Selectors

a. Type selector

```
/* A href that contains "example.com" */
[href*='example.com'] {
  color: red;
}

/* A href that starts with https */
[href^='https'] {
  color: green;
}

/* A href that ends with .com */
[href$='.com'] {
  color: blue;
}
```

b. Pseudo-elements
    i. ::before pseudo-element to insert content at the start of an element, or the ::after pseudo-element to insert content at the end of an element.
    ii. ::selection to style the content that has been highlighted by a user.

c. Combinators

- A combinator is what sits between two selectors.
    i. Descendant combinator
        1. allows us to target a child element
        2. .container h1 {}
    ii. Next sibling combinator
        1. element that immediately follows another element
        2. Div1 + div2
    iii. Child combinator
        1. direct descendant
        2. Parent > direct_child
    iv. Compound selectors
        1. <h1> elements, which will also have class “class_for_a”
            h1.class_for_a

Cascade

a. An inline style attribute with CSS declared in it will override all other CSS

b. If there are two divs inside a container with display flex column, first one with margin-bottom 10px, and second with margin-top 10px, then gap between them is 10px, due to margin collapsing.

Specificity

- Higher the points in specificity, more it will override lower-specificity elements.

a. A universal selector (*) has no specificity and gets 0 points. This means that any rule with 1 or more points will override it
   i. *
b. An element (type) or pseudo-element selector gets 1 point of specificity .
   i. Div, h1, p, etc
c. A class, pseudo-class or attribute selector gets 10 points of specificity.
   i. .my-class {}
   ii. :hover {}
   iii. [href='#'] {}
d. An ID selector gets 100 points of specificity
   i. #my-id {}
e. CSS applied directly to the style attribute of the HTML element, gets a specificity score of 1,000 points.
   i.
f. Lastly, an !important at the end of a CSS value gets a specificity score of 10,000 points.

```
.my-class {
  color: red !important; /* 10,000 points */
  background: white; /* 10 points */
}
```

g. this has 11 points of specificity

```
a.my-class {
  color: green;
}
```

h. This has 21 points of specificity

```
a.my-class.another-class {
  color: rebeccapurple;
}
```

i. 41 points of specificity

```
a.my-class.another-class[href]:hover {
  color: lightgrey;
}
```

Inheritance

a. Inherit
    i. make any property inherit its parent's computed value
b. Initial
    i. resets a property to its initial (default) value, which is the value defined by the CSS specification.
c. Unset
    i. acts as inherit for inherited properties and initial for non-inherited properties.

```
.parent {
  color: blue;
  margin: 20px;
}

.child {
  color: unset;  /* Acts like inherit for color, so it will be blue */
  margin: unset; /* Acts like initial for margin, so it will reset to 0 */
}
```

Flexbox

a. Wrap
    i. Nowrap : if there is not enough space in the container the items will overflow.
    ii. Wrap : cause items to wrap
b. flex-flow
    i. Flex direction + flex wrap
    ii. Eg : flex-flow : column wrap
c. Flex [Controlling space inside flex items]
    i. The flex-basis property defines the default size of an element before the remaining space is distributed
    ii. Flex-grow : 2 will take twice much space as flex-grow : 1 would take.
    iii. Flex- shrink
        1. The flex-shrink property defines the ability of a flex item to shrink if necessary

2. Flex-shrink : 2 will shrink twice much space as flex-shrink : 1 would shrink.

iv. Combining these three

```
.item {
  flex: 1 0 100px; /* flex-grow: 1, flex-shrink: 0, flex-basis: 100px */
}
```

d. Flex alignment
   i. justify -content : main axis
   ii. Align-items : cross axis
   iii. Place-content : both justify content and align items
   iv. align-self: aligns a single item on the cross axis.

```
.container {
  place-content: space-between;
  /* sets both to space-between */
}

.container {
  place-content: center flex-end;
  /* wrapped lines on the cross axis are centered,
  on the main axis items are aligned to the end of the flex container */
}
```

