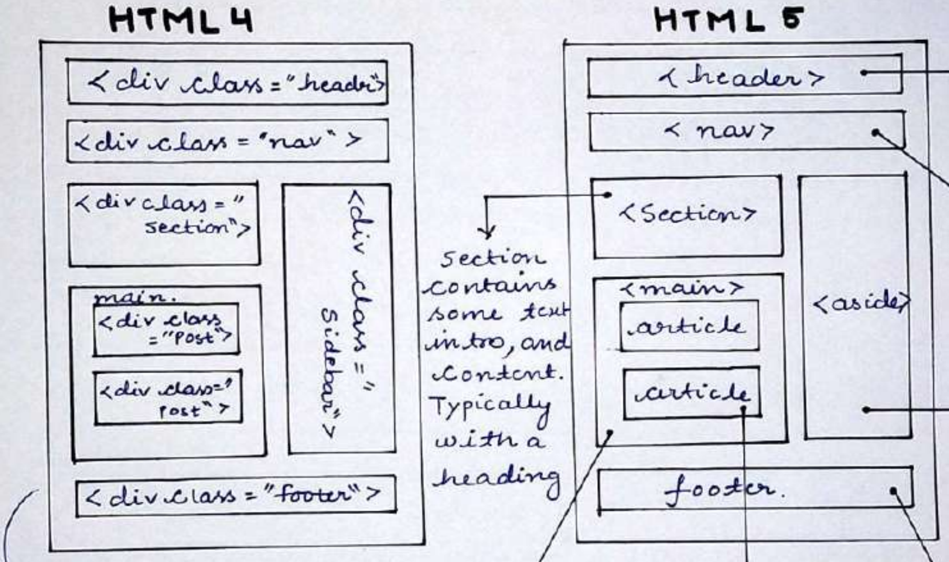
SEMANTIC ELEMENTS.

- Semantic elements are elements with some meaning.
- Semantic elements clearly describes its meaning.
- Semantic elements increase accessibility.
- Semantic elements improve the code structure and make code more readable

Some semantic elements are:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>



<header>: Introductory content for group of link. (logo, icons, owner info).

<nav>: nav is a one major group of navigation links.

<Section>: Section contains some text intro, and content. Typically with a heading

<aside>: It defines some content aside from the content it is placed in.

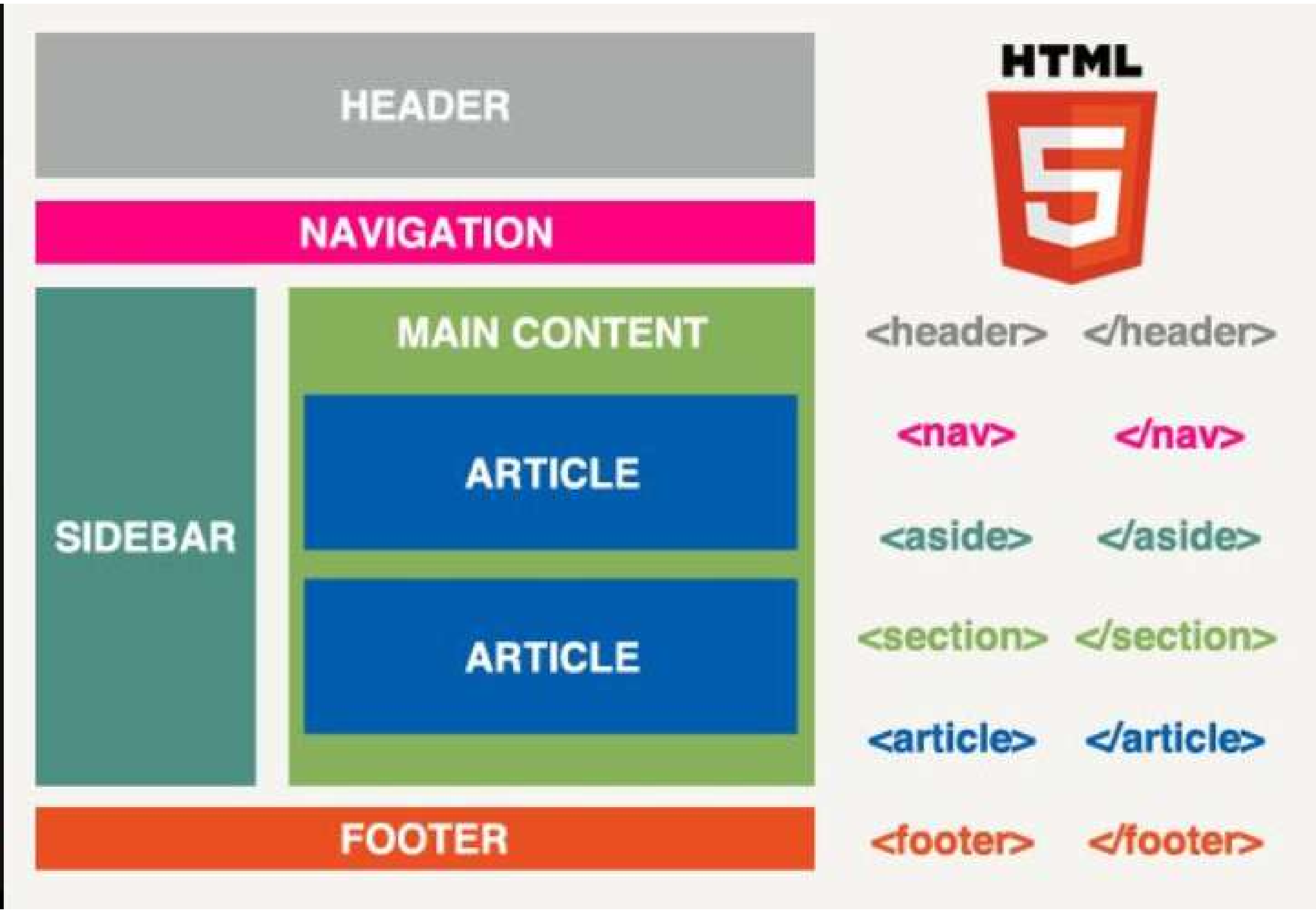
non-semantic elements like div and span. There are lots of classes and IDs which tells nothing about its content.

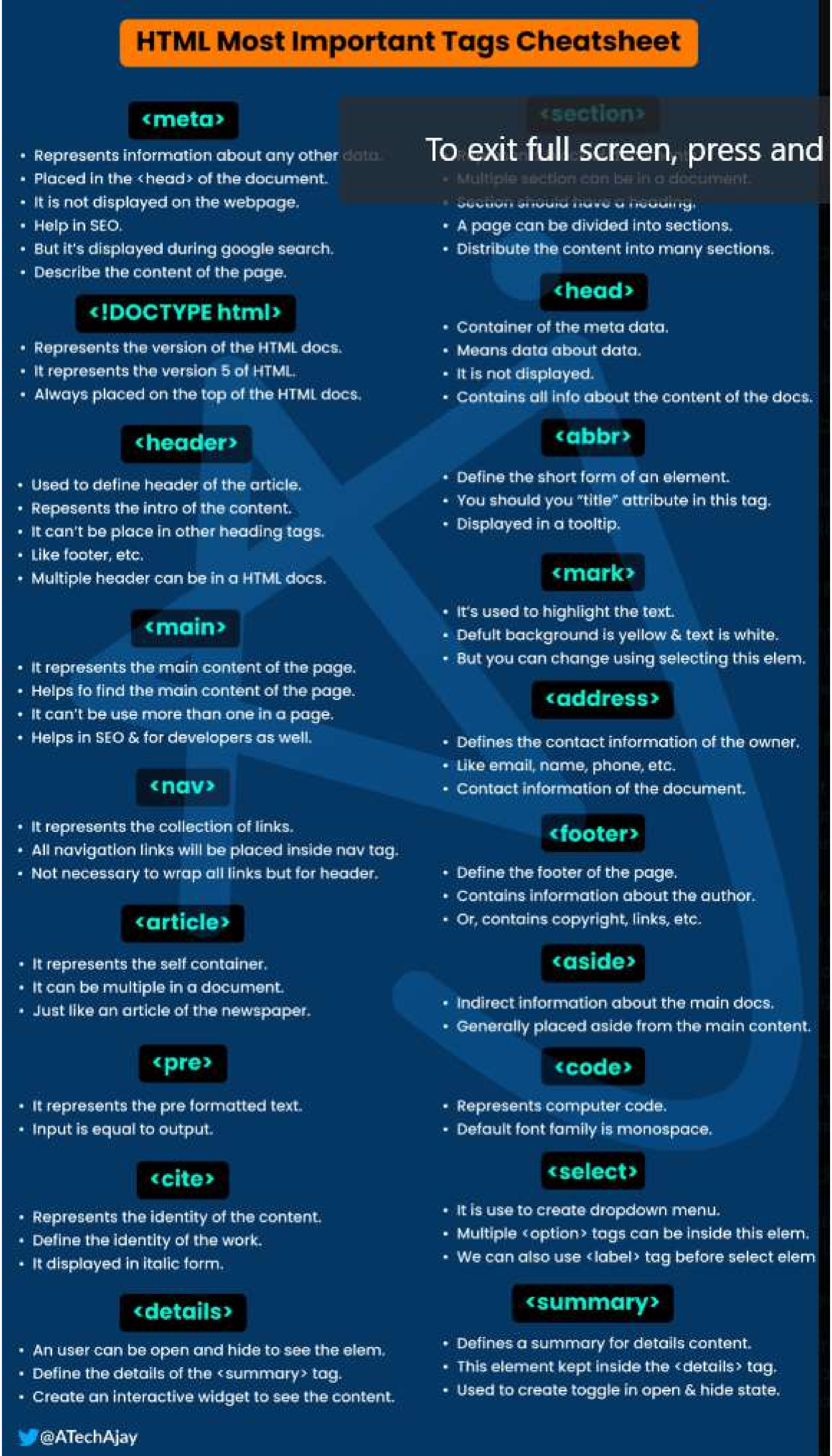
The main content of doc is specified by main element

Article contain independent, self-contained content Like a typical blog.

It defines the foot of the document

- back to top.
- copyright
- related document
- sitemap, etc...


HEADER
NAVIGATION
SIDEBAR
MAIN CONTENT
ARTICLE
ARTICLE
FOOTER
HTML
5
<header> </header>
<nav> </nav>
<aside> </aside>
<section> </section>
<article> </article>
<footer> </footer>



HTML Most Important Tags Cheatsheet

<meta>

- Represents information about any other data.
- Placed in the <head> of the document.
- It is not displayed on the webpage.
- Help in SEO.
- But it's displayed during google search.
- Describe the content of the page.

<!DOCTYPE html>

- Represents the version of the HTML docs.
- It represents the version 5 of HTML.
- Always placed on the top of the HTML docs.

<header>

- Used to define header of the article.
- Repesents the intro of the content.
- It can't be place in other heading tags.
- Like footer, etc.
- Multiple header can be in a HTML docs.

<main>

- It represents the main content of the page.
- Helps fo find the main content of the page.
- It can't be use more than one in a page.
- Helps in SEO & for developers as well.

<nav>

- It represents the collection of links.
- All navigation links will be placed inside nav tag.
- Not necessary to wrap all links but for header.

<article>

- It represents the self container.
- It can be multiple in a document.
- Just like an article of the newspaper.

<pre>

- It represents the pre formatted text.
- Input is equal to output.

<cite>

- Represents the identity of the content.
- Define the identity of the work.
- It displayed in italic form.

<details>

- An user can be open and hide to see the elem.
- Define the details of the <summary> tag.
- Create an interactive widget to see the content.

<section>

To exit full screen, press and

- [illegible]
- Multiple section can be in a document.
- Section should have a heading.
- A page can be divided into sections.
- Distribute the content into many sections.

<head>

- Container of the meta data.
- Means data about data.
- It is not displayed.
- Contains all info about the content of the docs.

<abbr>

- Define the short form of an element.
- You should you "title" attribute in this tag.
- Displayed in a tooltip.

<mark>

- It's used to highlight the text.
- Defult background is yellow & text is white.
- But you can change using selecting this elem.

<address>

- Defines the contact information of the owner.
- Like email, name, phone, etc.
- Contact information of the document.

<footer>

- Define the footer of the page.
- Contains information about the author.
- Or, contains copyright, links, etc.

<aside>

- Indirect information about the main docs.
- Generally placed aside from the main content.

<code>

- Represents computer code.
- Default font family is monospace.

<select>

- It is use to create dropdown menu.
- Multiple <option> tags can be inside this elem.
- We can also use <label> tag before select elem

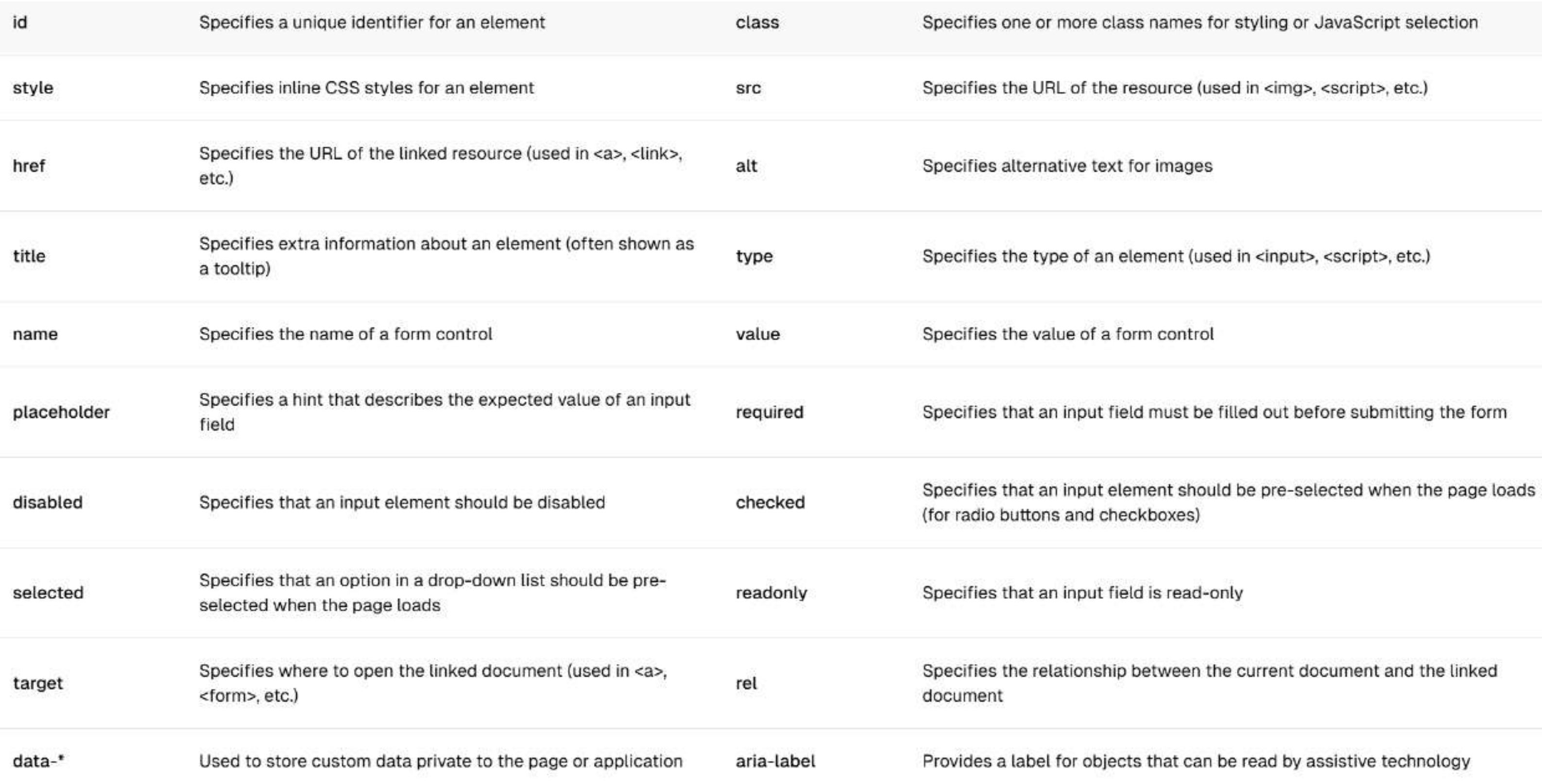
<summary>

- Defines a summary for details content.
- This element kept inside the <details> tag.
- Used to create toggle in open & hide state.

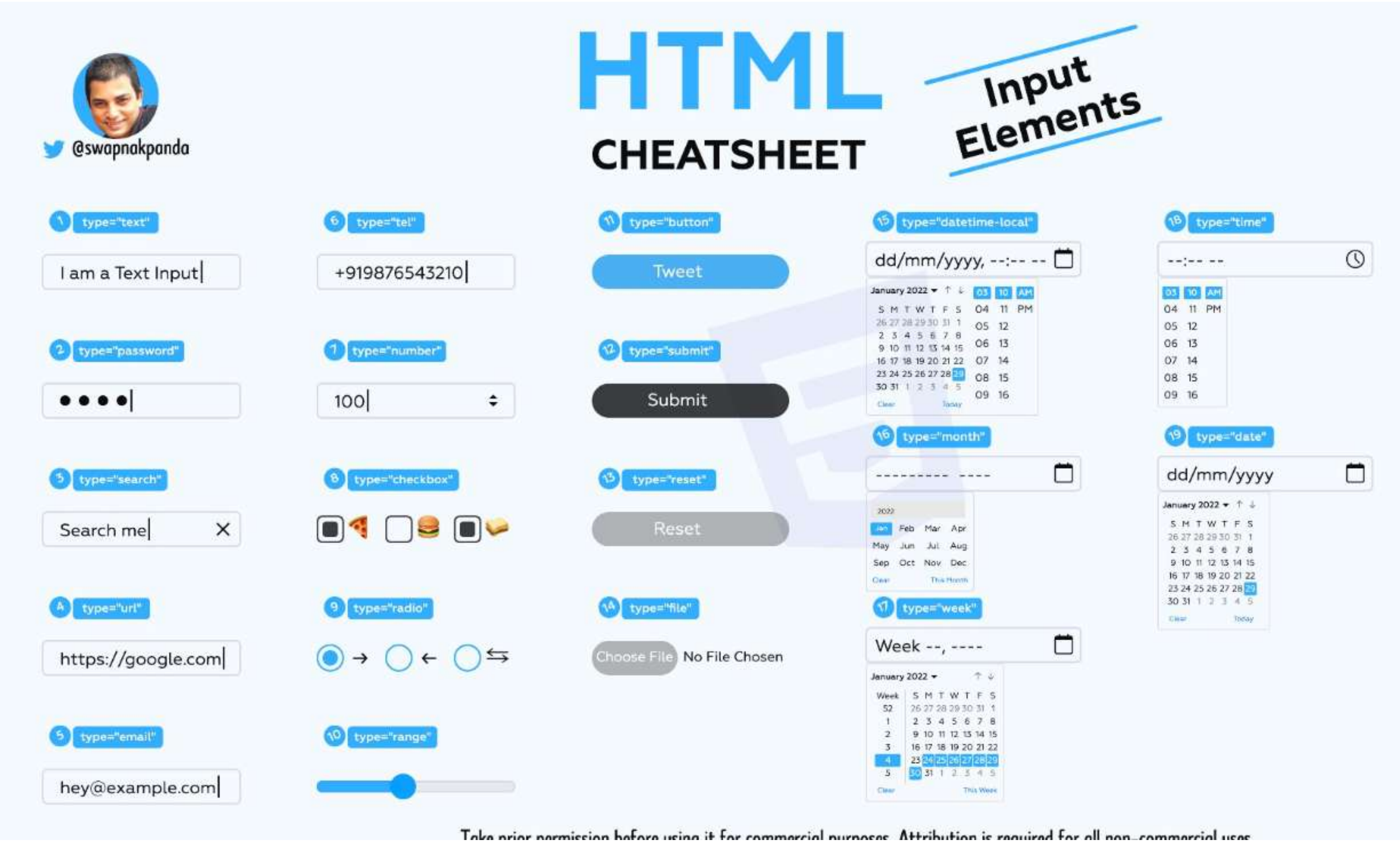
@ATechAjay

Chapter Summary

- All HTML elements can have **attributes**
- The `href` attribute of `<a>` specifies the URL of the page the link goes to
- The `src` attribute of `<img>` specifies the path to the image to be displayed
- The `width` and `height` attributes of `<img>` provide size information for images
- The `alt` attribute of `<img>` provides an alternate text for an image
- The `style` attribute is used to add styles to an element, such as color, font, size, and more
- The `lang` attribute of the `<html>` tag declares the language of the Web page
- The `title` attribute defines some extra information about an element



id	Specifies a unique identifier for an element	class	Specifies one or more class names for styling or JavaScript selection
style	Specifies inline CSS styles for an element	src	Specifies the URL of the resource (used in <img>, <script>, etc.)
href	Specifies the URL of the linked resource (used in <a>, <link>, etc.)	alt	Specifies alternative text for images
title	Specifies extra information about an element (often shown as a tooltip)	type	Specifies the type of an element (used in <input>, <script>, etc.)
name	Specifies the name of a form control	value	Specifies the value of a form control
placeholder	Specifies a hint that describes the expected value of an input field	required	Specifies that an input field must be filled out before submitting the form
disabled	Specifies that an input element should be disabled	checked	Specifies that an input element should be pre-selected when the page loads (for radio buttons and checkboxes)
selected	Specifies that an option in a drop-down list should be pre-selected when the page loads	readonly	Specifies that an input field is read-only
target	Specifies where to open the linked document (used in <a>, <form>, etc.)	rel	Specifies the relationship between the current document and the linked document
data-*	Used to store custom data private to the page or application	aria-label	Provides a label for objects that can be read by assistive technology



@swapnakpanda

HTML CHEATSHEET

Input Elements

1 type="text"
I am a Text Input

2 type="password"
●●●●

3 type="search"
Search me ✕

4 type="url"
https://google.com

5 type="email"
hey@example.com

6 type="tel"
+919876543210

7 type="number"
100

8 type="checkbox"
☒ 🍕 □ 🍔 ☒ 🧀

9 type="radio"
◉ → ○ ← ○ ⇆

10 type="range"

