FRONTEND WEB DEVELOPMENT PROGRAM

Lecture 2: Layout, Responsive Web Design

BeaconFire

OUTLINE

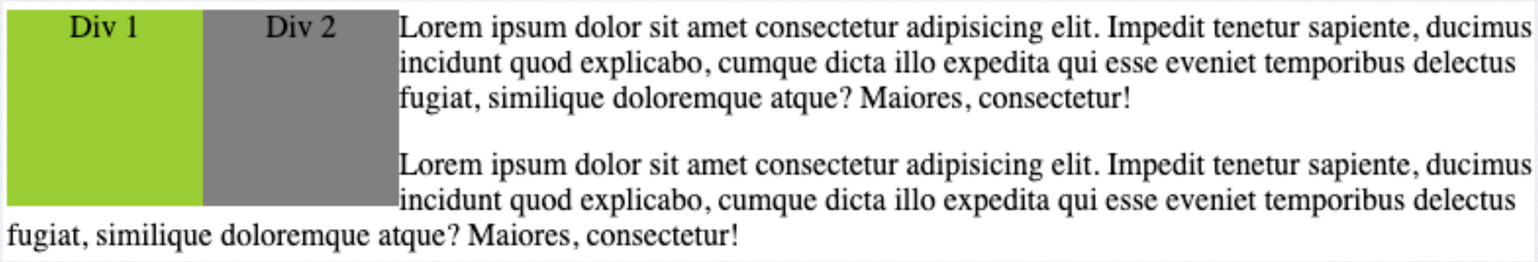
- Layout (Flow, Float, Clear, Position, Z-index)
- CSS Layout Modules (Flexbox, Grid)
- Transforms, Transitions, Animations
- Responsive Web Design
- CSS Preprocessors
- Bootstrap vs Tailwind
- Design Principles

FLOW

- The **flow** of a page refers to whether elements on the page affect or are affected by the position of other elements.
  - In flow: block and inline elements work as expected, they're stacked along each other
  - Out of flow: element positions are independent of each other; they act like they're on their own layer, and don't take up space on the page

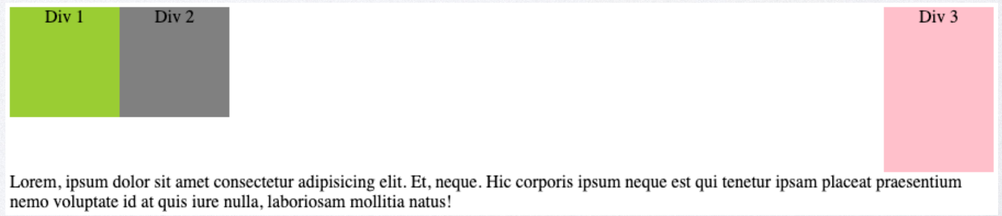
FLOAT

- **float**: controls position and formatting of content on a page, often for wrapping text around images or other elements
  - **none (default), left, right, inherit**



CLEAR

- **clear**: used on an element next to a floating element, making this element appear below
  - **none** (default), inherit
  - **left**: appear below "float: left" element
  - **right**: appear below "float:right" element
  - **both**: appear below the larger of the "float:left" & "float:right" elements



POSITION

- This property is required if you want to set the properties top, right, bottom, and left, which specify a distance from that side of an initial position. If none are specified, the element uses the normal position.
  - Positive values are away from that side, and negative values are towards that side.
  https://css-tricks.com/almanac/properties/t/top-right-bottom-left/

- **position**: specifies the initial position from which the element is offset
  - **static** (default): not affected by top, right, bottom, left
  - **relative**: offset is relative to the normal position
  - **absolute**: offset is relative to nearest positioned ancestor or viewport (out of flow)
  - **fixed**: offset is relative to the viewport, stays when scrolling (out of flow)
  - **sticky**: acts like relative until you scroll past the offset position, then becomes fixed, unless it leaves the parent (out of flow?)

Z-INDEX

- **z-index**: specifies the stack order of an element (which element is placed in front of or behind others).
  - Only works on positioned elements (position: absolute/relative/fixed/sticky) and flex items.

FLEXBOX

- The **Flexible Box Layout Module** makes it easier to design flexible, responsive layouts (changes based on viewport or content size).
  - It focuses on a **1-D** layout (dimensions of items, spacing between them).
  - Flexbox uses a **content-first approach**, where the contents are defined first, resulting in a flexible layout.

- **Flex container**: wrapper elements that are styled with "**display: flex**".

- **Flex item**: direct child elements of a flex container.

```
.flex-container {
  display: flex;
}
```

BeaconFire

FLEX CONTAINER

- **flex-direction**: the way that the flex items will be arranged or stacked.
- **flex-wrap**: whether the flex items should wrap around to the next line or not.
- **flex-flow**: shorthand property for setting flex-direction and flex-wrap.
- **justify-content**: align the flex items in their main axis (direction they're stacked).
- **align-items**: align the flex items in their cross axis.
- **align-content**: align the flex lines (rows or columns).