Grid

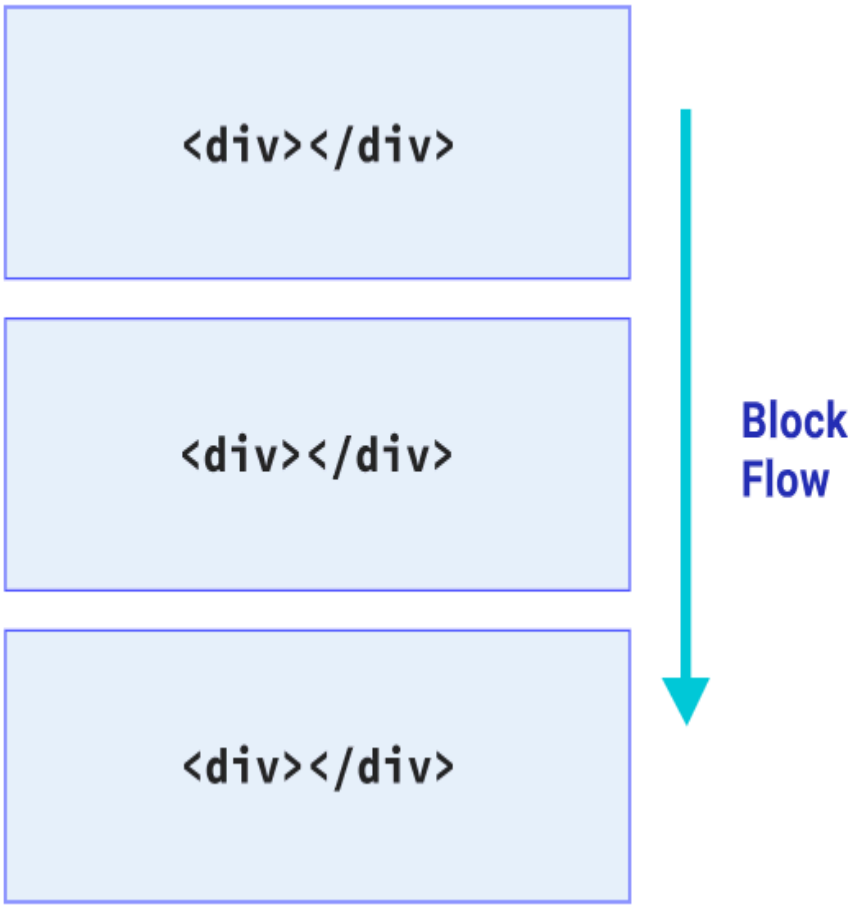
```
.container {
  display: grid;
  grid-template-columns: 100px 200px 100px; /* Three columns with specified widths
  grid-template-rows: 100px 200px; /* Two rows with specified heights */
  gap: 10px; /* Gap between rows and columns */
}
```

```css
.item1 {
  grid-column: 1 / 3; /* Spans from column 1 to 3 */
  grid-row: 1 / 2; /* Spans from row 1 to 2 */
}
```

```css
.container {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-template-rows: repeat(2, 200px 100px);
}

.item {
    grid-column-start: 1; /* start at column line 1 */
    grid-column-end: 4; /* end at column line 4 */
    grid-row-start: 2; /*start at row line 2 */
    grid-row-end: 4; /* end at row line 4 */
}
```

```css
.container {
  display: grid;
  grid-template-areas:
    'header header header'
    'sidebar main main'
    'footer footer footer';
  grid-template-columns: 1fr 2fr 1fr;
  grid-template-rows: auto 1fr auto;
  gap: 10px;
  height: 100vh;
}

.header {
  grid-area: header;
  background-color: #f8b400;
  padding: 20px;
  text-align: center;
}

.sidebar {
  grid-area: sidebar;
  background-color: #4caf50;
  padding: 20px;
}
```

Logical Properties



Pseudo elements

a. Both the ::before and ::after pseudo-elements create a child element inside an element only if you define a content property.

b. ::first-letter
c. ::first-line : needs to have display value block, inline-block, list-item, table-caption or table-cell.
d. ::backdrop
e. ::selection
f. ::placeholder
g. ::cue : for captions of a video element

Pseudo classes

a. :hover
b. :active : click and hold
c. :focus
d. :link : for links that has not been visited yet
e. :visited : for links that has been visited
f. If you define a :visited style, it can be overridden by a link pseudo-class with at least equal specificity. Because of this, it's recommended that you use the LVHA rule for styling links with pseudo-classes in a particular order: :link, :visited, :hover, :active.
g. :disabled and :enabled
h. :first-child and :last-child
i. :only-child
j. :first-of-type and :last-of-type
k. :nth-child and :nth-of-type
l. :empty
m. :is

```
.post :is(h2, li, img) {
    …
}
```

n. :not()

```
img:not([alt]) {
    outline: 10px red;
}
```

i. Styles all img which dont have alt attribute

Border

a. Making blob

```
.my-element {
    border: 2px solid;
  border-radius: 95px 155px 148px 103px / 48px 95px 130px 203px;
}
```