11 type="button"
Tweet

12 type="submit"
Submit

13 type="reset"
Reset

14 type="file"
Choose File No File Chosen

15 type="datetime-local"
dd/mm/yyyy, --:-- --

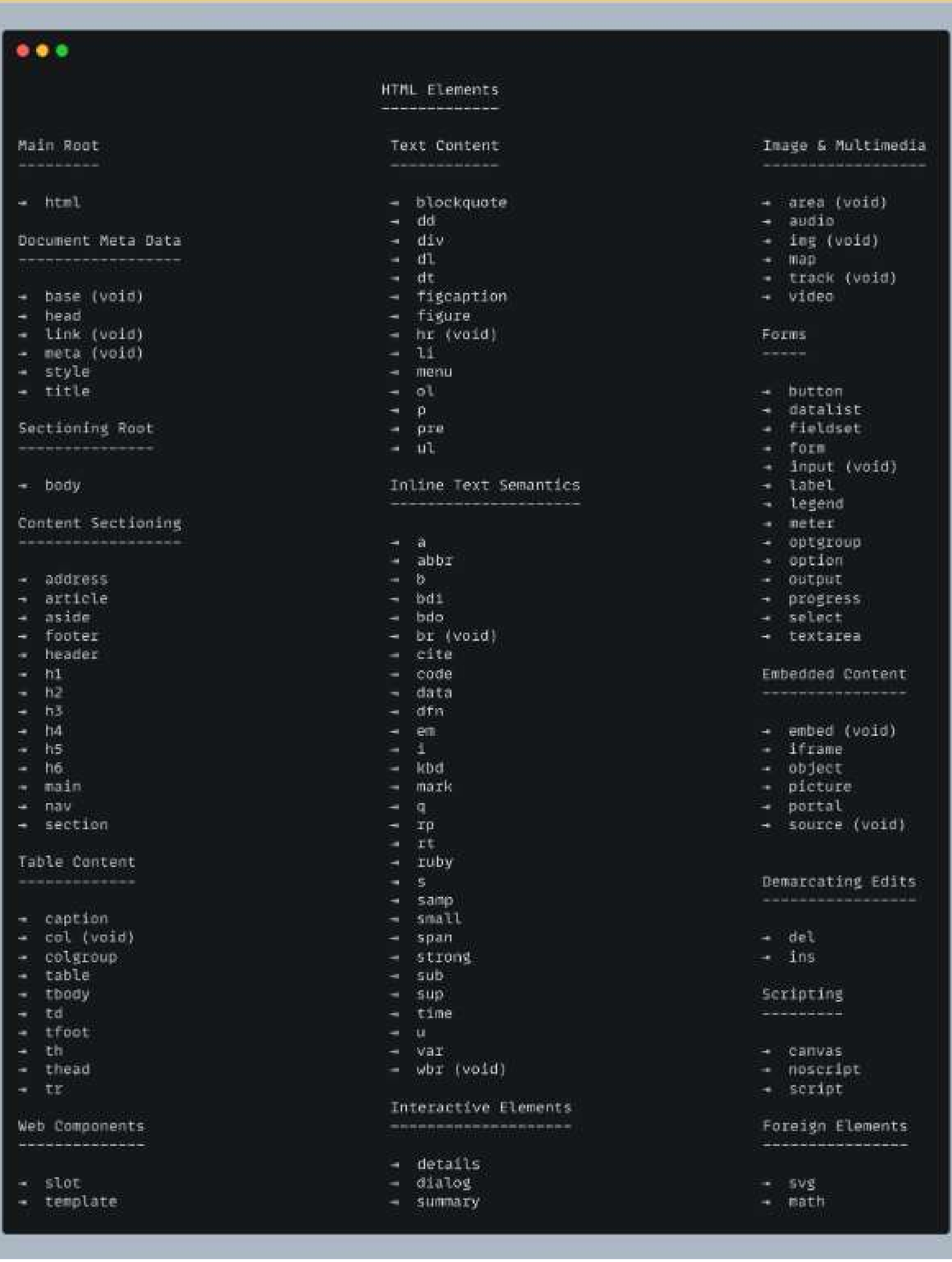
16 type="month"
--------- ----

17 type="week"
Week --, ----

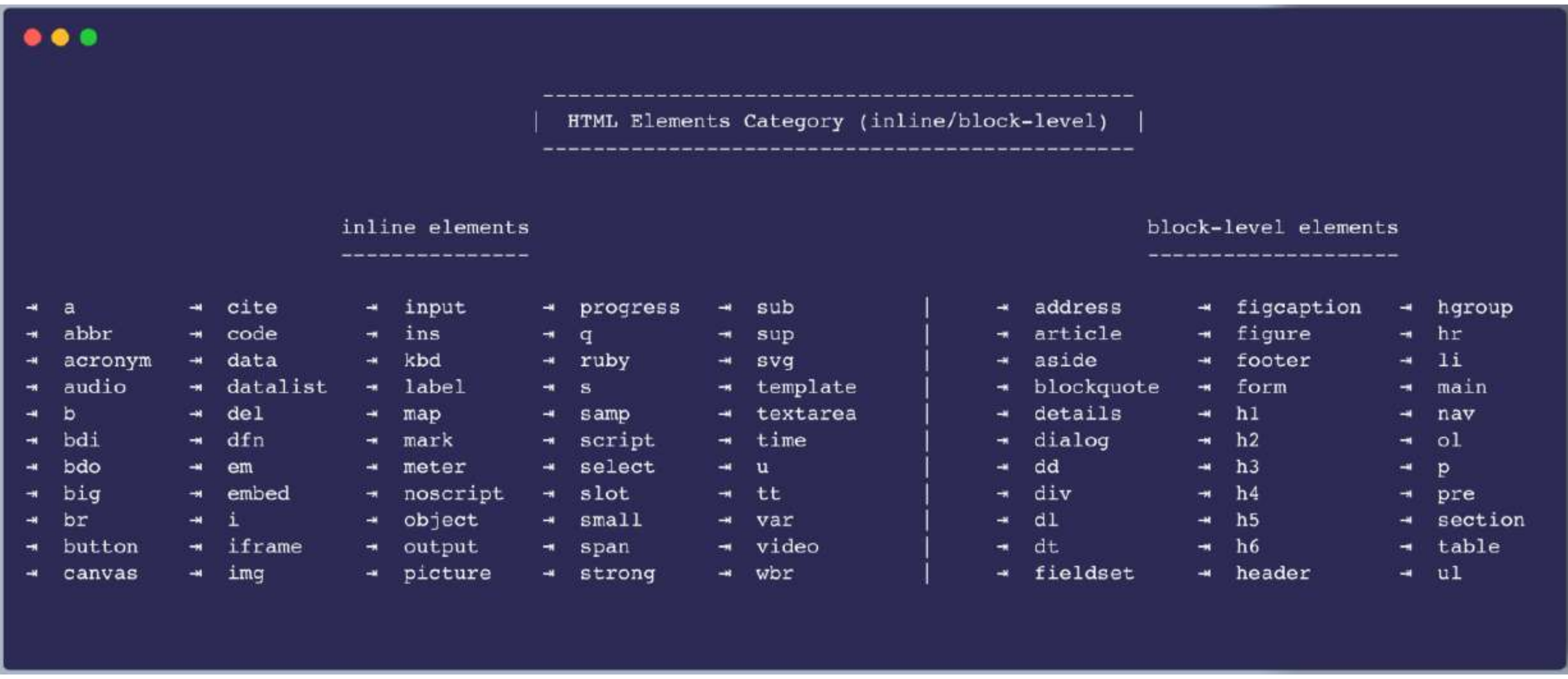
18 type="time"
--:-- --

19 type="date"
dd/mm/yyyy

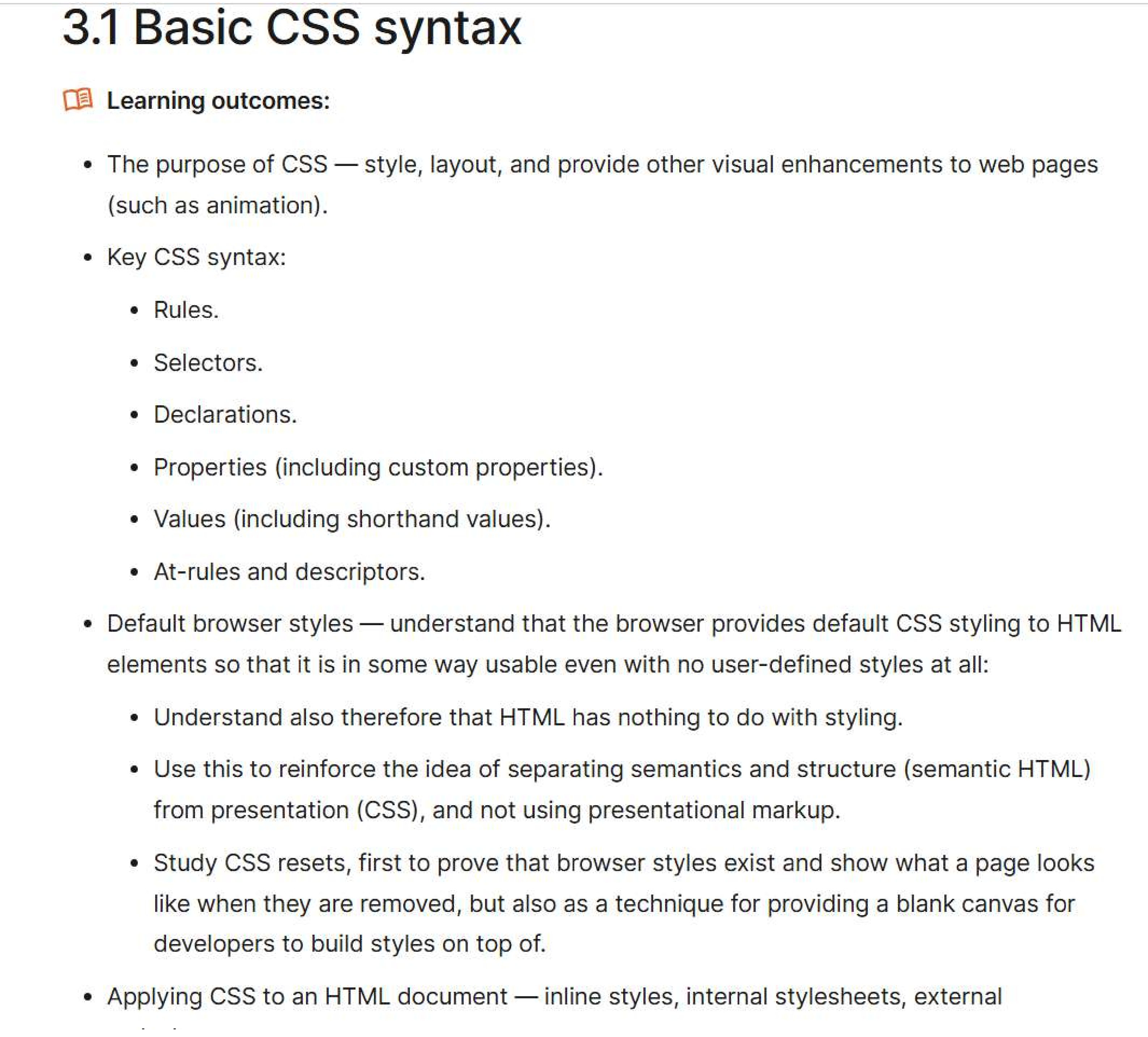
Take prior permission before using it for commercial purposes. Attribution is required for all non-commercial uses.



```
HTML Elements
-------------

Main Root
---------

→  html

Document Meta Data
------------------

→  base (void)
→  head
→  link (void)
→  meta (void)
→  style
→  title

Sectioning Root
---------------

→  body

Content Sectioning
------------------

→  address
→  article
→  aside
→  footer
→  header
→  h1
→  h2
→  h3
→  h4
→  h5
→  h6
→  main
→  nav
→  section

Table Content
-------------

→  caption
→  col (void)
→  colgroup
→  table
→  tbody
→  td
→  tfoot
→  th
→  thead
→  tr

Web Components
--------------

→  slot
→  template

Text Content
------------

→  blockquote
→  dd
→  div
→  dl
→  dt
→  figcaption
→  figure
→  hr (void)
→  li
→  menu
→  ol
→  p
→  pre
→  ul

Inline Text Semantics
---------------------

→  a
→  abbr
→  b
→  bdi
→  bdo
→  br (void)
→  cite
→  code
→  data
→  dfn
→  em
→  i
→  kbd
→  mark
→  q
→  rp
→  rt
→  ruby
→  s
→  samp
→  small
→  span
→  strong
→  sub
→  sup
→  time
→  u
→  var
→  wbr (void)

Interactive Elements
--------------------

→  details
→  dialog
→  summary

Image & Multimedia
------------------

→  area (void)
→  audio
→  img (void)
→  map
→  track (void)
→  video

Forms
-----

→  button
→  datalist
→  fieldset
→  form
→  input (void)
→  label
→  legend
→  meter
→  optgroup
→  option
→  output
→  progress
→  select
→  textarea

Embedded Content
----------------

→  embed (void)
→  iframe
→  object
→  picture
→  portal
→  source (void)

Demarcating Edits
-----------------

→  del
→  ins

Scripting
---------

→  canvas
→  noscript
→  script

Foreign Elements
----------------

→  svg
→  math
```



```
                                    ------------------------------------------------
                                   |  HTML Elements Category (inline/block-level)  |
                                    ------------------------------------------------


                      inline elements                                         block-level elements
                      ---------------                                         --------------------

⇥ a         ⇥ cite      ⇥ input     ⇥ progress  ⇥ sub       |   ⇥ address    ⇥ figcaption ⇥ hgroup
⇥ abbr      ⇥ code      ⇥ ins       ⇥ q         ⇥ sup       |   ⇥ article    ⇥ figure     ⇥ hr
⇥ acronym   ⇥ data      ⇥ kbd       ⇥ ruby      ⇥ svg       |   ⇥ aside      ⇥ footer     ⇥ li
⇥ audio     ⇥ datalist  ⇥ label     ⇥ s         ⇥ template  |   ⇥ blockquote ⇥ form       ⇥ main
⇥ b         ⇥ del       ⇥ map       ⇥ samp      ⇥ textarea  |   ⇥ details    ⇥ h1         ⇥ nav
⇥ bdi       ⇥ dfn       ⇥ mark      ⇥ script    ⇥ time      |   ⇥ dialog     ⇥ h2         ⇥ ol
⇥ bdo       ⇥ em        ⇥ meter     ⇥ select    ⇥ u         |   ⇥ dd         ⇥ h3         ⇥ p
⇥ big       ⇥ embed     ⇥ noscript  ⇥ slot      ⇥ tt        |   ⇥ div        ⇥ h4         ⇥ pre
⇥ br        ⇥ i         ⇥ object    ⇥ small     ⇥ var       |   ⇥ dl         ⇥ h5         ⇥ section
⇥ button    ⇥ iframe    ⇥ output    ⇥ span      ⇥ video     |   ⇥ dt         ⇥ h6         ⇥ table
⇥ canvas    ⇥ img       ⇥ picture   ⇥ strong    ⇥ wbr       |   ⇥ fieldset   ⇥ header     ⇥ ul
```



3.1 Basic CSS syntax

Learning outcomes:

- The purpose of CSS — style, layout, and provide other visual enhancements to web pages (such as animation).
- Key CSS syntax:
  - Rules.
  - Selectors.
  - Declarations.
  - Properties (including custom properties).
  - Values (including shorthand values).
  - At-rules and descriptors.
- Default browser styles — understand that the browser provides default CSS styling to HTML elements so that it is in some way usable even with no user-defined styles at all:
  - Understand also therefore that HTML has nothing to do with styling.
  - Use this to reinforce the idea of separating semantics and structure (semantic HTML) from presentation (CSS), and not using presentational markup.
  - Study CSS resets, first to prove that browser styles exist and show what a page looks like when they are removed, but also as a technique for providing a blank canvas for developers to build styles on top of.
- Applying CSS to an HTML document — inline styles, internal stylesheets, external

CSS SELECTORS

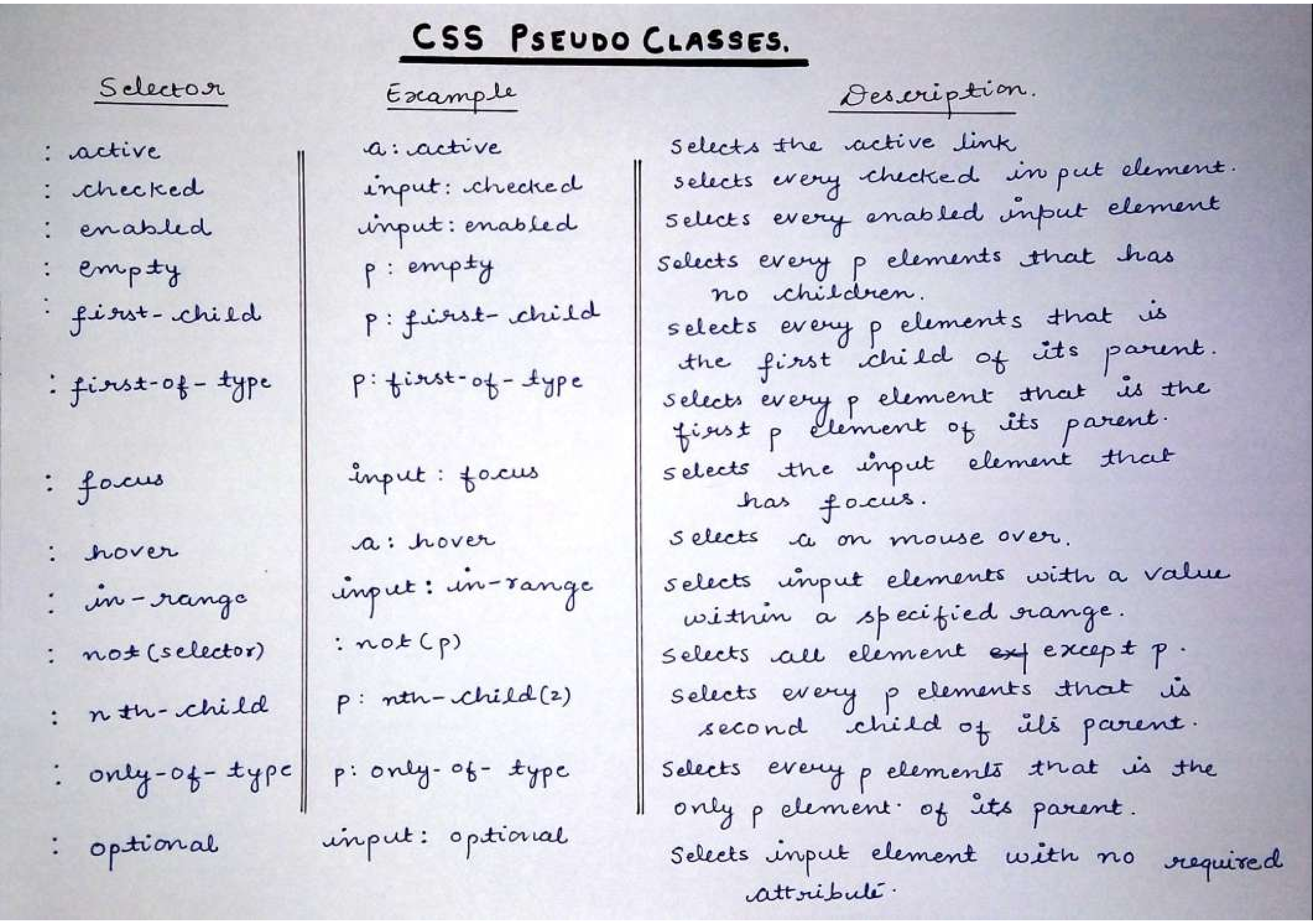
css selectors are usedto select element so that we can style them.

```
.example   //selects all elements
                with example class

# id   // selects the element
           with id = "id"

h1   // selects all h1 element

P.class // selects all p elements
              with class = "class"

div, p  // selects all div and
              p elements

div > h2  // selects all h2
                element whose
                parent is div

p ~ ul   // selects all ul that
               are preceded by P.
```

```
[target] = selects every element
                 with target attribute

[target = _parent]  // selects every
                           element with
                           att target = "_parent"

[title~ = Pratham]  // selects every
                           element with title
                           att. containing word
                           "Pratham"

[href ^= "https"]  // selects every element
                         whose href starting
                         with https

[href $ = ".png"]   // ends with . png
```

```
:not(h1)  // selects every element
               that is not h1

: root   // Selects the document
              roots elements
```

```
P: nth-child(2)   // second child of its
                        parent

P: nth-of-typ(2) // selects every p element
                        ie, second p element
                        of its parent

P: only-child // selects p thats only child
```

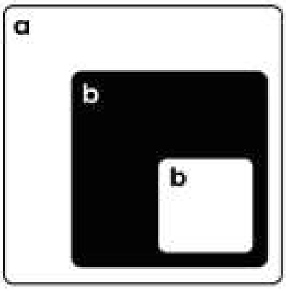


CSS PSEUDO CLASSES.

Selector	Example	Description.
: active	a : active	Selects the active link
: checked	input : checked	selects every checked in put element.
: enabled	input : enabled	Selects every enabled input element
: empty	p : empty	Selects every p elements that has no children.
: first-child	p : first-child	selects every p elements that is the first child of its parent.
: first-of-type	p : first-of-type	Selects every p element that is the first p element of its parent.
: focus	input : focus	selects the input element that has focus.
: hover	a : hover	Selects a on mouse over.
: in-range	input : in-range	selects input elements with a value within a specified range.
: not(selector)	: not(p)	Selects all element ~~ex~~ except p.
: nth-child	p : nth-child(2)	selects every p elements that is second child of its parent.
: only-of-type	p : only-of-type	Selects every p elements that is the only p element of its parent.
: optional	input : optional	Selects input element with no required attribute.

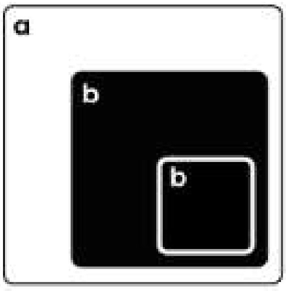
To exit full screen, press and hold Esc

CSS Selectors



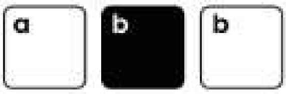
a > b Child Combinator

Select all *b* elements that are directly inside of *a* elements.



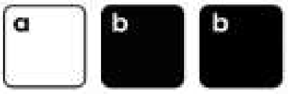
a b Descendent Combinator

Select all *b* elements that are anywhere inside of *a* elements.



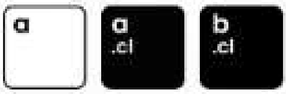
a + b **Adjacent sibling combinator**

Select all *b* elements that are immediately next to *a* elements.



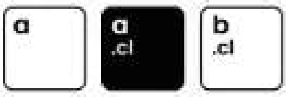
a ~ b **General sibling combinator**

Select all *b* elements that are anywhere after *a* elements.



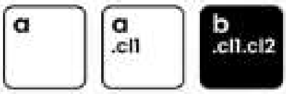
.cl **Class selector**

Select all elements that have the *cl* class name.



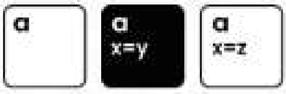
a.cl **Tag + Class selector**

Select all *a* elements that have the *cl* class name.



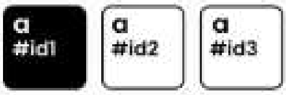
.cl1.cl2 **Multiclass selector**

Select all elements that have both the *cl1* and *cl2* class names.



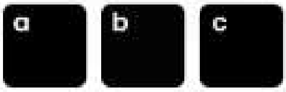
a[x=y] **Attribute selector**

Select all *a* elements that have the *x* attribute set to *y*.



#id1 **ID selector**

Select the element with the *id1* ID name.



***** **Universal selector**

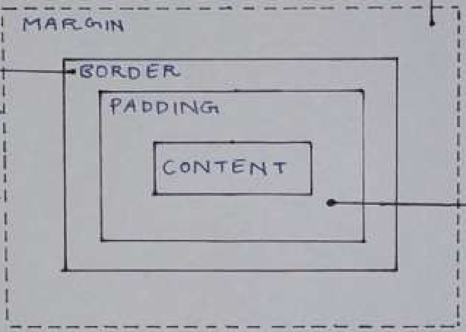
Select all elements.

eluda

CSS BOX-MODEL

- Everything in css is a box or rather everything in HTML is a box-model which is surrounded by 4 different box virtually.

1. Content (original content inside element)
2. Padding (create space b/w ~~an~~ content and element's border)
3. Border (create border around element)
4. Margin (space between element)



You can create border around element by specifying the width, color and style.

for ex: border: 1px solid black

Solid: □

dotted : ⬚

dashed : ⬚

double : ⧈

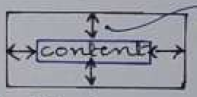
Margin defined the space between element.

for ex. margin: 10px;

It will create 10px empty space around element in all direction.

```
margin: 25px 5px 6px 10px;
         ↓    ↓   ↓    ↓
        top right bottom left
margin: 25px 10px 25px
        // Top, right and left, bottom
margin: 25px, 10px;
        // Top, bottom; and right, left
```

Padding allows you to create space between content and element's boundary for ex.

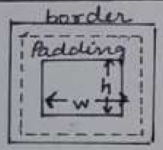


padding: 10px;

// similarly as margin, you can pass four, three, two or one value in padding as well.

you can write padding: 10%;

%- specify a padding or margin in % of the containing element.



