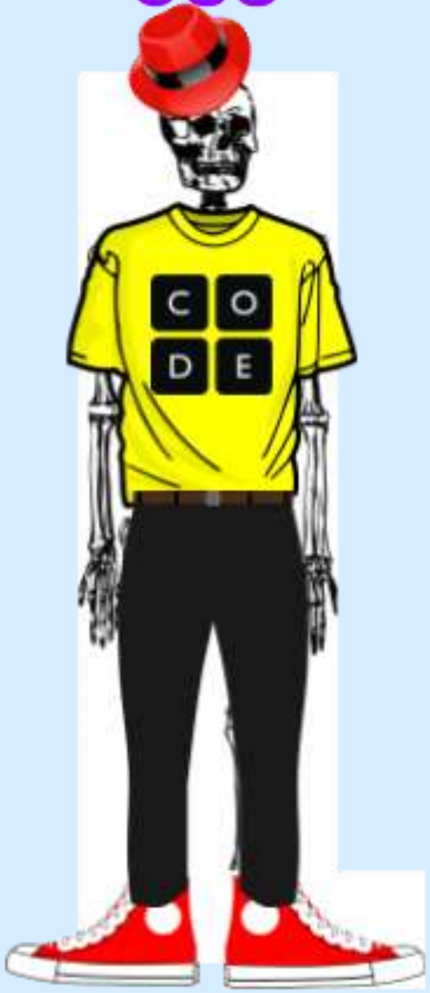
CASCADING STYLE SHEETS (CSS)

WORDS
&
IMAGES

HTML

CSS



THE CONTENT THE STRUCTURE THE STYLE

HTML HTML + CSS

UNIT - 1

Syllabus:

- **CSS 3:** Syntax structure, using style sheets, Box model, Grid, Flexbox. Responsive Web Design using Media Queries, use of viewport, Transition, Animation. CSS Framework: Bootstrap.
- **XML:** Introduction, syntax, Validating XML with Document type definition and XML Schemas.

INTRODUCTION

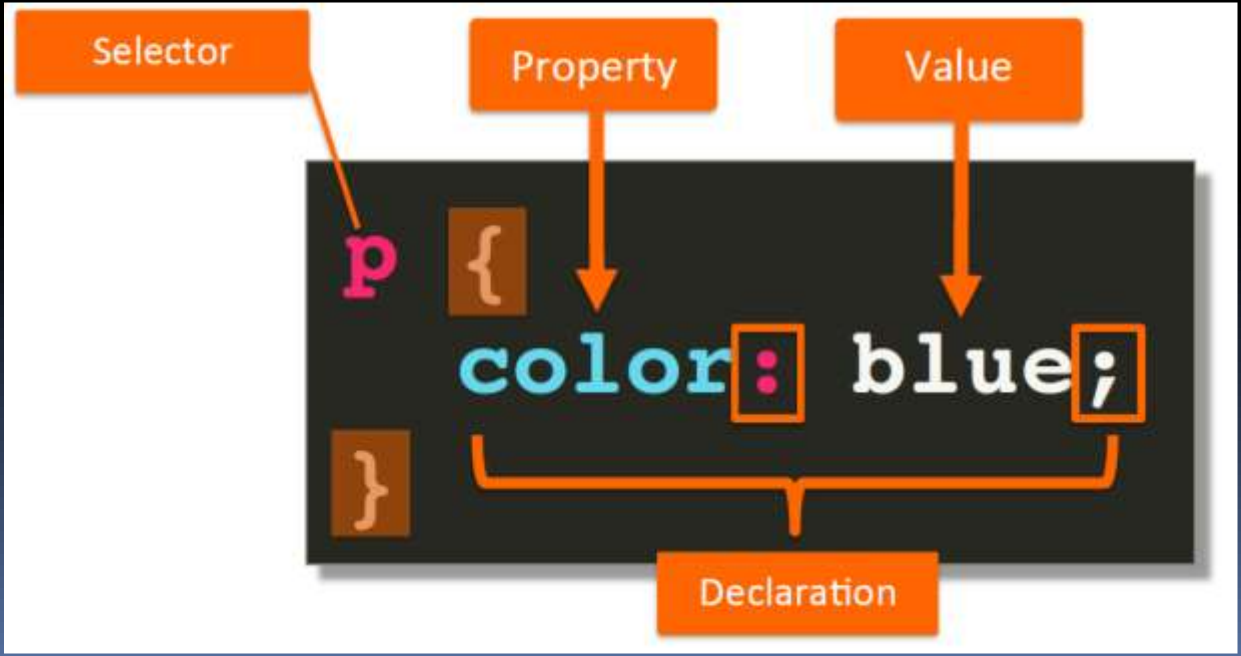
- STYLESHEET: set of RULES that expresses the presentation and layout of web pages.
- CSS:
  - a style sheet language used **for describing the presentation of a document** written in a markup language (HTML).
  - describes **how** HTML elements should be displayed the user.
  - Separates content from presentation.
  - **Latest version – CSS 3**

- Collection of CSS rules -> Stylesheet
- CSS works by associating rules with HTML elements.
- <u>CSS Rule Syntax:</u>

Selector

$$\underbrace{\{\ \text{Property1: value ; property2: value;...}\ \}}_{\text{Declaration}}$$

Example:

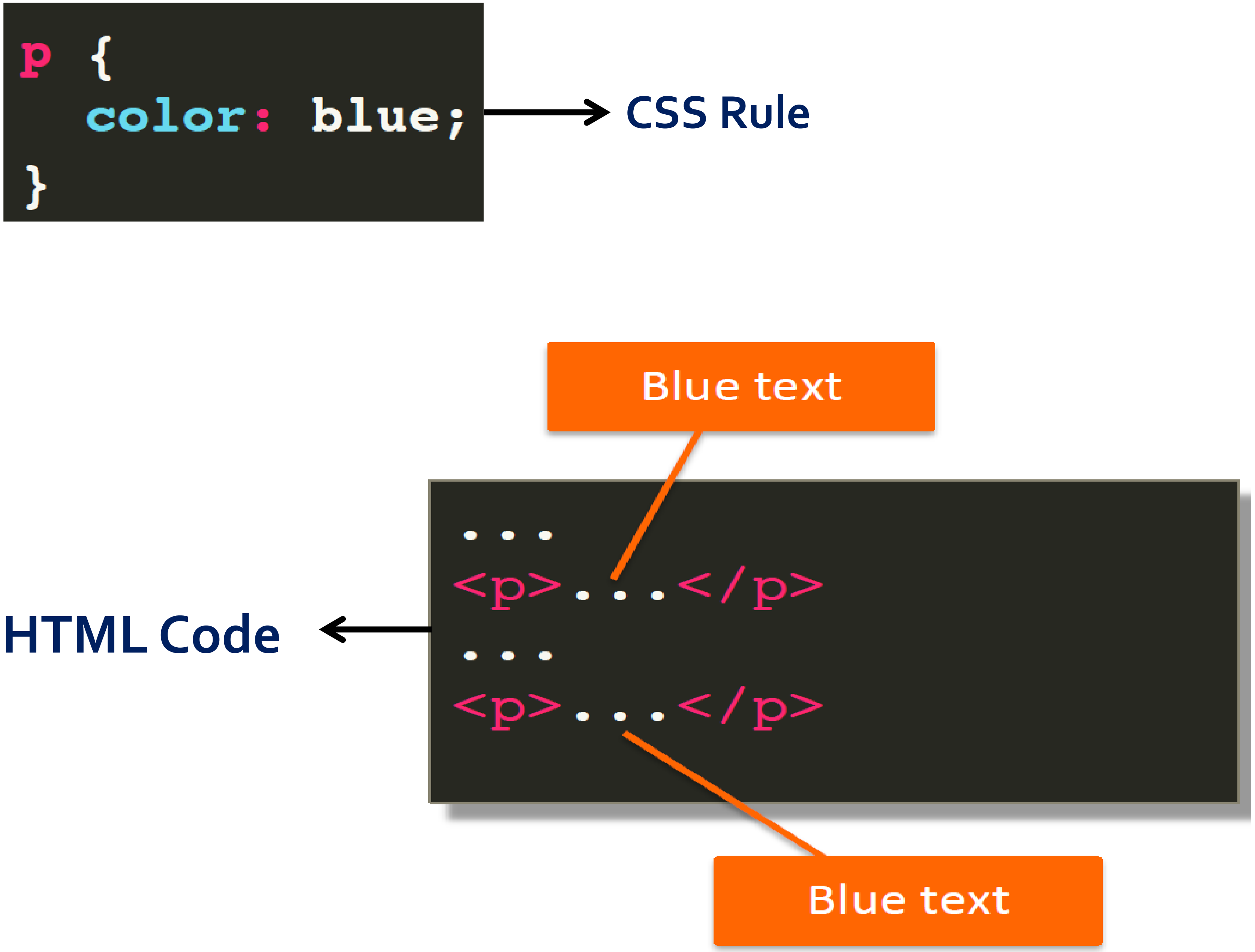


CSS Selectors

- used to "**find**" **(or select) the HTML element(s)** you want to style / **apply CSS rules**.
- **Basic Selectors:**
  - **Element name/type Selector**
  - **Class Selector**
  - **Id Selector**
  - **Universal Selector**
  - **Grouping Selector**
  - **Pseudo-Class Selector**

- **Element name/type Selector:**
  - Selects HTML elements based on the element name and applies CSS rules.
    - Syntax:

```
element-name {
            property1 : value; property2 : value;
            ……
}
```



```
p {
   color: blue;
}
```

```
...
<p>...</p>
...
<p>...</p>
```

Class Selector / Stylesheet 'class':

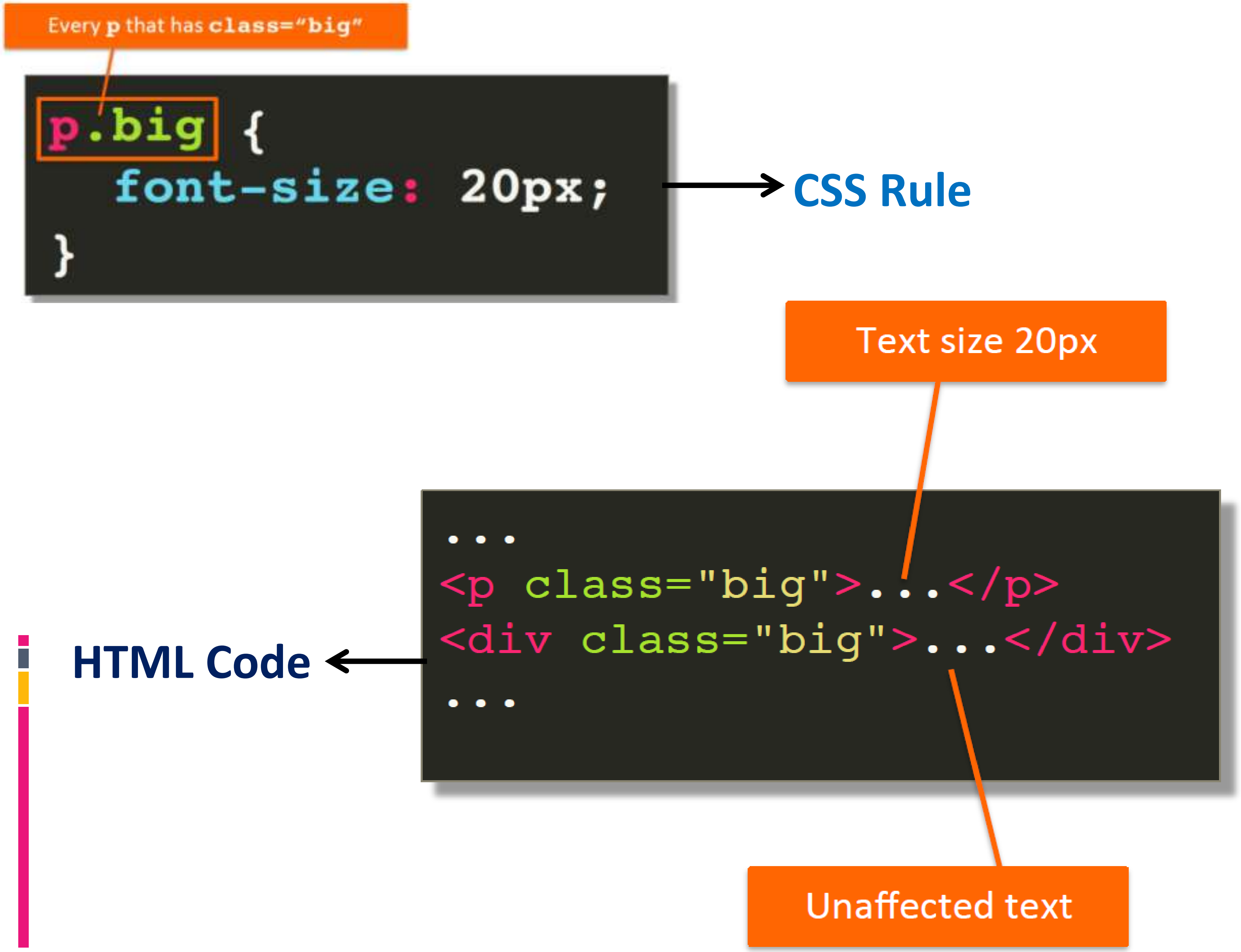
- Selects HTML elements with a specific **'class'** attribute.
- **Allows to define multiple styles for the same type of HTML element.**

Syntax 1:

```
Selector.classname
{  Property1: value; property2: value;  }
```

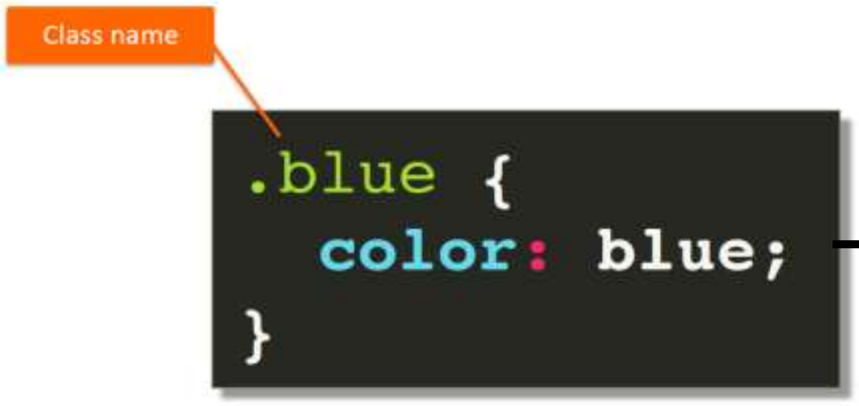
Syntax 2:

```
.classname
{  Property1: value; property2: value;  }
```



```css
p.big {
   font-size: 20px;
}
```

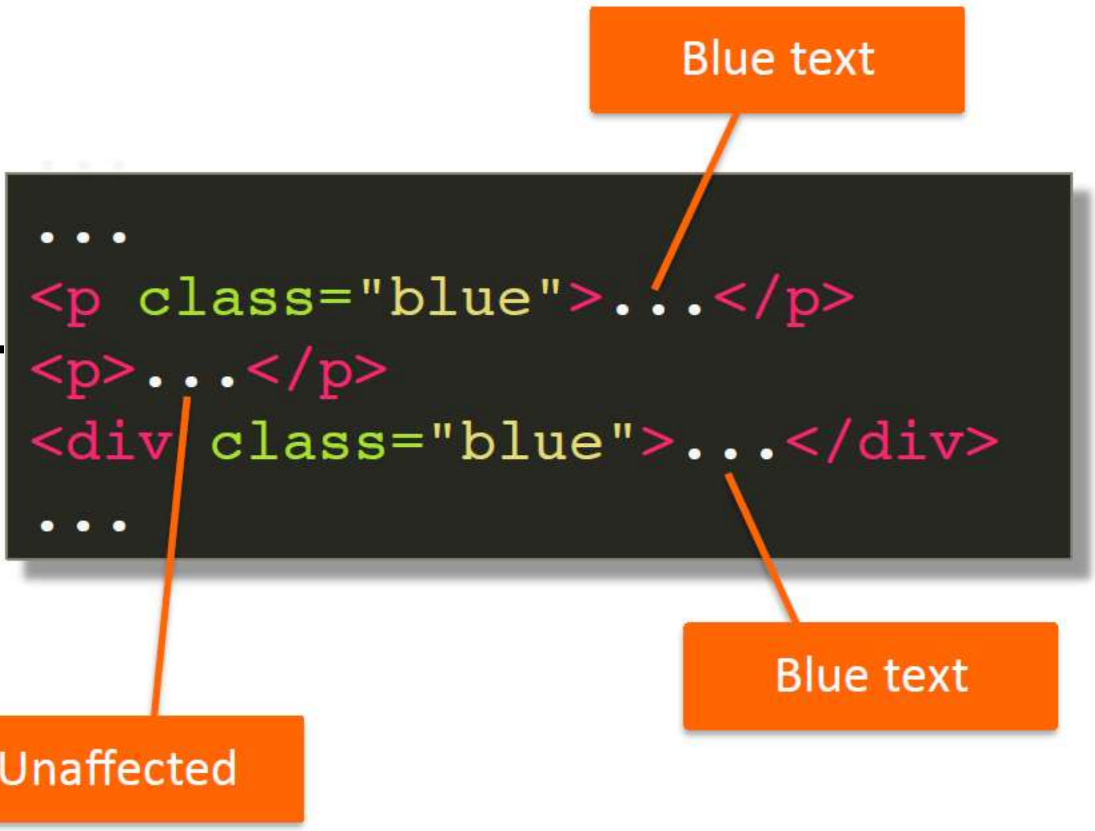
```html
...
<p class="big">...</p>
<div class="big">...</div>
...
```



```
.blue {
  color: blue;
}
```