Day5

Shadow

a. Box shadow
   CSS is creating a shadow based on the shape of the box **as if light is pointing at it.**
   i. Horizontal offset: a positive number pushes it towards right and a negative number will push it towards left.
   ii. Vertical offset: a positive number pushes it towards bottom, and a negative number will push it towards top.
   iii. Blur radius: a larger number produces a more blurred shadow, whereas a small number produces a sharper shadow.
   iv. Spread radius (optional): a larger number increases the size of the shadow and a smaller number decreases it, making it the same size as the blur radius if it's set to 0.
   v. Color
   vi. To make a box shadow an inner shadow, add an inset keyword before the other properties.

```
.my-element {
  box-shadow: 5px 5px 20px 5px darkslateblue, -5px -5px 20px 5px dodgerblue,
    inset 0px 0px 10px 2px darkslategray, inset 0px 0px 20px 10px steelblue;
}
```

b. Text shadow
   i. Works same as box shadow
c. Drop Shadow
   i. follows any potential curves of an image

ii. Gives shadow to content of image.
iii. inset keyword and spread value are not allowed.

```
.my-image {
  filter: drop-shadow(0px 0px 10px rgba(0 0 0 / 30%))
}
```

Focus

a. :focus: Styles an element when it has received focus.
b. :focus-within: Styles an element if it or any of its descendants have received focus.
c. :focus-visible: Styles an element when it has received focus and the focus is visible (typically through keyboard interaction).

Functions

a. calc() : takes a single mathematical expression as its parameter
b. min() and max() : returns the smallest/largest computed value of the one or more passed arguments.
c. clamp(min, preferred value, max)
    i. It's useful for responsive design, ensuring that a value does not go below or above certain limits.
    ii. font-size: clamp(1rem, 2.5vw, 2rem);
        1. At least 1rem
        2. Preferably 2.5vw (2.5% of the viewport width)
        3. At most 2rem
d. Clip-path
    i. .polygon {
    clip-path: polygon(0% 0%, 100% 0%, 100% 75%, 75% 75%, 75% 100%, 50% 75%, 0% 75%);
    }
        1. Start at the top-left corner (0% 0%).
        2. Move to the top-right corner (100% 0%).
        3. Move down to 75% height on the right edge (100% 75%).
        4. Move left to 75% width and 75% height (75% 75%).
        5. Move down to the bottom at 75% width (75% 100%).
        6. Move up to 75% height at 50% width (50% 75%).
        7. Move left to 75% height at the left edge (0% 75%).
    ii. CSS Shape functions are inset, polygon, circle, ellipse
e. Rotate
    i. transform: rotate3d(1, 0, 0, 45deg); rotate by x axis
    ii. transform: rotate3d(1, 1, 0, 45deg); rotate by x and y axis

iii. transform: rotate3d(1, 1, 1, 45deg); rotate by all three axes (normalized axes)
    1. transform: rotateX(10deg) rotateY(10deg) rotateZ(10deg);

f. Scale
    i. transform: scaleX(1.2) scaleY(1.3);
g. Translate
    i. transform: translatex(40px) translatey(25px);
h. Skew
    i. Tilts the object
    ii. transform: skew(10deg);

Gradients

a. Linear gradient
    i. To right, means left to right, left color will be first color, and right color will be second color.
    ii. background: linear-gradient(45deg, darkred 20%, crimson, darkorange 60%, gold, bisque);

b. Radial gradient
    i. Circular fashion
    ii. background: radial-gradient(darkred 20%, crimson, darkorange 60%, gold, bisque);

c. Conic gradient
    i. A conic gradient has a center point in your box and starts from the top (by default), and goes around in a 360 degree circle.
    ii. conic-gradient(from 10deg at 20% 30%, white, black);

1. ,from 10deg
    a. the gradient starts at 10 degrees from the top (clockwise).
2. At 20% 30%
    a. position of the center of the gradient within the element.
3. Color stops
    a. The transition happens around the circle, starting at 10 degrees and continuing clockwise.