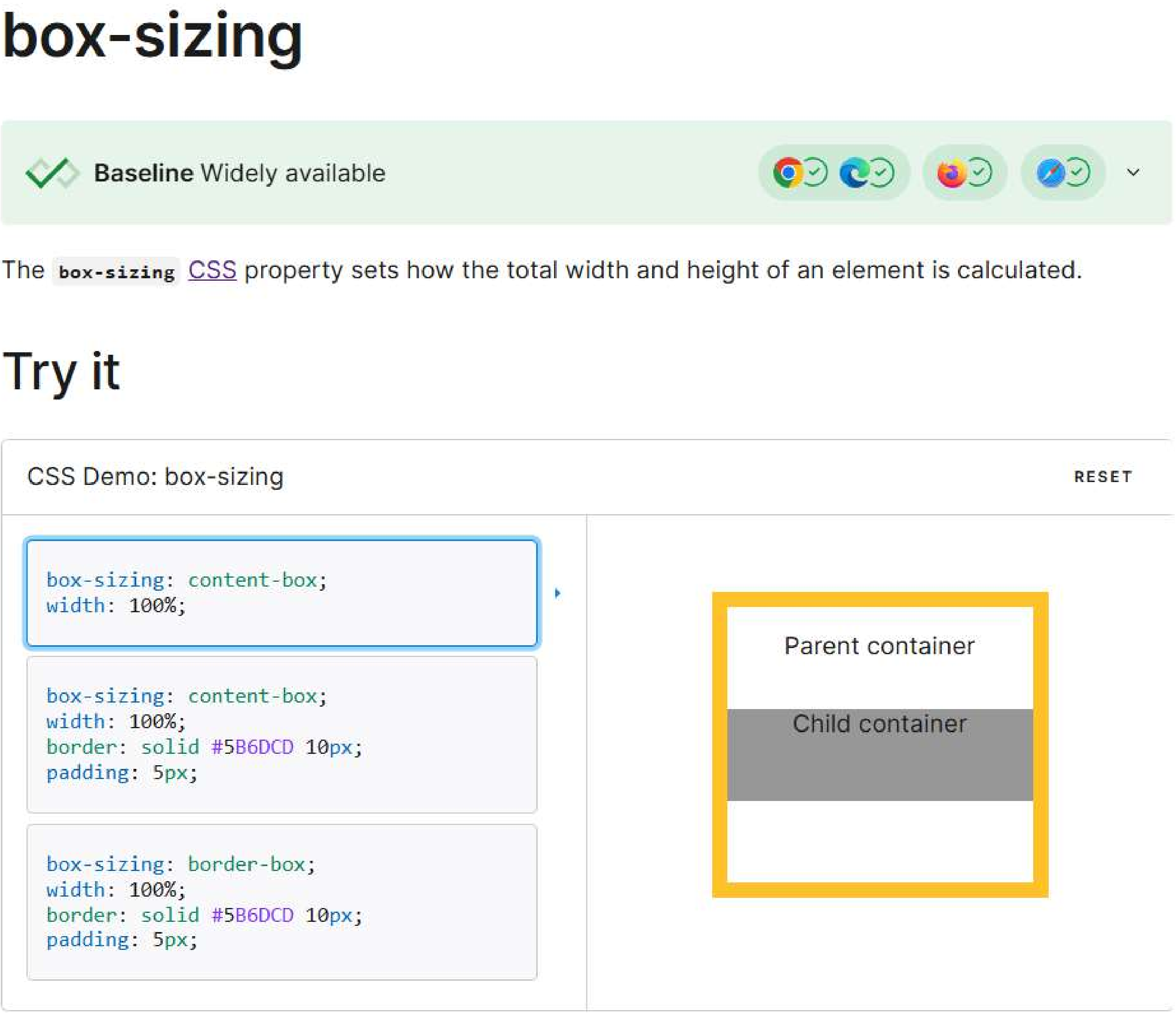
If you add width as 100px, padding as 10px and border as 2px then the entire width becomes 112px (100+10+2).

Box-sizing: border-box;

The box-sizing property defines how the width and height of an element are calculated:

if we apply box-sizing: border-box then the padding and border will be adjusted in the width and height of an element.

PRATHAM :)



box-sizing

Baseline Widely available

The `box-sizing` CSS property sets how the total width and height of an element is calculated.

Try it

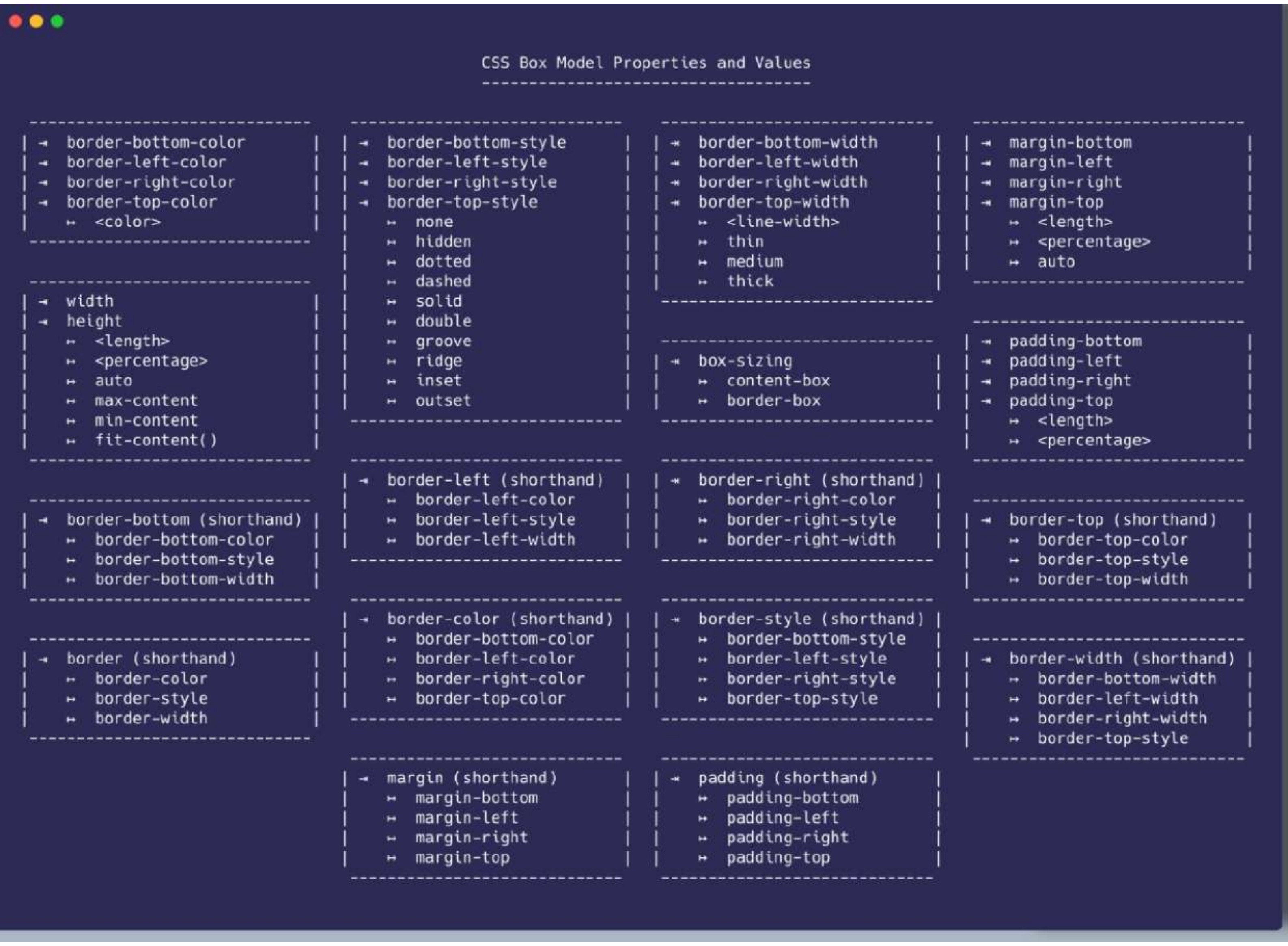
CSS Demo: box-sizing

RESET

```
box-sizing: content-box;
width: 100%;
```

```
box-sizing: content-box;
width: 100%;
border: solid #5B6DCD 10px;
padding: 5px;
```

```
box-sizing: border-box;
width: 100%;
border: solid #5B6DCD 10px;
padding: 5px;
```

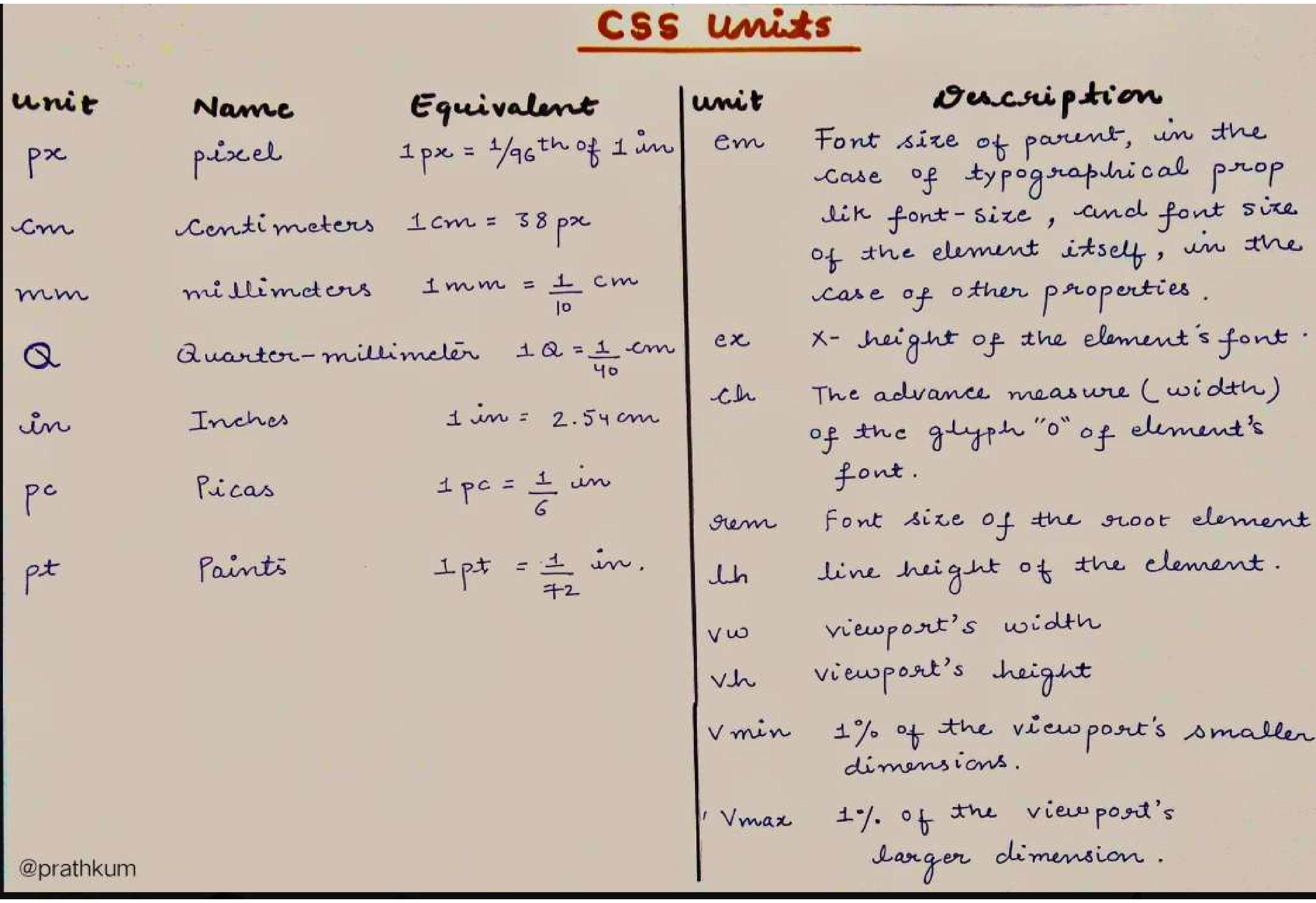


```
                                                 CSS Box Model Properties and Values
                                                 -----------------------------------

 ------------------------------    ------------------------------    ------------------------------    ------------------------------ 
| ⇥ border-bottom-color        |  | ⇥ border-bottom-style        |  | ⇥ border-bottom-width        |  | ⇥ margin-bottom              |
| ⇥ border-left-color          |  | ⇥ border-left-style          |  | ⇥ border-left-width          |  | ⇥ margin-left                |
| ⇥ border-right-color         |  | ⇥ border-right-style         |  | ⇥ border-right-width         |  | ⇥ margin-right               |
| ⇥ border-top-color           |  | ⇥ border-top-style           |  | ⇥ border-top-width           |  | ⇥ margin-top                 |
|   ↦ <color>                  |  |   ↦ none                     |  |   ↦ <line-width>             |  |   ↦ <length>                 |
 ------------------------------   |   ↦ hidden                   |  |   ↦ thin                     |  |   ↦ <percentage>             |
                                  |   ↦ dotted                   |  |   ↦ medium                   |  |   ↦ auto                     |
 ------------------------------   |   ↦ dashed                   |  |   ↦ thick                    |   ------------------------------ 
| ⇥ width                      |  |   ↦ solid                    |   ------------------------------ 
| ⇥ height                     |  |   ↦ double                   |                                      ------------------------------ 
|   ↦ <length>                 |  |   ↦ groove                   |   ------------------------------   | ⇥ padding-bottom             |
|   ↦ <percentage>             |  |   ↦ ridge                    |  | ⇥ box-sizing                 |  | ⇥ padding-left               |
|   ↦ auto                     |  |   ↦ inset                    |  |   ↦ content-box              |  | ⇥ padding-right              |
|   ↦ max-content              |  |   ↦ outset                   |  |   ↦ border-box               |  | ⇥ padding-top                |
|   ↦ min-content              |   ------------------------------    ------------------------------   |   ↦ <length>                 |
|   ↦ fit-content()            |                                                                      |   ↦ <percentage>             |
 ------------------------------    ------------------------------    ------------------------------    ------------------------------ 
                                  | ⇥ border-left (shorthand)    |  | ⇥ border-right (shorthand)   |
 ------------------------------   |   ↦ border-left-color        |  |   ↦ border-right-color       |   ------------------------------ 
| ⇥ border-bottom (shorthand)  |  |   ↦ border-left-style        |  |   ↦ border-right-style       |  | ⇥ border-top (shorthand)     |
|   ↦ border-bottom-color      |  |   ↦ border-left-width        |  |   ↦ border-right-width       |  |   ↦ border-top-color         |
|   ↦ border-bottom-style      |   ------------------------------    ------------------------------   |   ↦ border-top-style         |
|   ↦ border-bottom-width      |                                                                      |   ↦ border-top-width         |
 ------------------------------    ------------------------------    ------------------------------    ------------------------------ 
                                  | ⇥ border-color (shorthand)   |  | ⇥ border-style (shorthand)   |
 ------------------------------   |   ↦ border-bottom-color      |  |   ↦ border-bottom-style      |   ------------------------------ 
| ⇥ border (shorthand)         |  |   ↦ border-left-color        |  |   ↦ border-left-style        |  | ⇥ border-width (shorthand)   |
|   ↦ border-color             |  |   ↦ border-right-color       |  |   ↦ border-right-style       |  |   ↦ border-bottom-width      |
|   ↦ border-style             |  |   ↦ border-top-color         |  |   ↦ border-top-style         |  |   ↦ border-left-width        |
|   ↦ border-width             |   ------------------------------    ------------------------------   |   ↦ border-right-width       |
 ------------------------------                                                                       |   ↦ border-top-style         |
                                   ------------------------------    ------------------------------    ------------------------------ 
                                  | ⇥ margin (shorthand)         |  | ⇥ padding (shorthand)        |
                                  |   ↦ margin-bottom            |  |   ↦ padding-bottom           |
                                  |   ↦ margin-left              |  |   ↦ padding-left             |
                                  |   ↦ margin-right             |  |   ↦ padding-right            |
                                  |   ↦ margin-top               |  |   ↦ padding-top              |
                                   ------------------------------    ------------------------------ 
```

3.4 Handling conflicts in CSS

📖 **Learning outcomes:**

- Understand how rules can conflict in CSS.
- Inheritance.
- The cascade.
- The concepts that govern the outcome of CSS conflicts:
    - Specificity.
    - Source order.
    - Importance.



CSS Units

unit	Name	Equivalent
px	pixel	1 px = 1/96th of 1 in
cm	Centimeters	1 cm = 38 px
mm	millimeters	1 mm = $\frac{1}{10}$ cm
Q	Quarter-millimeter	1 Q = $\frac{1}{40}$ cm
in	Inches	1 in = 2.54 cm
pc	Picas	1 pc = $\frac{1}{6}$ in
pt	Paints	1 pt = $\frac{1}{72}$ in.

unit	Description
em	Font size of parent, in the case of typographical prop lik font-size, and font size of the element itself, in the case of other properties.
ex	X- height of the element's font.
ch	The advance measure (width) of the glyph "0" of element's font.
rem	Font size of the root element
lh	line height of the element.
vw	viewport's width
vh	viewport's height
v min	1% of the viewport's smaller dimensions.
Vmax	1%. of the viewport's larger dimension.

@prathkum

3.6 Sizing

Learning outcomes:

- Intrinsic size.
- Setting absolute and percentage sizes.
- `min-` / `max-width` and `min-` / `max-height` .
- Viewport units.

Resources:

[Sizing items in CSS]

[Handling different text directions > Logical properties]

CSS BACKGROUND

background

background-color
background-image
background-repeat
background-attachment
background-position
background-size

background-color

color | transparent

background-image

url | gradients | none

background-repeat

no-repeat | repeat |
repeat-x | repeat-y

background-attachment

no-repeat | repeat |
repeat-x | repeat-y

background-position

top right | left center |
bottom center top left |
right center | bottom right
top center | center center |
bottom left

background-size

cover | auto | contain |
length

BACKGROUND SHORTHAND

Traditional way

```
background-color: #ffffff;
background-image: url("img_tree.png");
background-repeat: no-repeat;
background-position: right top;
```

Shorthand property

```
background: #ffffff url("img_tree.png")
            no-repeat right top;
```

Alfaiz Ali
@imAlfaiz

CSS BORDER CHEATSHEET

border

```
border-width
border-style
border-color
```

border-width

```
border-width: thin;
border-width: medium;
border-width: thick;
border-width: 2px; //length
```

border-style

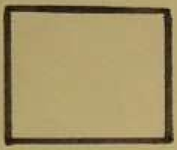
```
border-style: none;
border-style: hidden;
border-style: dotted;
border-style: dashed;
border-style: solid;
border-style: double;
border-style: groove;
border-style: ridge;
border-style: inset;
border-style: outset;
```

border-color

```
border-color: red; //color
```

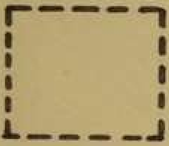
Alfaiz Ali
@ImAlfaiz

border-style



Solid

Displays a single straight solid line

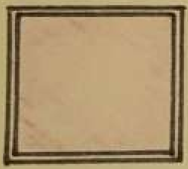


Dashed

Displays the small square of same length.

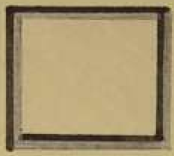
Dotted

Displays a series of rounded dots.



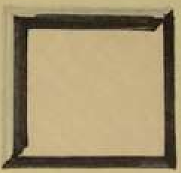
Double

Displays two straight line



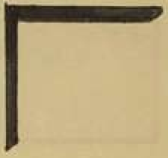
Groove

Displays the border with carved appearance.



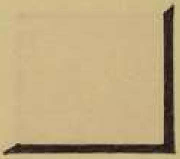
Ridge

Displays the border with an extruded appearance



Inset

makes the element appear embedded.



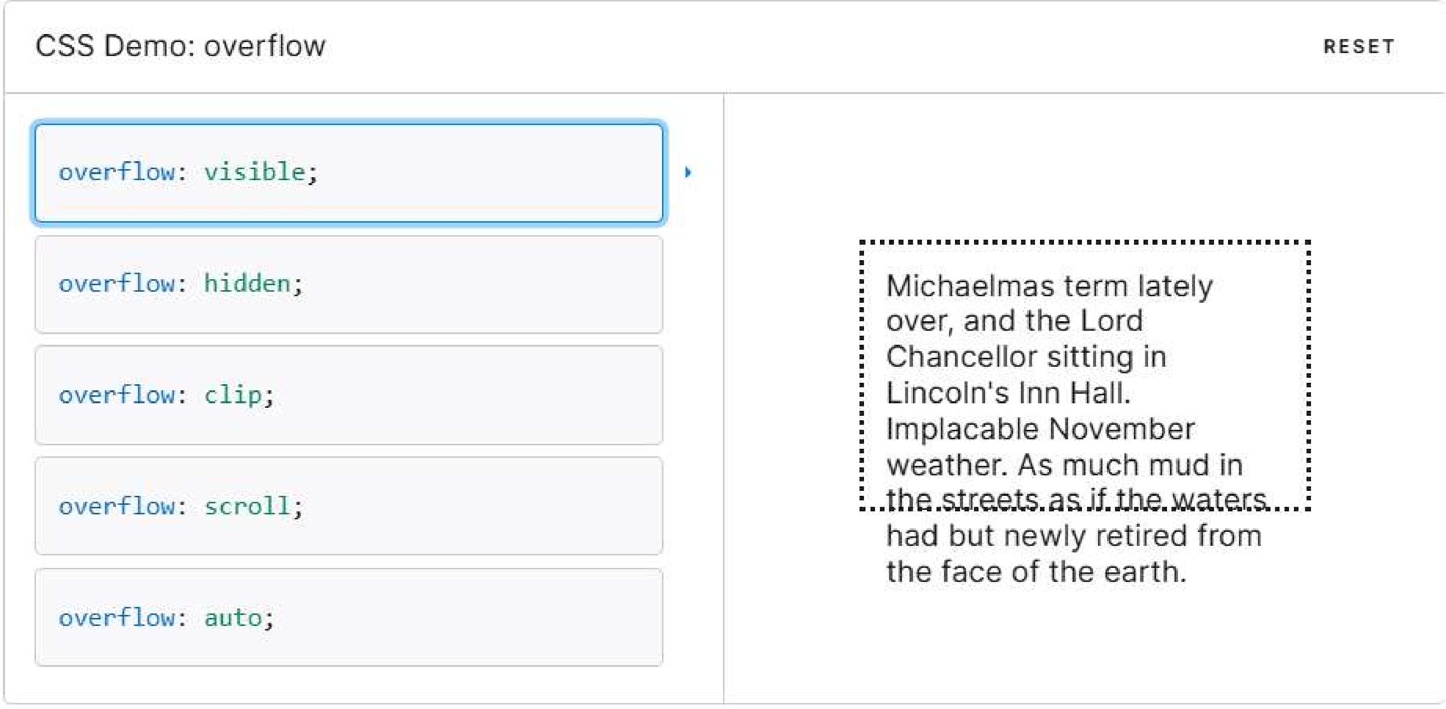
Outset

Displays a border that makes element appear embossed.

overflow

The `overflow` CSS shorthand property sets the desired behavior when content does not fit in the element's padding box (overflows) in the horizontal and/or vertical direction.

Try it



CSS Demo: overflow

```
overflow: visible;
```

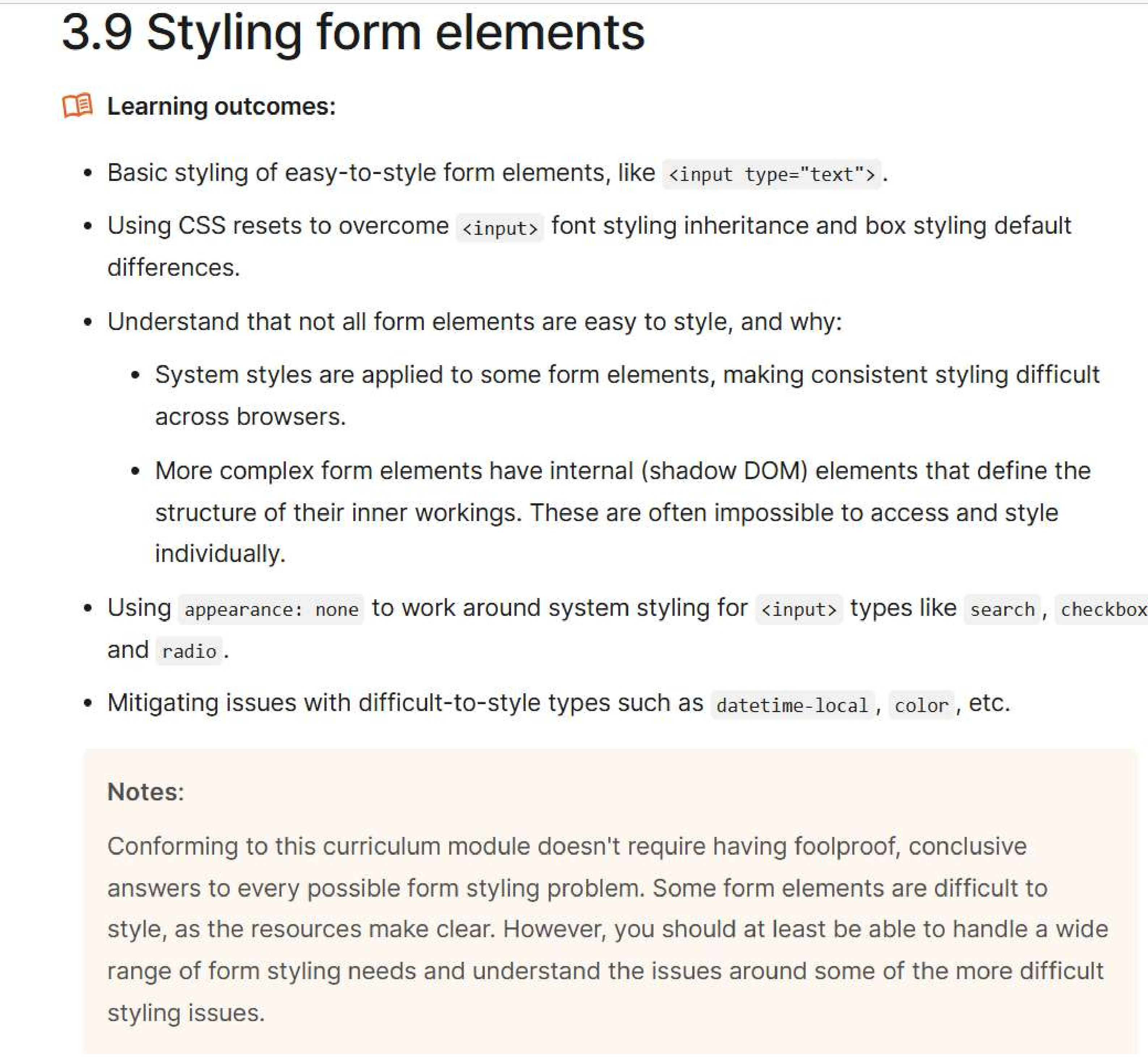
```
overflow: hidden;
```

```
overflow: clip;
```

```
overflow: scroll;
```

```
overflow: auto;
```

Michaelmas term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets as if the waters had but newly retired from the face of the earth.



