use ‘object-position: x% y%;’ where x represents the starting scaled point for x and same for y
- You can apply mask layers to images; this could be a gradient, a transparent or semi-transparent image, or some sort of stencil like image.

CSS Variables

- The var() function is used to insert the value of a CSS variable. They have access to the DOM, so you can create variables with local (defined in a selector) or global scopes (defined in entire document; define it in a :root{} selector), and change the variables with JS and media queries.
    - Ex: putting color values in variables so you can put them in one place and not have to copy and paste them everywhere.
    - Global variable definition:

```
:root {
 --blue: #1e90ff;
 --white: #ffffff;
}
```

    - Local variable definition (local variable will override global if it exists, otherwise creates new variable with local scope):

```
button {
 --blue: #0000ff;
}
```

    - Variable use:

```
body { background-color: var(--blue); }
```

    - You can get or set the variables with JS:

```
//Get root element
var r = document.querySelector(':root');
//get the styles for root
var rs = getComputedStyle(r);
//get the root styles value of --blue var
alert("The value of --blue is: " + rs.getPropertyValue('--blue'));
//changes the root’s var --blue
r.style.setProperty('--blue', 'lightblue');
```

    - You can also use media queries (CSS3 Media Queries - Examples (w3schools.com)) to adjust variables on the fly:

```
@media screen and (min-width: 450px) {
 .container {
  --fontsize: 50px;
 }
}
```

CSS Counter

- Counters are like variables that can be incremented, reset, and are used to track how many times certain items are used.
- Example:

```
body {
  counter-reset: section;
}

h2::before {
  counter-increment: section;
  content: "Section " counter(section) ": ";
}
```

- Before each <h2> element, we increment 'x' and add the "Section {x}:" string.
- You can nest counters to create outlined lists (i.e. 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, …)

CSS Math Functions

- calc(expression) – can be used as a property value
- max(val1, val2, …) – largest of comma separated values
- min(val1, val2, …) – smallest of comma separated values

CSS Animations

- 2D Transforms
  - Move, rotate, scale, skew
  - Ex: 'transform: translate(50px, 100px);'
- 3D Transforms
  - Same as 2d, but can also move, rotate, scale, skew in z-direction
- Transitions
  - Allow you to change property values smoothly over time
  - transition-delay /* sets a delay time before starting transition */
  - transition-duration /* time of transition */
  - transition-property /* property value to change */
  - transition-timing-function /*linear, ease, ease-in, ease-out, ease-in-out, cubic-bezier(n, n, n, n) */
  - Can be set separately or use the shorthand property:

```
transition: width 2s linear 1s /*property, duration, function, delay */
```

- Animations
  - Lets an element gradually change from one style to another.
  - @keyframes

- defines the ‘from’ and ‘to’ styles; for example:

```
@keyframes example {
 from {background-color: red;}
 to {background-color: yellow;}
}
```

- You can also define the styles with % from 0 to 100

```
@keyframes example {
 0%   {background-color:red; left:0px; top:0px;}
 25%  {background-color:yellow; left:200px; top:0px;}
 50%  {background-color:blue; left:200px; top:200px;}
 75%  {background-color:green; left:0px; top:200px;}
 100% {background-color:red; left:0px; top:0px;}
}
```

- animation-name – reference the name of the @keyframes
- animation-duration – tells how long the animation should take
- animation-delay – delay for animation
- animation-iteration-count – Number of times the animation should repeat (or infinite)
- animation-direction – normal, reverse, alternate, alternate-reverse
- animation-timing-function – same as transforms (ease, linear, etc.)
- animation-fill-mode -lets the element get the style values from the first keyframe (backwards) or maintain the style values after the animation (forwards) or both (both)
- animation shorthand can combine all of these, for example:

```
animation: example 5s linear 2s infinite alternate;
```