d. Elements can also have multiple gradients at background
    i. background: linear-gradient(to right, red, blue), radial-gradient(circle at center, yellow, green);

Animation

a. .element {
animation: animationName duration timing-function delay iteration-count direction fill-mode play-state;
}

a. animationName: The name of the animation (defined in @keyframes).
b. duration: The duration of the animation in seconds (e.g., 1s, 0.5s).
c. timing-function: Specifies the speed curve of the animation (e.g., ease, linear, ease-in-out, cubic-bezier(x1, y1, x2, y2)).
d. delay: Optional. The time to wait before starting the animation (e.g., 1s, 500ms).
e. iteration-count: Optional. The number of times the animation should repeat (e.g., infinite, 3, 2.5).
f. direction: Optional. Specifies whether the animation should play forwards, backwards, alternate, or alternate-reverse (e.g., normal, reverse, alternate).
g. fill-mode: Optional. Specifies what styles are applied to the element when the animation is not playing (e.g., forwards, backwards, both, none).
h. play-state: Optional. Specifies whether the animation is running or paused (e.g., running, paused).

Eg :

```
.my-element {
    animation: my-animation 10s ease-in-out 1s infinite forwards forwards running;
}
```

Filter

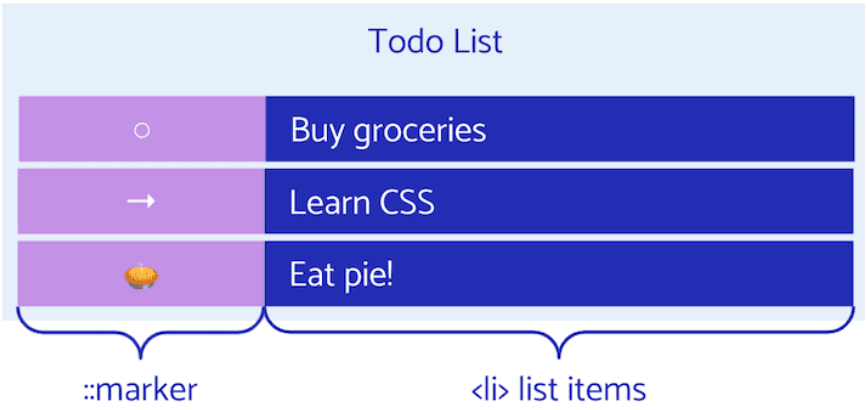
a. blur(10px)
b. brightness(80%)
c. filter: contrast(160%);
d. filter: grayscale(80%);
    i. 100% means black and white, with less contrast
e. filter: invert(1);
f. filter: opacity(0.3);
g. filter: saturate(155%);
h. The difference between backdrop-filter and filter is that the backdrop-filter property only applies the filters to the background, where the filter property applies it to the whole element.

Blend

a. normal: The default mode where no blending occurs; the element's content is rendered as usual.
b. multiply: Multiplies the background and the content colors, resulting in a darker effect.
c. screen: Multiplies the inverses of the background and the content colors, resulting in a lighter effect.
d. overlay: Combines multiply and screen modes. The background colors are multiplied or screened, depending on the content color.
e. darken: Keeps the darker color between the background and the content.

Lists

a. list-style-type: "🛒";



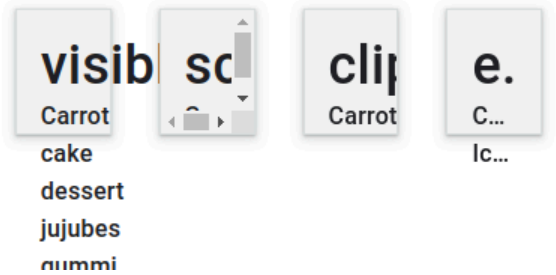
Transition

a. transition: transform 300ms ease-in-out 0s;

b. .longhand {
  transition-property: transform;
  transition-duration: 300ms;
  transition-timing-function: ease-in-out;
  transition-delay: 0s;
}