3.9 Styling form elements

Learning outcomes:

- Basic styling of easy-to-style form elements, like `<input type="text">`.
- Using CSS resets to overcome `<input>` font styling inheritance and box styling default differences.
- Understand that not all form elements are easy to style, and why:
  - System styles are applied to some form elements, making consistent styling difficult across browsers.
  - More complex form elements have internal (shadow DOM) elements that define the structure of their inner workings. These are often impossible to access and style individually.
- Using `appearance: none` to work around system styling for `<input>` types like `search`, `checkbox` and `radio`.
- Mitigating issues with difficult-to-style types such as `datetime-local`, `color`, etc.

Notes:

Conforming to this curriculum module doesn't require having foolproof, conclusive answers to every possible form styling problem. Some form elements are difficult to style, as the resources make clear. However, you should at least be able to handle a wide range of form styling needs and understand the issues around some of the more difficult styling issues.

3.10 Debugging CSS

Learning outcomes:

- Use the HTML validator to see if you have any invalid markup on your page — this could be causing your CSS to not apply as desired.
- Use the CSS validator to check for badly-formed CSS code. A missing semi-colon can cause a whole section of CSS declarations to not apply.
- Use browser developer tools to inspect the CSS that is applied to HTML elements on a page.
- Modify the applied CSS to figure out what changes are needed to get what you want. This includes enabling and disabling declarations, modifying values, and adding new declarations.
- Use layout inspection tools to inspect the box model, grids, flexbox, and other layout features (see also CSS Layout).
- Use responsive design mode tools to check responsive layouts (see also 5.5 Responsive design specifics).

Resources:

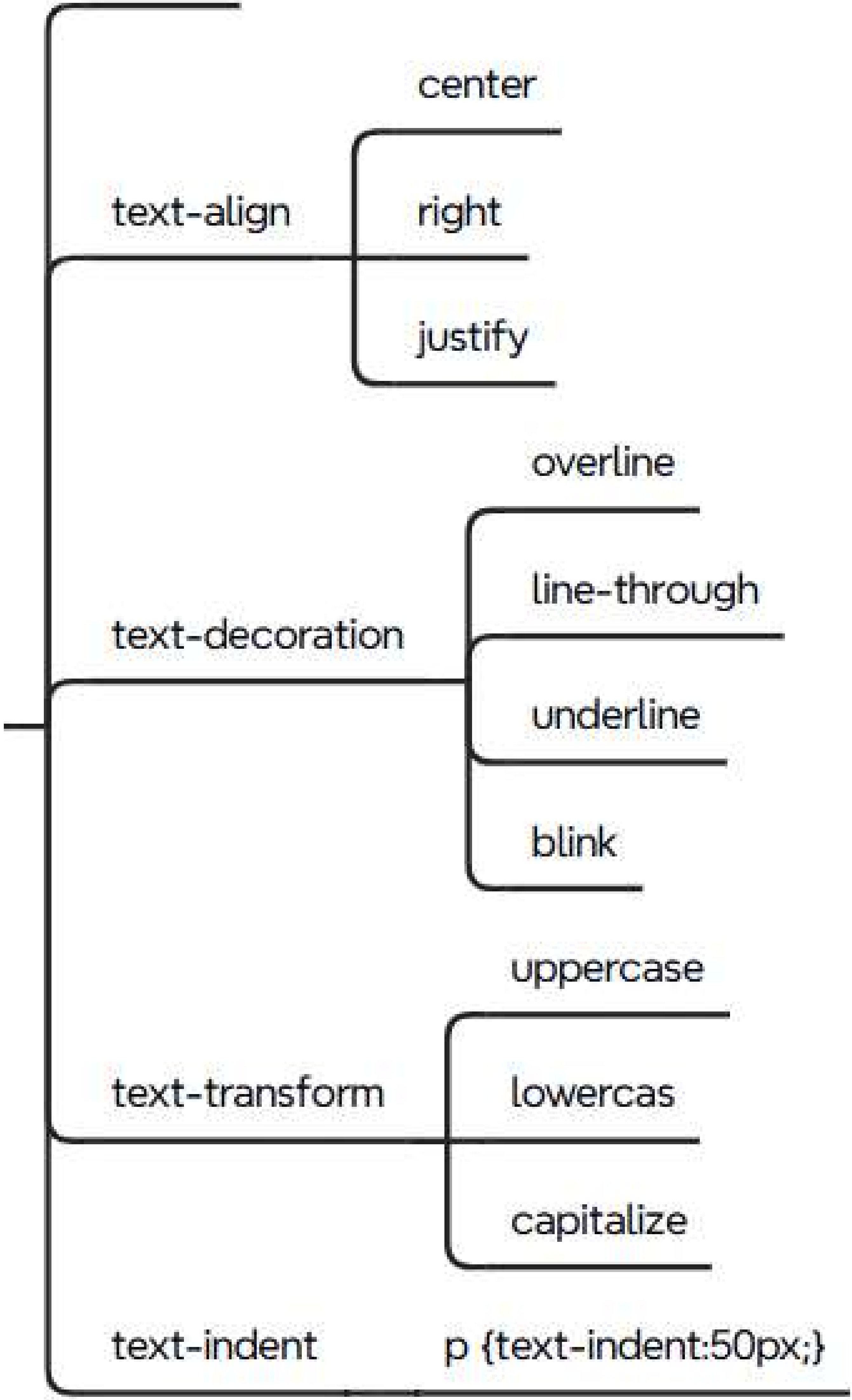
- Debugging CSS
- Handling common HTML and CSS problems
- Firefox > Examine and edit CSS, Firefox Source Docs
- Firefox > Responsive design mode, Firefox Source Docs
- Chrome > View and change CSS, developer.chrome.com)

4.1 Text and font styling

Learning outcomes:

- `color`.
- Font family, font stacks, web safe fonts.
- `font-size`, `font-weight`, and `font-style`.
- `text-align`, `text-transform`, and `text-decoration`.
- `text-shadow`.
- `line-height`.

> **Notes:**
>
> There are several other font and text styling properties, and students should be encouraged to explore more of them as part of their constant learning.



text-align

center

right

justify

text-decoration

overline

line-through

underline

blink

text-transform

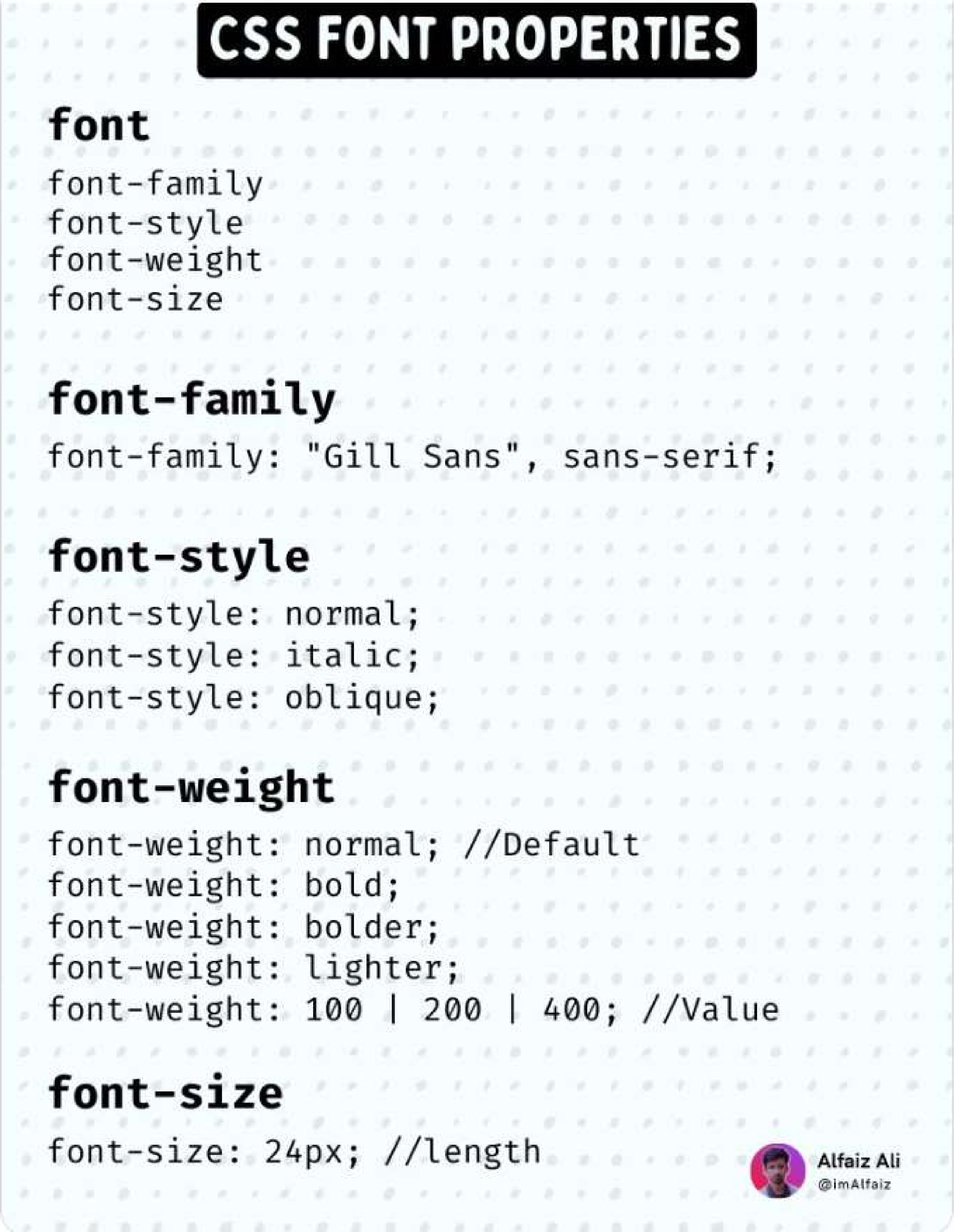
uppercase

lowercas

capitalize

text-indent

p {text-indent:50px;}



CSS FONT PROPERTIES

font

```
font-family
font-style
font-weight
font-size
```

font-family

```
font-family: "Gill Sans", sans-serif;
```

font-style

```
font-style: normal;
font-style: italic;
font-style: oblique;
```

font-weight

```
font-weight: normal; //Default
font-weight: bold;
font-weight: bolder;
font-weight: lighter;
font-weight: 100 | 200 | 400; //Value
```

font-size

```
font-size: 24px; //length
```

Alfaiz Ali
@imAlfaiz

4.2 Styling lists and links

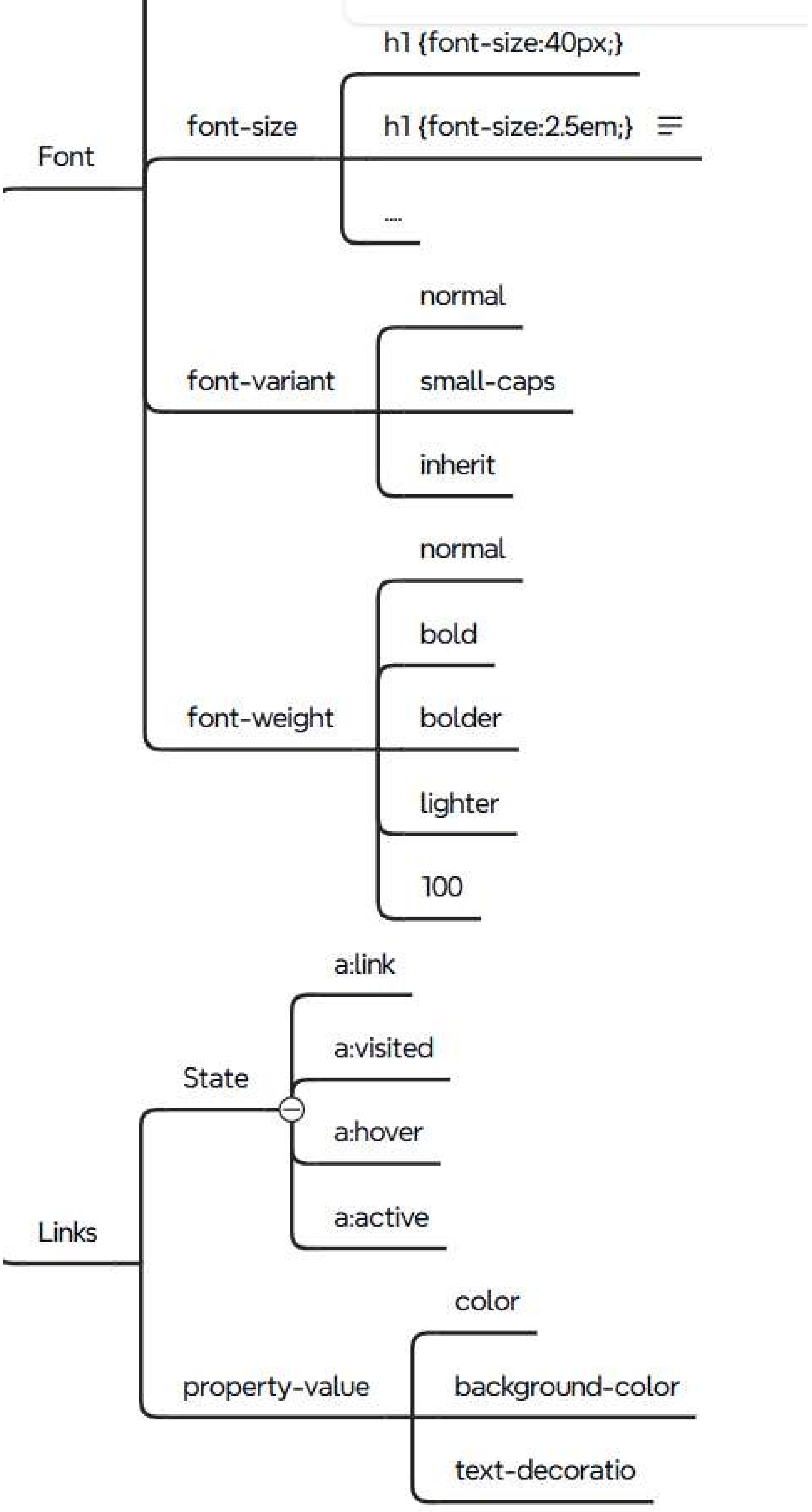
Learning outcomes:

- Spacing list items, for example, with `margin` or `line-height`.
- `list-style` properties.
- Understand why default link styles are important for usability on the web — they are familiar and help users recognize links.
- Styling link states: `:hover`, `:focus`, `:visited`, and `:active`:
  - Understand why these are necessary for usability and accessibility.
- Creating a navigation menu with lists and links.

Resources:

[Styling lists]

[Styling links]



- Font
  - font-size
    - h1 {font-size:40px;}
    - h1 {font-size:2.5em;}
    - ....
  - font-variant
    - normal
    - small-caps
    - inherit
  - font-weight
    - normal
    - bold
    - bolder
    - lighter
    - 100
- Links
  - State
    - a:link
    - a:visited
    - a:hover
    - a:active
  - property-value
    - color
    - background-color
    - text-decoratio

4.3 Web fonts

Learning outcomes:

- Understand that web fonts allow developers to go beyond the web safe font set and use custom fonts on their web apps.
- Basic setup — the `@font-face` at-rule, and `font-family` and `src` descriptors.
- Using a web font with the `font-family` property.
- Other descriptors — `font-weight`, `font-style`, etc.
- Using an online service to find web fonts and generate web font code, for example, Font Squirrel and Google Fonts.
- Usability implications of web fonts — using several of them can increase page download size.

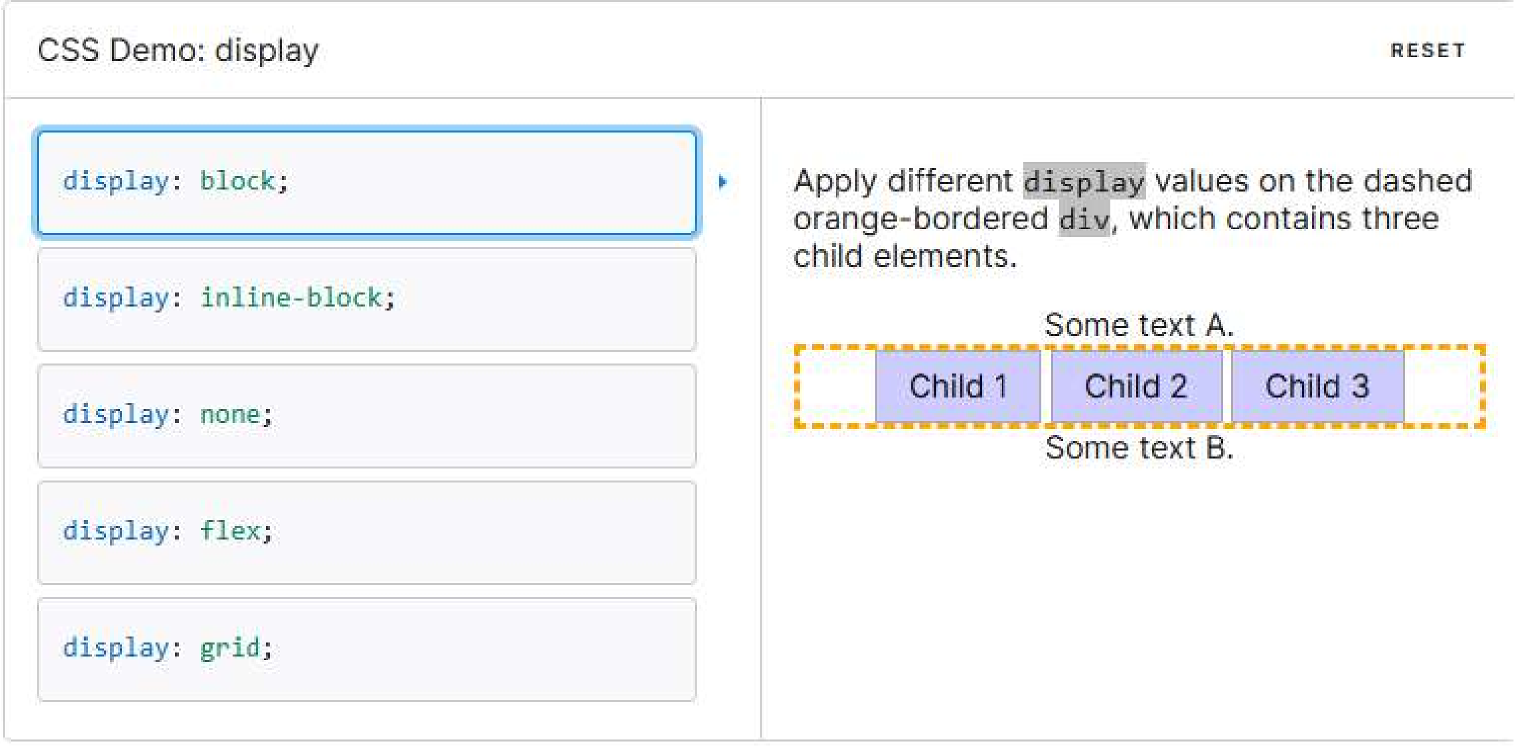
display

Baseline Widely available *

The `display` CSS property sets whether an element is treated as a block or inline box and the layout used for its children, such as flow layout, grid or flex.

Formally, the `display` property sets an element's inner and outer *display types*. The outer type sets an element's participation in flow layout; the inner type sets the layout of children. Some values of `display` are fully defined in their own individual specifications; for example the detail of what happens when `display: flex` is declared is defined in the CSS Flexible Box Model specification.

Try it



CSS Demo: display

```
display: block;
```

```
display: inline-block;
```

```
display: none;
```

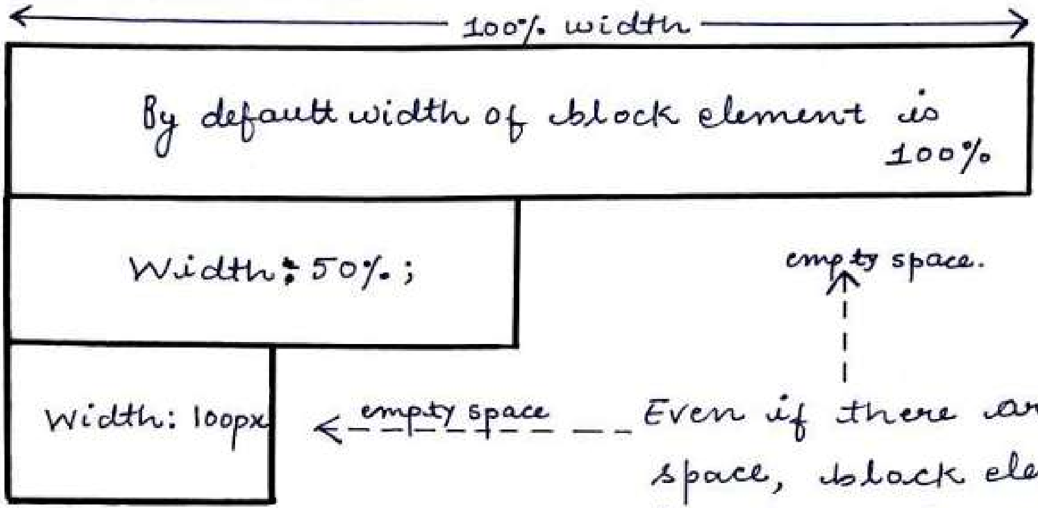
```
display: flex;
```

```
display: grid;
```

Apply different `display` values on the dashed orange-bordered `div`, which contains three child elements.

Display : Block, inline and inline-black.

Black Elements.



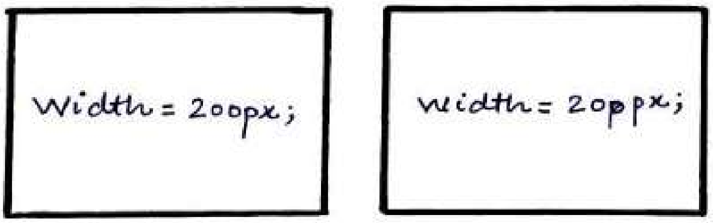
Even if there are space, block element do not render on a same line. In general they do not sit next to each other.

Inline

You can't control width and height of inline element. for ex. span.

Inline-block.

(As the term suggest, they are combination of block and inline)



The element is formatted as inline element but you can apply height and width as well.

5.1 CSS layout basics

Learning outcomes:

- Understand that normal flow is the default way a browser lays out block and inline content.
- Properties such as `display`, `float`, and `position` are intended to change how the browser lays out content.

Resources:

- Introduction to CSS layout
- Normal flow

5.2 Floats

Learning outcomes:

- Understand the purpose of floats — for floating images inside columns of text, or possibly other fun techniques like drop caps and floating inset information boxes.
- Understand that floats used to be used for multiple-column layouts, but this is no longer the case now better tools are available (see 5.4 Modern layout for details).
- Using the `float` property to create floats.
- Clearing floats using `clear`, and the `display: flow-root` value.