FLEX ITEM

- **order**: order the items by number, ascending
- **flex-basis**: the initial length of this item, in the direction of the main axis
  - flex-direction: row => width, column => height
- **flex-grow**: a factor by which this item grows relative to the others
- **flex-shrink**: a factor by which this item shrinks relative to the others
- **flex**: shorthand for grow/shrink/basis
- **align-self**: specify the item's vertical alignment, overriding container's align-items

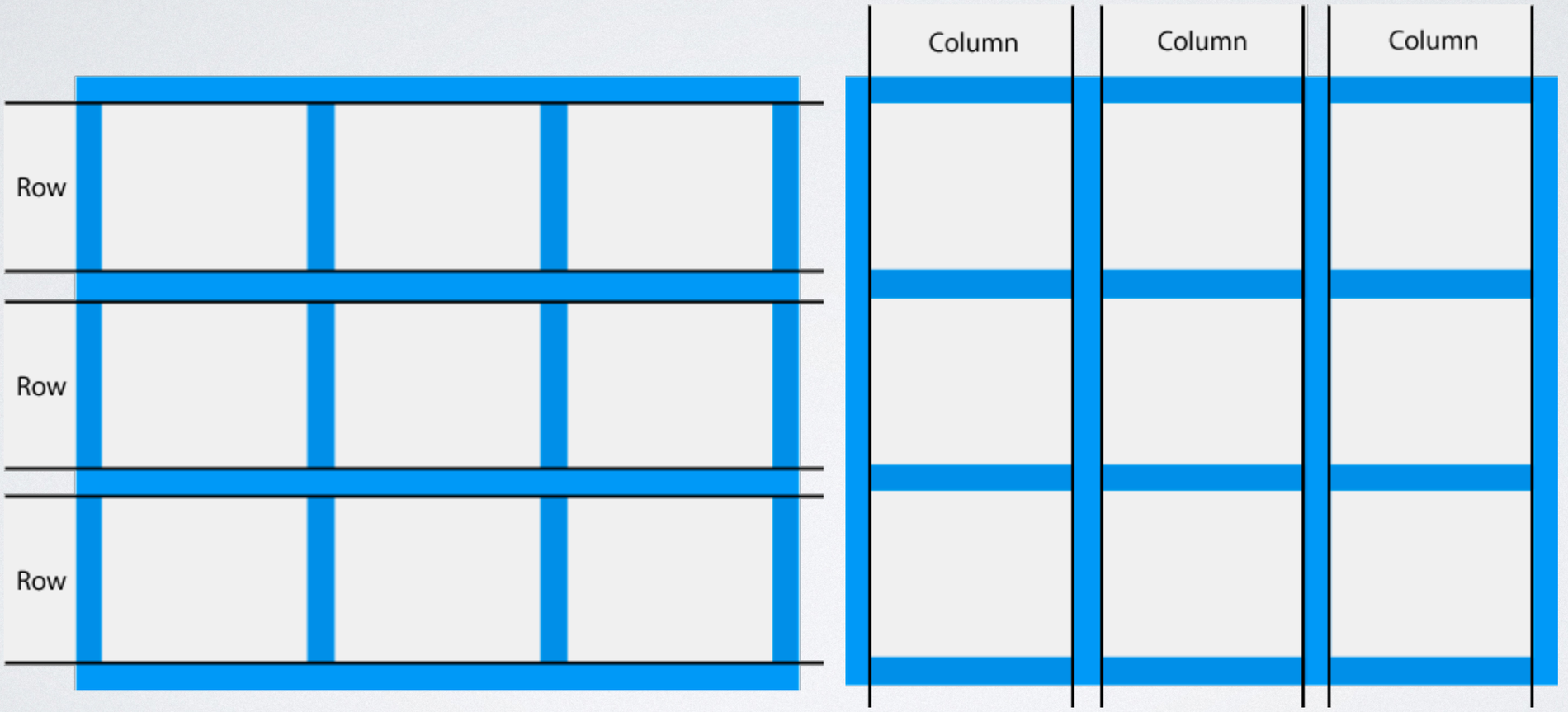
GRID

- The **Grid Layout Module** makes it easier to design a grid-based layout system via rows and columns.
  - It focuses on a **2-D** layout (dimensions of rows and columns, gaps between them).
  - Grid uses a **layout-first approach**, where the layout is defined first, then the content is placed in specific positions.
  - **fr**: grid-exclusive 'fraction' unit, accounts for weird behavior (overflowing items, gaps, margins).
  - **repeat(num, length)**: repeats a length to create many equal-sized rows & columns

- **Grid container**: an element styled with "**display:grid**" or "**display:inline-grid**"
  Grid gap: the space between rows and columns
  Grid item: direct child element of a grid container
  Grid line: the lines between and outside of rows and columns

https://www.w3schools.com/css/css_grid.asp

GRID ROWS & COLUMNS



GRID CONTAINER

- **grid-template-columns**: a space-separated list of lengths, each representing the width of a column

  grid-template-rows: a space-separated list, each representing the height of a row

```
grid-template-columns: auto auto auto;
grid-template-columns: repeat(3, auto);
```