CSS Rule

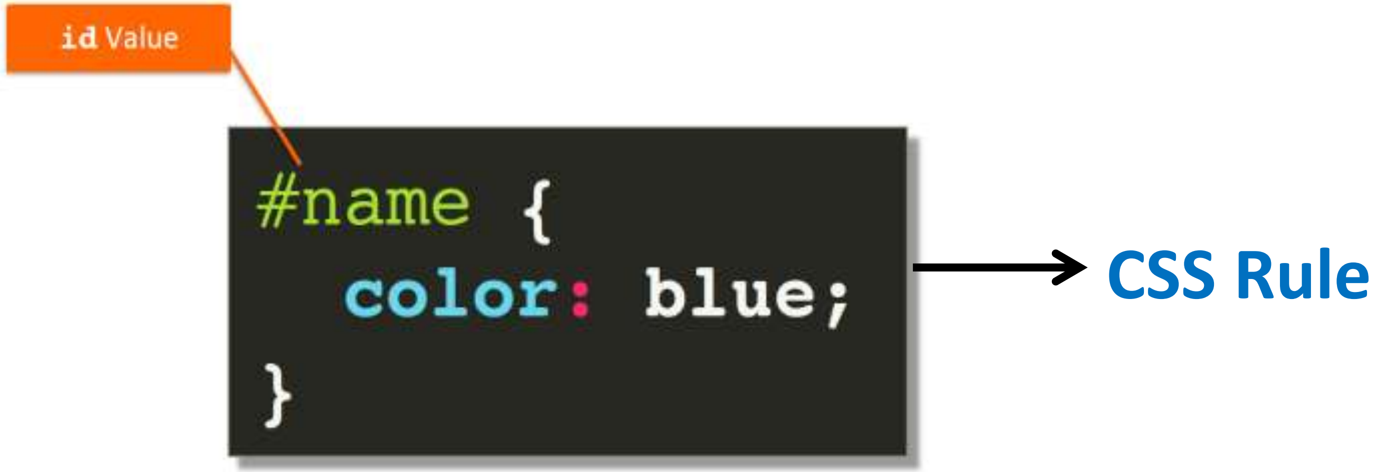
HTML Code



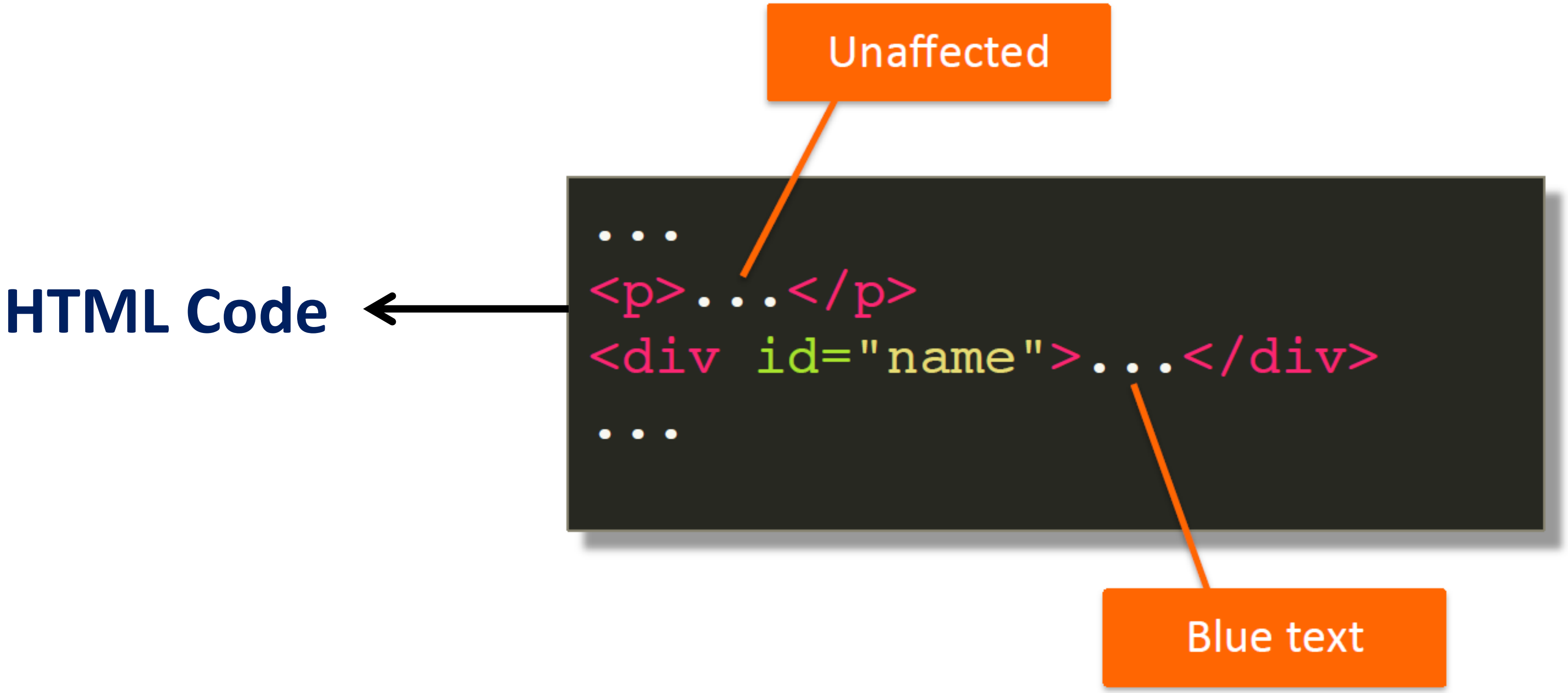
```
...
<p class="blue">...</p>
<p>...</p>
<div class="blue">...</div>
...
```

- Id Selector:
  - Uses the 'id' attribute of an HTML element to select a specific element.
  - IDs should be unique
- Syntax:

```
#idname
{
   Property1: value; property2: value;
}
```



```css
#name {
  color: blue;
}
```



```html
...
<p>...</p>
<div id="name">...</div>
...
```

- Universal Selector (*) :
  - selects **all** the HTML elements on the page.
  - **Syntax:**

```
* {
        property1 : value; property2 : value;
        ......
 }
```

- **Grouping of Selectors:**
  - Separate each selector with a comma

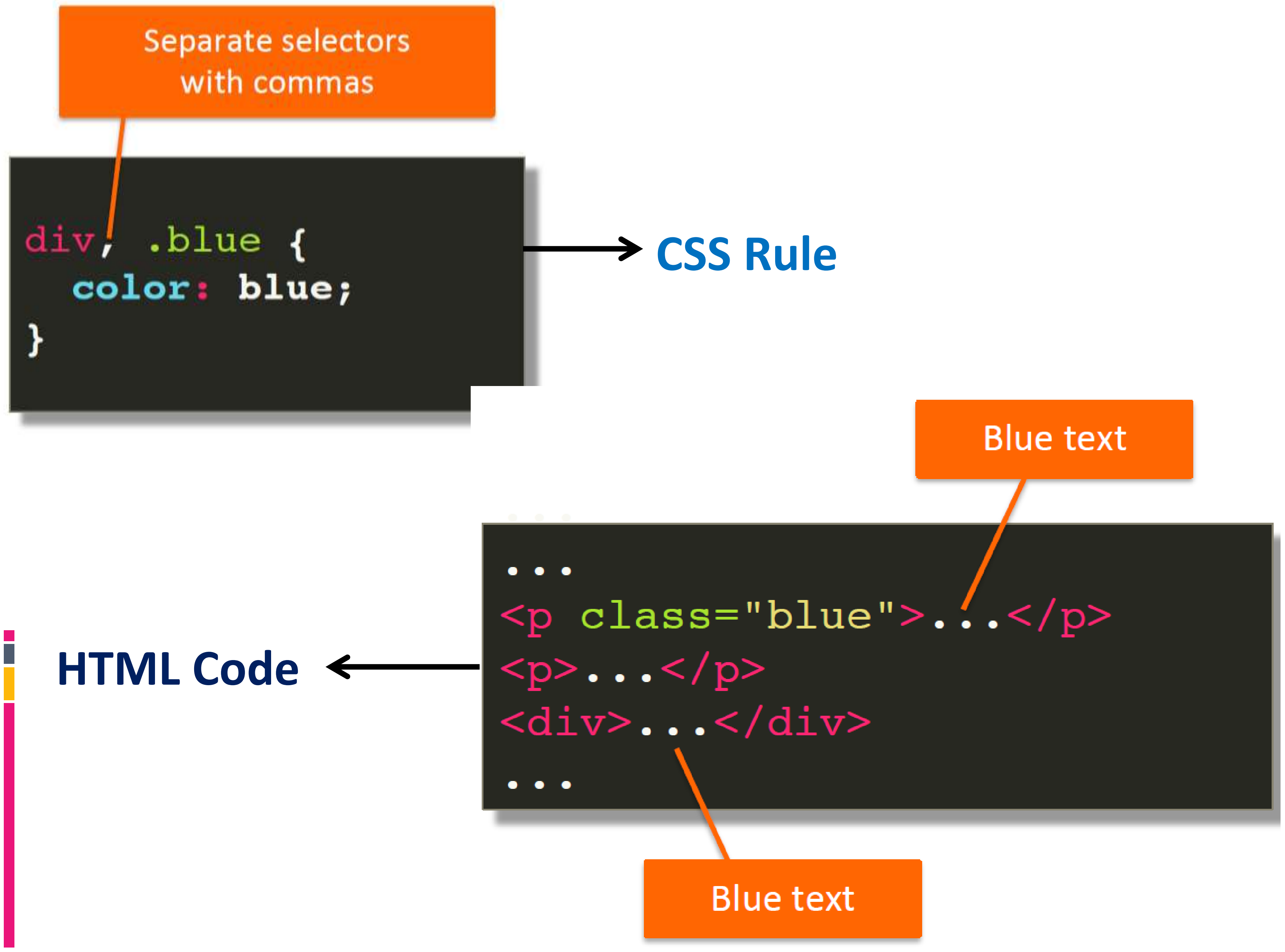
```
Selector 1, Selector 2, .....
{
    Property1: value; property2: value;
}
```

```
h1 { Color: red; }

h2 { Color: red; }
```

Grouping of selectors ←

```
h1, h2  { Color: red; }
```



```css
div, .blue {
  color: blue;
}
```

```html
...
<p class="blue">...</p>
<p>...</p>
<div>...</div>
...
```

Pseudo-class Selector

- A pseudo-class is **used to define a special state of an element**.
- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus

Syntax:

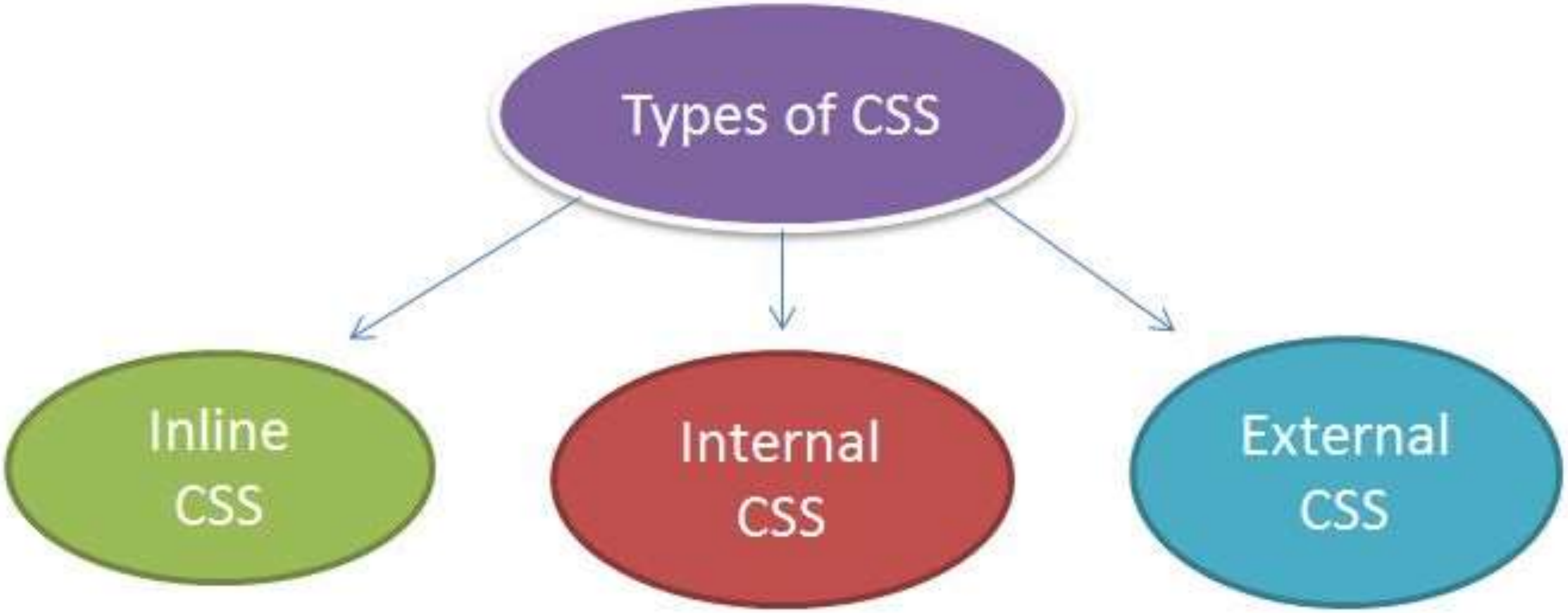
```
Selector : pseudo-class
 {
  property: value;
}
```

Some Psuedo-classes:

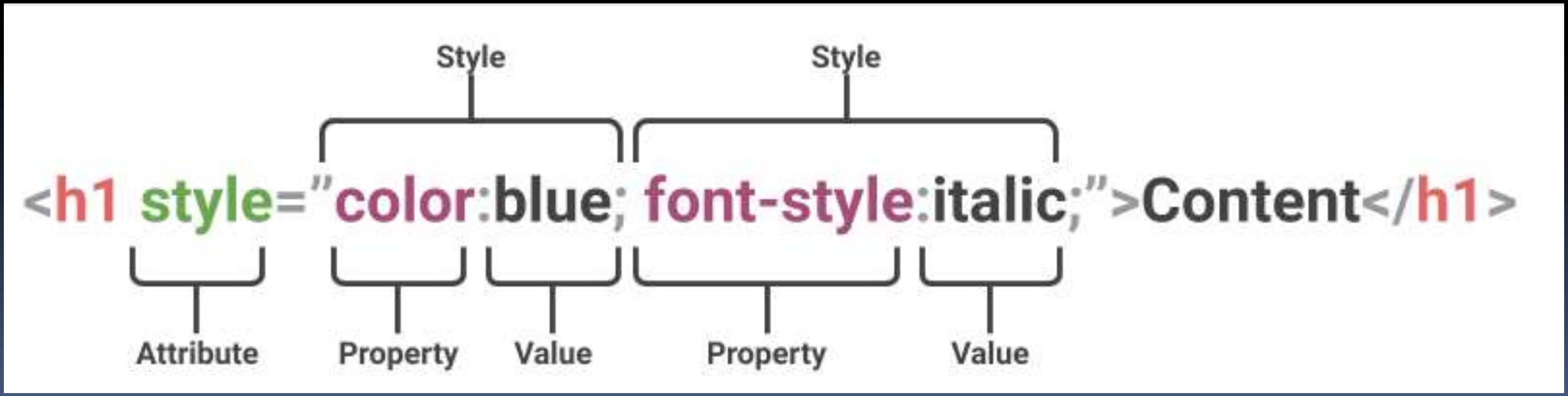
- :link
- :visited
- :active
- :hover
- :focus
- :first-child
- :last-child
- :nth-child(n) – n can be a number/even/odd

TYPES OF CSS

- There are **three ways** of inserting a style sheet to HTML pages:

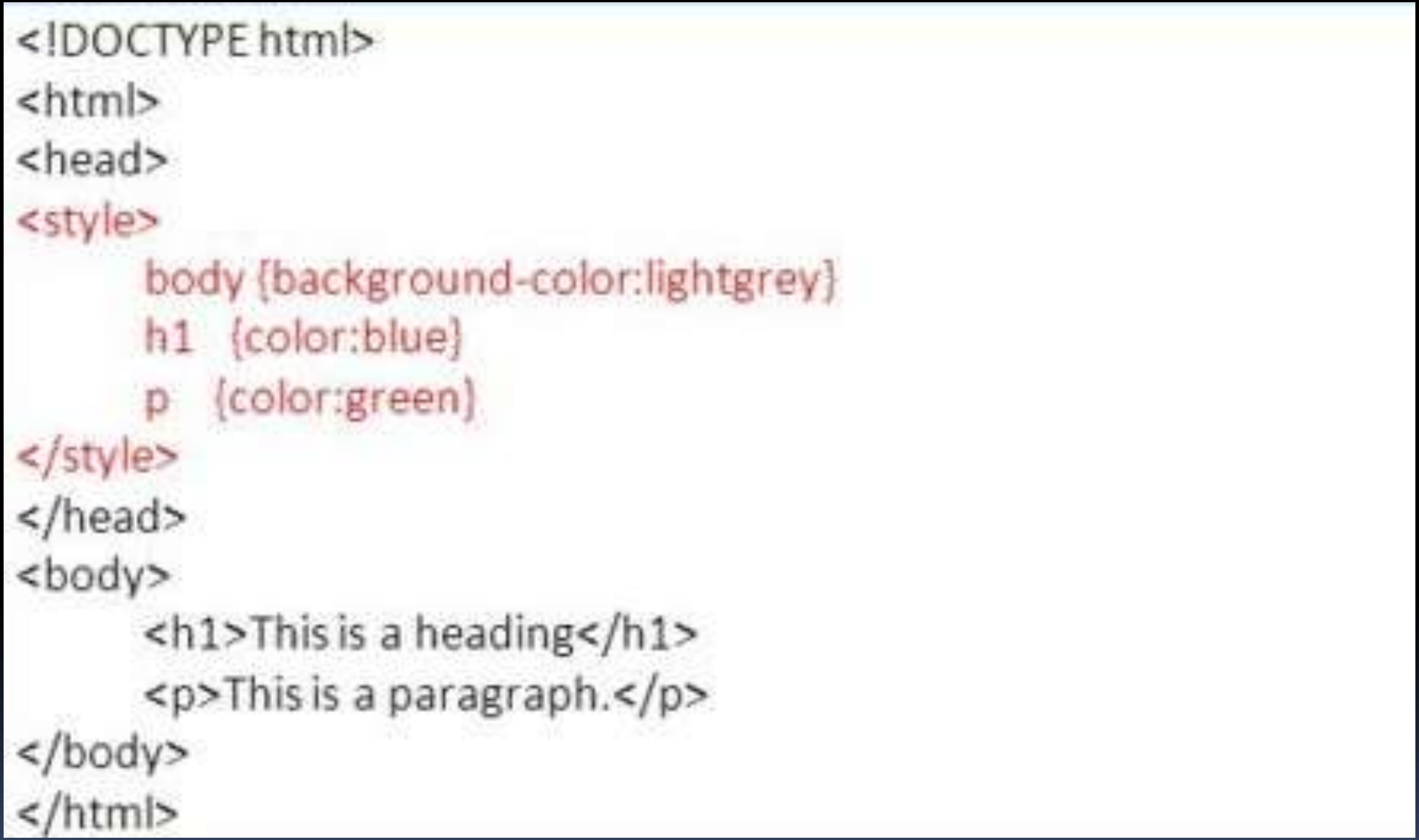


- **Inline CSS:**
  - Placed directly inside a specific HTML element in the code
  - "Style" attribute is used
  - Syntax: <element style=" cssproperty1:value; cssproperty2: value" >
  - Cannot be reused



- Internal/embedded CSS:
  - Used when a single document has a unique style
  - Head section of the same document
  - <style> tag is used

```
<html>
   <head>
      <style type="text/css>
         CSS Rules
      </style>
   </head>
   <body>
   </body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
<style>
        body {background-color:lightgrey}
        h1   {color:blue}
        p    {color:green}
</style>
</head>
<body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
</body>
</html>
```

- **External CSS:**
  - Used when we need to apply particular style to more than one web page
  - CSS rules stored in a separate file saved as .css
  - <link > tag is used to link css file to HTML page

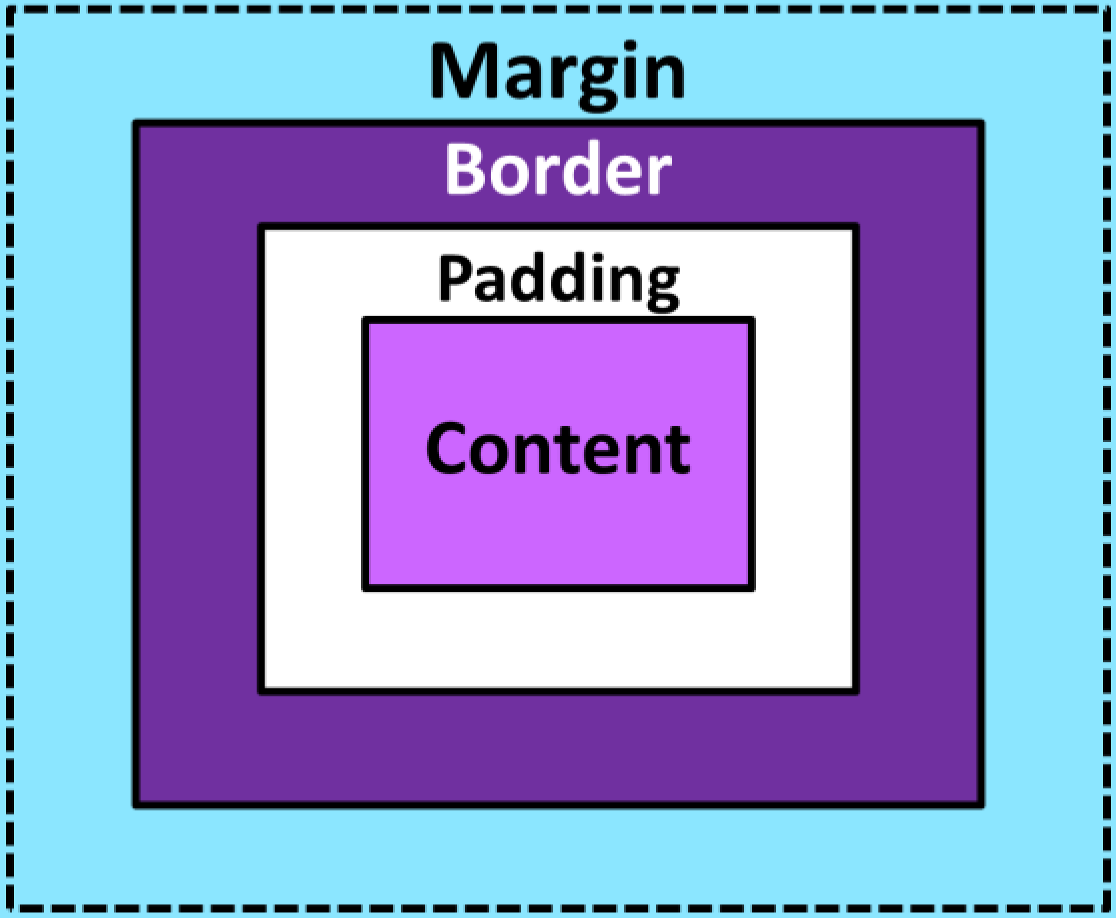
```
<html>
   <head>
     <link  rel="stylesheet"  type="text/css‟  href=‟.css">
   </head>
   <body>
   </body>
</html>
```