Resources:

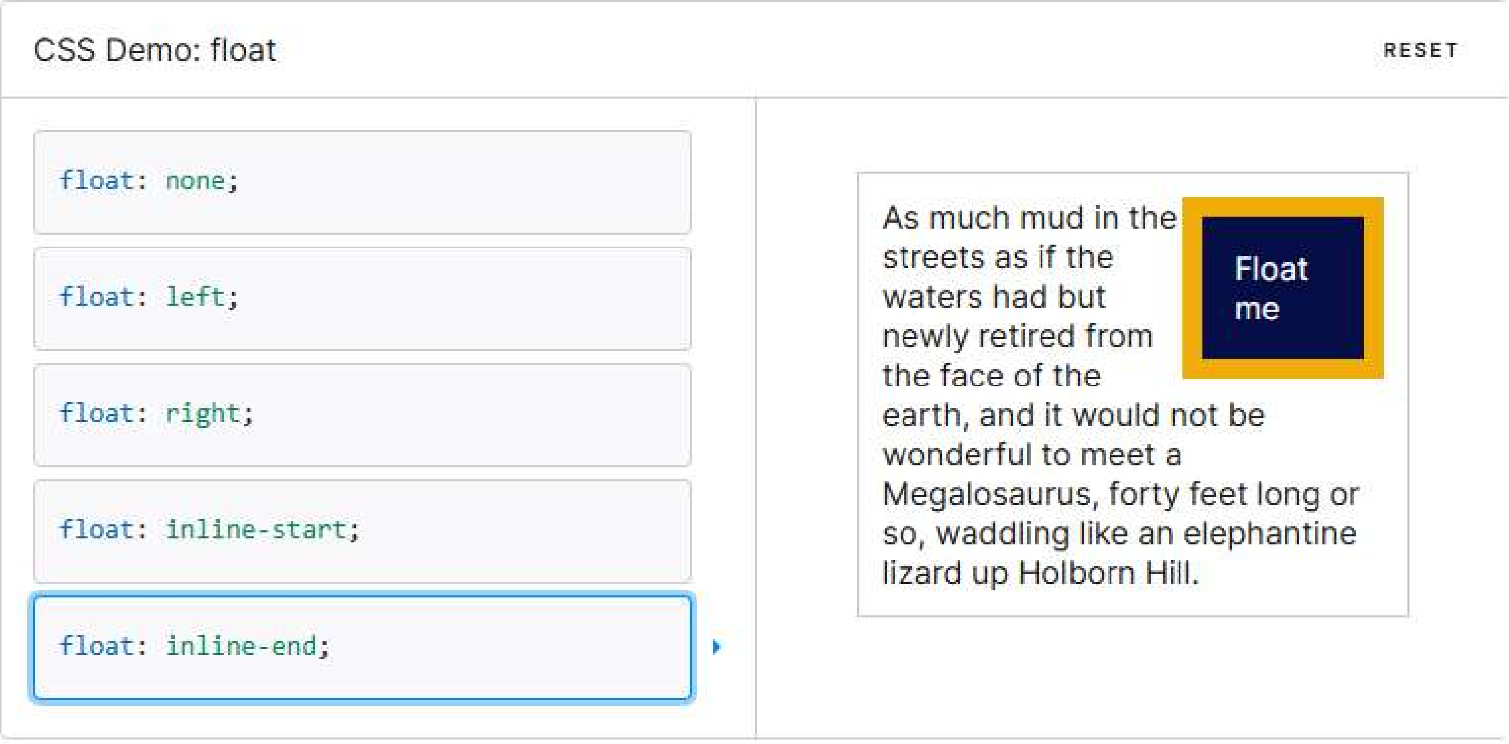
- Floats
- All About FLoats, CSS-Tricks (2021)

float

Baseline Widely available *

The `float` CSS property places an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element is removed from the normal flow of the page, though still remaining a part of the flow (in contrast to absolute positioning).

Try it



CSS Demo: float

```
float: none;
```

```
float: left;
```

```
float: right;
```

```
float: inline-start;
```

```
float: inline-end;
```

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

A *floating element* is one where the computed value of `float` is not `none`.

As `float` implies the use of the block layout, it modifies the computed value of the `display` values, in some cases:

5.3 Positioning

Learning outcomes:

- Understand that `static` positioning is the default way elements are positioned on the page.
- Relative positioning:
  - Understand that relatively positioned elements remain in the normal flow.
  - Final layout position can be modified using the `top`, `bottom`, `left`, and `right` properties.
- Absolute positioning:
  - Absolute (and fixed/sticky) positioning takes elements completely out of the normal flow to sit in a separate layer.
  - `top`, `bottom`, `left`, `right`, and `inset` have different effects on absolutely-positioned elements than on relatively-positioned ones.
  - Setting the positioning context of a positioned element by positioning an ancestor element.

- Fixed and sticky positioning:
  - Understand how these differ from absolute positioning.
- Understand what z-index is, and how to control the stacking of positioned elements with the `z-index` property.

Resources:

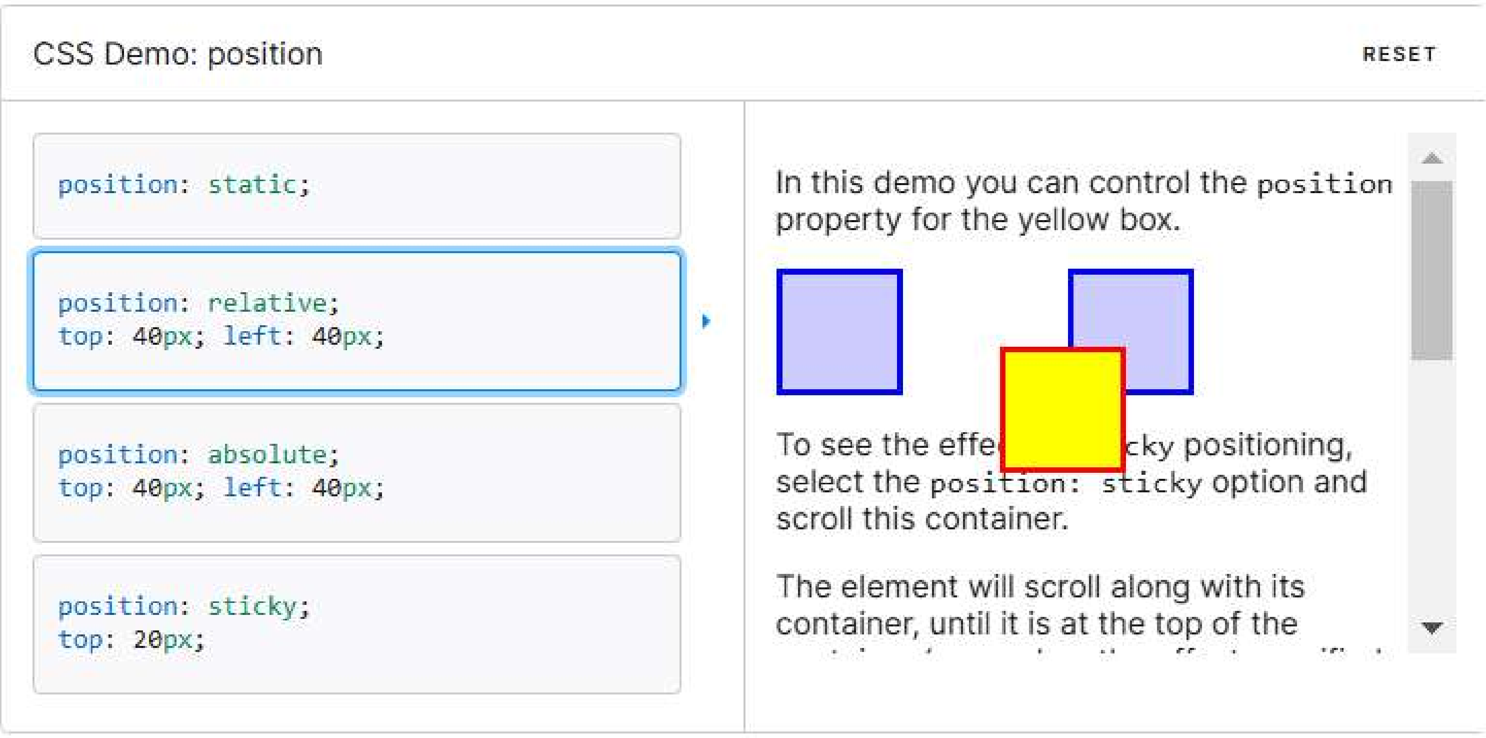
- [Positioning]()
- [Aside: Position: relative & absolute](), Scrimba COURSE PARTNER
- [Stacking context]()

position

Baseline Widely available

The `position` CSS property sets how an element is positioned in a document. The `top`, `right`, `bottom`, and `left` properties determine the final location of positioned elements.

Try it



CSS Demo: position

```
position: static;
```

```
position: relative;
top: 40px; left: 40px;
```

```
position: absolute;
top: 40px; left: 40px;
```

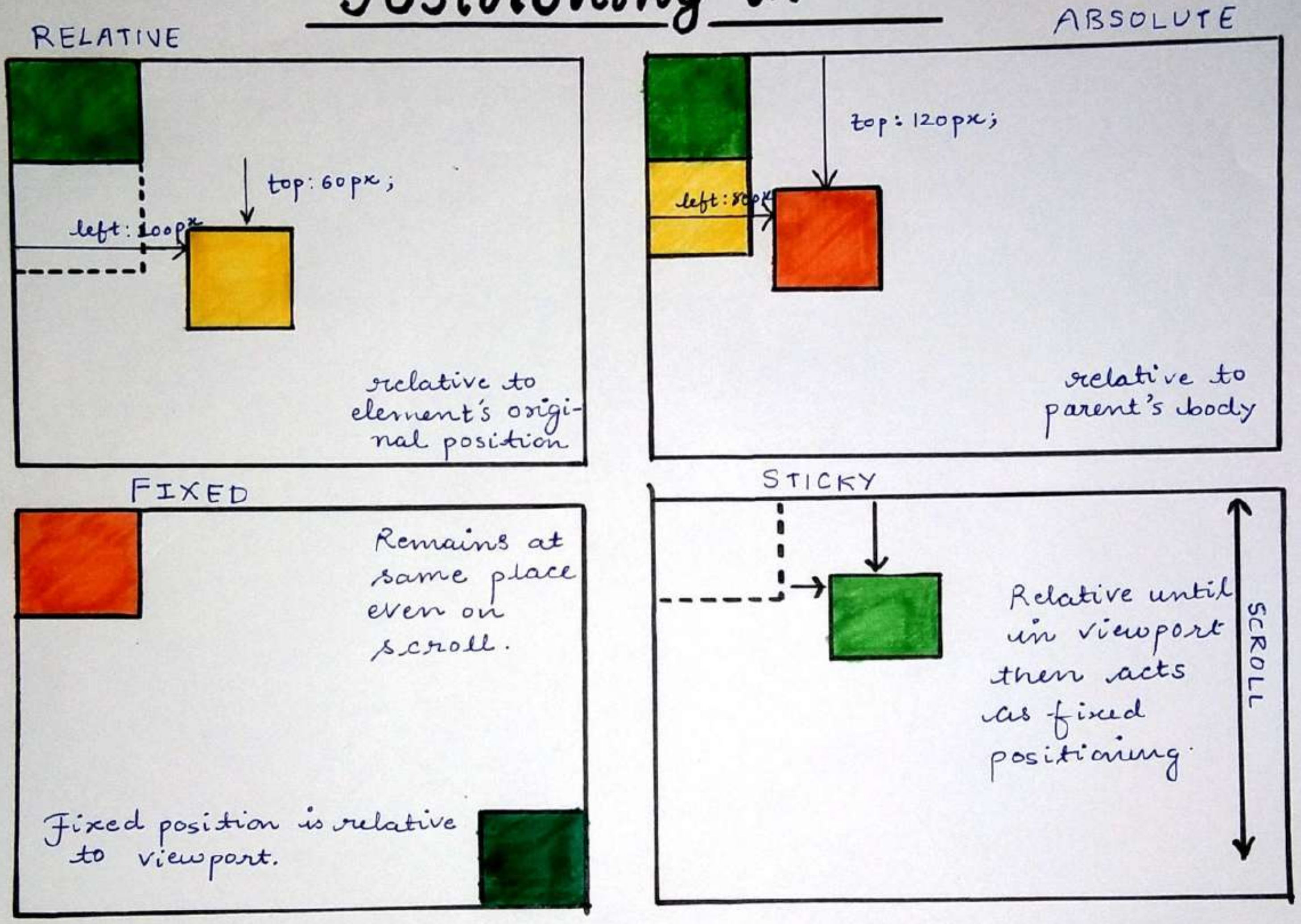
```
position: sticky;
top: 20px;
```

In this demo you can control the `position` property for the yellow box.

To see the effe cky positioning, select the `position: sticky` option and scroll this container.

The element will scroll along with its container, until it is at the top of the

Positioning in Css



relative to element's original position

relative to parent's body

Remains at same place even on scroll.

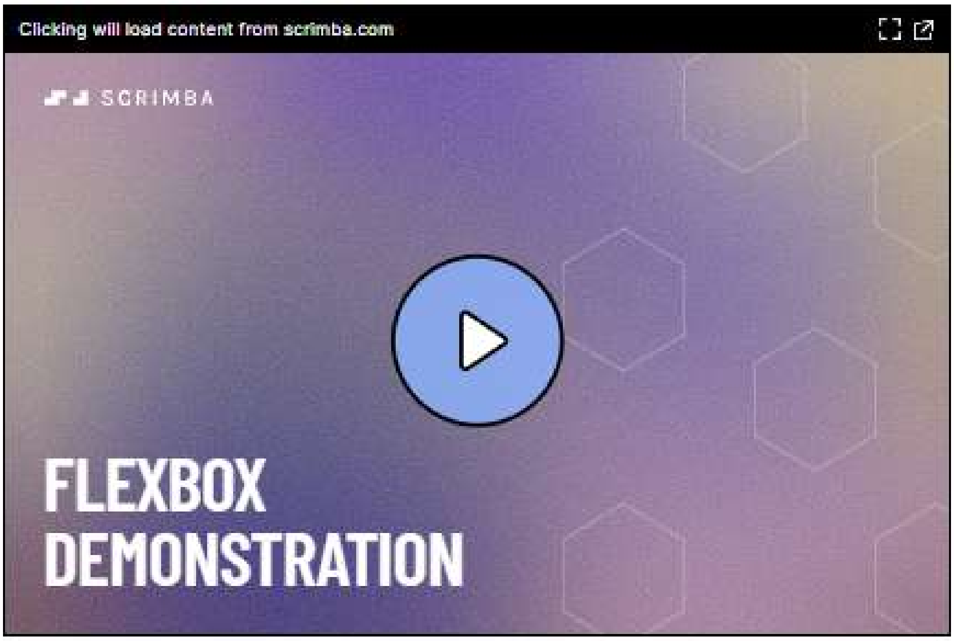
Fixed position is relative to viewport.

Relative until in viewport then acts as fixed positioning.

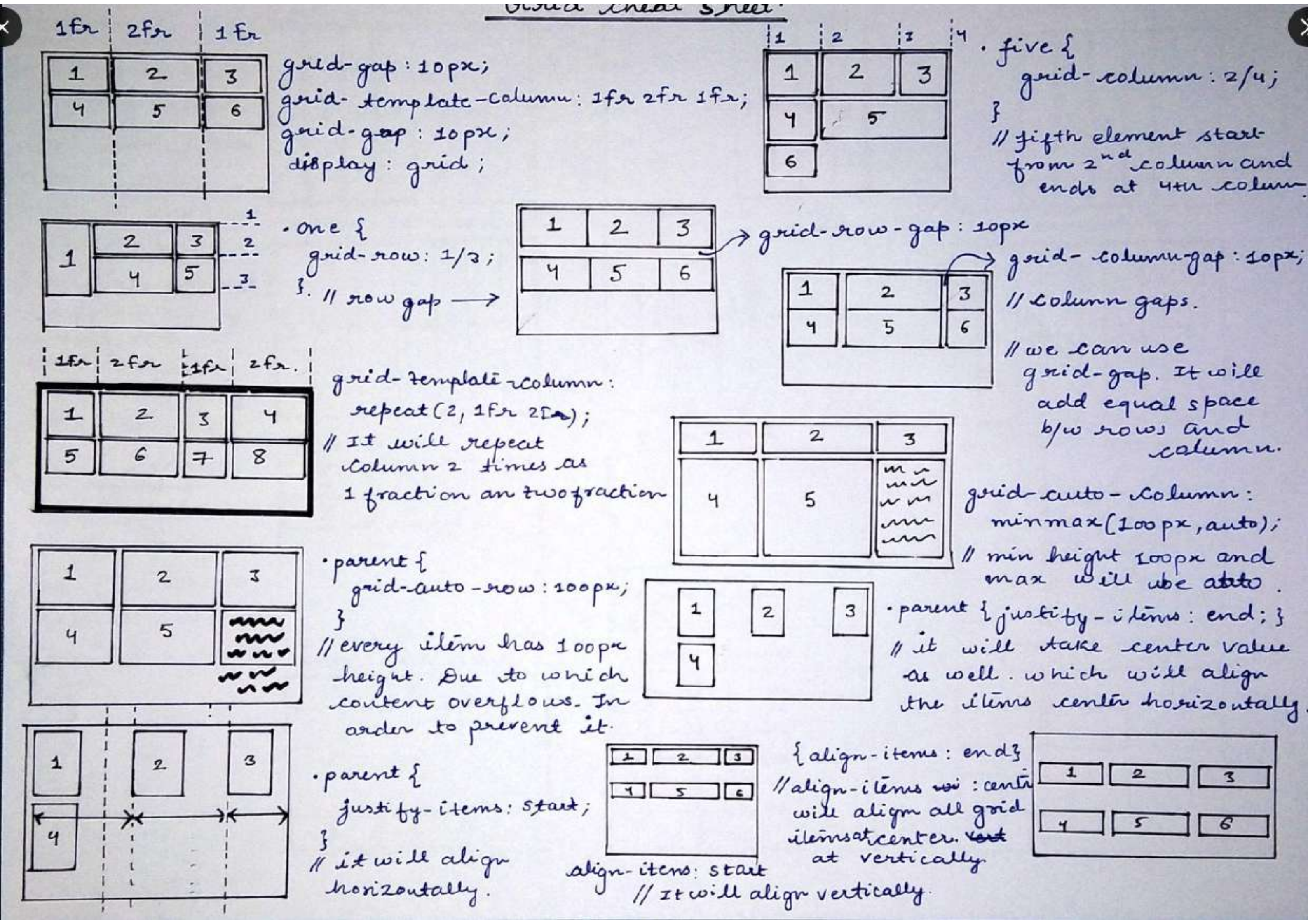
5.4 Modern layout

Learning outcomes:

- In general, gain an understanding of modern CSS layout techniques.
- Understand that, for basic placement tasks, the below tools could be overkill. Learn simple old-school techniques and where they are still effective:
  - Margins and padding for spacing.
  - Auto margins for horizontal centering tasks (e.g. `margin: 0 auto`).
- Flexbox:
  - Understand the purpose of flexbox — flexibly lay out a set of block or inline elements in one dimension.
    - See [We have a problem that flexbox can fix]() by Scrimba *Course Partner* for a use case example.
  - Understand flex terminology — flex container, flex item, main axis, and cross axis.
  - `display: flex`, and what it gives you by default.
  - Rows and columns, and how to wrap content onto new rows and columns.
  - Flexible sizing of flex items.
  - Justifying and aligning content.
  - Adjusting flex item ordering.



- CSS Grid:
  - Understand the purpose of CSS Grid — flexibly lay out a set of block or inline elements in two dimensions.
  - Understand grid terminology — rows, columns, gaps, and gutters.
  - `display: grid`, and what it gives you by default.
  - Defining grid rows and columns:
    - The `fr` unit.



Grid cheat sheet

```
grid-gap : 10px;
grid-template-column : 1fr 2fr 1fr;
grid-gap : 10px;
display : grid ;
```

```
.five {
  grid-column : 2/4;
}
// fifth element start from 2nd column and ends at 4th colum.
```

```
.one {
  grid-row: 1/3;
}. // row gap →
```

→ grid-row-gap : 10px

→ grid-column-gap : 10px;

// column gaps.

// we can use grid-gap. It will add equal space b/w rows and column.

```
grid-template-column :
  repeat(2, 1fr 2f~~r~~);
// It will repeat column 2 times as 1 fraction an twofraction
```

```
grid-auto-column :
  minmax(100px,auto);
// min height 100px and max will be auto.
```

```
.parent {
  grid-auto-row : 100px;
}
// every item has 100px height. Due to which content overflows. In order to prevent it.
```

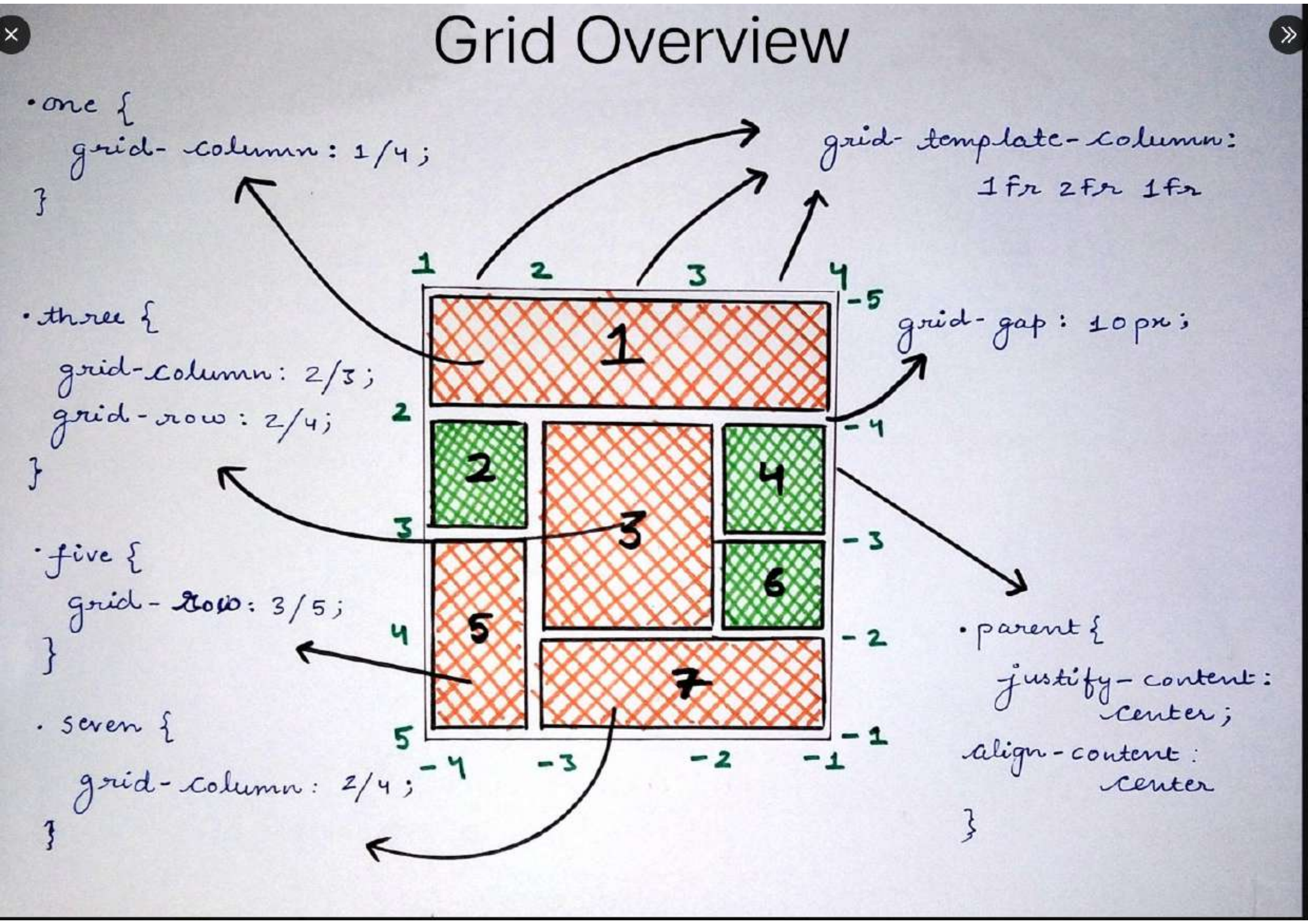
```
.parent {justify-items : end; }
// it will take center value as well. which will align the items center horizontally.
```

```
.parent {
  justify-items: start;
}
// it will align horizontally.
```

```
align-items: start
// It will align vertically.
```

```
{align-items : end}
//align-items ~~w~~ : center will align all grid items at center. ~~vert~~ at vertically.
```