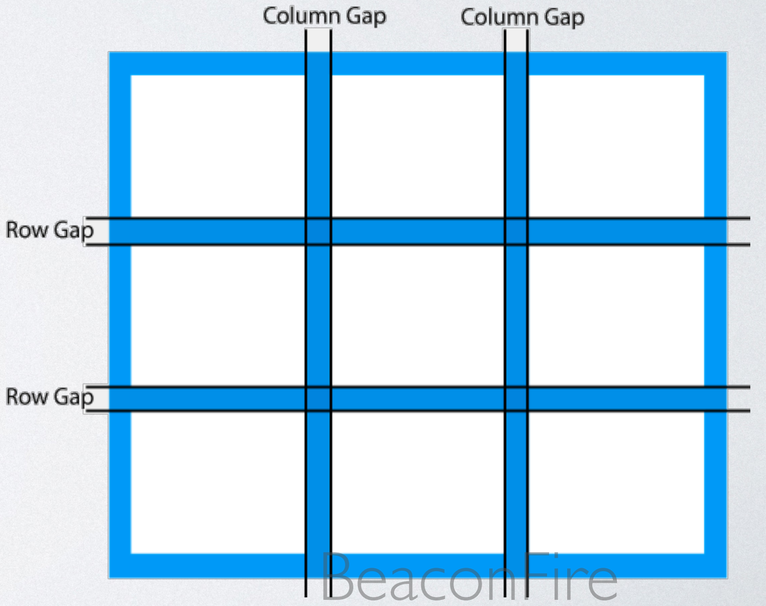
- **justify-content**: horizontally align the entire grid. (grid width < container width)

  align-content: vertically align the entire grid. (grid height < container height)

- **column-gap**: specify length for the gaps between columns

  row-gap: specify length for the gaps between rows

  gap: shorthand for row-gap, column-gap

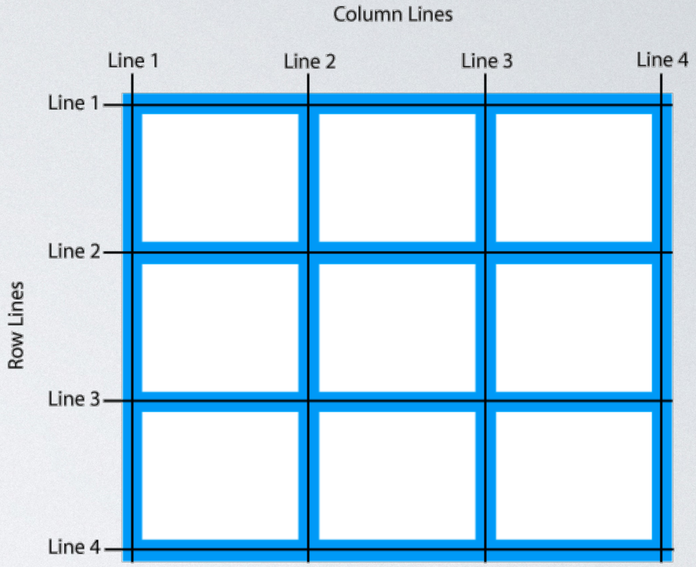


GRID ITEM (LINES)

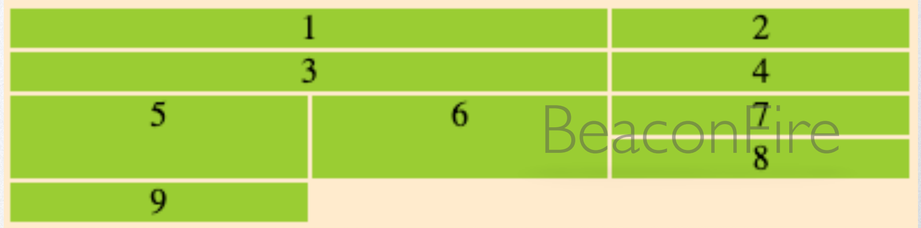
- Use grid lines to make them span multiple rows or columns and/or rearrange the order of the items.

- **grid-column-start**: an item's starting column line number
  grid-column-end: an item's ending column line number
  grid-column: shorthand, use "starting line / ending line"
  - "starting line / span (number of columns)"

- **grid-row-start**: an item's starting row line number
  grid-row-end: an item's starting row line number
  grid-row: shorthand, use "starting line / ending line"
  - "starting line / span (number of rows)"




```
<div class="grid-container">
  <div class="grid-item" style="grid-column: 1 / 3;">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item" style="grid-column: 1 / span 2;">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item" style="grid-row: 3 / 5;">5</div>
  <div class="grid-item" style="grid-row: 3 / span 2;">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

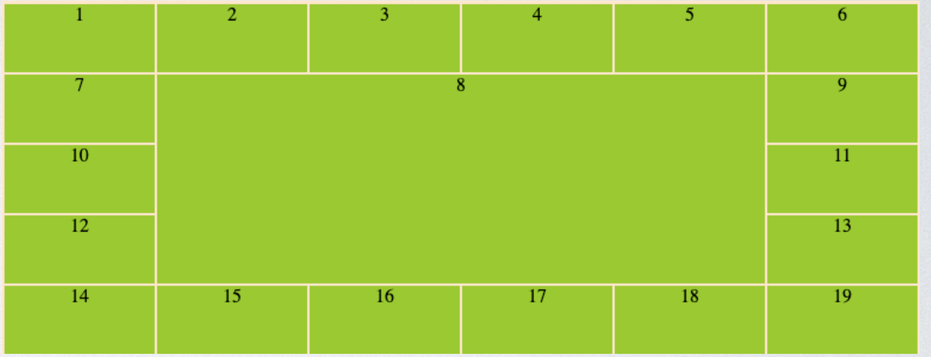


GRID ITEM (AREA)

- Use grid-area as a shorthand for adjusting grid lines (grid-row and grid-column) OR to assign a name to a grid item that'll be used for grid-template-areas.

- **grid-area**: row-start / col-start / row-end / col-end (top / left / bottom / right)
  - areaName

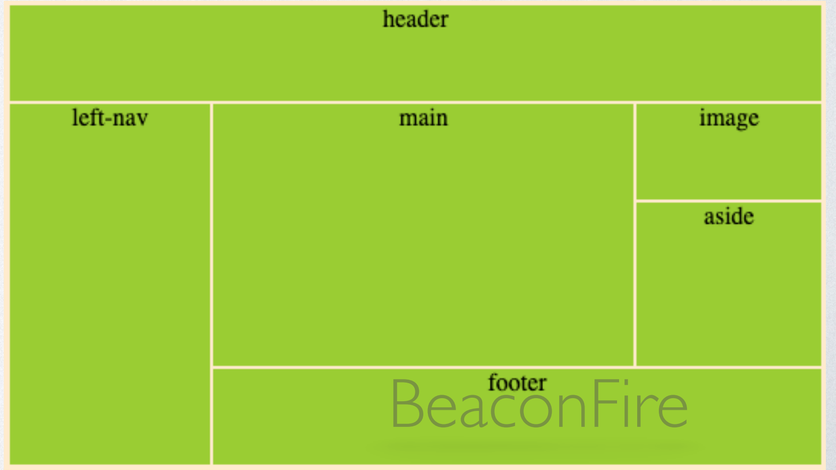
```
<div class="grid-item">6</div>
<div class="grid-item">7</div>
<div class="grid-item" style="grid-area: 2/2/5/6;">8</div>
<div class="grid-item">9</div>
```



- **grid-template-areas**: textual representation of the grid, where each cell is the areaName or . (no item)