- The **<link>** tag defines the relationship between the current document and an external resource.
- It is most often used to link to external style sheets.
- Attributes:
  - href - Specifies the URL of the linked document
  - rel - Specifies the relationship between the current document and the linked document
  - type - Specifies the media type of the linked document

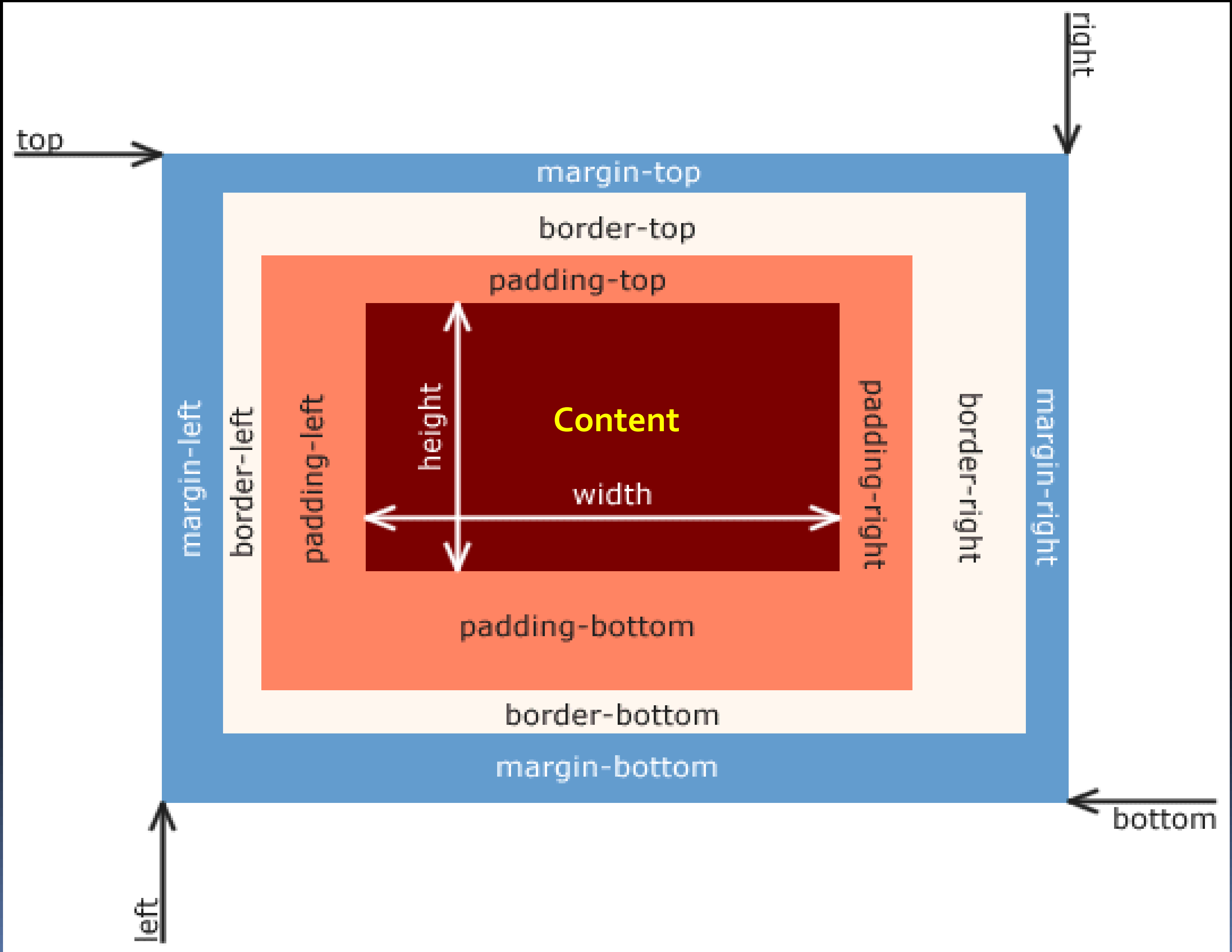
- Advantages of CSS:
  - Reduced Complexity and Repetition :
    - Supports reusability – Same stylesheet can be used in multiple pages
    - Saves the time of developer
  - Reduced Document Size
  - Reduced Clutter:
    - Can change the appearance of several pages by altering style sheet rather than individual page
  - Multiple style sheets can be included in a single page.
  - Improves formatting capability of a HTML page.
  - One style sheet can import and use styles from other style sheets
    - @import cssfilename

CSS Box Model

- Browser's rendering engine represents **each element as a rectangular box** according to the standard CSS basic box model.
- The CSS box model is essentially **a box that wraps around every HTML element.**
- Every box is composed of **four areas**/parts, defined by their respective edges:
  - Content
  - Padding
  - Border **and**
  - Margin


Margin
Border
Padding
Content

Width and Height of the box:



Box Size	CSS Properties
Total Width	width + padding-left + padding-right + border-left + border-right + margin-left + margin-right
Total Height	height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

CSS Dimension Properties

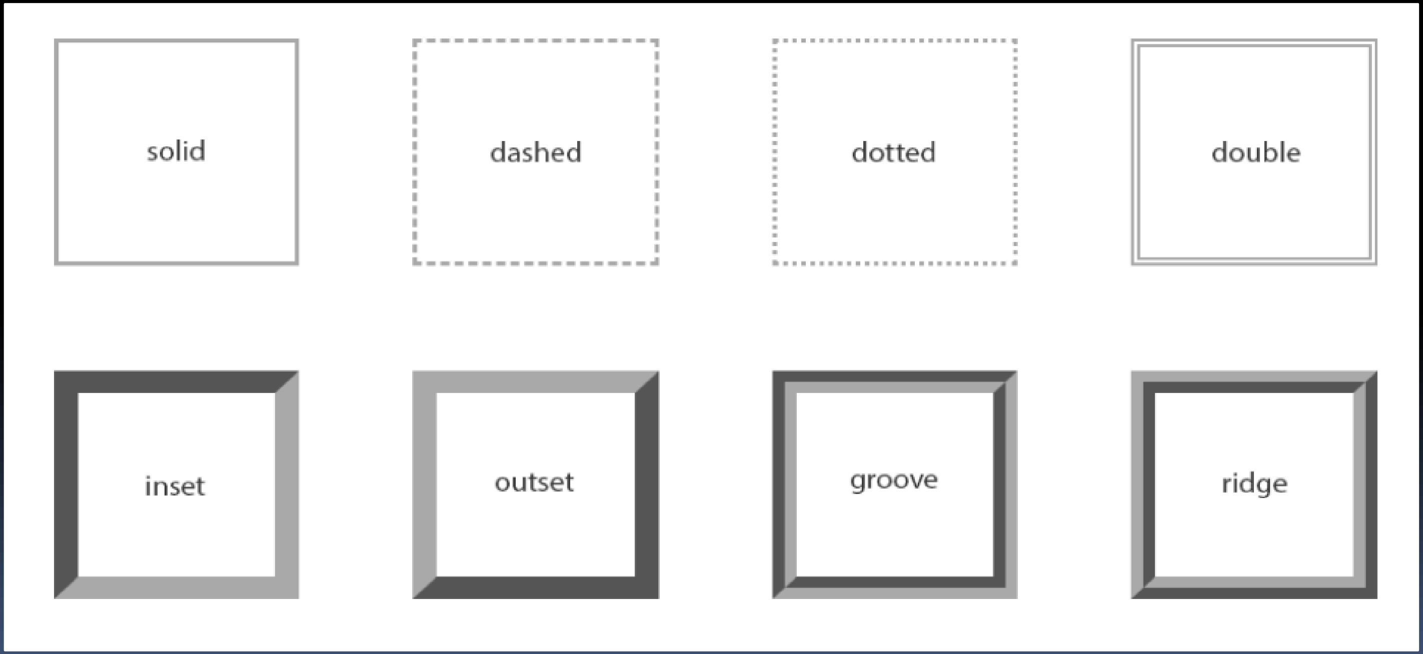
- width
- height
- min-width
- min-height
- max-width
- max-height
  - The CSS ***height*** and ***width*** properties are used to set the height and width of the content area of an element.
  - This width and height does not include paddings, borders, or margins.

CSS Padding Properties

- The CSS padding properties are used to generate space around an element's content, inside of any defined borders.
- **Properties:**
  - ***padding-top***
  - ***padding-right***
  - ***padding-bottom***
  - ***padding-left***
  - The ***padding*** shorthand property:
    - padding: 25px 50px 75px 100px;

CSS Border Properties

- The CSS border properties allow you to define the border area of an element's box.
  - ***border-style***:



- ***border-width:***
  - Specifies the width of the border area.
  - Shorthand property for setting the thickness of all the four sides of an element's border at the same time.
- ***border-color*** :
  - specifies the color of the border area.
  - This is also a shorthand property for setting the color of all the four sides of an element's border.

- ***border*** property is a shorthand property for the following individual border properties:
  - border-width
  - border-style (required)
  - border-color

  Example: border: 5px solid red

CSS Margin Properties

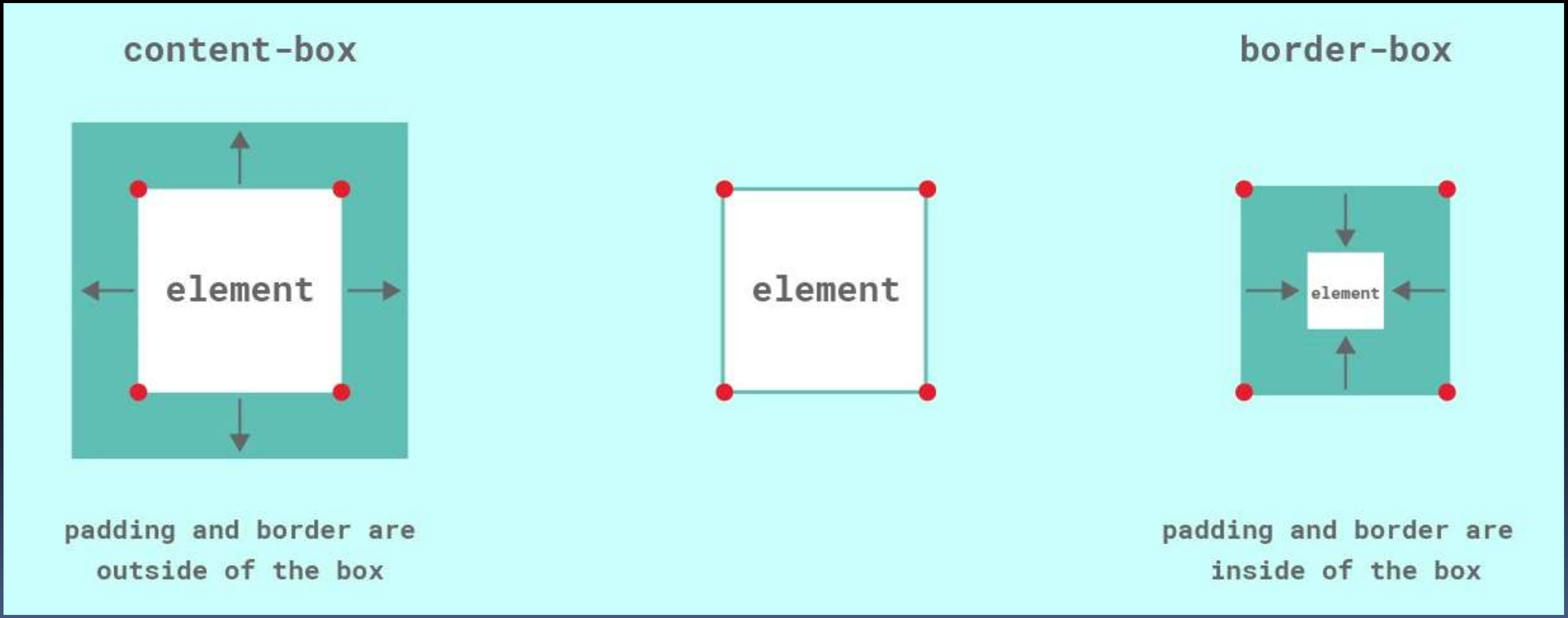
- The CSS margin properties allow you to set the spacing around the border of an element's box.
- margin is always transparent.
- Properties:
  - ***margin-top***
  - ***margin-right***
  - ***margin-bottom***
  - ***margin-left***

- The ***margin*** property is a shorthand property for the following individual margin properties.

Box-sizing property

- The box-sizing property defines **how the width and height of an element are calculated**: should they include padding and borders, or not.
- Syntax:
  - **box-sizing: content-box | border-box**

Value	Description
content-box	Default. The width and height properties includes only the content. Border and padding are not included
border-box	The width and height properties includes content, padding and border


content-box
border-box
element
element
element
padding and border are outside of the box
padding and border are inside of the box

RESPONSIVE WEB DESIGN

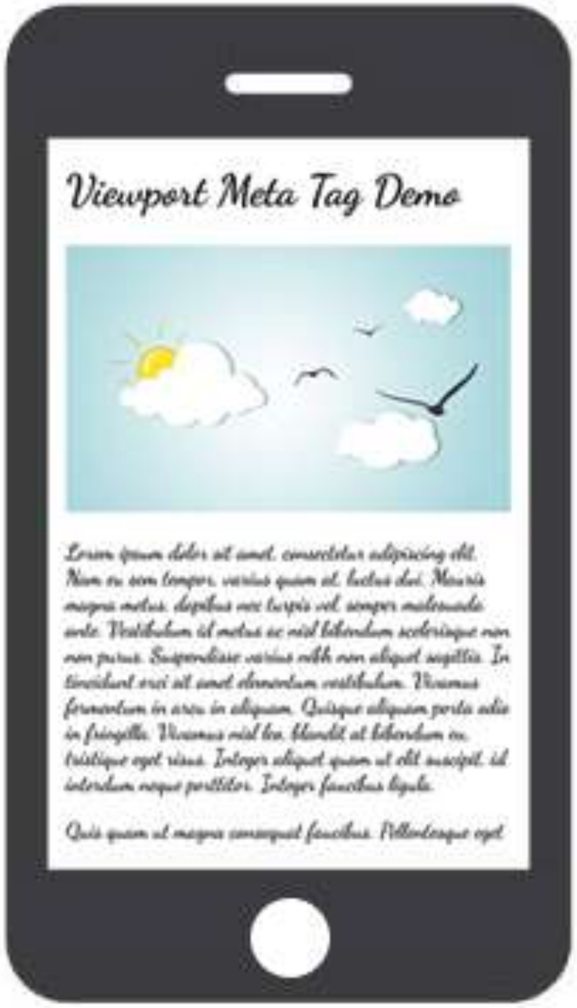
- It is about creating web sites which **automatically adjust themselves to look good on all devices**, from small phones to large desktops.
- Responsive Web Design is about **using HTML and CSS** to **automatically resize, hide, shrink, or enlarge, a website**, to make it look good on all devices (desktops, tablets, and phones).
- A responsive web design will **automatically adjusts for different screen sizes and viewports.**

VIEWPORT

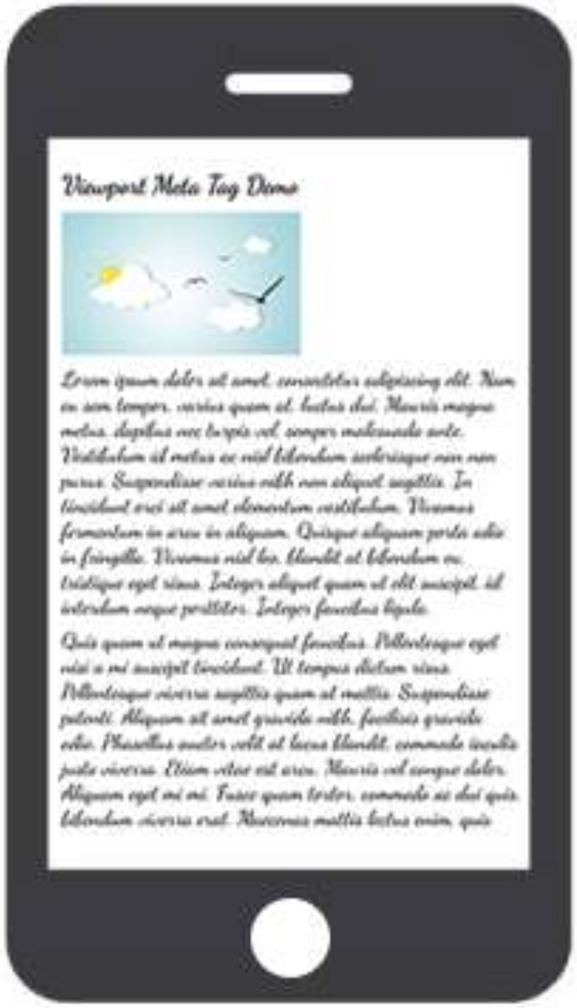
- That is the **part of the document you are viewing** on a browser window.
- The **viewport** is the **"user's visible area of a web page".**
- The browser's viewport is the **area of the window in which web content can be seen.**
- often **not** the same size as the rendered page.
- The viewport **varies with the device**, and will be smaller on a mobile phone than on a computer screen.

Setting The Viewport:

- Using HTML5 **<meta>** tag.
- **<meta name="viewport" content="width=device-width, initial-scale=1.0">**
    - *width=device-width* sets the width of the page to follow the screen-width of the device.
    - *initial-scale=1.0* sets the initial zoom level when the page is first loaded by the browser.



With Viewport Meta Tag



Without Viewport Meta Tag

Responsive Images:

- Responsive images are images that scale nicely to fit any browser size.
- If the **CSS width** & **height** properties were set to **100%**, the image will be responsive and scale up and down.
- **Drawback**:
  - the image can be scaled up to be larger than its original size

- Using the max-width & max-height Properties
  - If the max-width & **max-height** property is set to **100%**, the image will scale down if it has to, but never scale up to be larger than its original size.

Responsive Text Size:

- The text size can be set with a "**vw**" unit, which means the "viewport width".
- text size will follow the size of the browser window

  font-size: 10vw;

Media Queries:

- Media query is a CSS technique **introduced in CSS3**
- In addition to resize text and images, it is also **common to use media queries in responsive web pages.**
- With media queries you **can define completely different styles for different browser sizes**.
- Media queries can be used to check many things, such as:
  - width and height of the viewport
  - width and height of the device
  - orientation (is the tablet/phone in landscape or portrait mode)
  - resolution
- It uses the ***@media*** rule to include a block of CSS properties only if a certain condition is true.

@media rule: included in <style> - </style>

- used in media queries to apply different styles for different media types/devices.

@media not|only *mediatype* and (*mediafeature* and|or|not *mediafeature*)

{

CSS-Code;

}

- **not:** inverts the meaning of an entire media query.
- **only:** prevents older browsers that do not support media queries with media features from applying the specified styles. **It has no effect on modern browsers.**
- **and:** combines a media feature with a media type or other media features.

- **Media Types:**

Value	Description
all	Default. Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

- **Some Media Features:**
  - *height* : The viewport height.
  - *width* : The viewport width.
  - *max-height:* The maximum height of the display area, such as a browser window.
  - *max-width:* The maximum width of the display area, such as a browser window.
  - *min-height:* The minimum height of the display area, such as a browser window.
  - *min-width:* The minimum width of the display area, such as a browser window.
  - *orientation:* The orientation of the viewport (landscape or portrait mode)

CSS TRANSITIONS

- CSS3 transitions **allow you to change CSS property values smoothly, over a given duration**.
- They **provide a way to control animation speed** when changing CSS properties.
- let you decide
  - which properties to animate/change
  - when the transition will start
  - how long the transition will last and
  - how the transition will run.