Grid Overview



```
.one {
  grid- column: 1/4;
}
```

```
.three {
  grid-column: 2/3;
  grid-row: 2/4;
}
```

```
.five {
  grid- row: 3/5;
}
```

```
. seven {
  grid- column: 2/4;
}
```

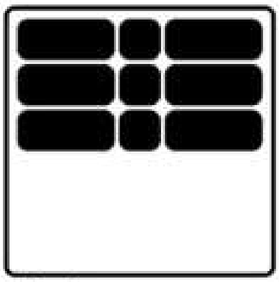
```
grid- template- column:
        1fr 2fr 1fr
```

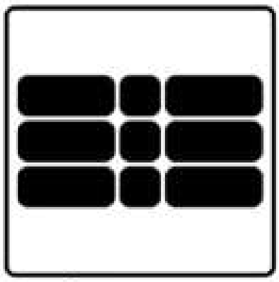
```
grid- gap: 10px;
```

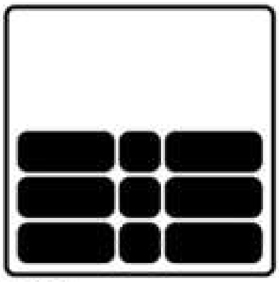
```
.parent {
  justify- content:
        center;
 align- content:
        center
}
```

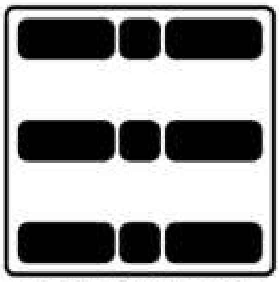
CSS Grid Layout

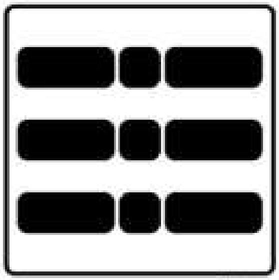
align-content

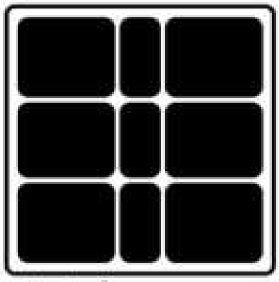

start


center

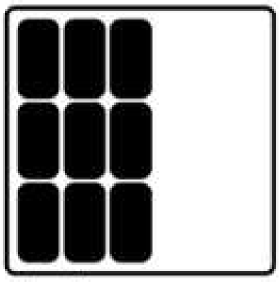

end

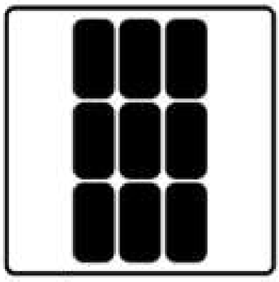

space-between

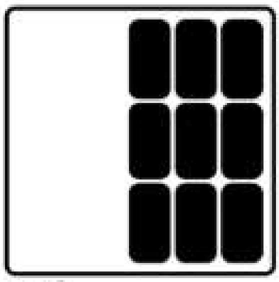

space-around

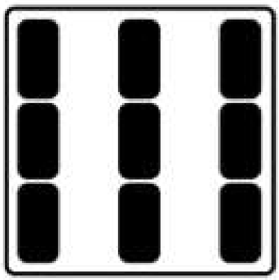

stretch

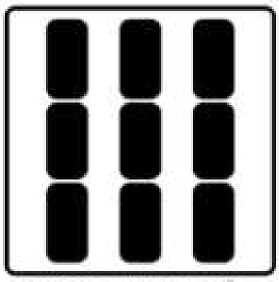
justify-content

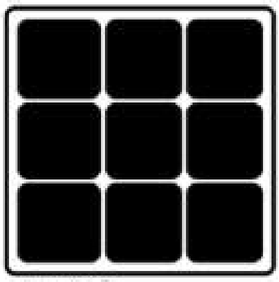

start


center

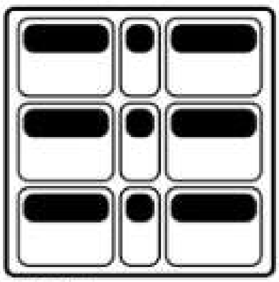

end

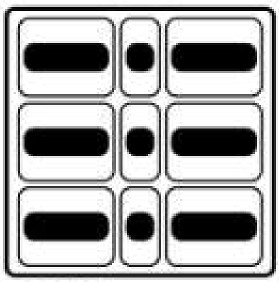

space-between

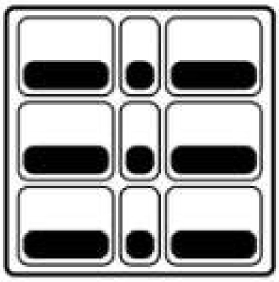

space-around

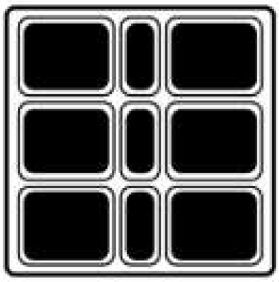

stretch

align-items

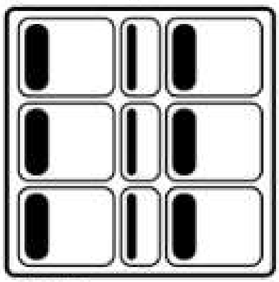

start

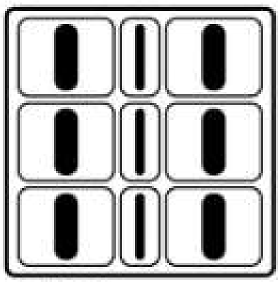

center

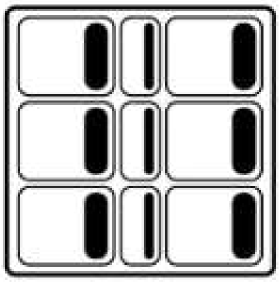

end

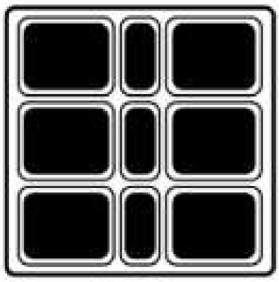

stretch

justify-items

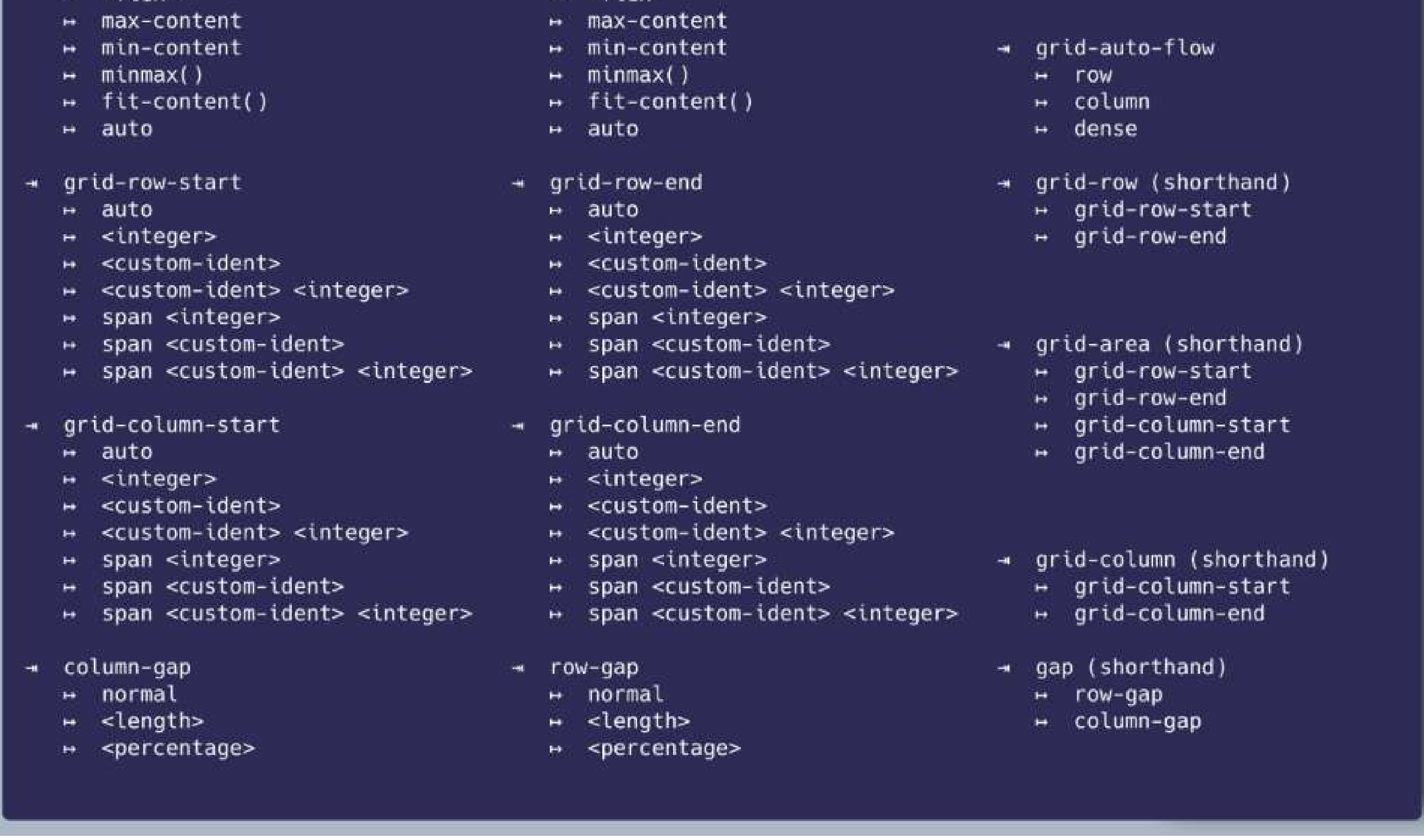

start


center

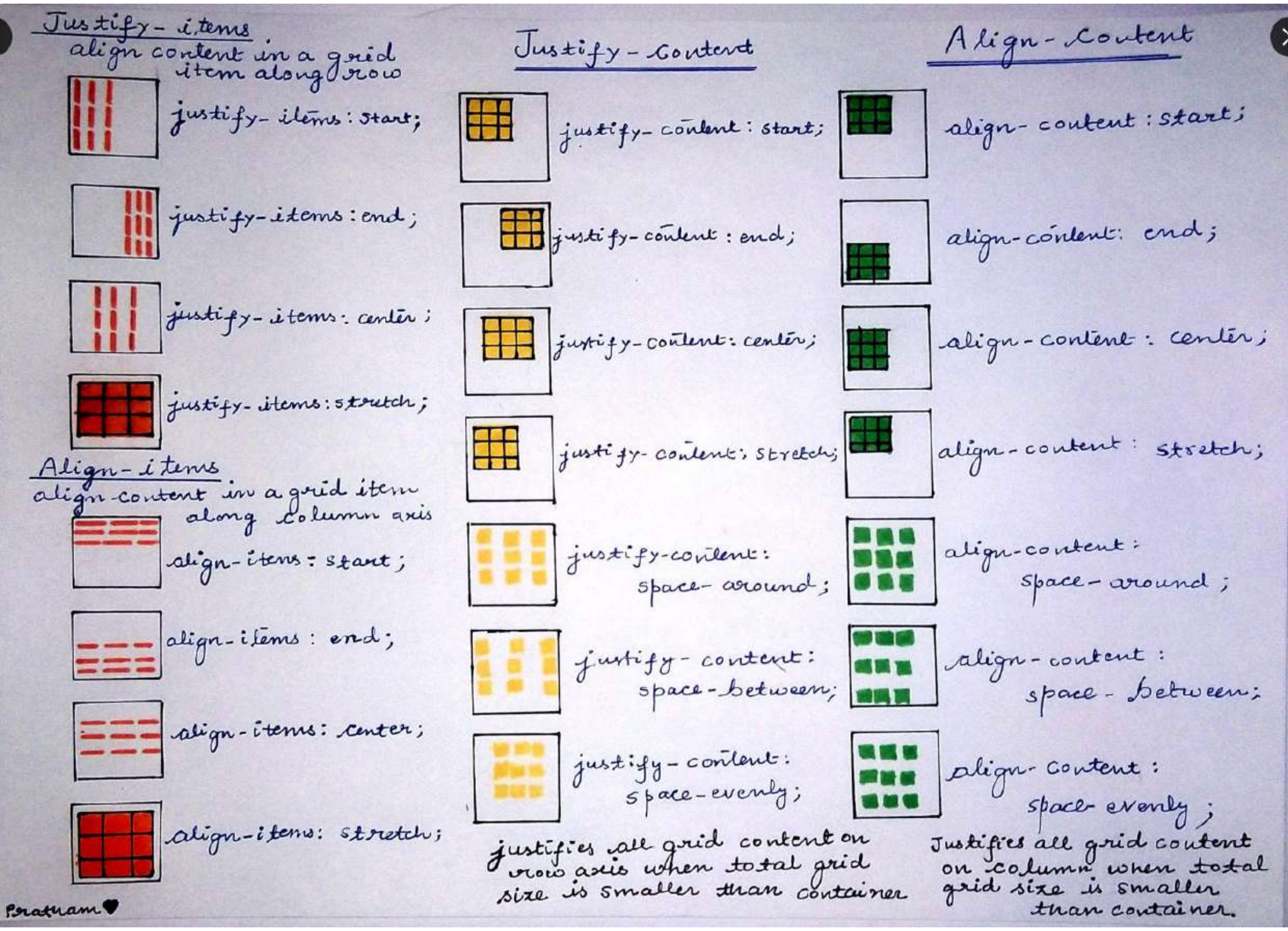

end


stretch

github.com/eludadev/css-docs



```
CSS Grid Layout Properties and Values
-------------------------------------

⇥ grid-template-columns
   ↦ none
   ↦ [linename]
   ↦ <length>
   ↦ <percentage>
   ↦ <flex>
   ↦ minmax()
   ↦ fit-content()
   ↦ repeat()
   ↦ max-content
   ↦ min-content
   ↦ auto

⇥ grid-auto-columns
   ↦ <length>
   ↦ <percentage>
   ↦ <flex>
   ↦ max-content
   ↦ min-content
   ↦ minmax()
   ↦ fit-content()
   ↦ auto

⇥ grid-row-start
   ↦ auto
   ↦ <integer>
   ↦ <custom-ident>
   ↦ <custom-ident> <integer>
   ↦ span <integer>
   ↦ span <custom-ident>
   ↦ span <custom-ident> <integer>

⇥ grid-column-start
   ↦ auto
   ↦ <integer>
   ↦ <custom-ident>
   ↦ <custom-ident> <integer>
   ↦ span <integer>
   ↦ span <custom-ident>
   ↦ span <custom-ident> <integer>

⇥ column-gap
   ↦ normal
   ↦ <length>
   ↦ <percentage>

⇥ grid-template-rows
   ↦ none
   ↦ [linename]
   ↦ <length>
   ↦ <percentage>
   ↦ <flex>
   ↦ minmax()
   ↦ fit-content()
   ↦ repeat()
   ↦ max-content
   ↦ min-content
   ↦ auto

⇥ grid-auto-rows
   ↦ <length>
   ↦ <percentage>
   ↦ <flex>
   ↦ max-content
   ↦ min-content
   ↦ minmax()
   ↦ fit-content()
   ↦ auto

⇥ grid-row-end
   ↦ auto
   ↦ <integer>
   ↦ <custom-ident>
   ↦ <custom-ident> <integer>
   ↦ span <integer>
   ↦ span <custom-ident>
   ↦ span <custom-ident> <integer>

⇥ grid-column-end
   ↦ auto
   ↦ <integer>
   ↦ <custom-ident>
   ↦ <custom-ident> <integer>
   ↦ span <integer>
   ↦ span <custom-ident>
   ↦ span <custom-ident> <integer>

⇥ row-gap
   ↦ normal
   ↦ <length>
   ↦ <percentage>

⇥ grid-template-areas
   ↦ none
   ↦ <string>+

⇥ grid-template (shorthand)
   ↦ grid-template-areas
   ↦ grid-template-columns
   ↦ grid-template-rows

⇥ grid (shorthand)
   ↦ grid-auto-columns
   ↦ grid-auto-rows
   ↦ grid-auto-flow
   ↦ grid-template-columns
   ↦ grid-template-rows
   ↦ grid-template-areas

⇥ grid-auto-flow
   ↦ row
   ↦ column
   ↦ dense

⇥ grid-row (shorthand)
   ↦ grid-row-start
   ↦ grid-row-end

⇥ grid-area (shorthand)
   ↦ grid-row-start
   ↦ grid-row-end
   ↦ grid-column-start
   ↦ grid-column-end

⇥ grid-column (shorthand)
   ↦ grid-column-start
   ↦ grid-column-end

⇥ gap (shorthand)
   ↦ row-gap
   ↦ column-gap
```



Justify-items

align content in a grid item along row

justify-items: start;

justify-items: end;

justify-items: center;

justify-items: stretch;

Align-items

align-content in a grid item along column axis

align-items: start;

align-items: end;

align-items: center;

align-items: stretch;

Justify-Content

justify-content: start;

justify-content: end;

justify-content: center;

justify-content: stretch;

justify-content: space-around;

justify-content: space-between;

justify-content: space-evenly;

justifies all grid content on row axis when total grid size is smaller than container

Align-Content

align-content: start;

align-content: end;

align-content: center;

align-content: stretch;

align-content: space-around;

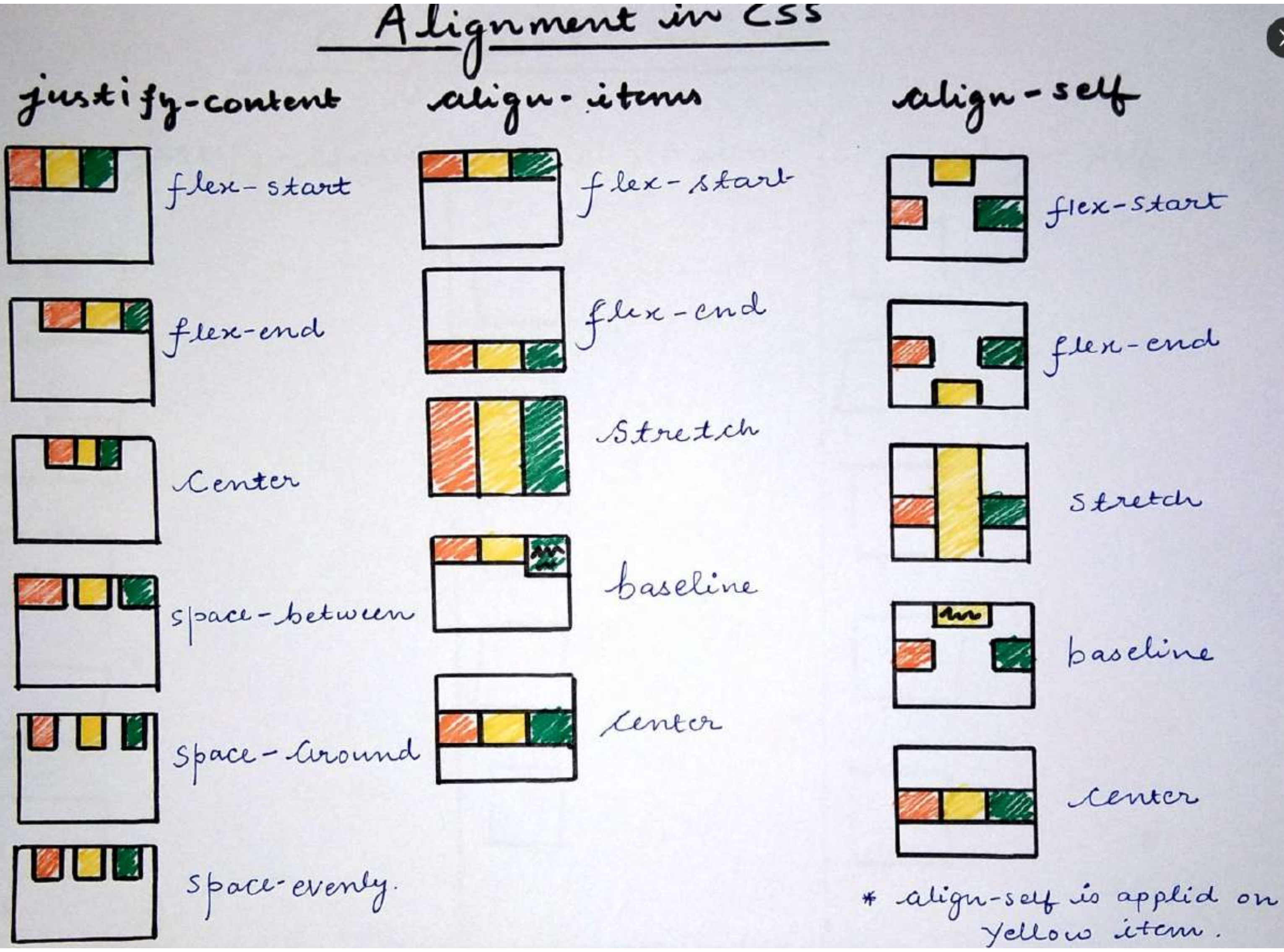
align-content: space-between;

align-content: space-evenly;

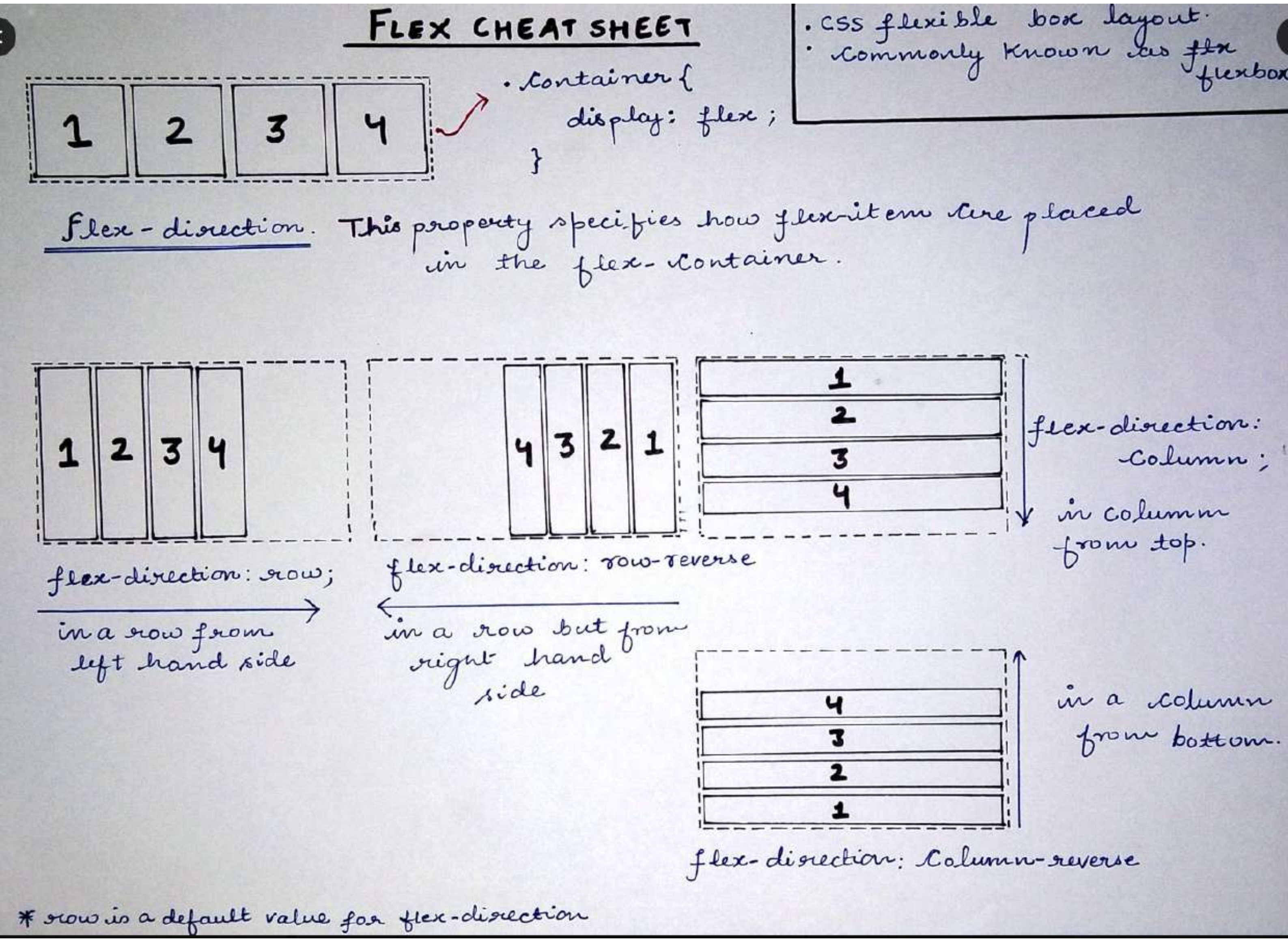
Justifies all grid content on column when total grid size is smaller than container.