```
<div class="grid-container template">
  <div class="grid-item" style="grid-area: header;">header</div>
  <div class="grid-item" style="grid-area: footer;">footer</div>
  <div class="grid-item" style="grid-area: main;">main</div>
  <div class="grid-item" style="grid-area: image;">image</div>
  <div class="grid-item" style="grid-area: aside;">aside</div>
  <div class="grid-item" style="grid-area: left-nav;">left-nav</div>
</div>
```

```
.template {
  grid-template-areas:
    'header header header header header'
    'left-nav main main main image'
    'left-nav main main main aside'
    'left-nav main main main aside'
    'left-nav footer footer footer footer';
  height: 300px;
}
```



TRANSFORMS

- CSS Transforms: move, rotate, scale, skew elements' shape and/or position

- **transform**: apply 2D or 3D transformations to an element
  - **translate(x,y)**: move an element in the direction specified by the x, y distances
  - **rotate(angle)**: rotate an element clockwise or counter-clockwise by the given degree
  - **scale(width, height)**: increase or decrease size of an element by the given factors for width & height
  - **skew(angleX, angleY)**: skew an element along the X,Y axis by the x,y angles
  - **matrix(scaleX, scaleY, skewX, skewY, translateX, translateY)**: shorthand for the others

- 3D transforms take another argument for the z-axis

- **transform-origin**: change the position of transformed elements
  - x-axis, y-axis, z-axis

```css
#transform {
  background-color: blue;
  /* transform-origin: x-axis y-axis z-axis; */
  /* transform-origin: bottom left; */
  /* transform-origin: 50px 50px; */
  /* transform-origin: bottom right 70px; */
}

.container:hover>#transform {
  transform: rotate(90deg);
  /* transform: rotate(.25turn); */
  /* transform: rotateX(180deg); */
  /* transform: rotateY(180deg); */

  /* transform: scale(0.5); */
  /* transform: scale(50%); */
  /* transform: scale(0.5, 2); */
  /* transform: scaleX(0.5); */
  /* transform: scaleY(2); */

  /* transform: translate(50px, 50px); */
  /* transform: translateX(100%); */
  /* transform: rotate(90deg) scale(0.5); */
}
```

TRANSITIONS

- CSS Transitions: create a smooth change in CSS property values over a given duration from A => B

- **transition**: shorthand for setting all 4 properties below
  transition-property: name of CSS property to change
  - all (default), none, {comma-separated list of properties}
  transition-duration: how many seconds/milliseconds the transition takes
  transition-timing-function: the speed curve for the transition
  - ease (default), linear, ease-in, ease-out, ease-in-out, cubic-bezier(n,n,n,n)
  transition-delay: how long to wait before beginning the transition

```css
#transition {
  background-color: chocolate;

  /* transition: property duration timing-function delay */

  transition: 1s ease;
  /* transition: 1s ease-in; */
  /* transition: 1s ease-out; */
  /* transition: 1s ease-in-out; */

  /* transition: 1s ease 1s; */
}

.container:hover>#transition {
  transform: rotate(90deg) scale(1.5);
  background-color: yellow;
  width: 500px;
  font-size: 100px;
}
```

ANIMATIONS

- CSS Animations: create a smooth change in CSS property values over a given duration from A => B => C => …

- **Keyframe**: describes what style an element has at a particular time (playing a sequence of these creates the animation)
  @keyframes: a rule that defines the changes in the style over time, which must be bound to an element

- **animation**: shorthand for setting all 6 properties below
  animation-name: name of the @keyframes animation
  animation-duration: how long the animation takes to finish 1 cycle (default: 0)
  animation-timing-function: the speed curve for the animation
  - ease (default), linear, ease-in, ease-out, ease-in-out, cubic-bezier(n,n,n,n)
  animation-delay: how long to wait before beginning the animation
  animation-iteration-count: how many times this animation should be played
  animation-direction: whether we play the animation forward/backward/in alternate cycles
  - normal (default), reverse, alternate, alternate-reverse

- **animation-fill-mode**: specify an element style for when the animation isn't active
  - none (default), forwards, backwards, both

