CSS

### What is CSS?

CSS stands for Cascading Style Sheets.

CSS is the language we use to style a Web page.

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

### Selectors

Selectors simply refer to the HTML elements to which CSS rules apply; they’re what is actually being “selected” for each rule. Selectors Include Universal, Type, Class and ID

#### Universal Selector \*

The universal selector will select elements of any type, hence the name “universal”, and the syntax for it is a simple asterisk. In the example below, every element would have the color: purple; style applied to it.

\* {

color: purple;

}

#### Type Selectors

A type selector (or element selector) will select all elements of the given element type, and the syntax is just the name of the element Ex:

Element Name {

}

Class Selectors .

Class selectors will select all elements with the given class, which is just an attribute you place on an HTML element.

#### **ID Selectors #**

ID selectors are similar to class selectors. They select an element with the given ID

### 

### **The Cascade of CSS**

A CSS declaration that is more specific will override ones that are less specific. Inline styles, have the highest specificity compared to selectors, while each type of selector has its own specificity level that contributes to how specific a declaration is. The higher a thing is, the more important and specific it is.

1. ! Important

1. Things labelled as ! important has the highest importance.

2. Type

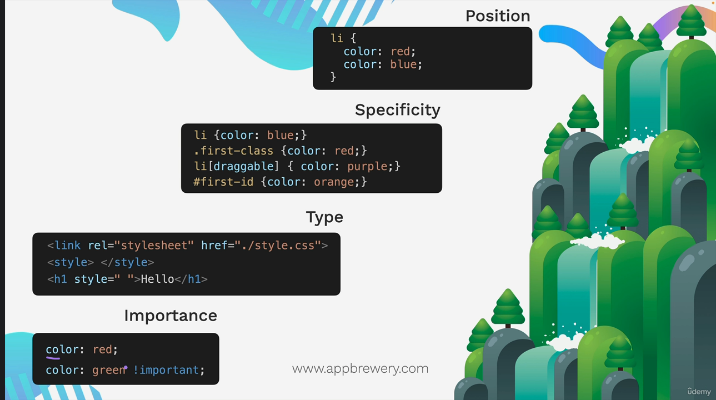
1. Inline styling (Most important type)
2. Internal styling
3. External styling

3. Selectors

1. ID selectors (most specific selector)
2. Class selectors
3. Type selectors
4. Universal selector (None)

4. Position

1. The thing coded last



The lower the code the more important it is

### The Box model

The box model is a concept within web programming that holds every single thing on a web page is a rectangular box. These boxes can have other boxes in them and can sit alongside one another. There are four attributes to the box model. Content, Padding, Border and Margin

* **Content box**: The area where your content is displayed, which can be sized using properties like [width](https://developer.mozilla.org/en-US/docs/Web/CSS/width) and [height](https://developer.mozilla.org/en-US/docs/Web/CSS/height).
* **Padding box**: The padding sits around the content as white space; its size can be controlled using [padding](https://developer.mozilla.org/en-US/docs/Web/CSS/padding) and related properties.
* **Border box**: The border box wraps the content and any padding. Its size and style can be controlled using [border](https://developer.mozilla.org/en-US/docs/Web/CSS/border) and related properties.
* **Margin box**: The margin is the outermost layer, wrapping the content, padding, and border as whitespace between this box and other elements. Its size can be controlled using [margin](https://developer.mozilla.org/en-US/docs/Web/CSS/margin) and related properties.A picture containing shape

  Description automatically generated

### Position

The position [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property sets how an element is positioned in a document. (static, relative, fixed, absolute or sticky)

#### postion static;

The element is positioned according to the normal flow of the document. The top, right, bottom, left, and z-index properties have no effect. This is the default value.

#### position : relative;

The element is positioned according to the normal flow of the document, and then offset relative to itself based on the values of top, right, bottom, and left.

#### position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

#### position: absolute;

It is positioned relative to its closest positioned ancestor, if any; otherwise, it is top left of the web page. Its final position is determined by the values of top, right, bottom, and left. This value creates a new stacking context when the value of z-index is not auto. The margins of absolutely positioned boxes do not collapse with other margins.

#### position: sticky;

An element with position: sticky; is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

#### The z-index Property

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative z-index.

The element with the highest z-index will stack in front.

Create circle in css using border radius

Border radius 50%; is a perfect circle

Combining selectors

You can use

Group ,

You can use multiple selectors at the same time using grouping. To do so separate a selector by a comma

h1, h2,{

color: blue;

}

Child >

The child selector selects all elements that is the direct child of a specified element. It applies to the direct child of the left side. (Only one generation) Not grand children etc

selector > selector {

}

Descendant

The descendant selector matches all elements that are descendants of a specified element. Applies to the descendant. To do this we separate them with a space. The selector to the left of the space is the ancestor while the selector to the right of the space is the descendant.