Defining transitions:

- CSS Transitions are controlled using the shorthand ***transition*** property.
- To create a transition effect, you must specify two things:
  - **the CSS property you want to add an effect to**
  - **the duration of the effect**

CSS Transition Properties:

- transition: A shorthand property for setting *the transition-property, transition-duration, transition-timing-function, and transition-delay.*
- transition-property: **Specifies the name of the CSS property the transition effect is for.**
- transition-duration: **Specifies how many seconds transition effect takes to complete**
- transition-timing-function
- transition-delay: **Specifies a delay (in seconds) for the transition effect**

transition-timing-function

- Specifies the speed curve of the transition effect.
  - *ease* - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
  - *linear* - specifies a transition effect with the same speed from start to end
  - *ease-in* - specifies a transition effect with a slow start
  - *ease-out -* specifies a transition effect with a slow end
  - *ease-in-out* - specifies a transition effect with a slow start and end

CSS Animations

- CSS3 allows **animation of HTML elements** without using JavaScript or Flash!.
- CSS animations make **it possible to animate transitions from one CSS style configuration to another**.
- "**An animation lets an element gradually change from one set of CSS styles to another**."
- During the animation, you **can change the set of CSS styles many times**.

- Animations consist of two components:
  - **A style describing the CSS animation** and
  - **A set of keyframes** that indicate the start and end states of the animation’s style, as well as possible intermediate waypoints.
- Steps in creating Animations:
  - Configuring the animation using animation properties.
  - Defining the animation sequence using keyframes.

CSS Animation Properties

To create a CSS animation sequence, style the element you want to animate with the animation property or its sub-properties.

- **animation-name**: Specifies the name of the @keyframes animation
- **animation-duration**: Specifies how long time an animation should take to complete one cycle
- **animation-delay**: Specifies a delay for the start of an animation
- **animation-play-state**: specifies whether the animation is running or paused.
- **animation-iteration-count:**
  - Specifies the number of times an animation should be played
  - Values: number, infinite

- **animation-direction**:
  - Specifies whether an animation should be played forwards, backwards or in alternate cycles.
  - **Values:**
    - **normal** - The animation is played as normal (forwards). This is default
    - **reverse** - The animation is played in reverse direction (backwards)
    - **alternate** - The animation is played forwards first, then backwards
    - **alternate-reverse** - The animation is played backwards first, then forwards

- **animation-timing-function**:
  - Specifies the speed curve of the animation.
  - **Values:**
    - **ease** - Specifies an animation with a slow start, then fast, then end slowly (this is default)
    - **linear** - Specifies an animation with the same speed from start to end
    - **ease-in** - Specifies an animation with a slow start
    - **ease-out** - Specifies an animation with a slow end
    - **ease-in-out** - Specifies an animation with a slow start and end

- **animation-fill-mode**:
  - Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both).
  - **Values:**
    - **none** - Default value. Animation will not apply any styles to the element before or after it is executing.
    - **forwards** - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count).
    - **backwards** - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period.
    - **both** - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions.

- **animation**: Shorthand property
  - *Syntax:*
  - **animation: name duration timing-function delay iteration-count direction fill-mode play-state;**

Keyframes:

- To use CSS animation, you must first **specify some keyframes for the animation**.
- Keyframes **hold what styles the element will have at certain times**.
- Each keyframe **describes how the animated element should render at a given time during the animation sequence.**
- Keyframes are **paired with the animation property** to set the duration, timing function, delay and direction.
- keyframes use a **<percentage>** to indicate the time during the animation sequence at which they take place.
- 0% indicates the first moment of the animation sequence, while 100% indicates the final state of the animation.

The @keyframes rule

- controls the intermediate steps in a CSS animation sequence by defining styles for keyframes along the animation sequence.
- **To use keyframes, create a @keyframes rule** with a name that is then used by the animation-name property to match an animation to its keyframe declaration.

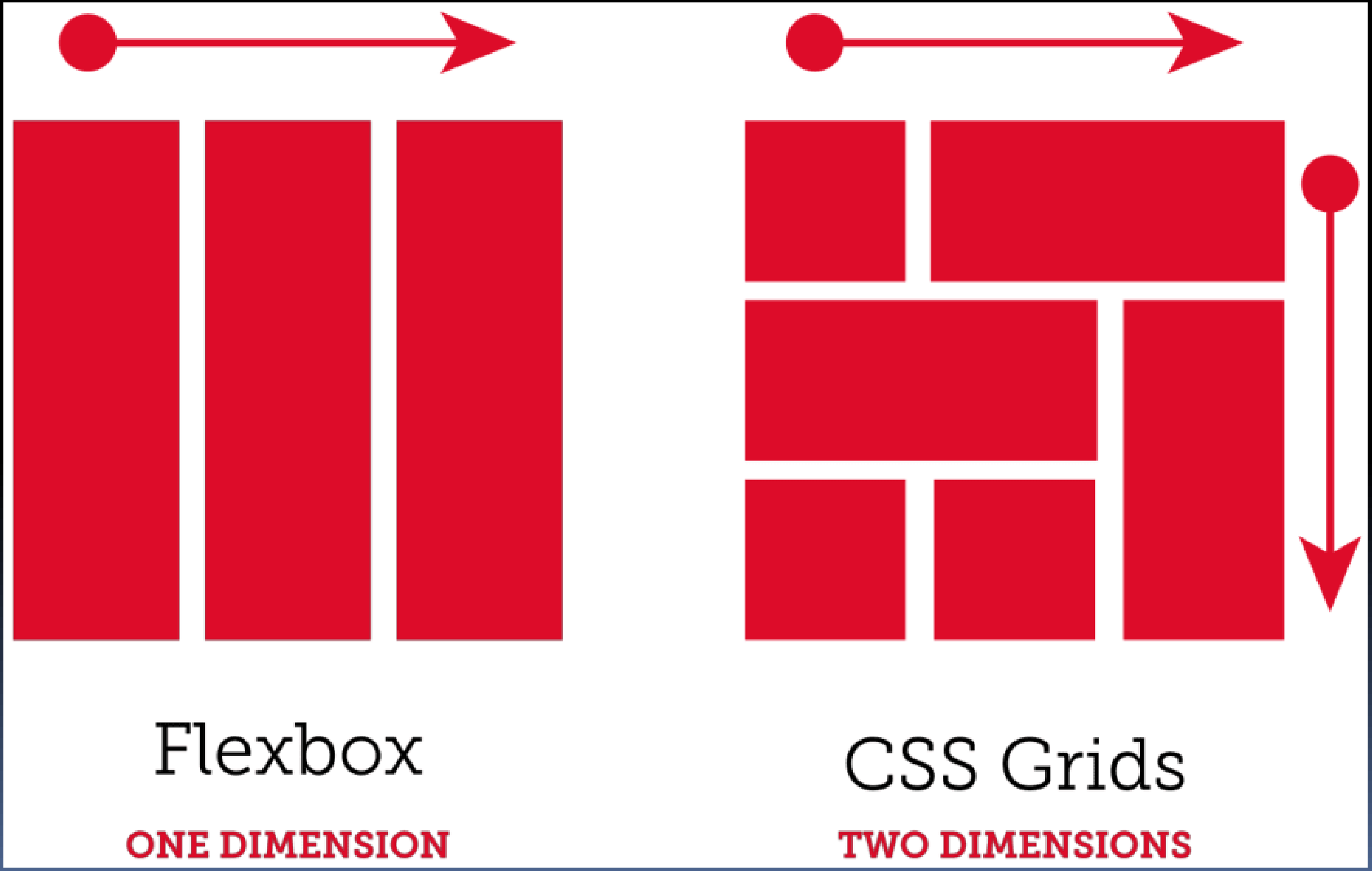
- Syntax:
  - **@keyframes** ***animationname*** *{**keyframes-selector** {css-styles;}}*
  - Each @keyframes rule contains a style list of keyframe selectors, which specify percentages along the animation when the keyframe occurs, and a block containing the styles for that keyframe.

Value	Description
animationname	Required. Defines the name of the animation.
keyframes-selector	Required. Percentage of the animation duration. **values:** 0-100% from (same as 0%) to (same as 100%) You can have many keyframes-selectors in one animation.
css-styles	Required. One or more legal CSS style properties

CSS LAYOUT TECHNIQUES

- allow us to **take elements** contained in a web page and **control where they're positioned** relative to the following factors:
  - their default position in normal layout flow,
  - the other elements around them,
  - their parent container, and
  - the main viewport/window.

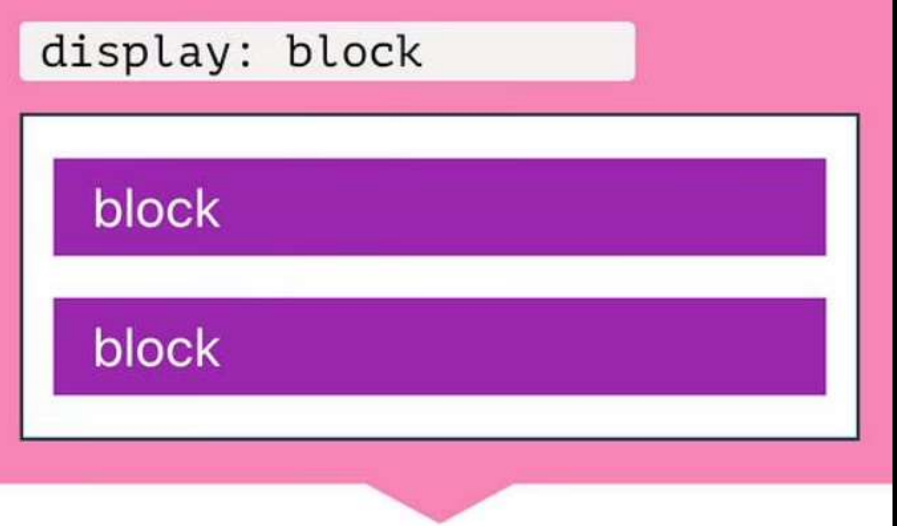
- Two such page layout techniques are:
  - **Flexbox**
  - **Grid**



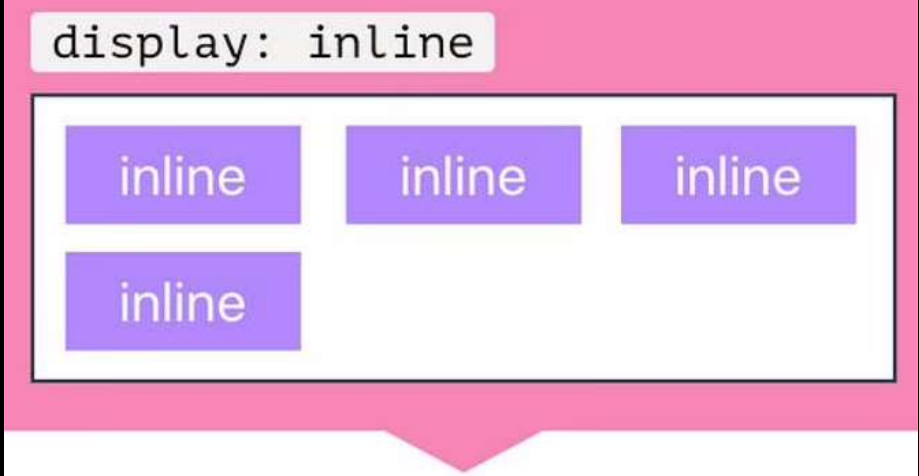
CSS ‘display’ property:

- most important CSS property for controlling layout.
- specifies if/how an element is displayed.
- CSS Syntax:
  - display: *value*;
- Property Values:

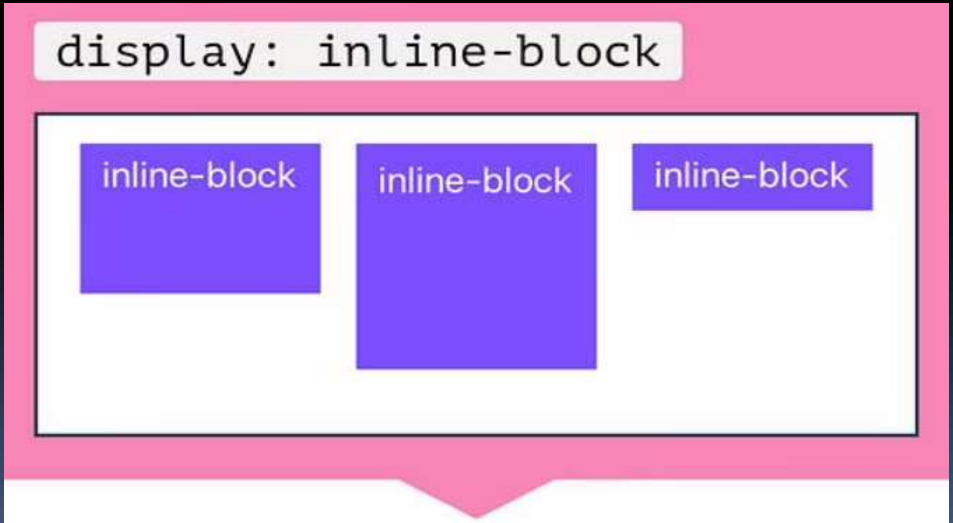
  inline | block | inline-block | flex | grid | inline-grid | inline-flex



Displays an element as a block element. It starts on a new line and takes up the whole width.



Displays an element as an inline element. Any height and width properties will have no effect

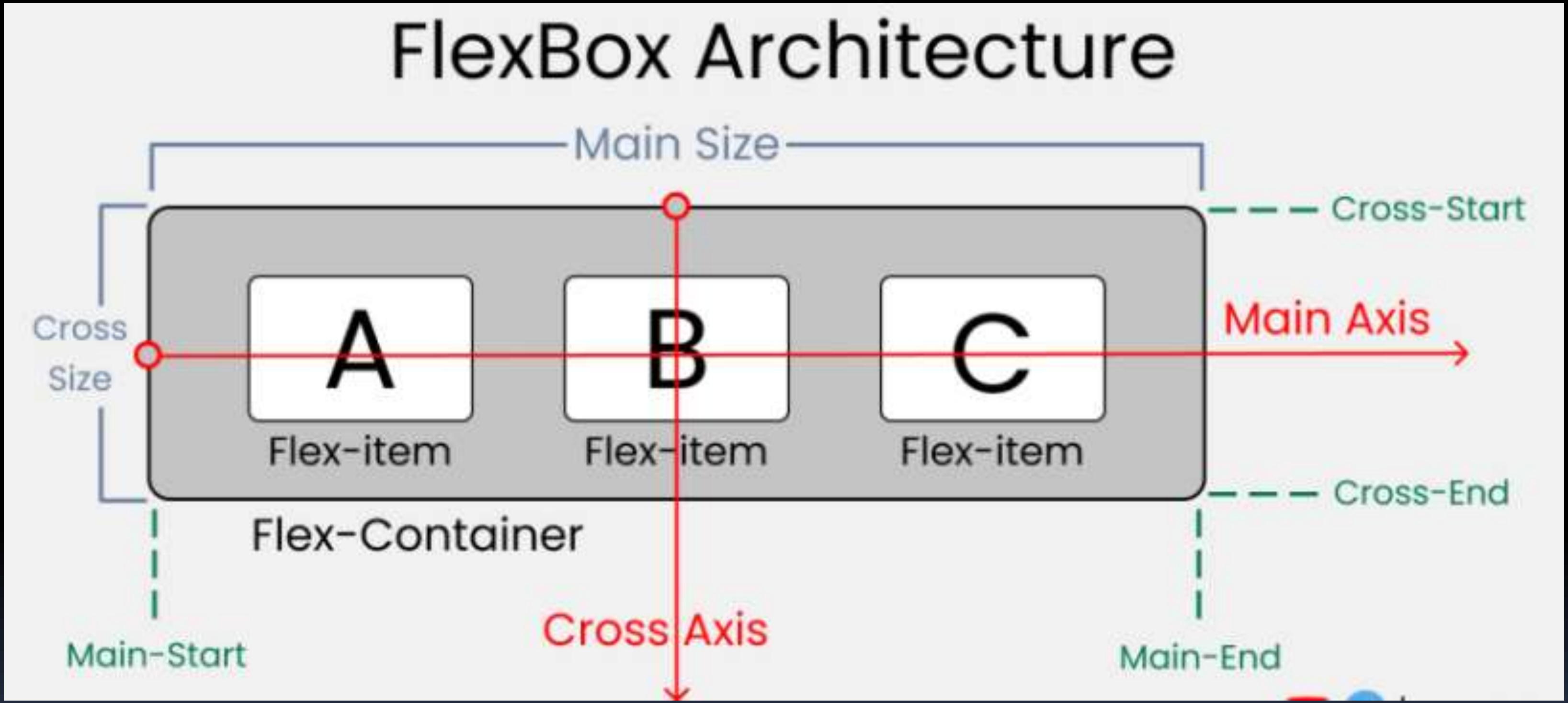


Displays an element as an inline-level block container. You CAN set height and width values.

FLEXBOX LAYOUT

- **FLEX**ible **BOX** layout
- It is a CSS 3 web layout model
- It is a **one-dimensional** layout model.
- Used to arrange elements in a **row or a column** at a time.
- allows responsive elements within a container to be automatically arranged depending upon screen size (or device)

- Two key terminologies in Flexbox are the **main axis** and the **cross axis**.



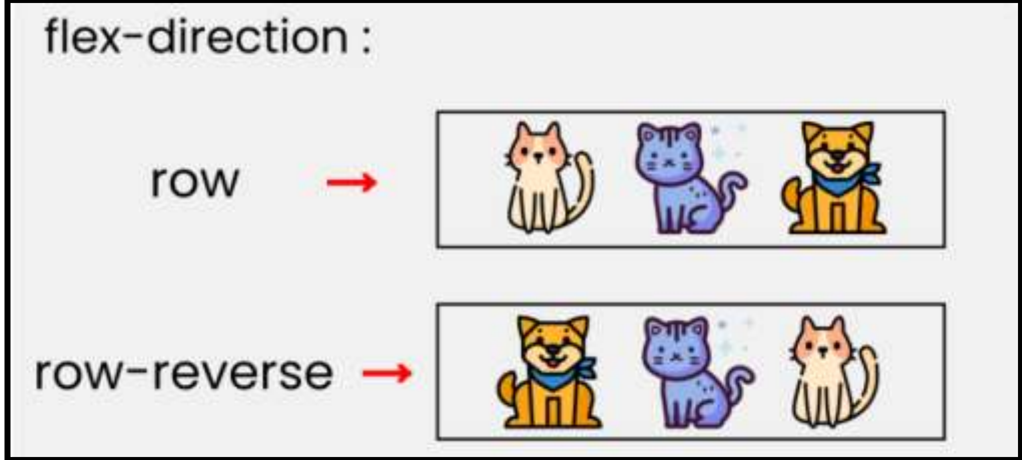
Using Flexbox:

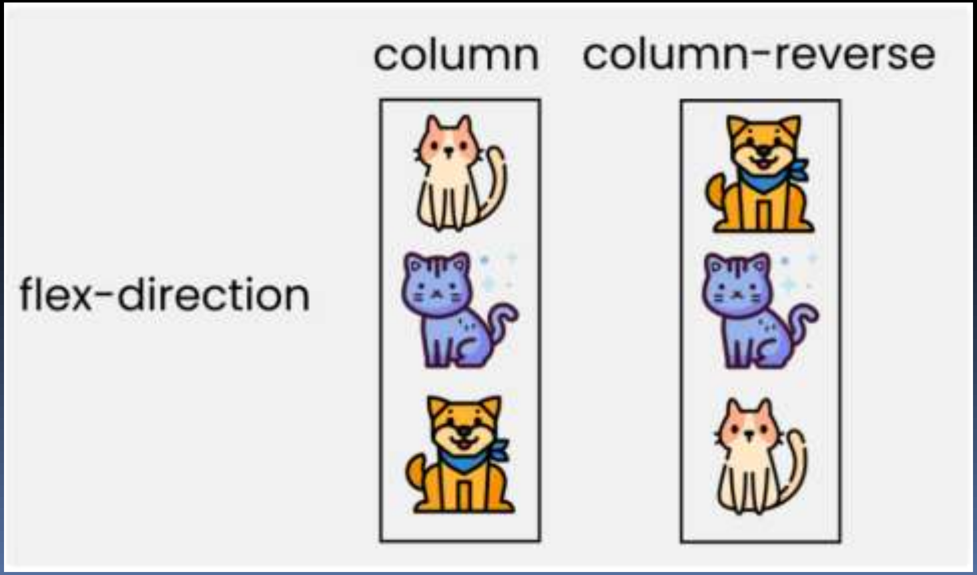
- apply ***display: flex*** to the parent element (container) of the elements you want to layout.
- all its direct children then become ***flex items***.
- The flex container properties are:
  - flex-direction
  - flex-wrap
  - flex-flow
  - justify-content
  - align-items
  - align-content