```css
@keyframes around {
  25% {
    transform: translateX(100%);
    background-color: aliceblue;
  }

  50% {
    transform: translate(100%, 100%);
    background-color: aqua;
  }

  75% {
    transform: translateY(100%);
    background-color: cornflowerblue;
  }

  100% {
    transform: translate(0);
    background-color: dodgerblue;
  }
}

.container:hover>#animation {
  /* animation: name duration timing-function delay iteration-count direction */

  animation: around 1s ease-in-out 2;
  /* animation: around 1s ease-in-out infinite; */
  /* animation: around 1s ease-in-out reverse; */
  /* animation: around 1s ease-in-out infinite alternate; */
}
```

TRANSITION VS ANIMATION

CSS Transitions

- Can only move from initial to final state — no intermediate steps
- Can only run once
- Require a trigger to run (like mouse hover)
- Run forwards when triggered and in reverse when trigger is removed
- Easier to use with JavaScript
- Best for creating a simple change from one state to another

CSS Animations

- Can move from initial to final state, with intermediate steps in between
- Can loop infinitely thanks to animation-iteration-count property
- Can be triggered but can also run automatically
- Can run forwards, in reverse, or alternate directions
- More difficult to use with JavaScript
- Best for creating a complex series of movements

RESPONSIVE WEB DESIGN

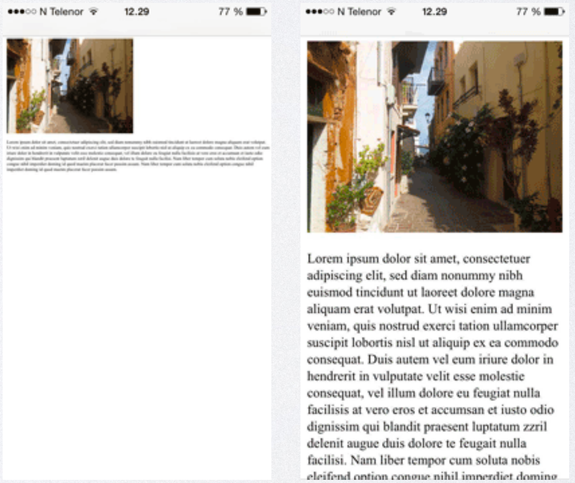
- **Responsive web design (RWD)** is a design approach where you use HTML & CSS to make the website UI look great on all devices and screen sizes.
  * Users scroll vertically — adapt to smaller devices with vertical layouts

- **Mobile-first approach**: design for mobile devices before any other devices (displays faster). Layout: single-column => multi-column.

- **Viewport**: the part of the webpage that the user can see
  - Need in **<meta>** for page to recognize device size.

- What can we do to implement RWD?
  - Flexbox (grow, shrink), Grid (fr)
  - Images with <picture>
  - Set font-sizes, width/height with rem, em, vw/vh, %, auto and min/max lengths
  - Media queries


```
<meta name="viewport" content="width=device-width, initial-scale=1">
```



BeaconFire

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design#historic_website_layouts

MEDIA QUERIES

- **@media**: a rule for applying a block of CSS properties when the given condition is true
  - ex: based on the viewport size, orientation (portrait/landscape)
- **Media types**: all, print (printers), screen (desktop, mobile devices), speech (screenreaders)
- **Media features**: min-width, min-height, orientation, resolution, …
- **Breakpoint**: the condition for when a media rule takes effect
- **Mobile-first approach**: the initial styles are for smaller devices, add rules for when the viewport lengths get larger (min-width, min-height)

```
@media not|only mediatype and (mediafeature and|or|not mediafeature) {
  CSS-Code;
}
```

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

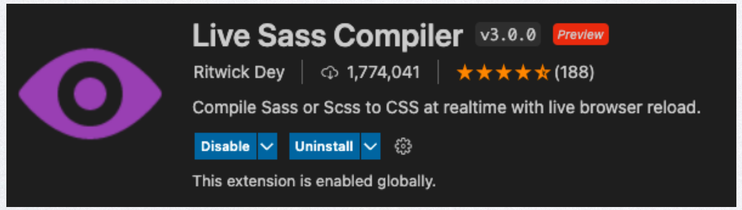
CSS PREPROCESSOR

- **CSS preprocessor**: a program with its own unique, CSS-based syntax that is transpiled into regular CSS.

- Most CSS preprocessors' syntax will include features that make the CSS structure more readable and easier to maintain.
  - ex: variables, mixins, nested rules, inheritance, extending rules

- Some of the most popular CSS preprocessors are SASS & SCSS, LESS, Stylus.

- Recommended:



SASS

- **Syntactically Awesome Style Sheets (SASS)**: "the most mature, stable, and powerful CSS extension language in the world". There are two SASS variations (SCSS and SASS), both with their own distinct file extensions (.scss and .sass).
  - CSS compatibility: you can write CSS in .scss, but not .sass files.

- SCSS: no strict indentation, curly braces, semicolons

- SASS: strict indentation, no braces, new lines

- https://sass-lang.com/install
  https://sass-guidelin.es/#the-7-1-pattern

```scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

```sass
$font-stack: Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```

VARIABLES

- **Variables ($)**: $varName
  - Not the same as CSS variables, so compatibility isn't an issue
  - Can be declared anywhere
  - Only one value at a time

```scss
$font-stack: Helvetica,sans-serif;
$primary-color: gray;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

NESTING SELECTORS

- You can nest CSS selectors to create a parent-child hierarchy, which will be converted into CSS with the corresponding selectors:
  - parent child

- **&**: refers to the parent selector in nested CSS blocks, which is useful for pseudo-class/elements

```scss
ul {
  background-color: aqua;
  li {
    color: coral;
    &:hover {
      background-color: palevioletred;
    }
    &:first-child {
      color: darkblue;
    }
    &::first-letter {
      font-size: 30px;
    }

    span {
      font-size: 20px;
      font-style: oblique;
    }
  }
}
```

```css
ul {
  background-color: aqua;
}

ul li {
  color: coral;
}

ul li:hover {
  background-color: palevioletred;
}

ul li:first-child {
  color: darkblue;
}

ul li::first-letter {
  font-size: 30px;
}

ul li span {
  font-size: 20px;
  font-style: oblique;
}
```

PARTIALS

- **Partials (_)**: stylesheets that can be imported into other style sheets (similar to JS modules being imported).
  - They will not be transpiled into CSS.
  - When importing them with **@import**, you don't need "_" or the file extension.

_header.scss

```
@import "header";
```

MIXIN

- **@mixin**: defines a rule that can be copied & reused throughout the stylesheet.
  - It can accept arguments.
  - Does not get transpiled to CSS.

- **@include**: copies all the mixin declarations into another block.

- Use this when you want reusable rules that can be customized with arguments.

```scss
@mixin flexCenter {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
  border: 2px solid orange;
}

.flex-lg {
  @include flexCenter;
  width: 80%;
}
.flex-sm {
  @include flexCenter;
  width: 50%;
}
```

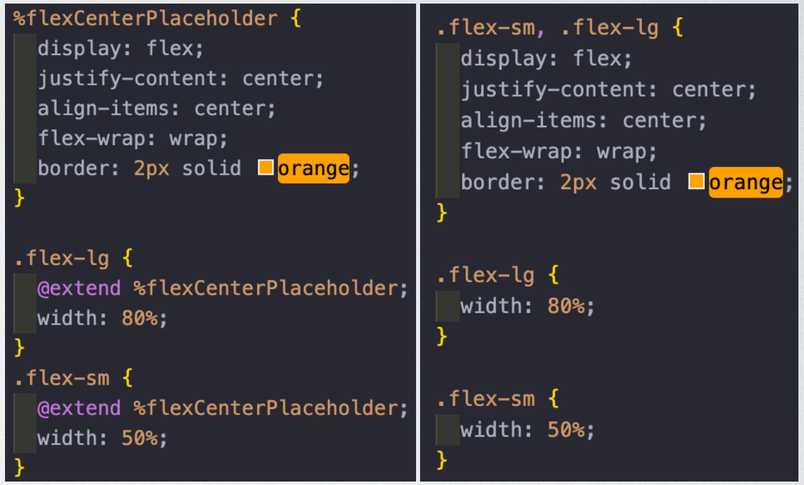
```scss
.flex-lg {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
  border: 2px solid orange;
  width: 80%;
}

.flex-sm {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
  border: 2px solid orange;
  width: 50%;
}
```

INHERITANCE

- **Placeholder (%)**: defines a rule that can be reused (extended) throughout the stylesheet.
  - Does not accept arguments.
  - Does not get transpiled to CSS.

- **@extend**: makes a rule inherit the declarations from another rule or placeholder.
  - Shares the common declarations via grouping selector (,).
  - Creates a separate rule for unshared declarations.

- Use this for inheritance (when there's a meaningful relationship between selectors).



```scss
%flexCenterPlaceholder {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
  border: 2px solid orange;
}

.flex-lg {
  @extend %flexCenterPlaceholder;
  width: 80%;
}
.flex-sm {
  @extend %flexCenterPlaceholder;
  width: 50%;
}
```

```scss
.flex-sm, .flex-lg {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
  border: 2px solid orange;
}

.flex-lg {
  width: 80%;
}

.flex-sm {
  width: 50%;
}
```

INTERPOLATION

- **Interpolation (#{})**: pass arguments to a @mixin and use it to create dynamic CSS selectors or styles.

```scss
@mixin corner-icon($name) {
  .icon-#{$name} {
    width: 100px;
    height: 100px;
    margin: 10px;
  }
}

#img-container {
  height: 200px;
  width: 700px;
  border: 2px orange solid;

  @include corner-icon("google");
}
```

```scss
#img-container {
  height: 200px;
  width: 700px;
  border: 2px orange solid;
}
#img-container .icon-google {
  width: 100px;
  height: 100px;
  margin: 10px;
}
```

BOOTSTRAP

- Bootstrap is one of the world's most popular front-end open-source toolkits, which features built-in UI components, pre-defined SASS variables and mixins, a responsive grid system, and powerful JavaScript plugins.