Overflow

```
.box {
  width: 15vw;
  height: 20vh;
}

.scroll {
  overflow: scroll;
}

.clip {
  overflow: clip;
}

.ellipsis p, .ellipsis h1 {
  text-overflow: ellipsis;
  overflow: hidden;
  white-space: nowrap;
}
```



a. overflow: hidden scroll;
    i. Here overflow-x : hidden and overflow-y: scroll
b. There is difference between text overflow and element overflow.
    i. Text overflow is generally about inline overflow, where element overflow is about block overflow.
c. Different values of overflow
    i. visible: The overflow is not clipped. It renders outside the element's box. This is the default value.
    ii. hidden: The overflow is clipped, and the rest of the content is invisible.
    iii. scroll: The overflow is clipped, but a scrollbar is added to see the rest of the content.
    iv. auto: Similar to scroll, but a **scrollbar is added only if the content overflows.**
d. Scroll behavior
    i. control the behavior for a scrolling box when scrolling is triggered by anchor links, JavaScript, or user actions.
    ii. Values:
        1. auto: Default instant scrolling.
        2. smooth: Smooth scrolling.
e. Overscroll bahevior

i. controls the behavior when the scroll position of a scroll container reaches the edge of the scroll range.
ii. Suppose you have a parent div , and a child div, where both are overflowing. While scrolling the child div, if we hit the top or bottom or child div, and scroll even further, then the scroll will be applied to parent div
iii. We wouldnt want such thing to happen, so to trap that scroll inside child div, we can use this overscroll bahevior.

Background

a. Background-repeat:

i. By default, background images repeat horizontally and vertically to fill the entire space of the background layer.
ii. So, we can use background-repeat property. It has following values :
    1. repeat:
        a. The image repeats within the space available, cropping as necessary.
    2. Round :
        a. The background image is repeated as many times as will fit without clipping. If it doesn't fit a whole number of times, the image is scaled down to fit.
        b. The image will be scaled if necessary to ensure that it fits exactly into the container without clipping.
    3. Space :
        a. background image is repeated as many times as will fit without clipping, and the extra space is distributed around the images.
iii. Repeat-x : The background image is repeated only horizontally.
iv. Repeat-y : The background image is repeated only vertically.
v. No-repeat : The background image is not repeated.

b. Background-position

i. set the initial position of a background image within its container.
ii. background-position: x-position y-position;
iii. Key words
    1. left, center, right: Horizontal positions.
    2. top, center, bottom: Vertical positions.
iv. Percentages
    1. 0% 0% places the background at the top-left corner.
    2. 100% 100% places it at the bottom-right corner.
    3. 50% 50% centers the background image.
v. Px values
    1. background-position: 10px 20px;
       This will specify exact positions from the top-left corner.

c. **Background-size :**
    i. control how a background image is scaled and displayed within an element.Keyword Values:
    ii. Values :
        1. auto: Default value. The background image is displayed at its original size.
        2. cover: Scales the background image to cover the entire container, potentially cropping the image to maintain its aspect ratio.
        3. contain: Scales the background image to be as large as possible without cropping or stretching the image, fitting entirely within the container.

d. **Background-attachment**
    i. Fixed
        1. **Parallax effect**
        2. The background image remains fixed in place relative to the viewport, even when the content is scrolled.
    ii. Scroll
        1. The background image scrolls with the rest of the content.
        2. default behavior.
    iii. Local
        1. The background image scrolls with the element's content.
        2. If the element has a scrolling mechanism (like overflow: auto), the background image will move with the element's content.

e. **Background-origin**
    i. border-box
        1. The background image starts from the upper-left corner of the border box (includes the border area).
    ii. Padding-box
        1. The background image starts from the upper-left corner of the padding box (excludes the border but includes the padding).
    iii. Content-box
        1. The background image starts from the upper-left corner of the content box (excludes the border and padding).

background-origin: border-box

background-origin: content-box

background-origin: padding-box

f. Background-clip

i. How far image extends
ii. determines the area within which the background image (or color) is visible
iii. Same values as background-origin
    1. Padding-box
    2. Border-box
    3. content-box

background-clip: padding-box

background-clip: content-box

background-clip: border-box

g. Multiple backgrounds

```
background-image: url("https://assets.codepen.io/7518/pngaaa.com-1272986.png"),
    url("https://assets.codepen.io/7518/blob.svg"),
    linear-gradient(hsl(191deg, 40%, 86%), hsla(191deg, 96%, 96%, 0.5));
  background-size: contain, cover, auto;
  background-repeat: no-repeat, no-repeat, no-repeat;
  background-position: 50% 50%, 10% 50%, 0% 0%;
  background-origin: padding-box, border-box, content-box;
```