Web Design

Advanced CSS Features

Transform

CSS Transform Property

- The transform property lets us move and distort elements on the page
- Takes functions arguments, and can take multiple (order matters)
- The transform-origin property sets where the transform should start from or reference (default is the center)

```
.transformer {
    transform:
        translate(4rem, 10rem)
        rotate(45deg)
        scale(2)
        skew(20deg, 30deg)
    ;
}
```

Pseudo classes and elements

CSS Pseudo-Classes

- Let us style based on the state of an element (like a class but not defined as a class)

```
.element:hover {} /* Any element being hovered over by the cursor */
.element:active {}
.element:disabled {}
.element:focus {}
.element:first-of-type {} /* Any .descendant that is the first .descendant of its siblings */
.element:first-child {} /* Any .descendant that is also the first child */
.element:nth-of-type(2n + 1) {} /* Any .descendant whose order fits the pattern 2n + 1 among its siblings */
.element:not(.active) {} /* Any .descendant that does not have the class active */
.element:not(:hover) {} /* Any .descendant that is not being hovered over */
```

CSS Pseudo-Elements

- Every element has an ::after and a ::before pseudo-element
- We can use these pseudo-elements to add additional styling to our elements and they allow us to do some tricks easily
- To use the after/before pseudo-elements, we first need to set the content property
- Then we can just interact with the pseudo-element like a normal element

```
.element::after {
    content: "";
    width: 100%;
    height: 100%;
    display: block;
}
```

CSS Pseudo-Elements (cont.)

- Other Pseudo-Elements

```
.element::first-letter {} /* Styles first letter of text */
.element::first-line {} /* Styles first line of text */
.element::selection {} /* Styles highlight color */
```

Selector Combinators

CSS Combinators

- Combinators tell how different selectors interact with each other
- Descendant combinator: “ “ (space character)
- Child combinator: “>”
- Adjacent sibling combinator: “+”
- General sibling combinator: “~”

```
.parent .descendant {} /* Any descendant of .parent */
.parent > .child {} /* Any child of .parent */
.sibling + .element {} /* Any .element that comes _right_ after a .sibling (at the same level) */
.sibling ~ .element {} /* Any .element that comes after a .sibling (at the same level) */
```

The Comma In CSS

- In CSS, the comma (“,”) means “or” and allows for applying the same styling to different selectors

```
.a, .b {} /* Any .a or .b */
h1, h2, h3 {}
```

Custom Properties

CSS Custom Properties (Variables)

- Can store values for reuse (or just to give them a name)
- Defined like any other property, but prefixed with “--”
- Use the var() function to use the value
- Can store multiple values
- Updating the value of a variable updates any references to that variable

```
.element {
    --size: 10rem;
    width: var(--size);
    height: var(--size);
    --fancy-border: 1px dashed;
}

.element .child {
    border: var(--fancy-border) gray;
}
```

CSS Custom Properties (cont.)

- Store global values in the :root
- Examples: colors, styles, sizes, pre-computed constants

```
:root {
    --fancy-red: #ff2323;
    --default-border: 1px solid gray;
    --header-height: 2rem;
}
```

Functions

CSS Calc Function

- The calc function can do math
- Works with values with different units (as long as they are the same type of value)
- Works with variables

```
.element {
    width: calc(50% - 5px);
    height: calc(2 * var(--default-height));
}
```

Other CSS Functions

- min (take the lesser value, good for setting a max value)
- max (take the greater value, good for setting a min value)
- clamp (preferred value with bounds)

```
.element {
    font-size: min(10vh, 16px);
    padding: max(5vw, 1rem);
    margin: clamp(1rem, 2.5vw, 2rem);
}
```