The flex-direction Property

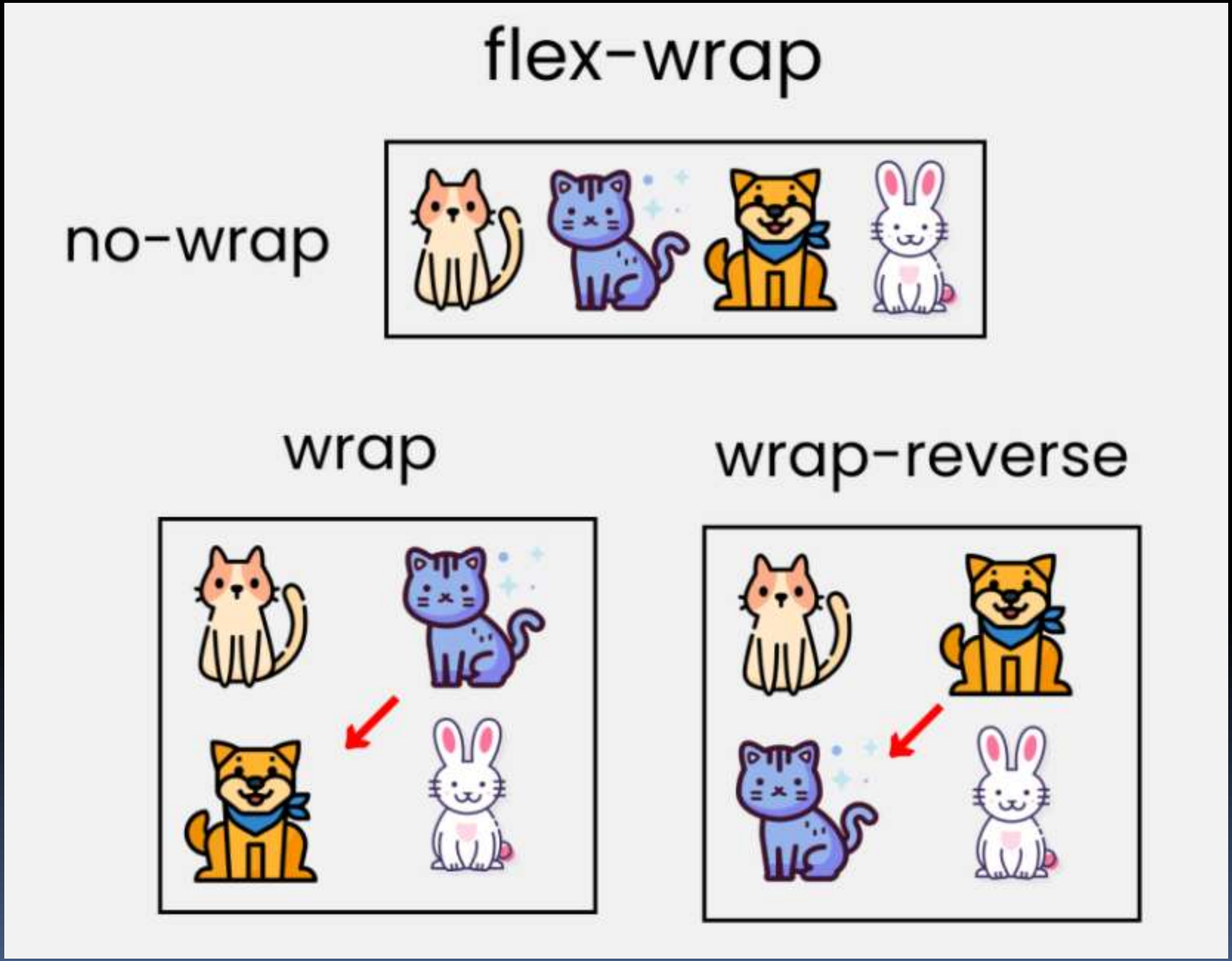
Syntax:

- ***flex-direction:*** **row | row-reverse | column | column-reverse;**





The flex-wrap Property:

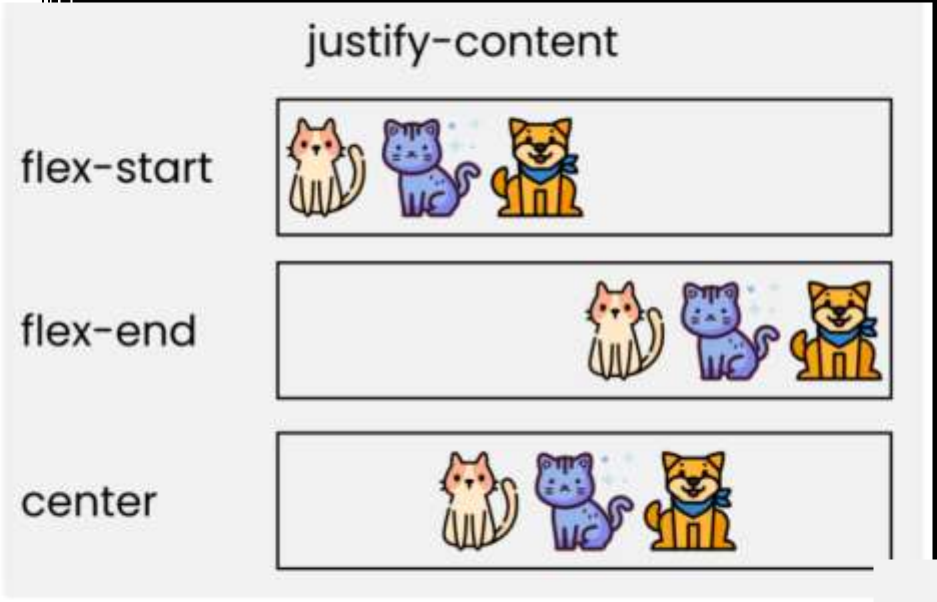


The flex-flow Property:

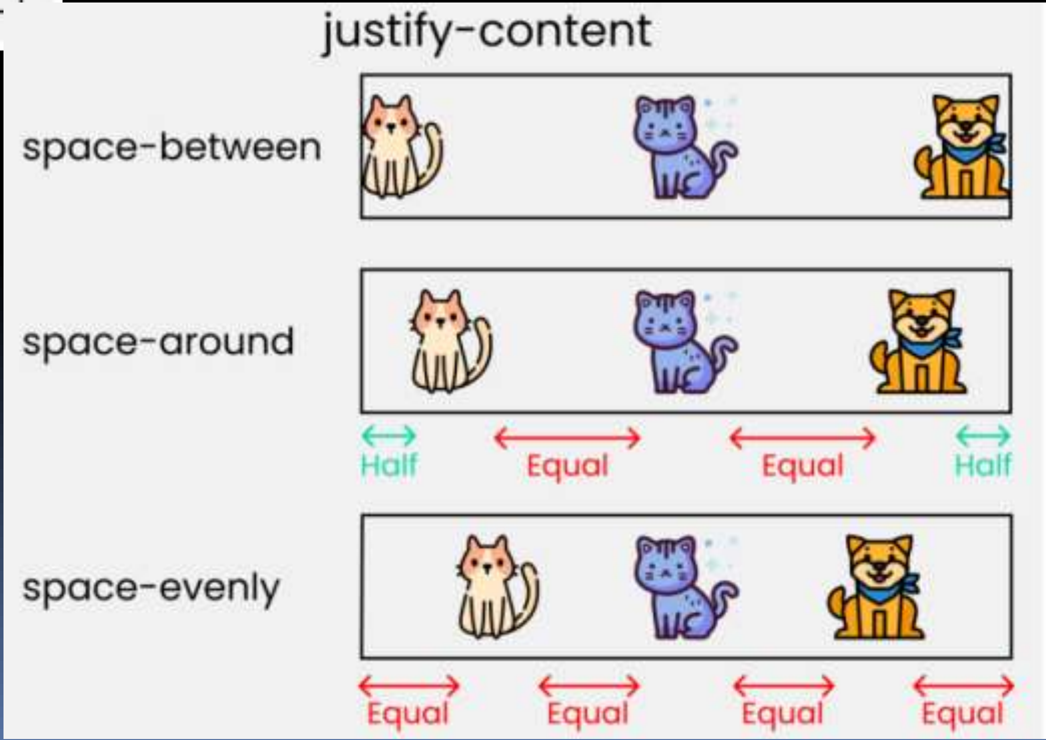
- This property is a shorthand for the following CSS properties:
  - flex-direction
  - flex-wrap

  - **Example:**
    - flex-flow: row wrap;

The justify-content Property:

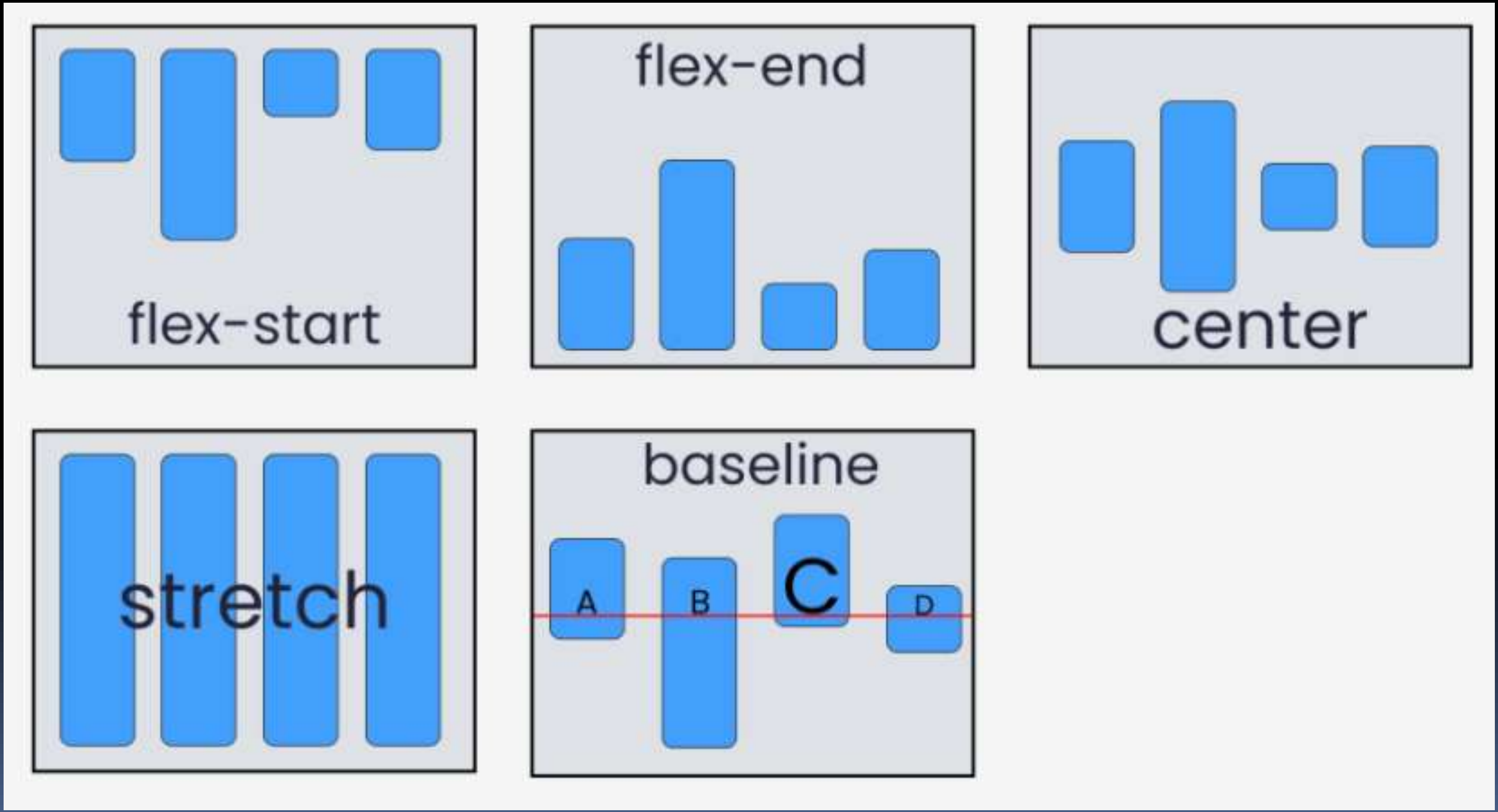


Aligns Items Horizontally



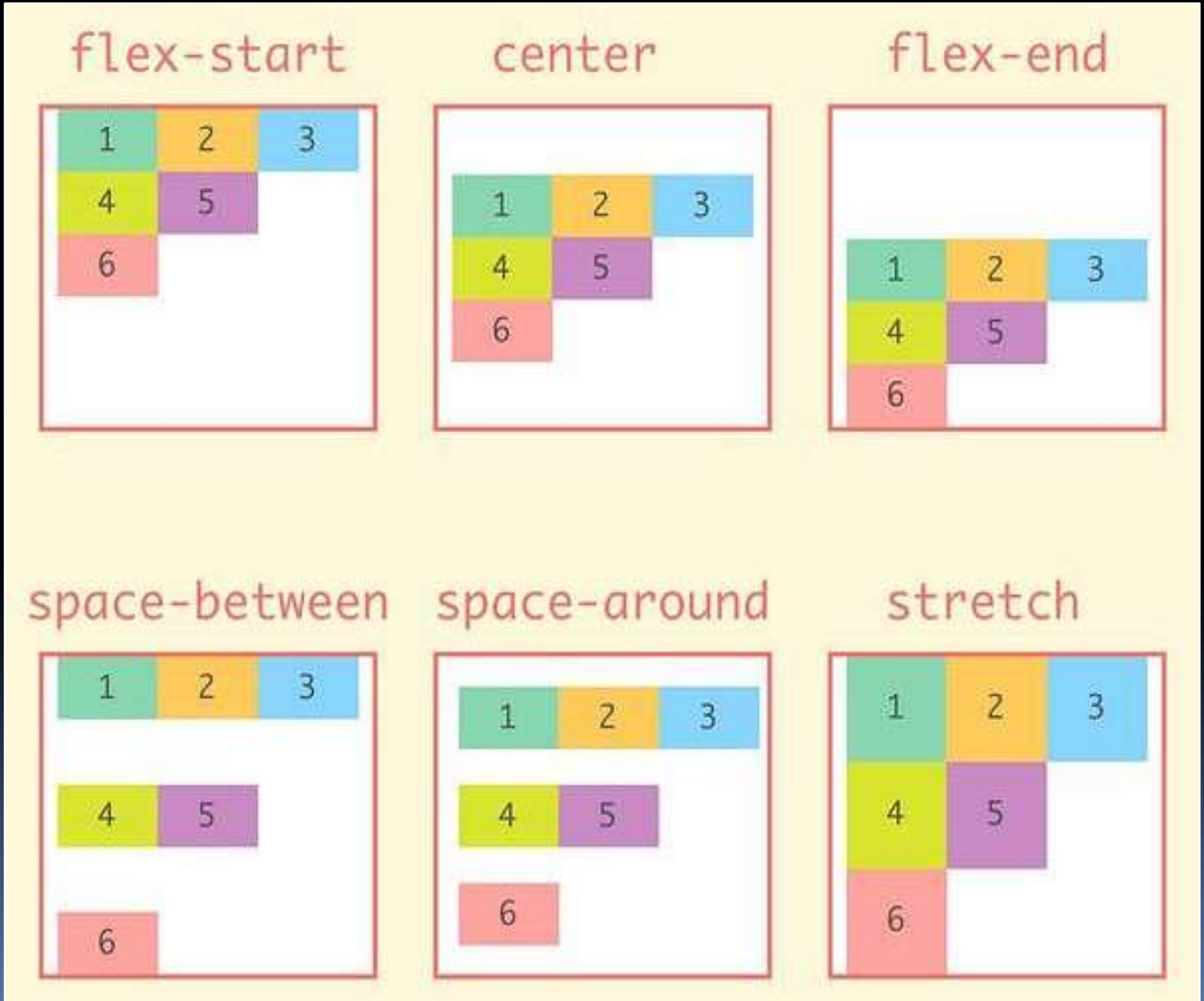
The align-items Property:

- distributes Flex-items along the **Cross Axis** - **Vertically**



The align-content Property:

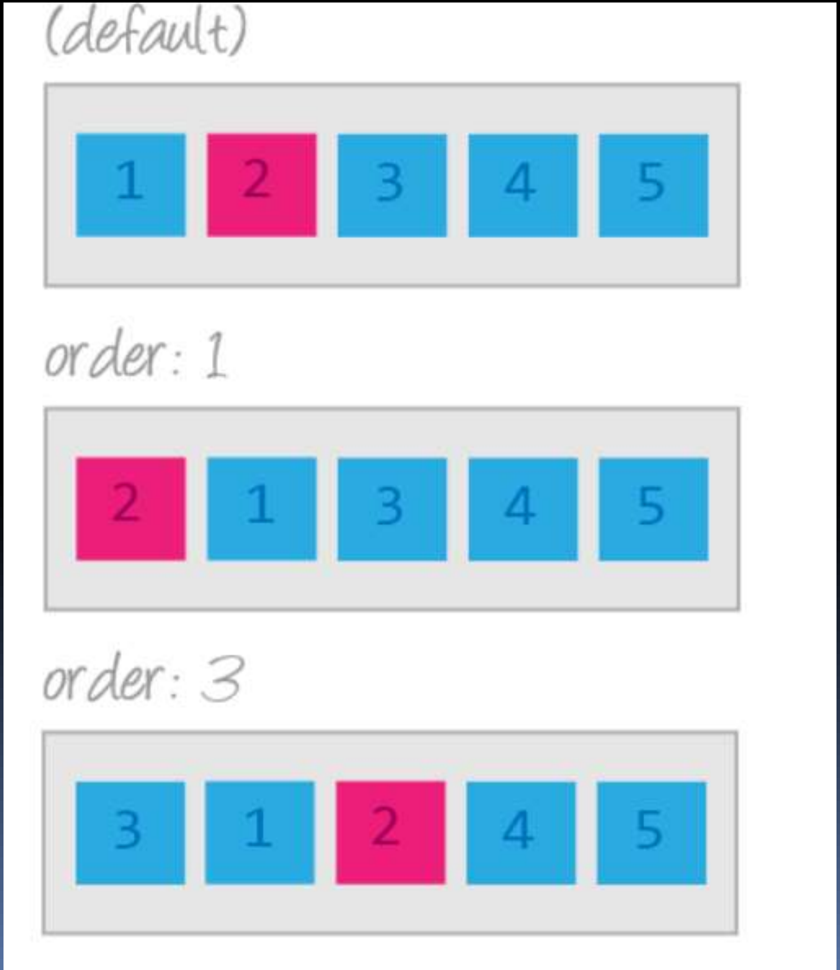
- It is similar to align-items, **but instead of aligning flex items, it aligns flex lines.**



- **The flex item properties are:**
  - order
  - flex-grow
  - flex-shrink
  - flex-basis
  - flex
  - align-self

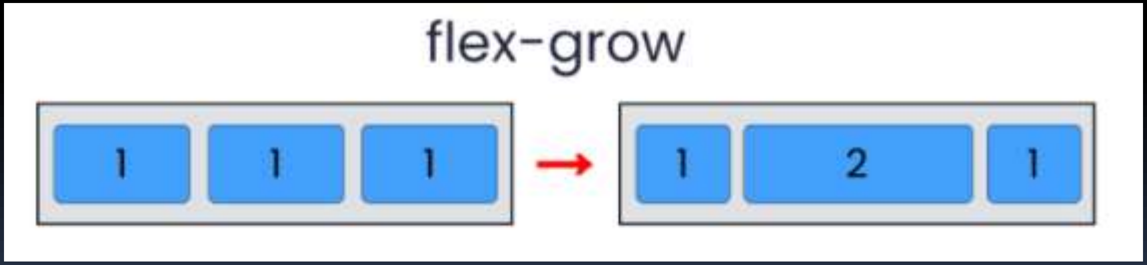
The order Property:

- Specifies the order of the flex items inside the same container.
- The value must be an integer, default value is 0.