Ancestor Descendant {

}

Chaining

No spaces

### Display

The display property is the most important CSS property for controlling layout.

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline and can be changed

#### Block

display: block;

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

<div> <h1> - <h6> <p> <form> <header> <footer> <section>

#### Inline

display: inline;

An inline element does not start on a new line and only takes up as much width as necessary.

Examples of inline elements:

<span> <a> <img>

#### Inline-block

display: inline-block

Inline-block allows to set a width and height on the element.

It does not add a line-break after the element, so the element can sit next to other elements.

The top and bottom margins/paddings are respected

#### None

display: none;

Does not display the element. Commonly used with JavaScript to hide and show elements without deleting and recreating them.

### Float and Clear

#### Float

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.In its simplest use, the float property can be used to wrap text around images

The float property can have one of the following values:

left - The element floats to the left of its container

right - The element floats to the right of its container

none - The element does not float (will be displayed just where it occurs in the text). This is default

inherit - The element inherits the float value of its parent

#### Clear

When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property. When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

The clear property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

none - The element is not pushed below left or right floated elements. This is default

left - The element is pushed below left floated elements

right - The element is pushed below right floated elements

both - The element is pushed below both left and right floated elements

inherit - The element inherits the clear value from its parent

.

# **Responsive Web Design**

## **What is Responsive Web Design?**

Responsive web design makes your web page look good on all devices.

Responsive web design uses only HTML and CSS.

Responsive web design is not a program or a JavaScript.

There are four main ways to achieve responsive web design.

CSS Media Queries

CSS Flexbox

CSS Grid

CSS Frameworks (Bootstrap)

### CSS Media Query

The CSS Media Query gives you a way to apply CSS only when the browser and device environment matches a rule that you specify.

Media queries are a key part of responsive web design, as they allow you to create different layouts depending on the size of the viewport, but they can also be used to detect other things about the environment your site is running on, for example whether the user is using a touchscreen rather than a mouse.

A media query is composed of an optional media type and any number of media feature expressions, which may optionally be combined in various ways using logical operators. Media queries are case-insensitive.

Media types define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.

Media features describe a specific characteristic of the user agent, output device, or environment such as max-width and min-width

@media (media-feature-rule) {

/\* CSS rules go here \*/

}

For example,

@media (max-width: 600px) {

h1{

color: blue

}

}

This says that when a screen’s width is 600px and lower then the h1 color will be blue.

If it was min-width then it would say that when a screen’s width is 600px andgreater then the h1 color would be blue.

max-width for smaller devices

min-width for larger devices

Media queries can be combined using and/or

@media (min-width: 600px) and (max-width: 900px) {

h1{

color: blue

}

}

We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

/\* 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) \*/

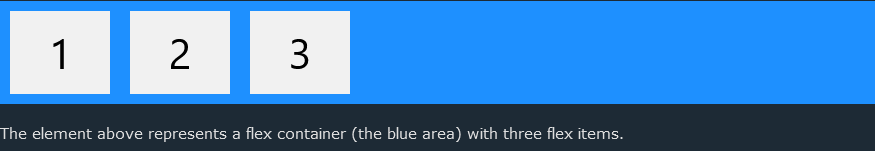
@media only screen and (min-width: 1200px) {...}

### Flexbox

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure

To start using the Flexbox model, you need to first define a flex container which will hold your items in HTML. Then select that container with CSS and do display: flex; to make them flexible.

-HTML

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

-CSS

.flex-container:{

display: flex;

}

#### Flex Container Properties

These Properties are applied to the flex container.

##### The flex-direction Property

The flex-direction property defines in which direction the container wants to stack the flex items.

**flex-direction: column;** The column value stacks the flex items vertically (from top to bottom):

**flex-direction: row;** The row value stacks the flex items horizontally (from left to right):

##### The justify-content Property

The justify-content property is used to align the flex items.

**justify-content: center;** The center value aligns the flex items at the center of the container

##### The align-items Property

The align-items property is used to align the flex items.