Pratham♥

Alignment in CSS



* align-self is applid on yellow item.



FLEX CHEAT SHEET

- CSS flexible box layout.
- commonly known as ~~flx~~ flexbox

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

1
2
3
4

flex-direction. This property specifies how flex-item are placed in the flex-container.

1 2 3 4
4 3 2 1
1
2
3
4
4
3
2
1

flex-direction: row;

in a row from left hand side

flex-direction: row-reverse

in a row but from right hand side

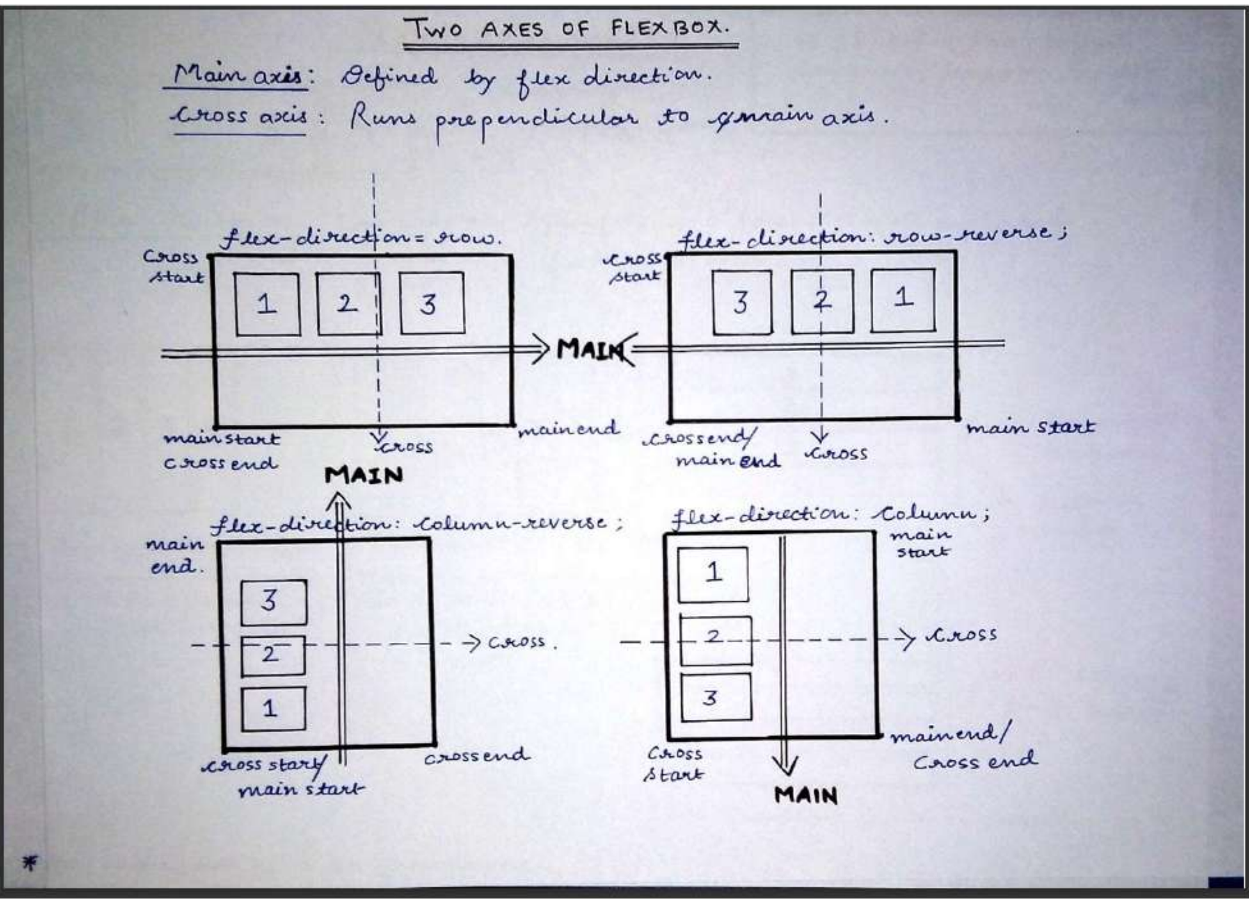
flex-direction: Column;

in column from top.

in a column from bottom.

flex-direction: Column-reverse

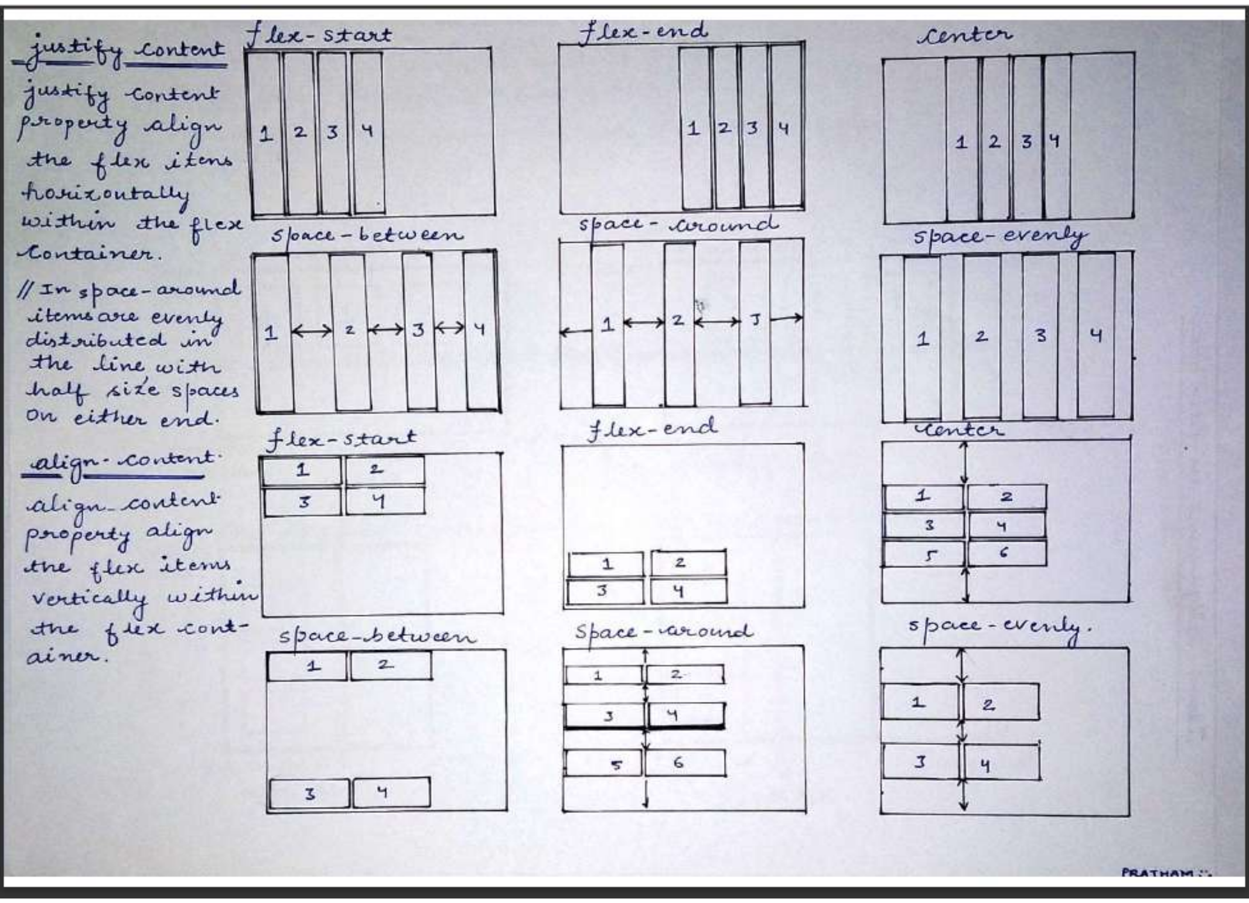
* row is a default value for flex-direction



TWO AXES OF FLEXBOX.

Main axis: Defined by flex direction.

Cross axis: Runs prependicular to gmain axis.



justify content

justify content property align the flex items horizontally within the flex container.

// In space-around items are evenly distributed in the line with half size spaces on either end.

align-content.

align-content property align the flex items vertically within the flex container.

flex-start

flex-end

center

space-between

space-around

space-evenly

flex-start

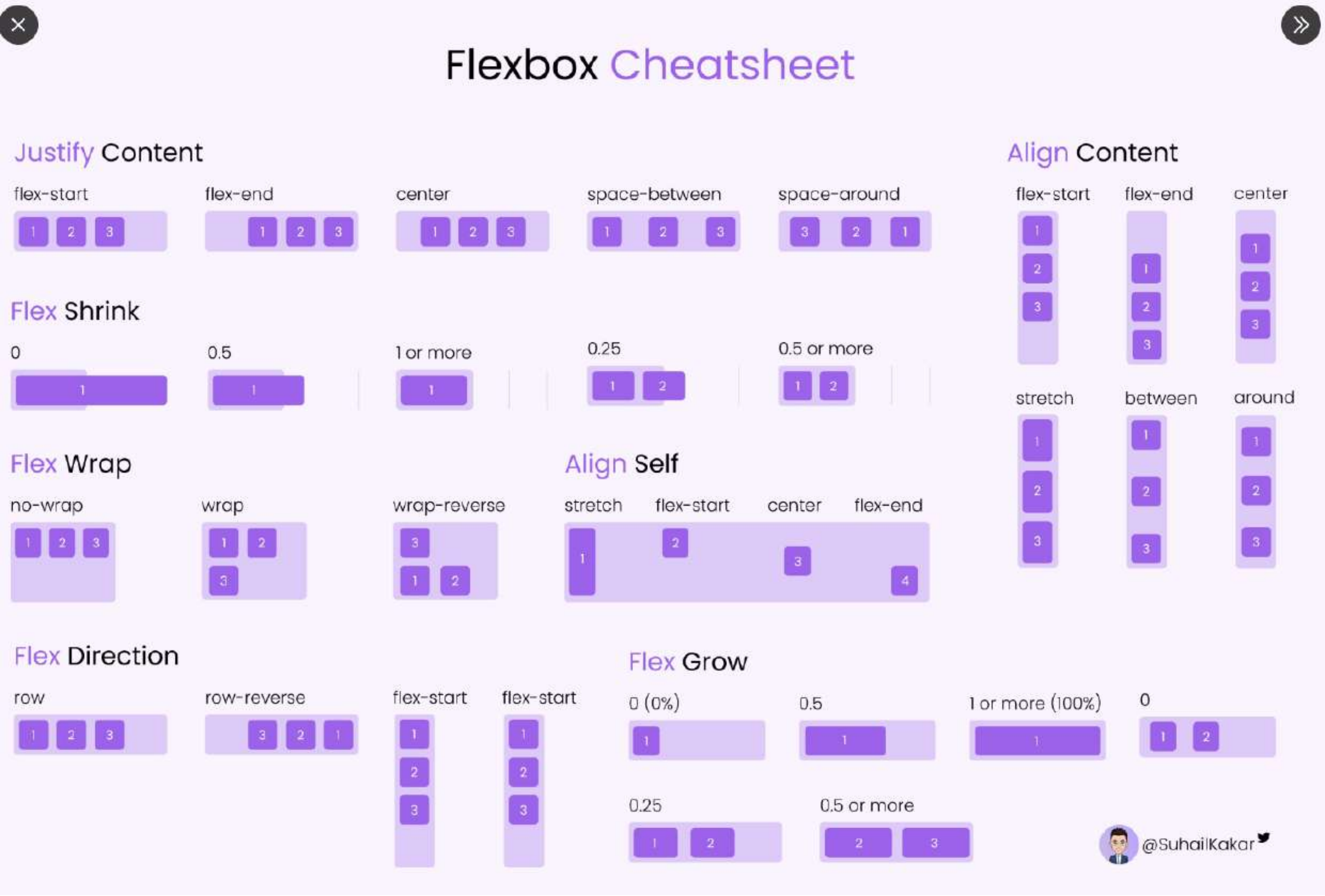
flex-end

center

space-between

Space-around

space-evenly.



Flexbox Cheatsheet

Justify Content

flex-start · flex-end · center · space-between · space-around

Flex Shrink

0 · 0.5 · 1 or more · 0.25 · 0.5 or more

Flex Wrap

no-wrap · wrap · wrap-reverse

Align Self

stretch · flex-start · center · flex-end

Align Content

flex-start · flex-end · center

stretch · between · around

Flex Direction

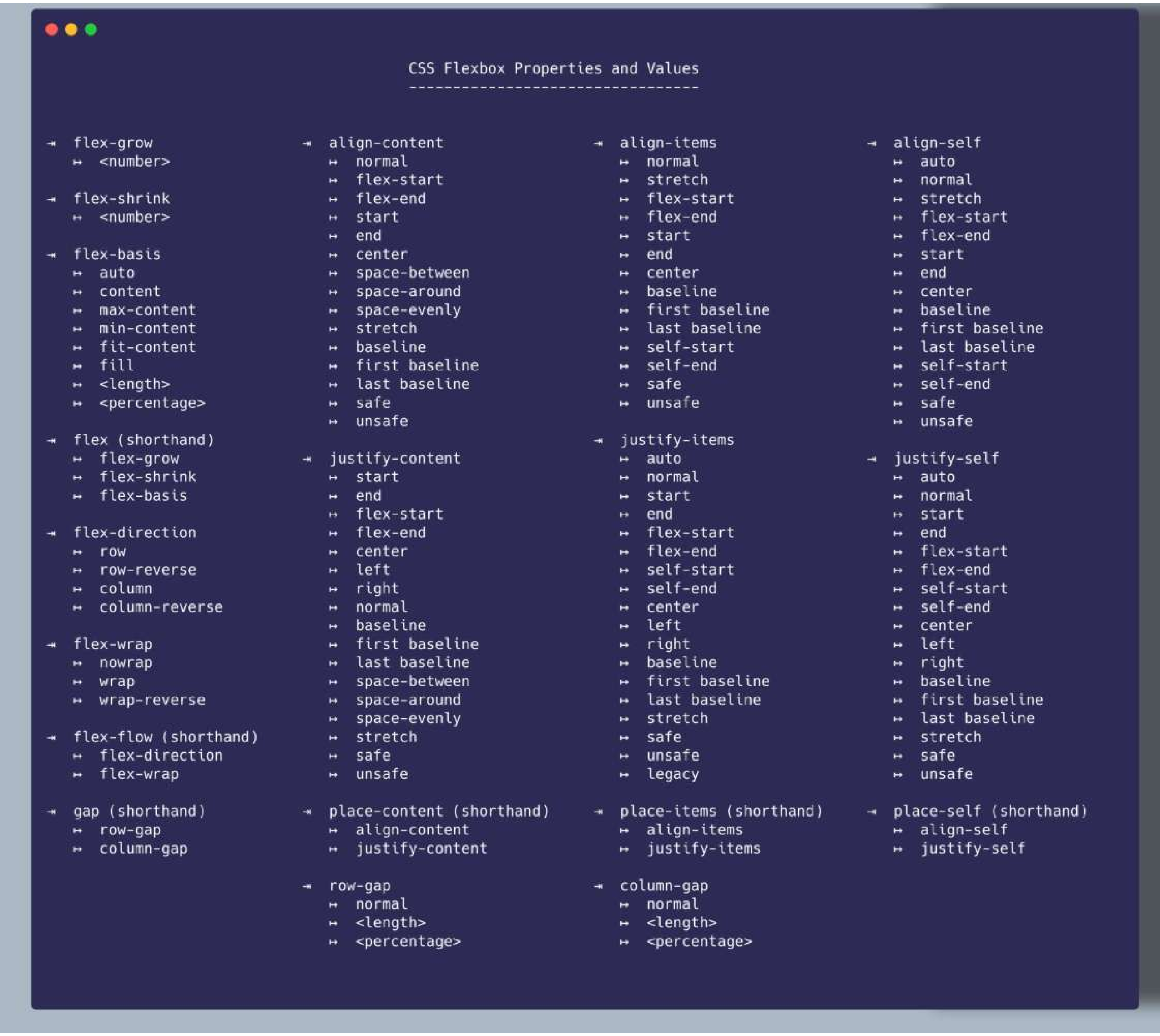
row · row-reverse · flex-start · flex-start

Flex Grow

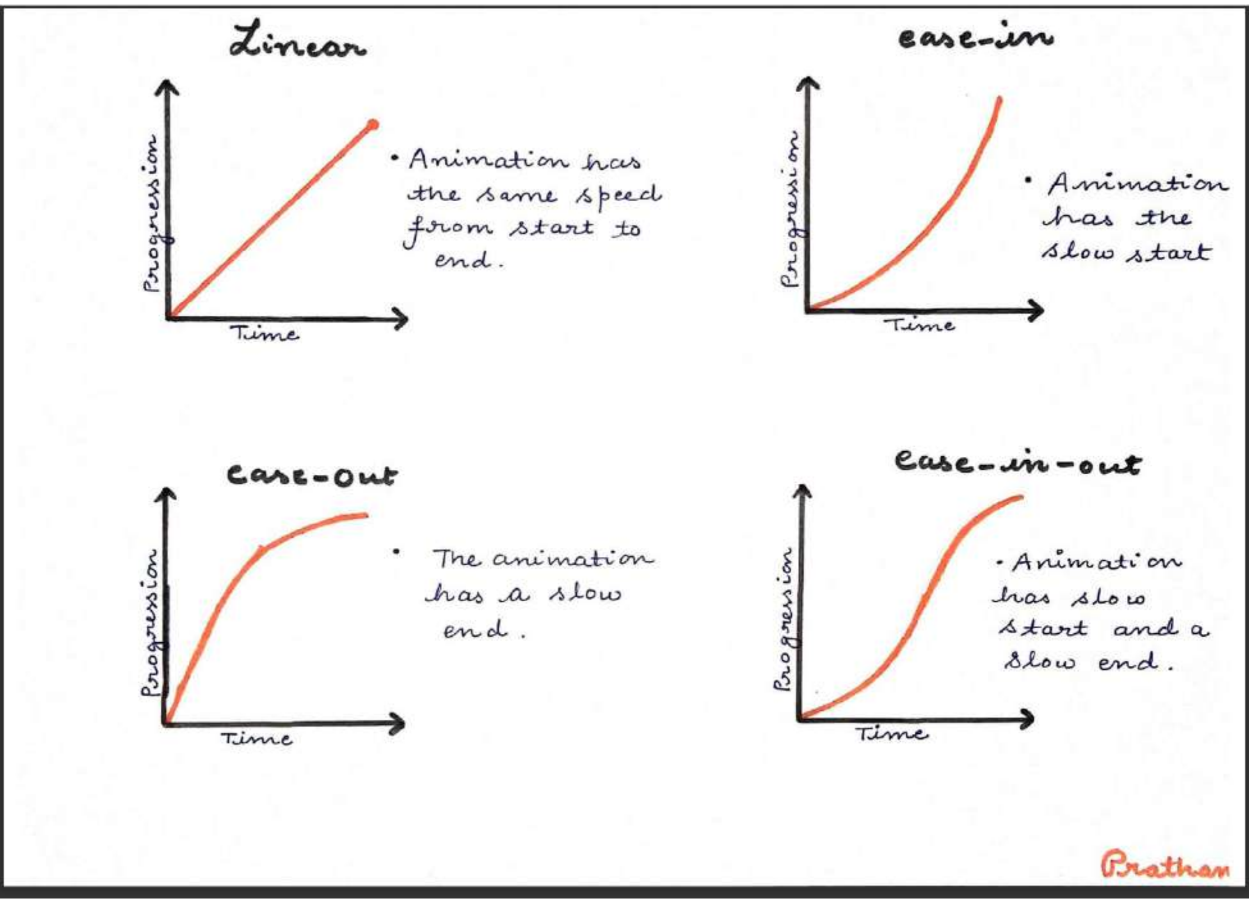
0 (0%) · 0.5 · 1 or more (100%) · 0

0.25 · 0.5 or more

@SuhailKakar



```
CSS Flexbox Properties and Values
---------------------------------

⇥ flex-grow
   ↦ <number>

⇥ flex-shrink
   ↦ <number>

⇥ flex-basis
   ↦ auto
   ↦ content
   ↦ max-content
   ↦ min-content
   ↦ fit-content
   ↦ fill
   ↦ <length>
   ↦ <percentage>

⇥ flex (shorthand)
   ↦ flex-grow
   ↦ flex-shrink
   ↦ flex-basis

⇥ flex-direction
   ↦ row
   ↦ row-reverse
   ↦ column
   ↦ column-reverse

⇥ flex-wrap
   ↦ nowrap
   ↦ wrap
   ↦ wrap-reverse

⇥ flex-flow (shorthand)
   ↦ flex-direction
   ↦ flex-wrap

⇥ gap (shorthand)
   ↦ row-gap
   ↦ column-gap

⇥ align-content
   ↦ normal
   ↦ flex-start
   ↦ flex-end
   ↦ start
   ↦ end
   ↦ center
   ↦ space-between
   ↦ space-around
   ↦ space-evenly
   ↦ stretch
   ↦ baseline
   ↦ first baseline
   ↦ last baseline
   ↦ safe
   ↦ unsafe

⇥ justify-content
   ↦ start
   ↦ end
   ↦ flex-start
   ↦ flex-end
   ↦ center
   ↦ left
   ↦ right
   ↦ normal
   ↦ baseline
   ↦ first baseline
   ↦ last baseline
   ↦ space-between
   ↦ space-around
   ↦ space-evenly
   ↦ stretch
   ↦ safe
   ↦ unsafe

⇥ place-content (shorthand)
   ↦ align-content
   ↦ justify-content

⇥ row-gap
   ↦ normal
   ↦ <length>
   ↦ <percentage>

⇥ align-items
   ↦ normal
   ↦ stretch
   ↦ flex-start
   ↦ flex-end
   ↦ start
   ↦ end
   ↦ center
   ↦ baseline
   ↦ first baseline
   ↦ last baseline
   ↦ self-start
   ↦ self-end
   ↦ safe
   ↦ unsafe

⇥ justify-items
   ↦ auto
   ↦ normal
   ↦ start
   ↦ end
   ↦ flex-start
   ↦ flex-end
   ↦ self-start
   ↦ self-end
   ↦ center
   ↦ left
   ↦ right
   ↦ baseline
   ↦ first baseline
   ↦ last baseline
   ↦ stretch
   ↦ safe
   ↦ unsafe
   ↦ legacy

⇥ place-items (shorthand)
   ↦ align-items
   ↦ justify-items

⇥ column-gap
   ↦ normal
   ↦ <length>
   ↦ <percentage>

⇥ align-self
   ↦ auto
   ↦ normal
   ↦ stretch
   ↦ flex-start
   ↦ flex-end
   ↦ start
   ↦ end
   ↦ center
   ↦ baseline
   ↦ first baseline
   ↦ last baseline
   ↦ self-start
   ↦ self-end
   ↦ safe
   ↦ unsafe

⇥ justify-self
   ↦ auto
   ↦ normal
   ↦ start
   ↦ end
   ↦ flex-start
   ↦ flex-end
   ↦ self-start
   ↦ self-end
   ↦ center
   ↦ left
   ↦ right
   ↦ baseline
   ↦ first baseline
   ↦ last baseline
   ↦ stretch
   ↦ safe
   ↦ unsafe

⇥ place-self (shorthand)
   ↦ align-self
   ↦ justify-self
```



Linear

- Animation has the same speed from start to end.