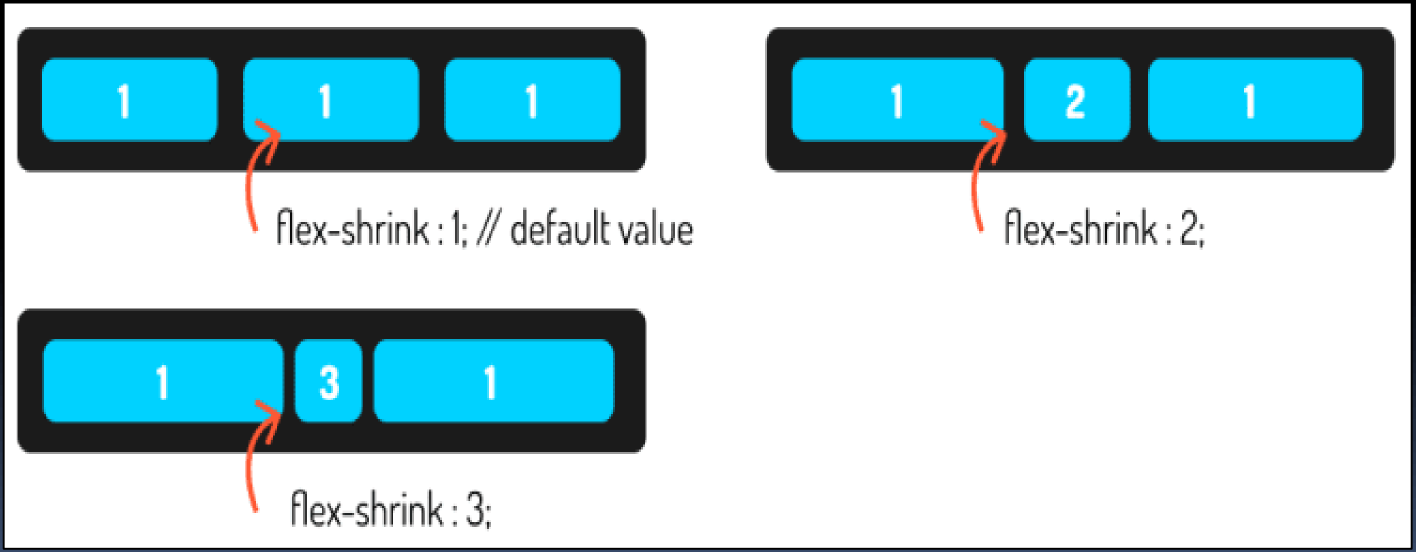
The flex-grow Property :

- Specifies how **much a flex item will grow relative to the rest of the flex items** inside the same container.
- The value must be a number, default value is 1.



The Flex-shrink Property :

- Specifies **how much a flex item will shrink relative to the rest of the flex items** inside the same container.
- The value must be a number, default value is 1.



The Flex-basis Property

- Specifies the initial length of a flex item.
- takes the same values as the width property

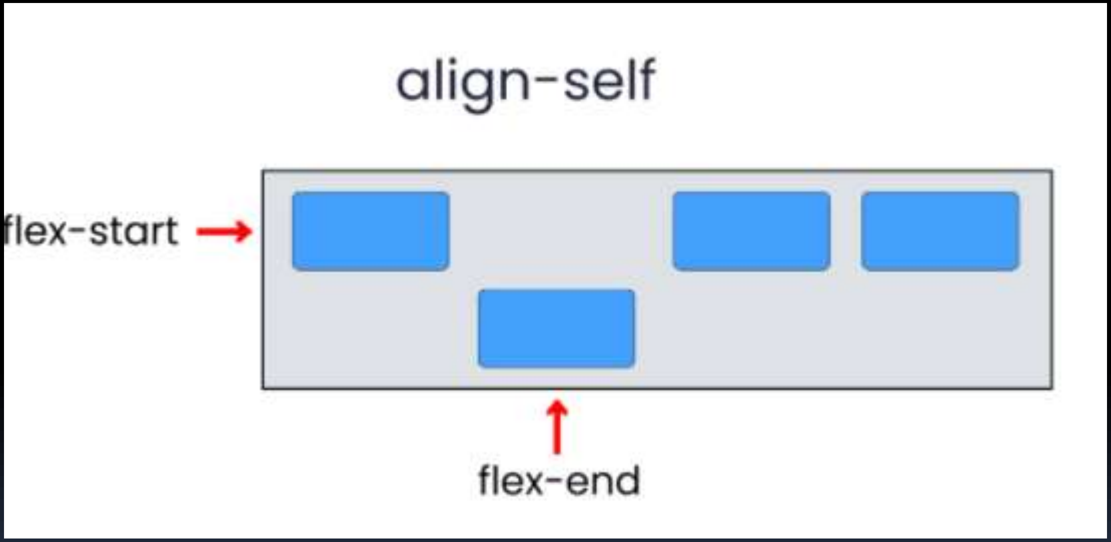
The **flex** Property :

- It is a shorthand to **flex-grow**, **flex-shrink** and **flex-basis** combined.

```
Flex : 0  1  150px;
```

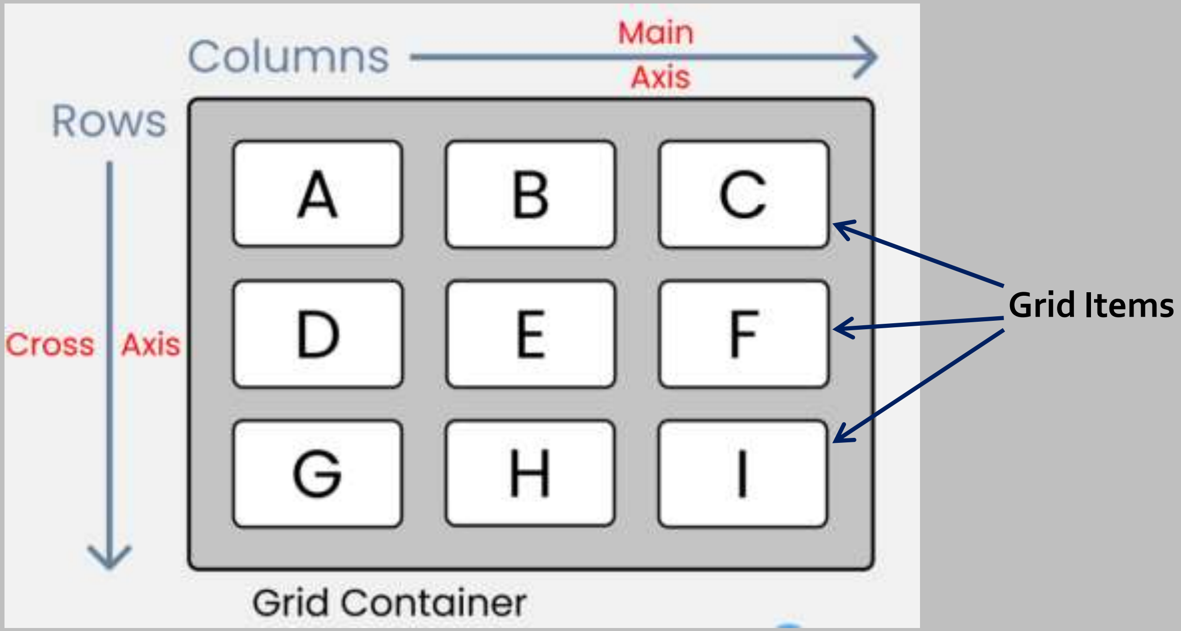
The align-self Property :

- Specifies the alignment for the selected item inside the flexible container.



GRID LAYOUT

- It is a **CSS 3 web layout model**.
- The CSS Grid Layout offers a **two-dimensional** grid layout system, with **rows** and **columns**.
- allows **responsive elements** within a container **to be automatically arranged** depending upon screen size (or device).
- consists of a parent element (**Grid Container**), with one or more child elements (**Grid Items**).

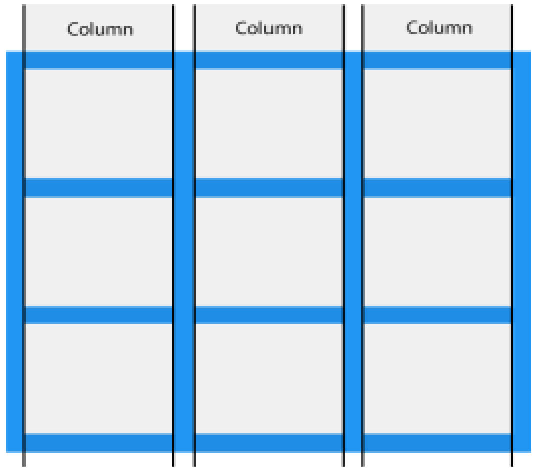

Columns
Main
Axis
Rows
Cross
Axis
A
B
C
D
E
F
G
H
I
Grid Items
Grid Container

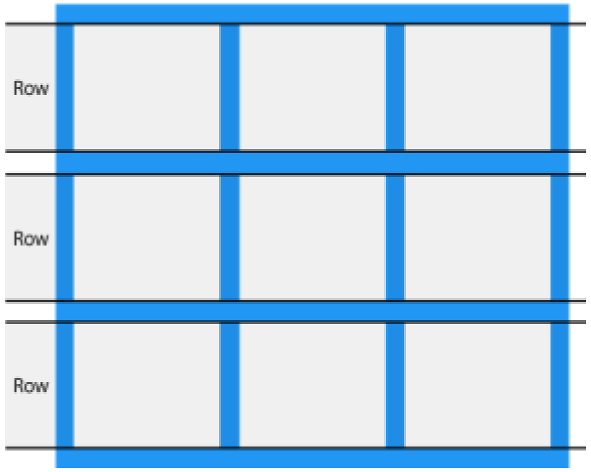
Grid Container:

- To make an HTML element behave as a grid container, you have to set the ***display*** property to ***grid*** or ***inline-grid*****.**
- All direct children of the grid container automatically become ***grid items*****.**
- Grid containers consist of grid items, placed inside columns and rows.
- ***Syntax:***

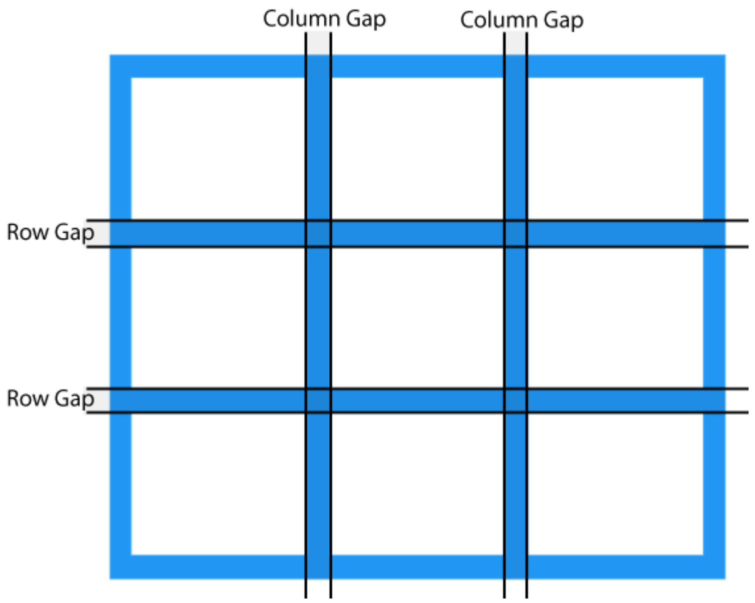
```
.container {
 display: grid | inline-grid;
}
```

- **Grid Columns:** The vertical lines of grid items are called *columns*.
- **Grid Rows:** The horizontal lines of grid items are called *rows*.



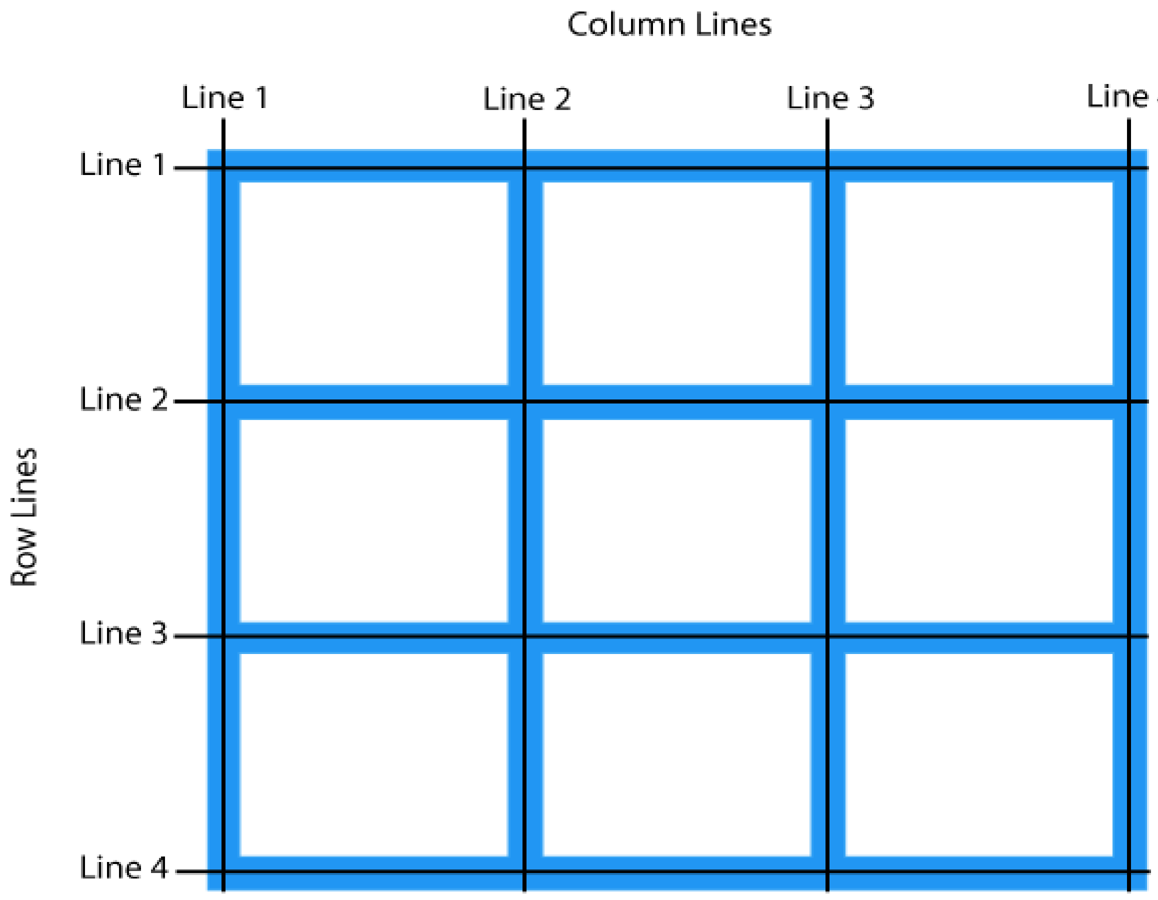


- **Grid Gaps:** The spaces between each column/row are called gaps or gutters.



Grid Lines:

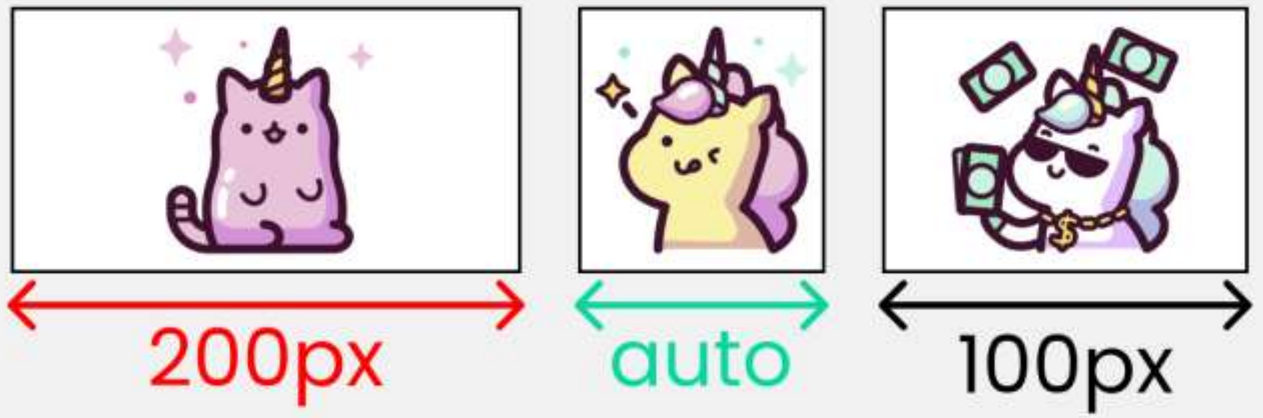
- The lines between columns are called column lines.
- The lines between rows are called row lines.



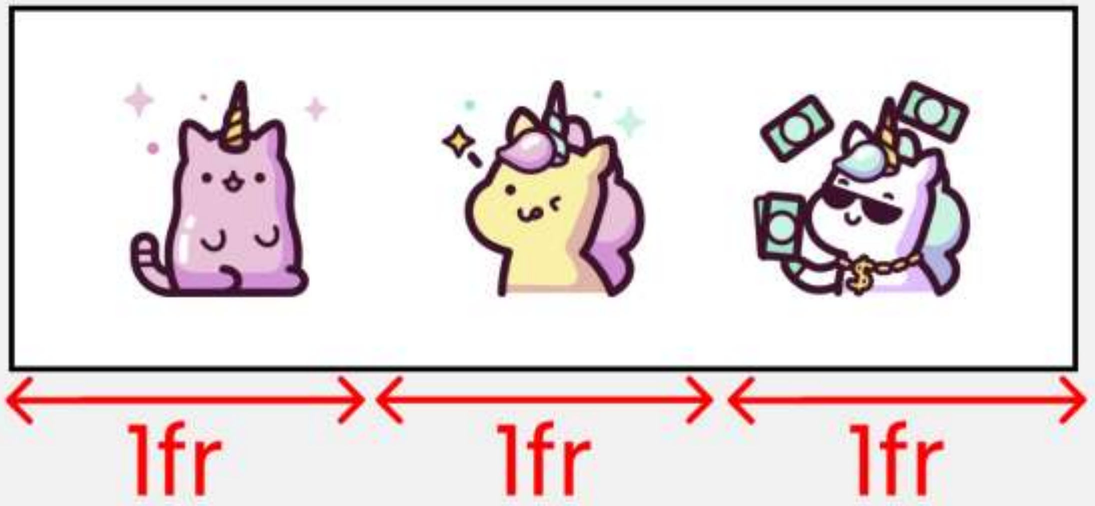
- Grid Container Properties:
  - grid-template-columns
  - grid-template-rows
  - column-gap
  - row-gap
  - gap
  - grid-template-areas
  - justify-items
  - align-items
  - justify-content
  - align-content

- **grid-template-columns property**:
  - specifies the **number** (and the **widths**) of columns in a grid layout.
  - The **values are a space separated list**, where each value specifies the size of the respective column.
  - ***Examples:***
    - grid-template-columns: 90px 50px 120px;
    - grid-template-columns: 1fr 2fr 1fr;
    - grid-template-columns: 100px auto 200px;
    - grid-template-columns: auto auto auto;
- **Fr** is a **fractional unit**. **It represents a fraction of the available space in the grid container**.
- Example: 1fr = 1 part of the available space.