- To import Bootstrap into your .html file, either:
  - Download Bootstrap from the official website and include it as a source file in your HTML.
  - Include the link from a Content Delivery Network (CDN)
    - https://cdnjs.com/

- https://getbootstrap.com/docs/5.2/getting-started/introduction/

.MIN

- When importing external libraries, there's always a .min and regular file.
  - The .min represents a compressed file (lightweight, unreadable, used for production).
  - The regular file is a normal-sized CSS file that is human-readable and used for development.

```
webkit-text-size-
adjust:100%}body{margin:0}article,aside,details,figcaption,figure,footer,header,hgroup,main,menu,nav,section,summary{display:block}aud
io,canvas,progress,video{display:inline-block;vertical-align:baseline}audio:not([controls]){display:none;height:0}
[hidden],template{display:none}a{background-color:transparent}a:active,a:hover{outline:0}abbr[title]{border-bottom:none;text-
decoration:underline;-webkit-text-decoration:underline dotted;-moz-text-decoration:underline dotted;text-decoration:underline
dotted}b,strong{font-weight:700}dfn{font-style:italic}h1{font-size:2em;margin:.67em 0}mark{background:#ff0;color:#000}small{font-
size:80%}sub,sup{font-size:75%;line-height:0;position:relative;vertical-
align:baseline}sup{top:-.5em}sub{bottom:-.25em}img{border:0}svg:not(:root){overflow:hidden}figure{margin:1em 40px}hr{-webkit-box-
sizing:content-box;-moz-box-sizing:content-box;box-sizing:content-box;height:0}pre{overflow:auto}code,kbd,pre,samp{font-
family:monospace,monospace;font-
size:1em}button,input,optgroup,select,textarea{color:inherit;font:inherit;margin:0}button{overflow:visible}button,select{text-
transform:none}button,html input[type=button],input[type=reset],input[type=submit]{-webkit-
appearance:button;cursor:pointer}button[disabled],html input[disabled]{cursor:default}button::-moz-focus-inner,input::-moz-focus-
inner{border:0;padding:0}input{line-height:normal}input[type=checkbox],input[type=radio]{-webkit-box-sizing:border-box;-moz-box-
```

```
html {
  font-family: sans-serif;
  -ms-text-size-adjust: 100%;
  -webkit-text-size-adjust: 100%;
}
body {
  margin: 0;
}
article,
aside,
details,
figcaption,
figure,
footer,
header,
hgroup,
main,
menu,
nav,
section,
summary {
  display: block;
}
```

BOOTSTRAP (REBOOT)

- **Reboot**: stylesheet with element-specific CSS changes and rules for pre-defined classes, such that all styling can be done with classes
  - globally sets "box-sizing: border-box"
  - globally sets font-family/weight/line-height/color
  - removes margin-top defaults, recommends a single direction of margin (bottom)

- Removes margin-top from and adds margin-bottom to the following elements:
  - <h1…h6>, <p>, <ul>, <ol>, <dl>

BOOTSTRAP (BREAKPOINTS)

- **Mobile-first responsiveness**: apply the bare minimum of styles to make the layout work at the smallest breakpoint, then layer on styles for larger devices

Breakpoint	Class infix	Dimensions
Extra small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

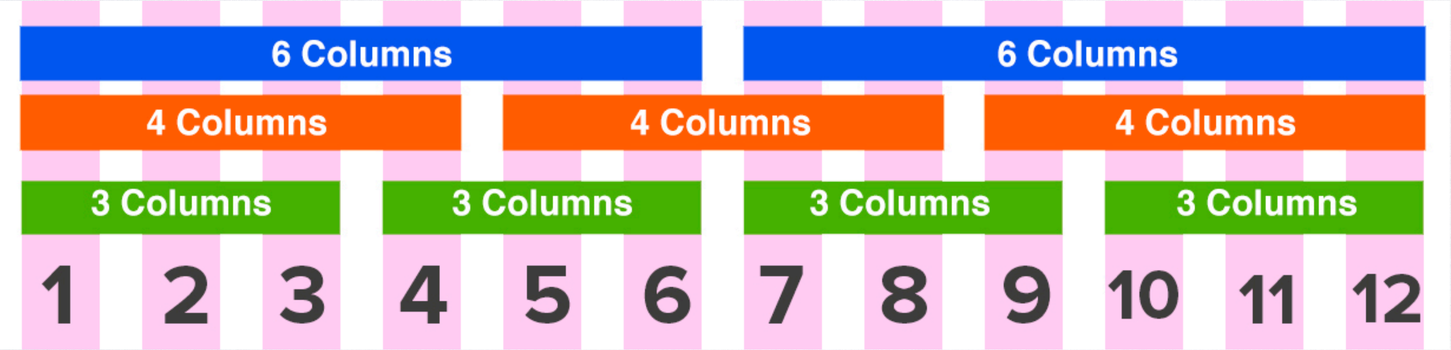
BOOTSTRAP (CONTAINERS)

- Container classes are the most basic layout element in Bootstrap and are required when using the default grid system. They add padding and alignment.
  - **.container:** width is fixed at each bootstrap breakpoint.
  - **.container-{breakpoint}:** width: 100% until this breakpoint.
  - **.container-fluid:** width: 100% at all breakpoints.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

BOOTSTRAP (GRID)

- The Bootstrap grid system is fully-responsive, based on a 12-column layout, and built with flexbox.
  - Requires a series of containers, rows, and column classes.
  - Column elements must be the children of row elements.




```
<div class="container-fluid">
  <h2 class="text-primary">Boot
  <div class="row">
    <div class="col-sm-1" style
    <div class="col-sm-2" style
    <div class="col-sm-3" style
  </div>
</div>
```