ease-in

- Animation has the slow start

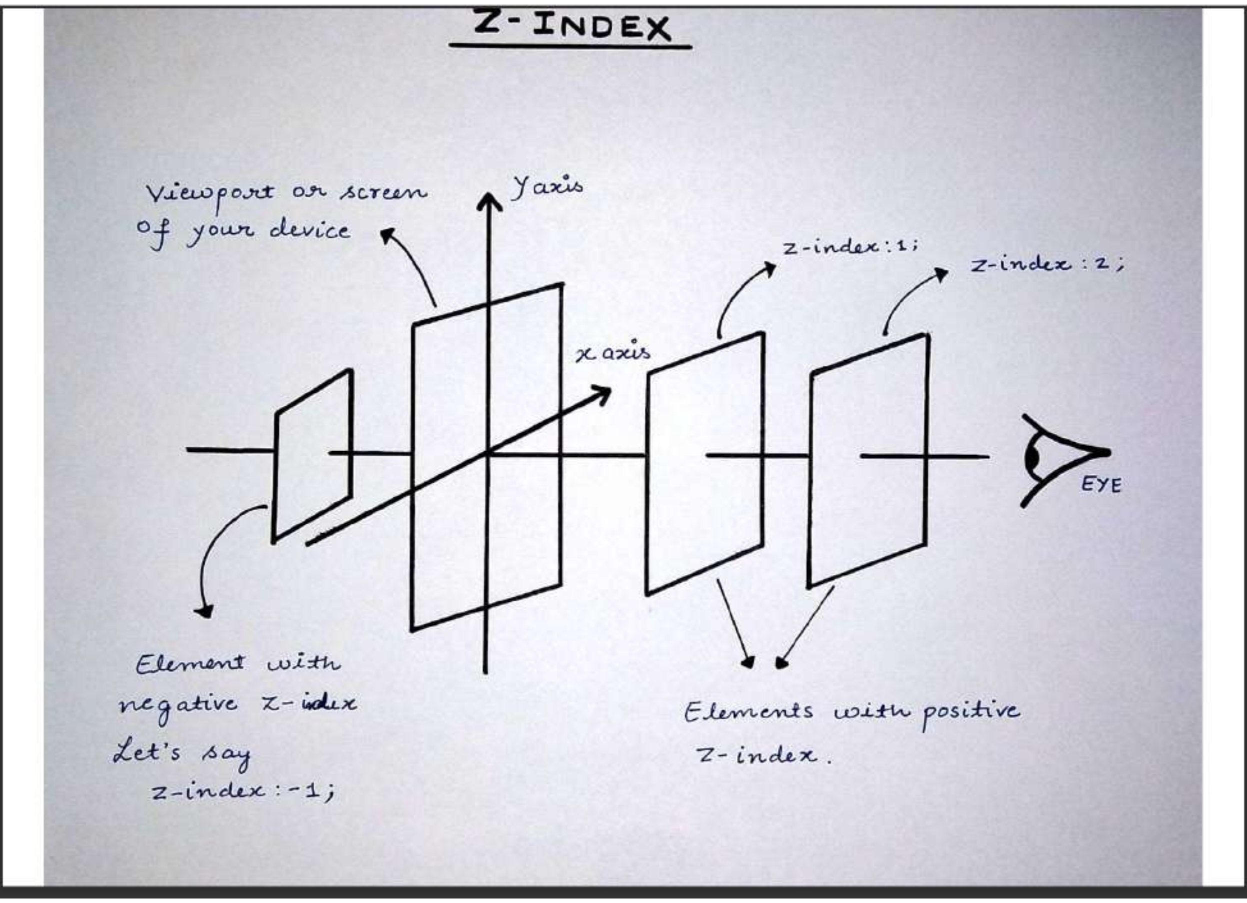
ease-out

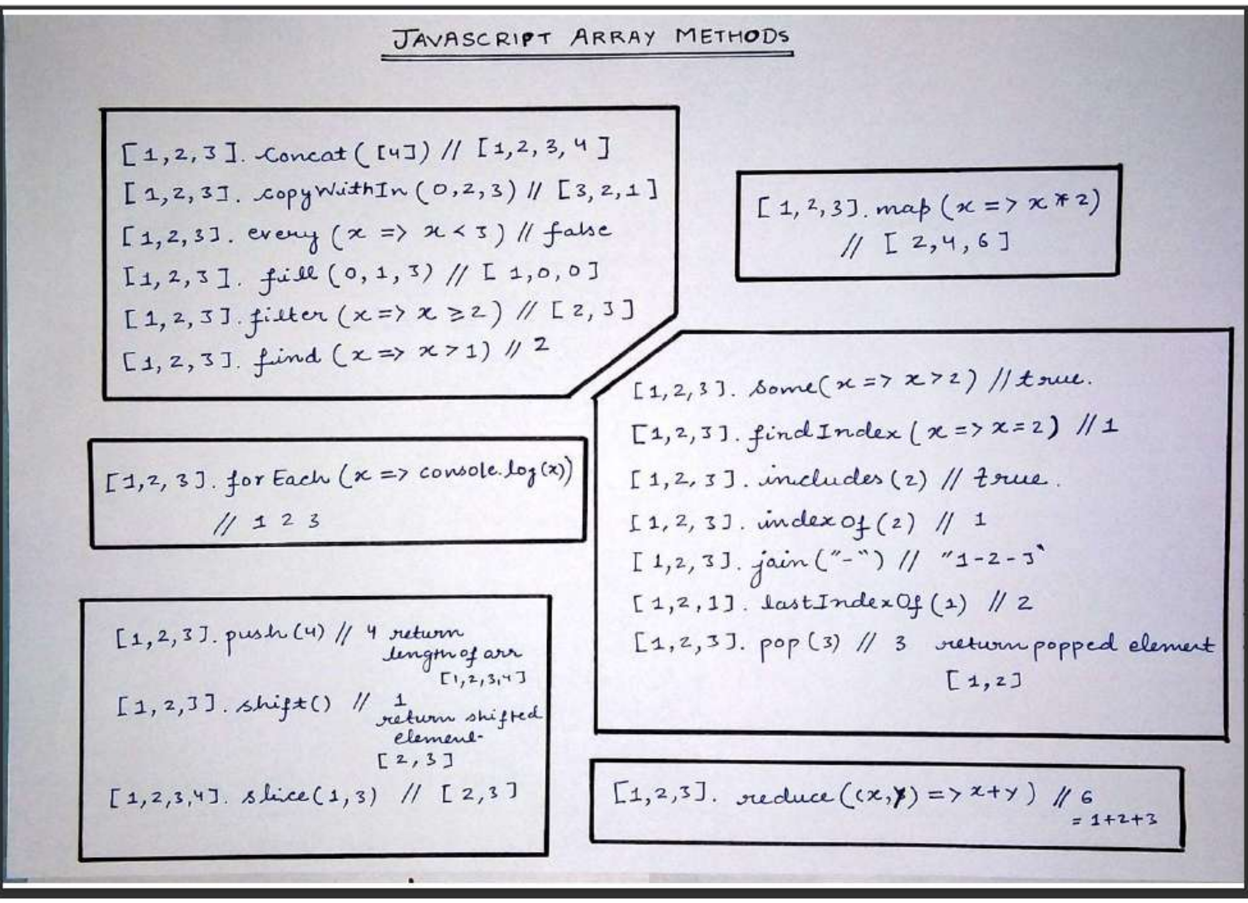
- The animation has a slow end.

ease-in-out

- Animation has slow start and a slow end.

Prathan


Z-INDEX
Viewport or screen
of your device
Y axis
z-index:1;
z-index:2;
x axis
EYE
Element with
negative z-index
Let's say
z-index:-1;
Elements with positive
z-index.



JAVASCRIPT ARRAY METHODS

```
[1,2,3].Concat([4]) // [1,2,3,4]
[1,2,3].copyWithIn(0,2,3) // [3,2,1]
[1,2,3].every(x => x < 3) // false
[1,2,3].fill(0,1,3) // [1,0,0]
[1,2,3].filter(x => x ≥ 2) // [2,3]
[1,2,3].find(x => x > 1) // 2
```

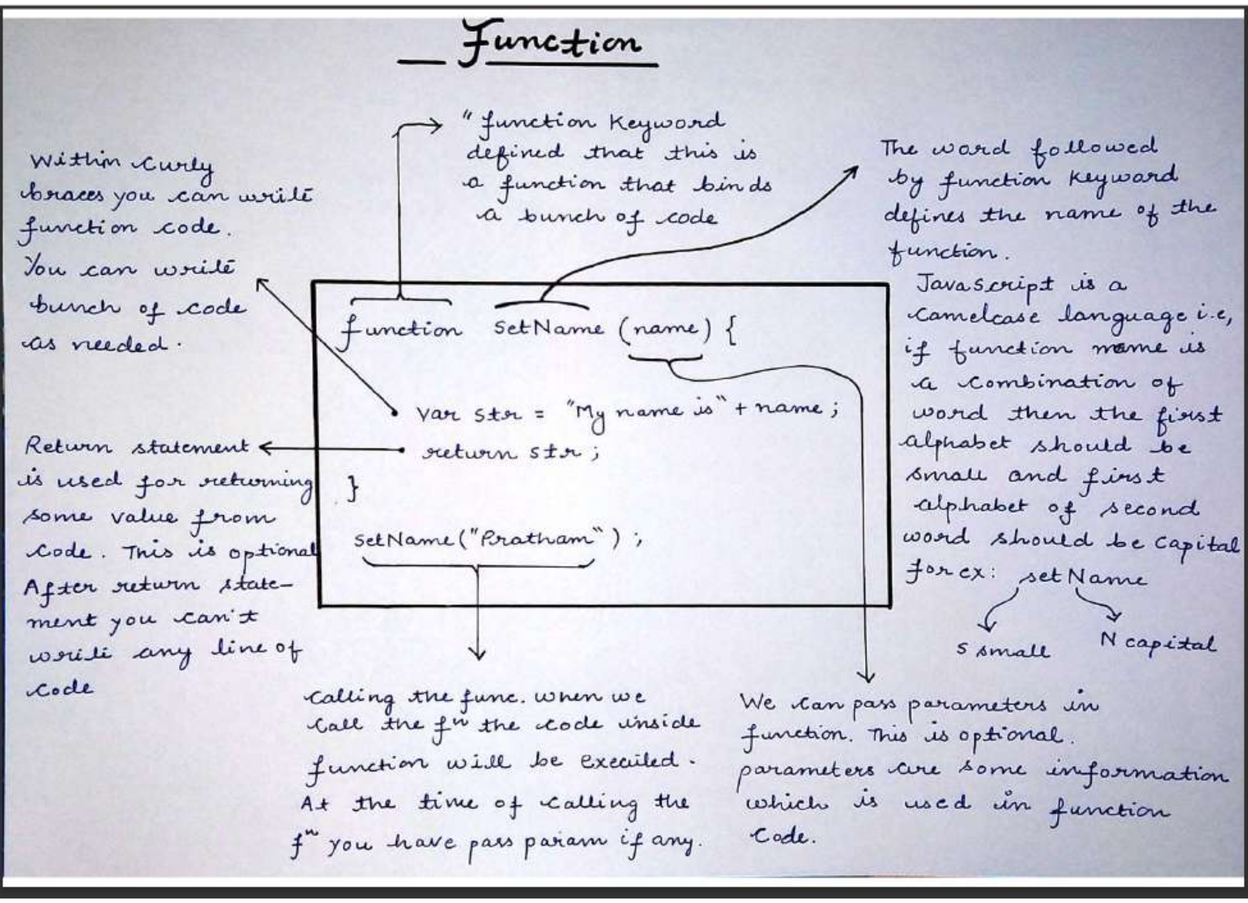
```
[1,2,3].forEach(x => console.log(x))
       // 1 2 3
```

```
[1,2,3].push(4) // 4 return
                  length of arr
                  [1,2,3,4]
[1,2,3].shift() // 1
                  return shifted
                  element
                  [2,3]
[1,2,3,4].slice(1,3) // [2,3]
```

```
[1,2,3].map(x => x * 2)
     // [2,4,6]
```

```
[1,2,3].some(x => x > 2) // true.
[1,2,3].findIndex(x => x = 2) // 1
[1,2,3].includes(2) // true.
[1,2,3].indexOf(2) // 1
[1,2,3].jain("-") // "1-2-3"
[1,2,1].lastIndexOf(1) // 2
[1,2,3].pop(3) // 3 return popped element
                   [1,2]
```

```
[1,2,3].reduce((x,y) => x+y) // 6
                                = 1+2+3
```



Function

"function Keyword defined that this is a function that binds a bunch of code

Within curly braces you can write function code. You can write bunch of code as needed.

The word followed by function keyword defines the name of the function.

Javascript is a camelcase language i.e, if function name is a combination of word then the first alphabet should be small and first alphabet of second word should be capital for ex: setName

s small

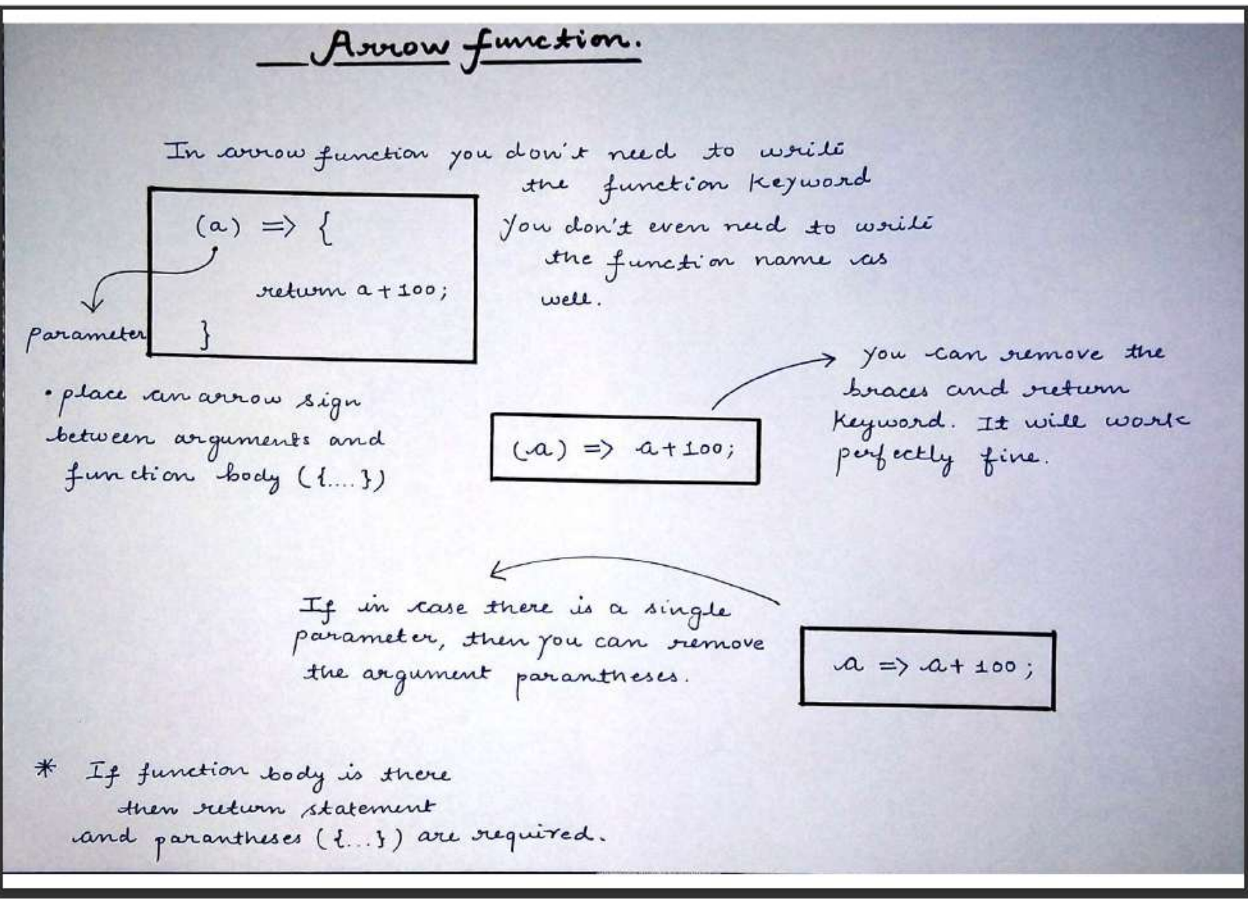
N capital

```
function SetName (name) {

    • var str = "My name is" + name;
    • return str;
}

SetName("Pratham");
```

Return statement is used for returning some value from code. This is optional After return statement you can't write any line of code

Calling the func. when we call the fn the code inside function will be exeuted. At the time of calling the fn you have pass param if any.

We can pass parameters in function. This is optional. parameters are some information which is used in function code.



Arrow function.

In arrow function you don't need to write the function keyword

```
(a) => {

      return a + 100;

}
```

parameter

You don't even need to write the function name as well.

- place an arrow sign between arguments and function body ({....})

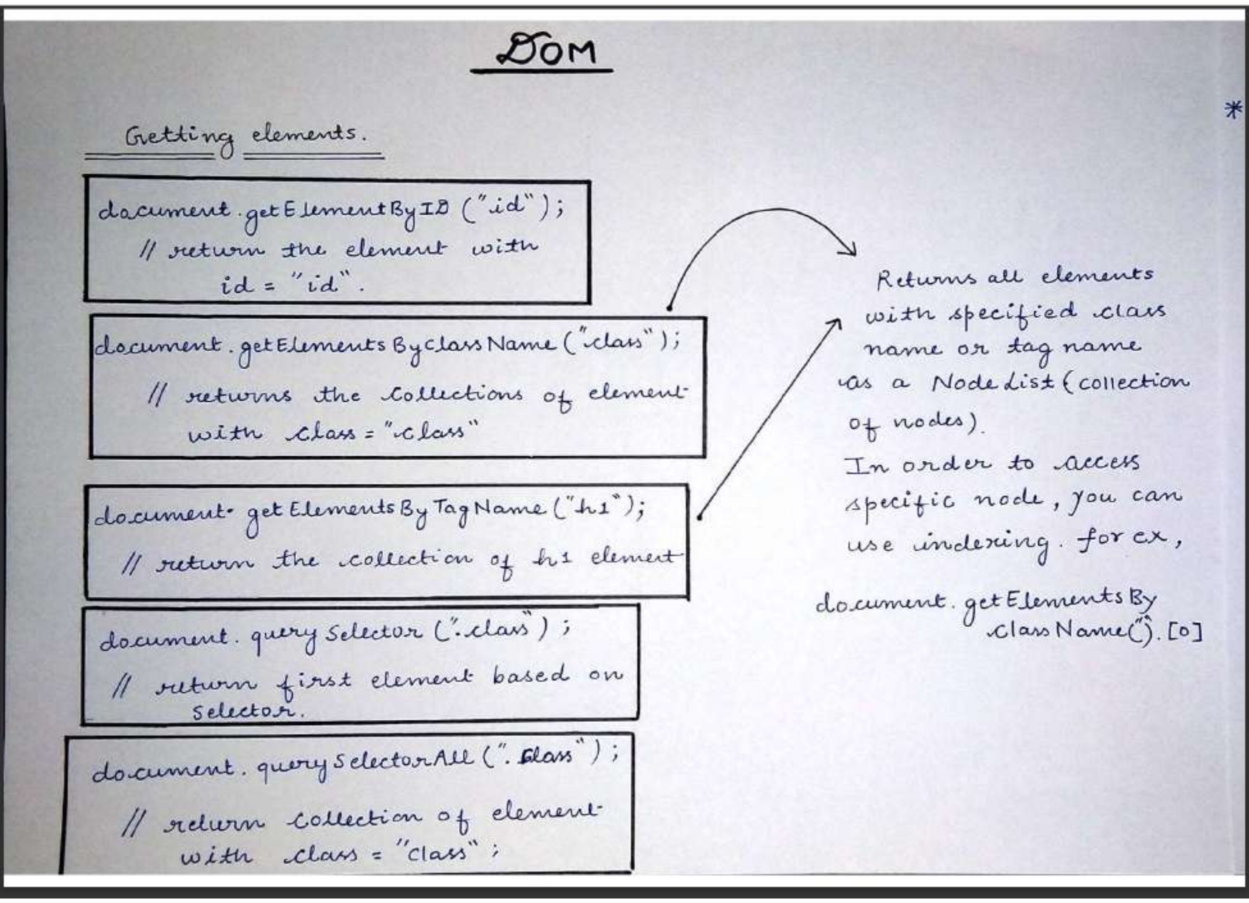
```
(a) => a+100;
```

you can remove the braces and return keyword. It will work perfectly fine.

If in case there is a single parameter, then you can remove the argument parantheses.

```
a => a+ 100;
```

* If function body is there then return statement and parantheses ({...}) are required.



DOM

*

Getting elements.

```
document.getElementByID ("id");
  // return the element with
      id = "id".
```

```
document.getElementsByclassName ("class");
  // returns the collections of element
      with class = "class"
```

```
document. getElementsByTagName ("h1");
  // return the collection of h1 element
```

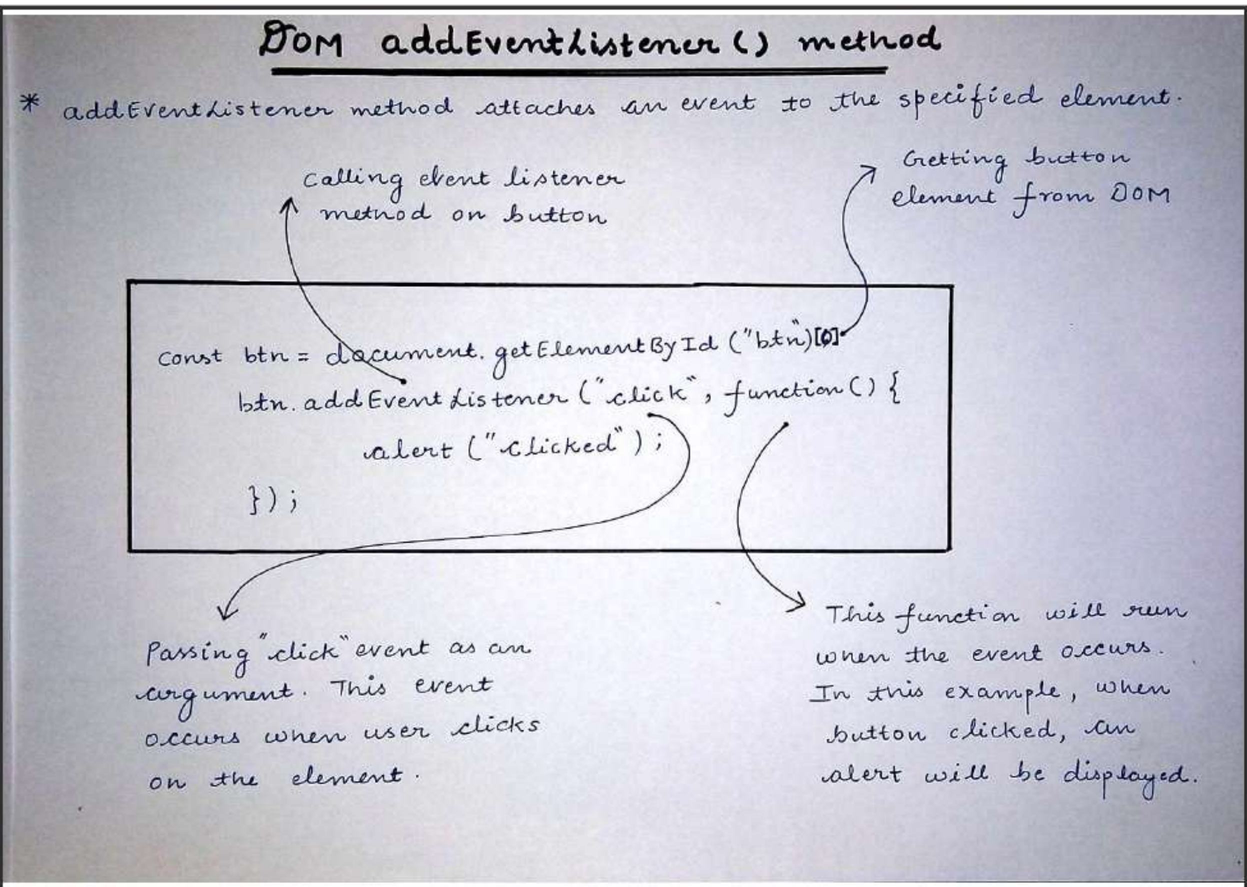
```
document. querySelector (".class");
  // return first element based on
      selector.
```

```
document. querySelectorAll (".class");
  // return collection of element
      with class = "Class";
```

Returns all elements with specified class name or tag name as a Node List (collection of nodes).

In order to access specific node, you can use indexing. for ex,

```
document. getElementsBy
          className(""). [0]
```



DOM addEventListener() method

* addEventListener method attaches an event to the specified element.

Calling event listener method on button

Getting button element from DOM

```
Const btn = document. getElementById ("btn")[0]
      btn. addEventListener ("click", function() {
              alert ("clicked");
      });
```

Passing "click" event as an argument. This event occurs when user clicks on the element.

This function will run when the event occurs. In this example, when button clicked, an alert will be displayed.