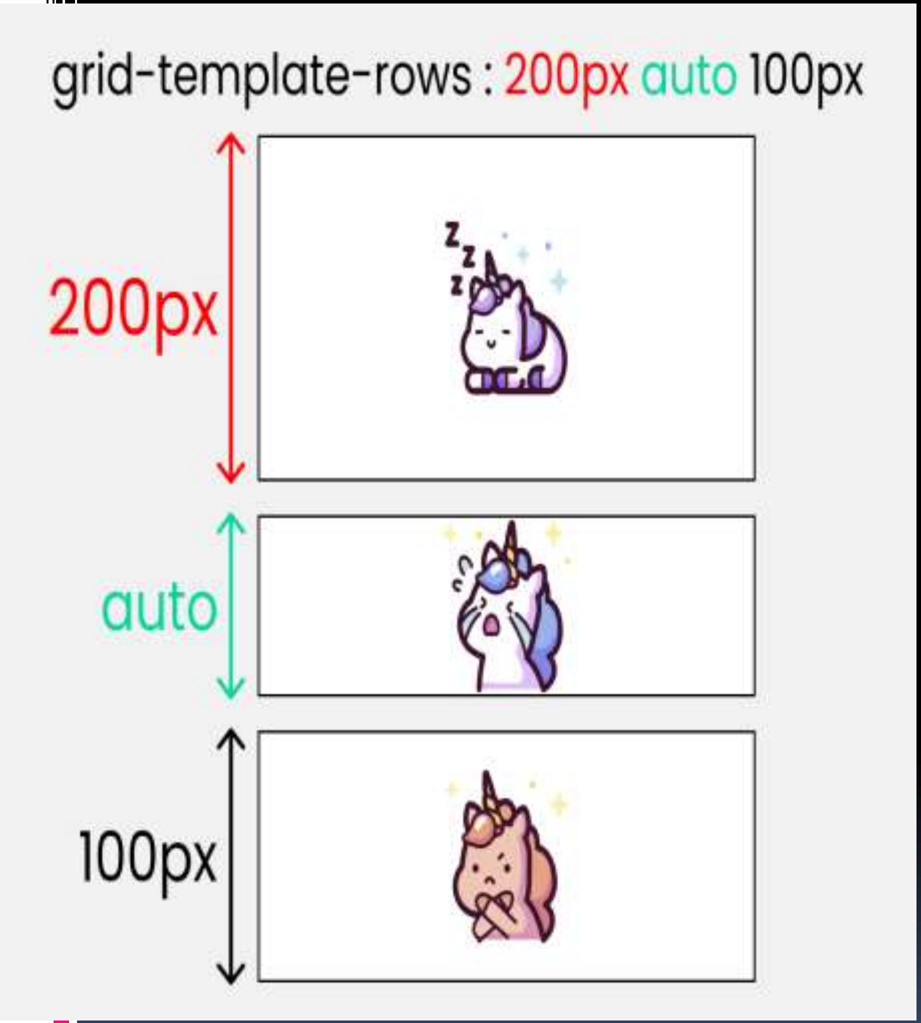
```
grid-template-columns : 200px auto 100px;
```

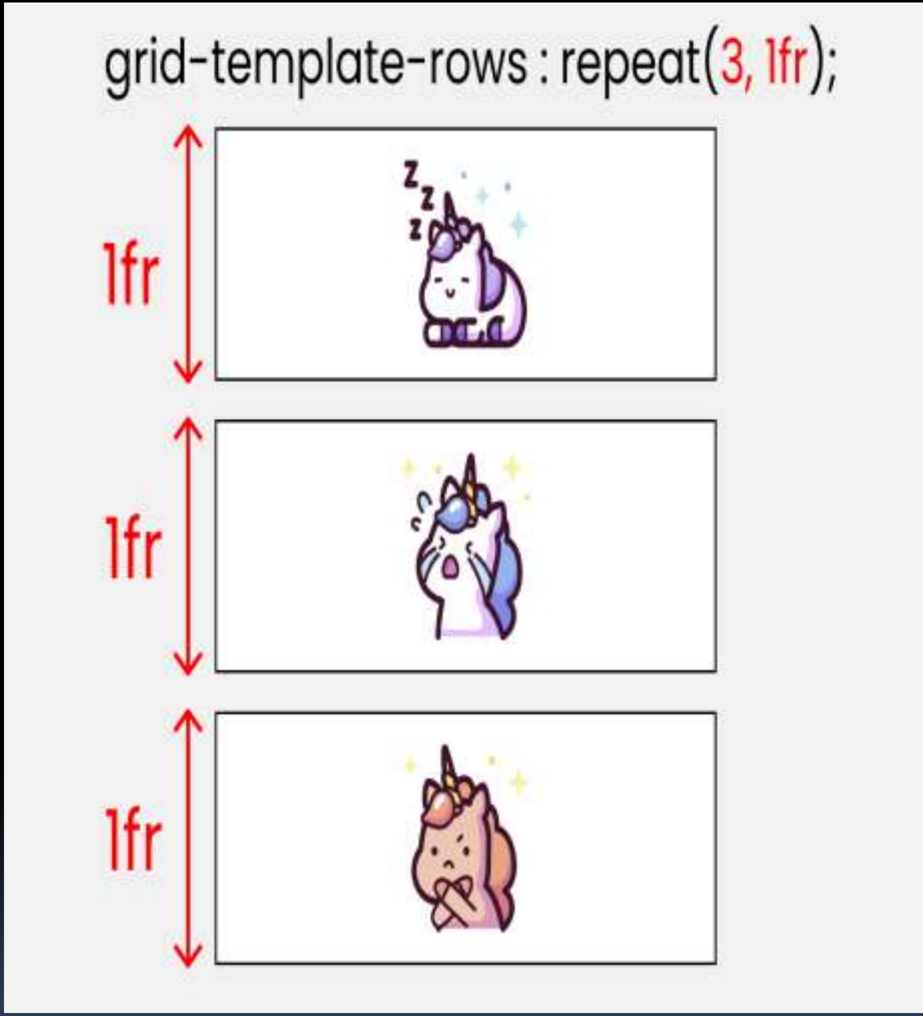


```
grid-template-columns : repeat(3, 1fr);
```



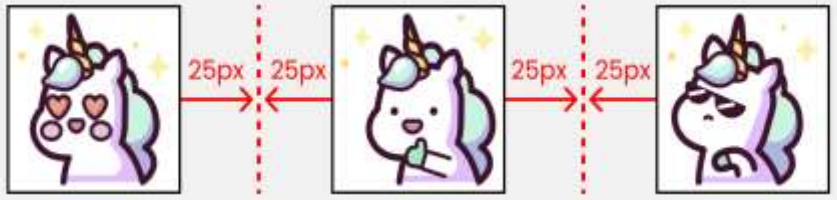
- **grid-template-rows**:
  - Defines the **number of rows** in a grid layout, and it can define the height of each row.
  - The value is a space-separated-list, where each value defines the height of the respective row.
  - ***Examples:***
    - grid-template-rows: 1fr 2fr;
    - grid-template-rows: 50px 100px;
    - grid-template-rows: 100px auto;


grid-template-rows : 200px auto 100px
200px
auto
100px


grid-template-rows : repeat(3, 1fr);
1fr
1fr
1fr

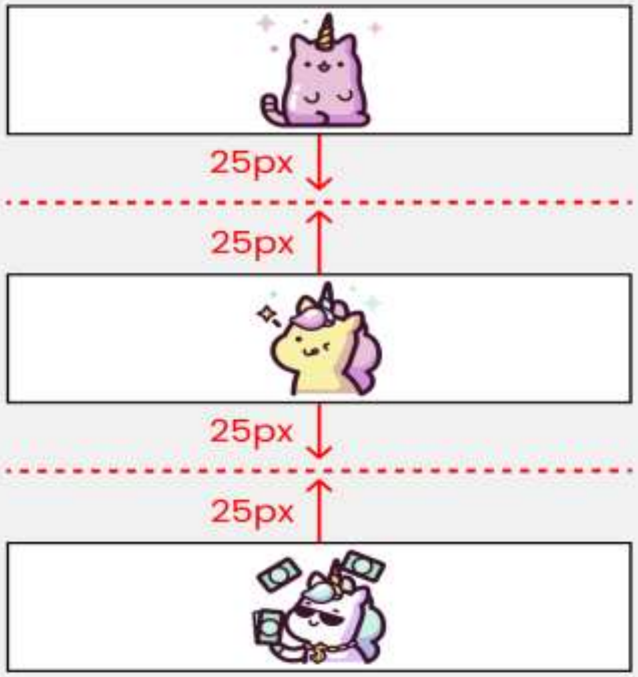
- **column-gap:**
  - Defines the size of the gap between the columns in a grid layout.
  - Syntax: column-gap: length;
- **row-gap:**
  - Defines the size of the gap between the rows in a grid layout.
  - Syntax: row-gap: length;
- **gap property:**
  - defines the size of the gap between the rows and columns in a grid layout, and is a shorthand property for the following properties:
    - row-gap
    - column-gap
  - Syntax: gap: row-gap column-gap;

column-gap: 50px



Red Dotted lines are called -> grid lines

row-gap: 50px

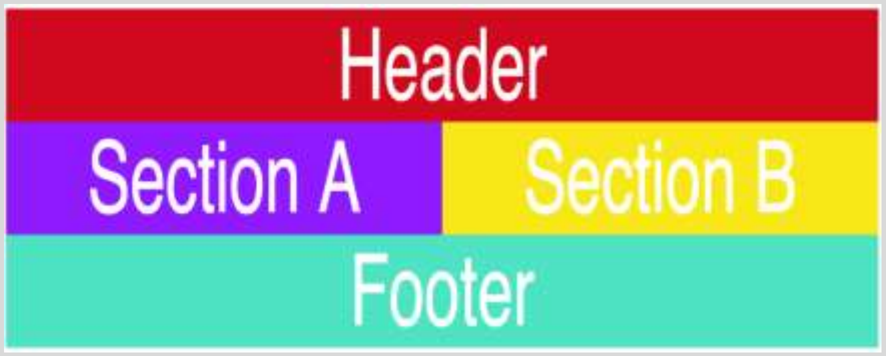


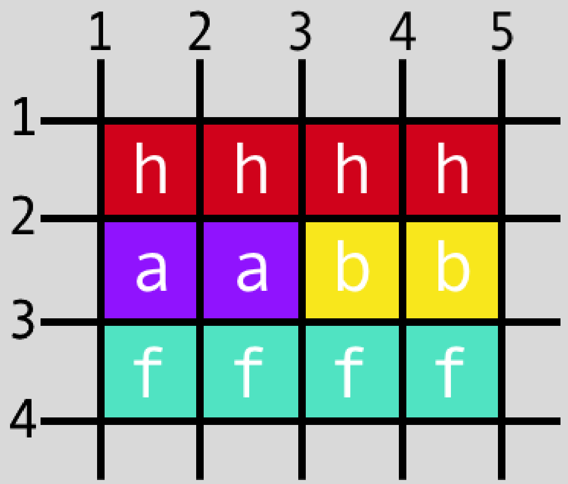
Red Dotted lines are called -> grid lines

- **grid-template-areas**:
  - Specifies **areas** within the grid layout.
  - We **can name grid items** by using the **grid-area** property and then **reference to the name in the grid-template-areas property**.

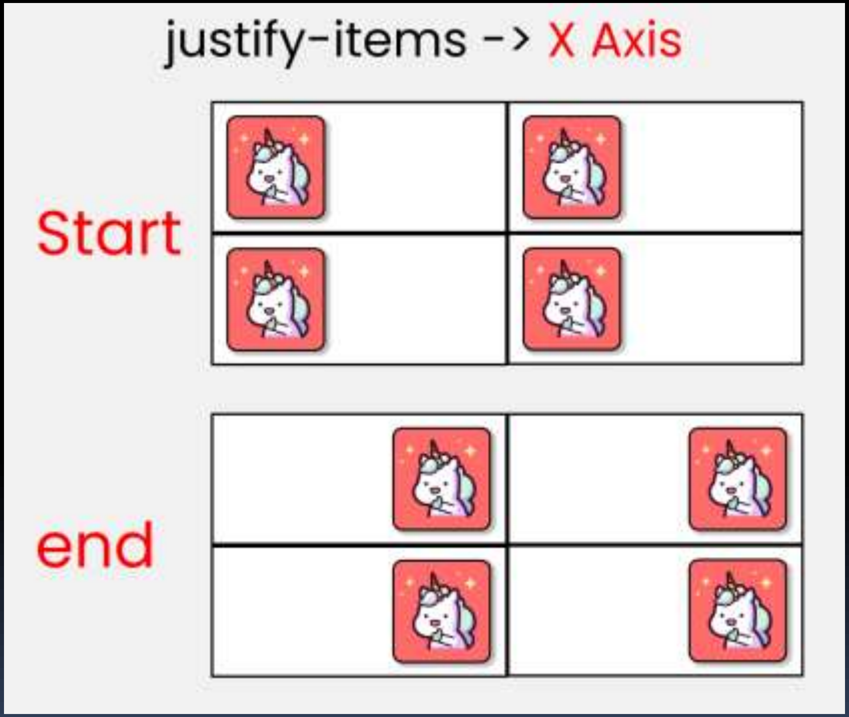
h = Header
a = section A
b = section B
f = Footer

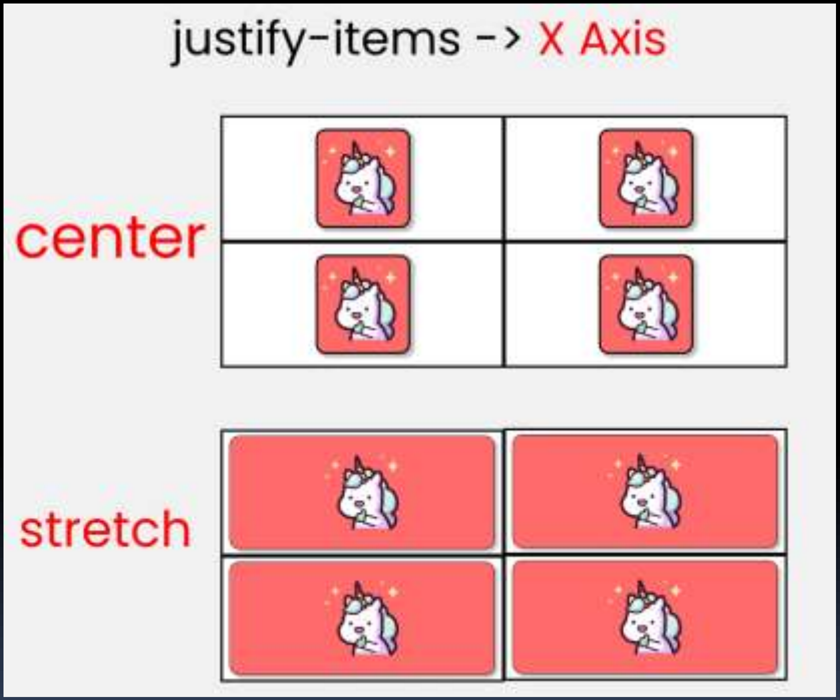
```
grid-template-areas:
      "h  h  h  h"
      "a  a  b  b"
      "f  f  f  f"
```



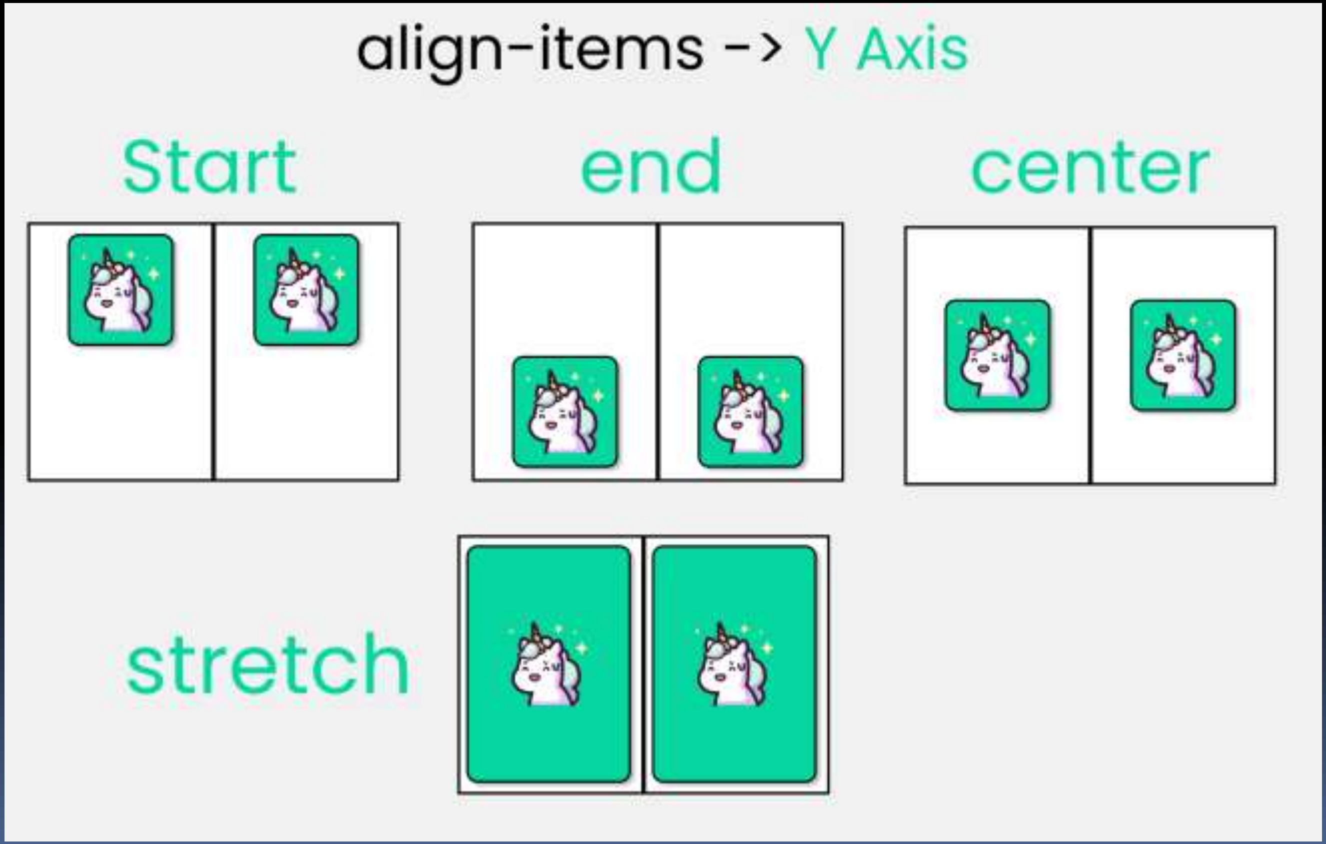


- **justify-items property:**
  - Used to position grid-items inside grid containers along the X-Axis (row).

