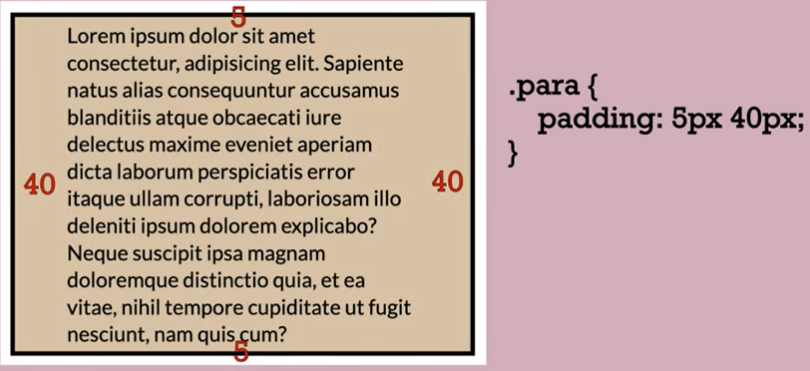
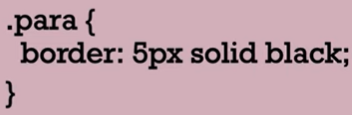
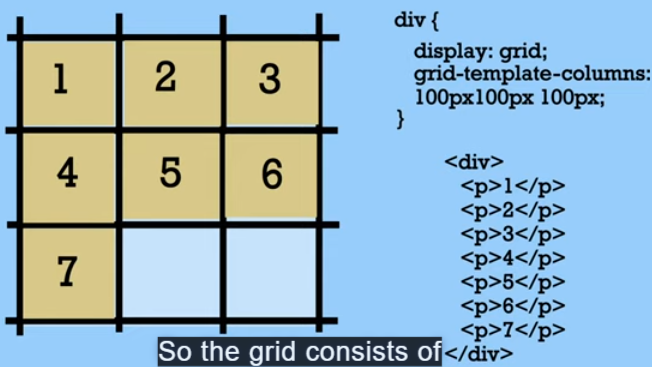
Useful notes CSS

* Units of measurement
  + Width, margin, font-size, etc.
  + px; em; or %
    - no space between number and unit (15em)
    - 1em = 24px on standard page
    - 1em = 100% of standard font size
    - 1.5em = 150%
  + absolute measurements – are same size no matter how large the page is
    - px (pixel)
  + relative measurements – scale based on certain things
    - em or %
    - better for programming because it will work with phones or computers
* Block vs Inline Elements
  + Every element is inside its own invisible box – the box can be a block or inline space
    - Boxes – default take up 100% of the space they have – don’t allow other elements to share the line they are on
      * Ex: div, header, main, footer, nav, p, ul, li, section, article, form, h1-h6
    - Inline – allow other elements to share the same line until its full
      * Ex: a, img, span, button, label
    - Turn a block into an inline by: target the code in css with {display: inline;}
    - Turn an inline into a block by: target the code in css with {display: block;}
    - You can also make something an inline-block: {display: inline-block}
  + Centering
    - Text-align: center
      * Content is centered inside of invisible box
    - Margin 0 auto:
      * The box itself is centered
      * If it takes up the whole page, you won’t see it unless you give the box itself a width
      * The first says to set the top and bottom margins to 0 and the second says to set the left and right margins to auto: meaning to split any margin available between the left and right.
    - For images, you need to make it a block or put it inside a block level element, then center it with margin 0 auto;
  + CSS Selectors and Specificity
    - Can target in CSS with many things; p, h1, div, ID names (#[name]), class names (.[name])
    - You can also be specific on the order the code was written
      * Useful for fonts; you can set a font for the whole page at the top of the body, but then later on you can give a h1 a different font that will override the base font, because it comes later in the code
    - Specificity
      * Most specific > ID > class > elements (p, div, h1) – more specific will override less specific
    - Can select multiple things with commas
      * Ex: h1, h4 {font-family: ‘Lemon’}
    - Can select all things within an element
      * Ex: article p {font-family: ‘Lemon’}
        + This will only select elements of “p” under the parent element article
    - Pseudo-selectors
      * You can just select the nth element
        + Ex: li:nth-of-type(2) {background-color: #ccc;}

2 selects the 2nd element within a list

* + - Pseudo-classes
      * Allow to change appearance of an element when a user is interacting with it
        + Hover over a button and it changes color
        + Ex: nav a:hover {color: black; background-color: blue} button:hover {background-color: white}
* Box Model
  + 
  + Block
    - Width is auto 100%
    - Height will expand or shrink dependent on content
  + Inline
    - Width will hug the size of the element
    - Height will expand or shrink dependent on content
  + Padding – white space, put space inside the elements border
    - Useful if you but a border or background on the element
    - If you give 2 values, the first value is given to the height and the second value is given to the width
      * 
    - You can give 4 values, it starts on the top value and goes clockwise
    - Can apply it to just one side with padding-right: padding-top, etc.
  + Border – to show a border you need 3 elements; width, style, and color
    - 
    - Styles: double, dashed, dotted
    - Border-radius – rounds the edges
      * Border-radius: 50% will make the element a circle if it was a square beforehand
  + Margin – gap between borders of two elements
  + If we have the width of an element at 500px, this does not include padding and border
    - If padding were 20px and the border was 10px, then the actual width of the element is 560px
      * Box-sizing: border-box; will make the padding and border included in the elements width
      * Default box-sizing: content-box
* Browser Developer Tools
  + Chrome developer tool – right click the element you want to inspect 🡪 click inspect
    - You can toggle things on and off to see how the page would look
    - You can see colors, fonts, sizes, etc.
    - You can also toggle to different devices and screen sizes to see how the page would look like on different screens
* CSS Layout
  + Page layout – layout of the page
  + Normal flow – elements follow the order of input
    - Take elements out of normal flow
      * Float: right, left, etc. – shift an element a direction and allow content wrapped around it
      * Position property (normally static) – relative – absolute – fixed (good for nav) – flex box (rows or columns, picss) – grid (what we will use most often, rows and columns)
        + Stacking order – need to be aware of the z index (bigger number will be in the front)
* Grid layout
  + 
  + Explicit – manually define which box an element goes into
    - Always setup columns this way
      * Grid-container {
        + Display: grid;
        + Grid-template-columns: 200px, 200px, 200px;
        + Grid-gap – space between elements}
  + Implicit – automatically defined which box what element goes in
  + ‘Fr’ measurement – fractional unit – only will take what is available – better than % (won’t make you horizontally scroll)
    - Fr fills space depending on how many columns you make
      * 1fr – takes the whole space
      * 1fr, 2fr, 3fr (1fr will take up 1/6th of the space)
      * If you want multiple columns of the same length; repeat(5, 1fr); will make 5 columns all with 1/5th the space.
  + You can nest grids inside grids
  + You want to give the parent element the declared ‘display grid’
  + Placing elements
    - [element] {
      * Grid-column-start: 1;
      * Grid-column-end: 3;
    - Or you can grid-column: 1/3; will do the same thing
      * Can do the same thing with row
  + Centering items
    - Align-self: center (to center vertically in a grid)
* **justify** aligns horizontal by default, **align** takes care of the other direction (vertical alignment by default). If we want to align the grid container, we use the **content** suffix. If we want to align the contents of the grid items within the grid cells, then we use **items**.
  + alignment properties we studied above (justify-items, justify-content, align-items, and align-content) **only work inside of a Grid**!