INHERITANCE, CASCADE, SPECIFICITY

- **Inheritance**: some styles are inherited throughout the HTML document tree (font properties, text color)
  - The **inherit** keyword gives a declaration the same value as its parent.
  - http://www.javascriptkit.com/dhtmltutors/cssreference.shtml

- **Cascade**: an algorithm that determines which style/stylesheet to prioritize and apply to a document (think, which is closest to the element?)
  1: inline style (author)
  2: internal style (author)
  3: external style (author)
  4: online external style (user)
  5: default browser

- **Specificity**: an algorithm that determines which rules to prioritize and apply to an element, represents how specific a selector is
  1: id selectors
  2: class & pseudo-class selectors
  3: element selectors

- **ORDER MATTERS**: In the same stylesheet, if two rules have the same specificity, we apply the rule that was declared **last.**
  - In all cases, the applied rule doesn't overwrite all declarations, only those that conflicted.
  - Inline styles are the highest priority & specificity doesn't apply (you don't use selectors).
  - The **!important** keyword makes a declaration have the highest specificity, so it overrides all other declarations (be careful).

!IMPORTANT

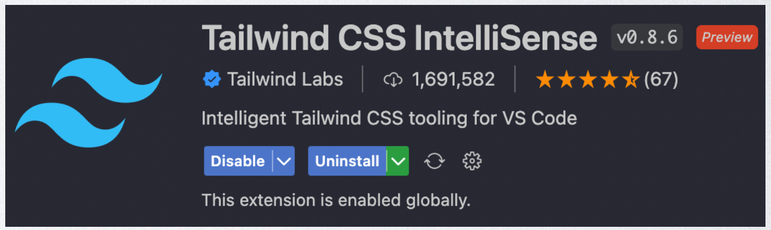
- The *!important* rule in CSS is used to prioritize a particular style (property:value).

- Avoid using *!important* if possible.
  - The only way to override an !important rule is to include another !important rule on a style property with the same or higher priority.
  - This may result in you trying to apply styles that you think should work, but they don't because a style was set elsewhere with !important.

TAILWIND

- **Tailwind**: a utility-first CSS framework that works on a lower-level than Bootstrap.
  - Instead of using "semantic class names", you add utility classes to an element based on the properties you want to change.
  - There are no default, element-specific CSS changes.
  - You don't need to write your own CSS and create class names.

```
<div class="p-6 max-w-sm mx-auto bg-white rounded-xl shadow-lg flex items-center
  <div class="shrink-0">
    <img class="h-12 w-12" src="/img/logo.svg" alt="ChitChat Logo">
  </div>
  <div>
    <div class="text-xl font-medium text-black">ChitChat</div>
    <p class="text-slate-500">You have a new message!</p>
  </div>
</div>
```



- https://tailwindcss.com/

WHAT IS "GOOD DESIGN"?

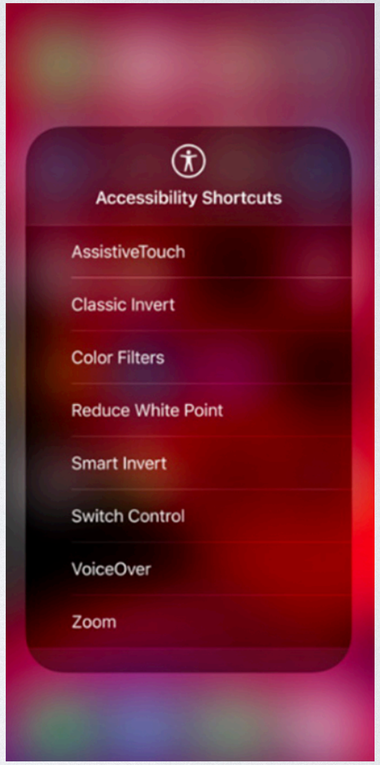
- *Accessibility, usability, and inclusion* are closely related aspects in creating a web that works for everyone.
- Their goals, approaches, and guidelines overlap significantly.
- It is most effective to address them together when designing and developing websites and applications.

ACCESSIBILITY

- **Accessibility**: addresses discriminatory aspects related to equivalent user experience for people with disabilities.
  - People with disabilities can equally perceive, understand, navigate, and interact with websites and tools.

- ex: captions for noisy environments, screen readers, screen/text magnification for the elderly, text contrasts & line height in lower ambient light, line length & wrapping

- <label>, aria-label, title



USABILITY

- **Usability**: designing products to be effective, efficient, and satisfying. Usability includes *user experience design*.
- This may include general aspects that impact everyone and do not disproportionally impact people with disabilities.
- Usability practice and research often does not sufficiently address the needs of people with disabilities.

INCLUSION

- **Inclusion**: universal design & design for all, diversity and ensuring the involvement of everyone to the greatest extent possible.
  - ex: consider socioeconomic differences, demographics, language/geographic barriers, technical constraints (hardware/internet connectivity), age, mental/physical impairments, …

- **Localization (l10n)**: adapting to meet the language, culture, and other requirements of some specified market/audience
  - Date & timezone, currency, culture-sensitive text & graphics

- **Internationalization (I18n)**: design the product with localization in mind (adaptable to various cultures, regions, and languages)

- **Globalization (G11n)**: targeting all audiences

ANY QUESTIONS?