**align-items: center;** The center value aligns the flex items in the middle of the container

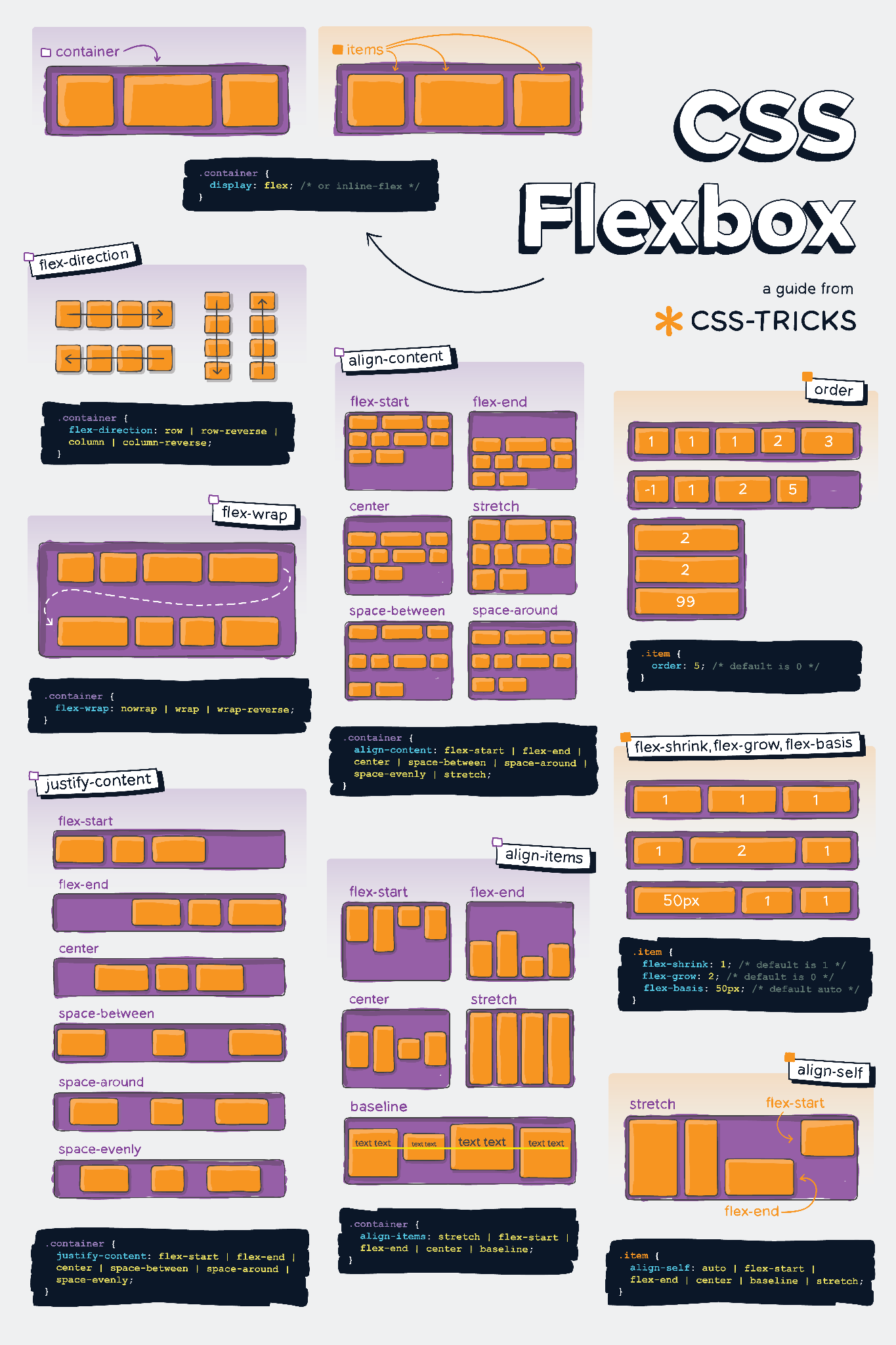
**align-items: baseline;** The baseline value aligns the flex items such as their baselines aligns

Perefectly center something with this code

display: flex;

justify-content: center;

align-items: center;



CSS Grid

## **Grid Layout**

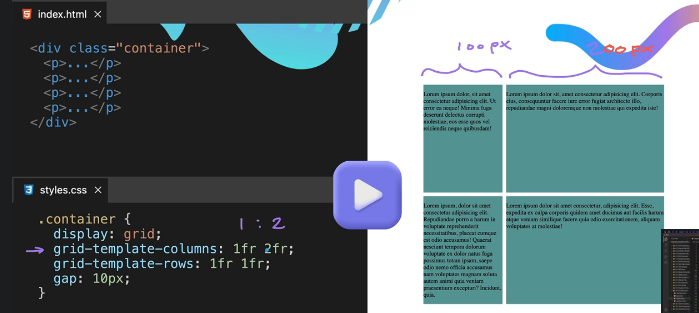
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

A grid layout consists of a parent element, with one or more child elements.

To start using the Grid model, you need to first define a grid container which will hold your items in HTML. Then select that container with CSS and do display: grid;. Lastly, set out the grid layout using grid-template-columns and grid-template-rows.

**grid-template-columns**  Specifies the size of the columns, and how many columns in a grid layout.

**grid-template-rows** Specifies the size of the rows in a grid layout and how many rows in a grid layout.



The css code creates a column with a size of two ( 1fr 2fr) and rows with a size of two (1fr 1fr)

\*Notice how the top left column is larger than the rest. This is because it is 2fr.

(The fr unit (a “fraction”) can be used when defining grids like any other CSS length such as %, px or em.)