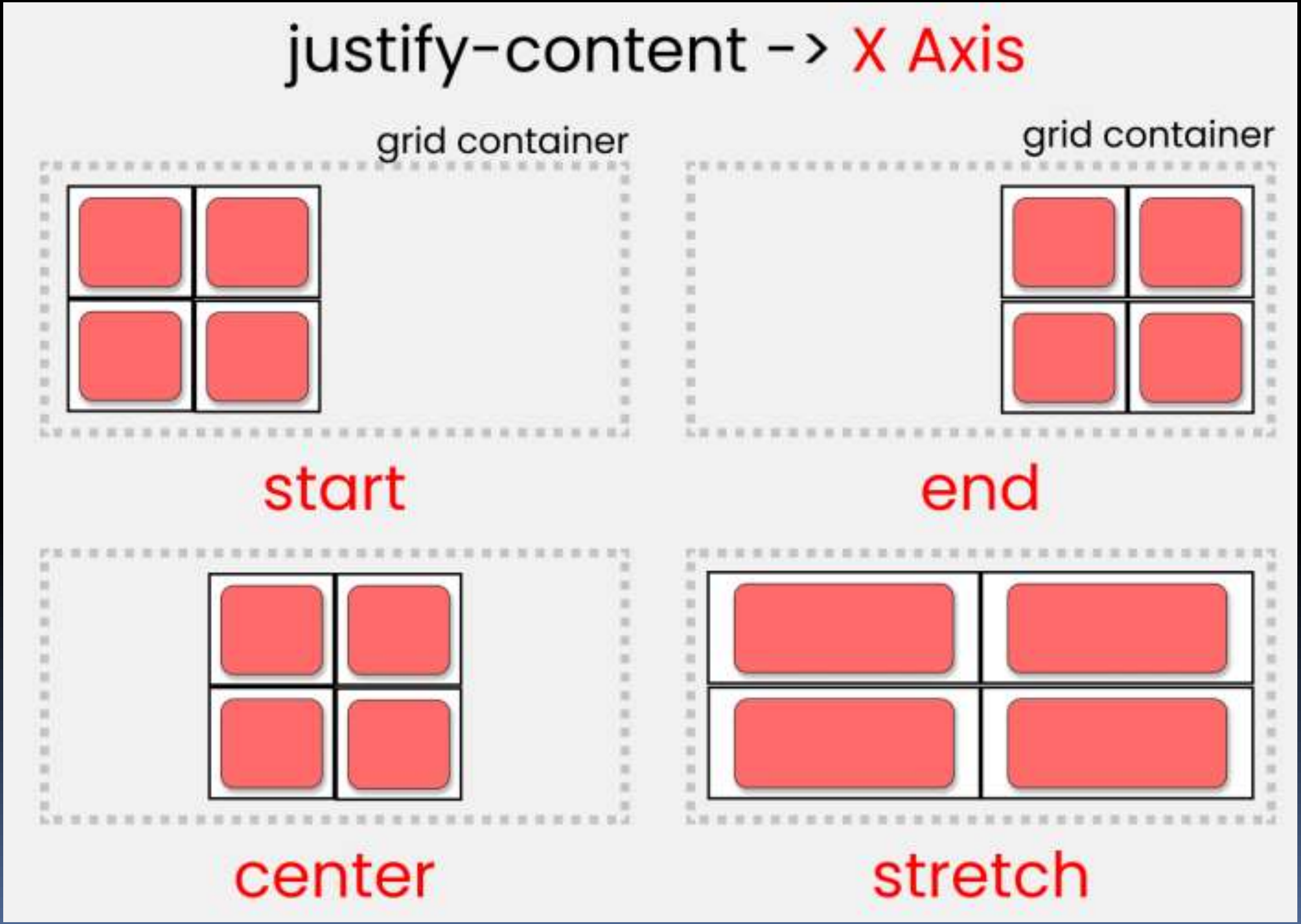


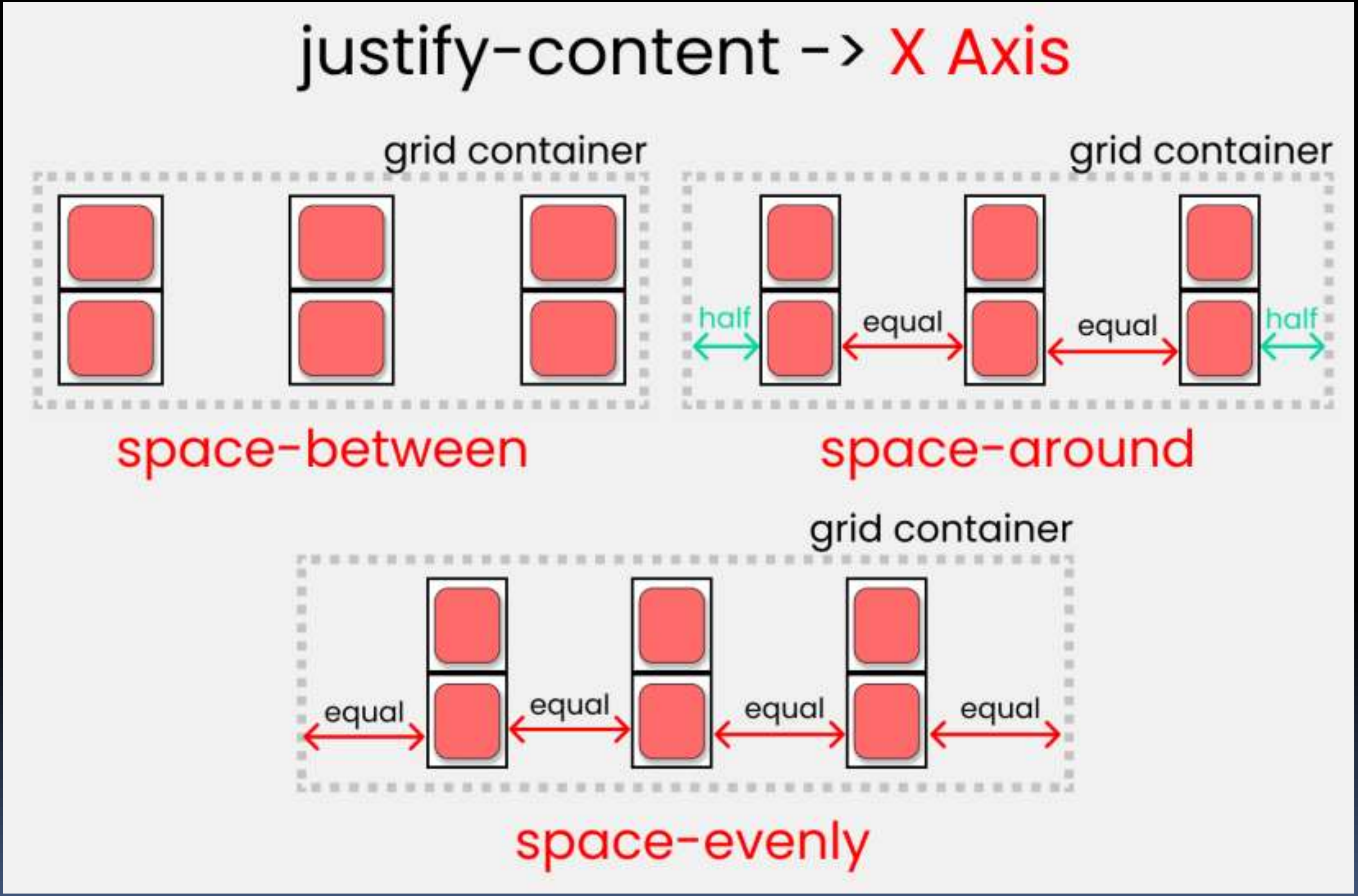


- **align-items:**
  - Used to position grid-items inside the grid container along the Y-Axis (column)

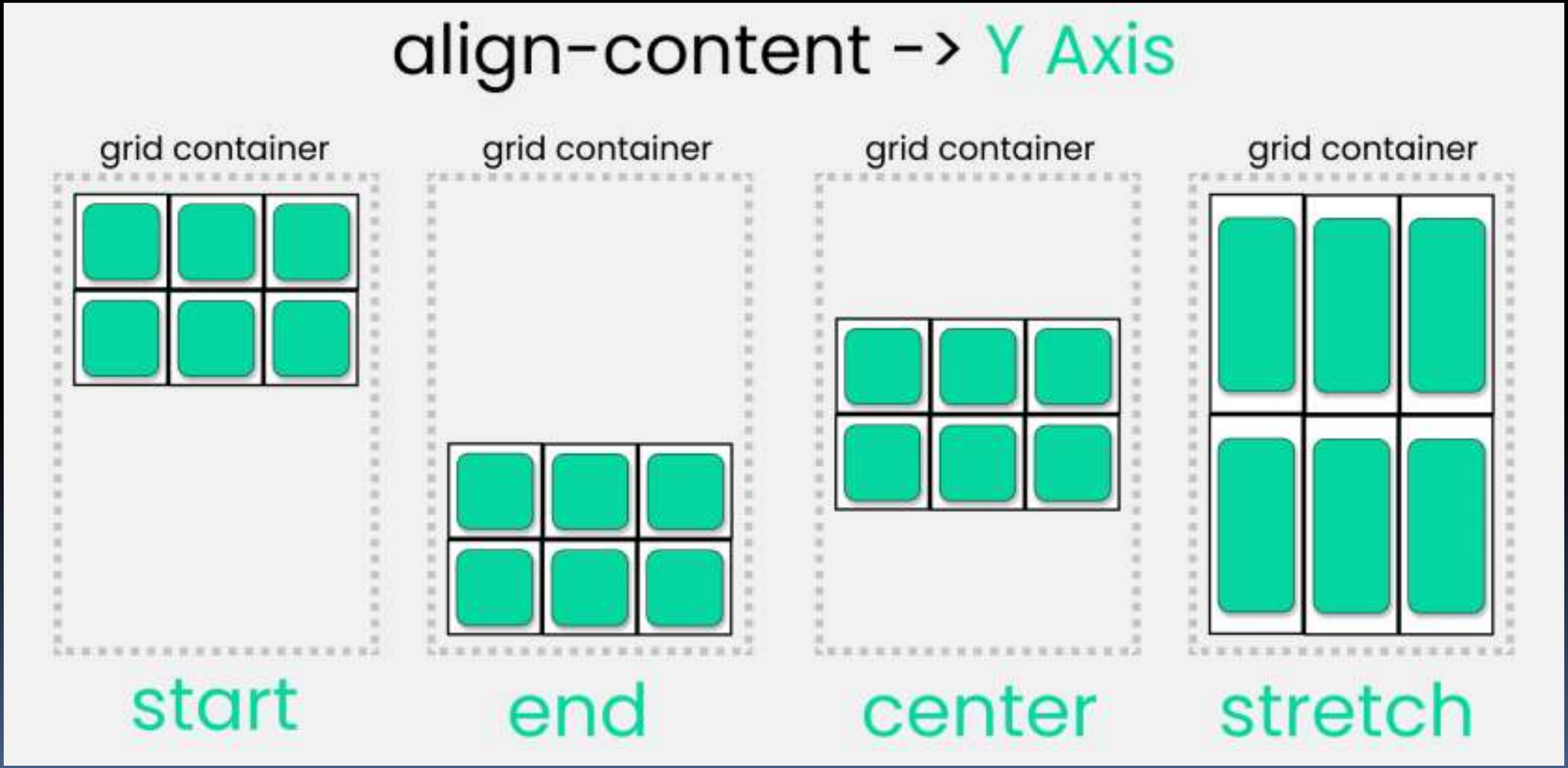


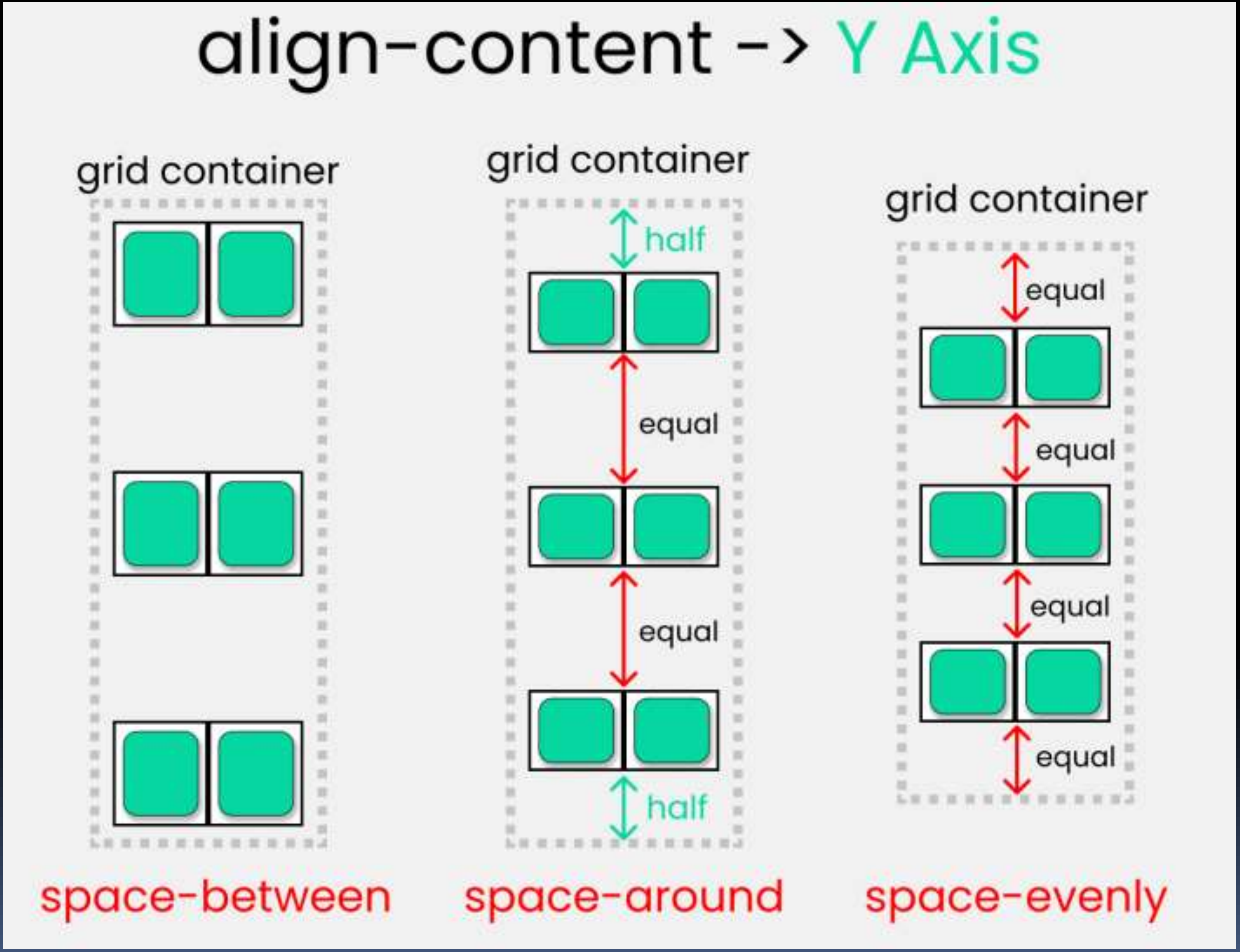
- **justify-content:**
  - Used to position grid inside the grid container along the X-Axis (row).




justify-content -> X Axis
grid container
space-between
grid container
half
equal
equal
half
space-around
grid container
equal
equal
equal
equal
space-evenly

- **align-content:**
  - Used to position grid inside the grid container along the Y-Axis (column).




align-content -> Y Axis
grid container
grid container
grid container
half
equal
equal
half
equal
equal
equal
equal
space-between
space-around
space-evenly

CSS Grid Items:

- All direct children of the grid container automatically become ***grid items***.
- Grid containers consist of grid items, placed inside columns and rows.
- By default, **a container has one grid item for each column, in each row**, but you can style the grid items so that they will span multiple columns and/or rows.

- **grid-column-start:**
  - Defines on which column-line the item will start.
- **grid-column-end:**
  - Defines how many columns an item will span, or on which column-line the item will end.
- **grid-column:**
  - shorthand property for the following properties:
  - grid-column-start / grid-column-end

- **grid-row-start:**
  - defines on which row-line the item will start.
- **grid-row-end:**
  - defines how many rows an item will span, or on which row-line the item will end
- **grid-row:**
  - shorthand property for the following properties:
  - grid-row-start/ grid-row-end
- use the keyword "**span**" to define how many columns/rows the item will span.

- **grid-area property or naming grid items:**
  - grid-area is also a shorthand property for *grid-row* and *grid-column* property.
  - The grid-area property can also be **used to assign a name to a grid item**.
  - **Named grid items can then be referenced to by the grid-template-areas property of the grid container.**

BOOTSTRAP

- Developed at **Twitter**, and released as an open source product in August 2011 on GitHub.
- In June 2014 Bootstrap was the **No.1** project on **GitHub**.
- **"Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites."**
- Open Source – free to download and use
- **Includes HTML and CSS and JavaScript-based design templates** for typography, forms, buttons, tables, navigation and other interface components., as well as optional JavaScript plugins.

- **Latest Version** :
  - **BOOTSTRAP 5 with new components, faster stylesheet and more responsiveness**.
  - Bootstrap 5 **supports the latest, stable releases of all major browsers and platforms**.
  - The main differences between Bootstrap 5 and Bootstrap 3 & 4, is that Bootstrap 5 has switched to ***JavaScript*** instead of jQuery.

Using Bootstrap 5:

- Two ways :
  - Include Bootstrap 5 from a CDN
    - jsDelivr provides CDN support for Bootstrap's CSS and JavaScript.
    - <!-- Latest compiled and minified CSS -->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
    - <!-- Latest compiled JavaScript -->
      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></script>
  - Download Bootstrap 5 from https://getbootstrap.com/

Designing a Web page using Bootstrap 5:

1. Add the HTML5 doctype:
   - Include the HTML5 doctype at the beginning of the page, along with the lang attribute and character set.

   -
     ```
     <!DOCTYPE html>
     <html lang="en">
      <head>
       <title>Bootstrap 5 Example</title>
       <meta charset="utf-8">
      </head>
     </html>
     ```
2. Responsive meta tag:
   - To ensure proper rendering and touch zooming, add the following <meta> tag inside the <head> element.
     - `<meta name="viewport" content="width=device-width, initial-scale=1">`

3. **Containers**:

- There are two container classes to choose from:
  - The **.container** class provides a responsive fixed width container
  - The **.container-fluid** class provides a full width container, spanning the entire width of the viewport.

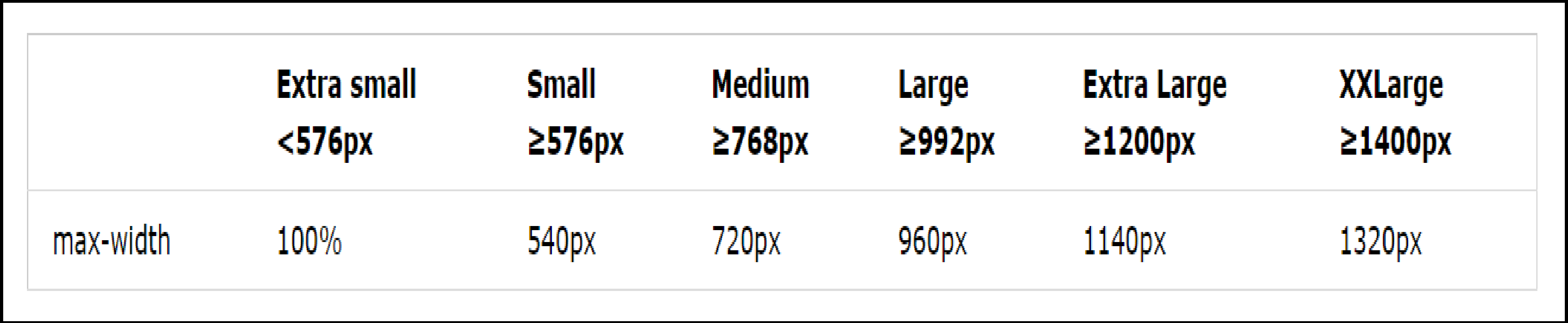
Bootstrap Containers:

- Containers are the most basic layout element in Bootstrap.
- Bootstrap 5 also requires a containing element to wrap site contents.
- There are two main containers:

1. Fixed Container
2. Fluid Container

Fixed Container (Default Container):

- *.container* class is **used to create a responsive, fixed-width container**.
- Its **max-width changes on different screen sizes**



	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra Large ≥1200px	XXLarge ≥1400px
max-width	100%	540px	720px	960px	1140px	1320px

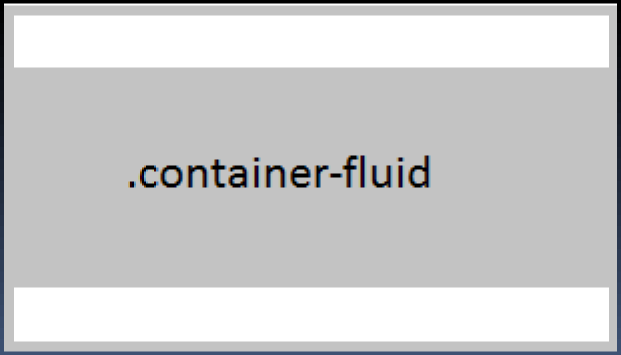
- *Syntax:*

```
<div class="container">
 ...
</div>
```

Fluid Container:

- Use the .*container-fluid* class to create a **full width container that will always span the entire width of the screen.**
- The width of a fluid container is always **100%**.
- Syntax:

```
<div class="container-fluid">
  ...
</div>
```



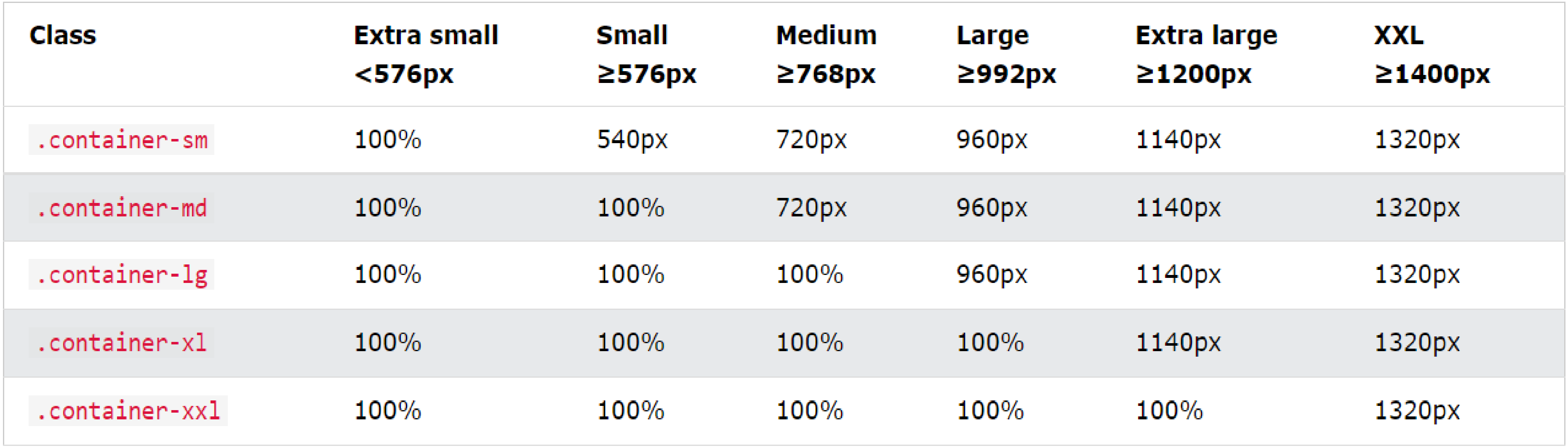
Bootstrap 5 Text/Typography

- Bootstrap 5 uses a **default font-size of 16px** by default, and its line-height is 1.5.
- **<h1> - <h6>**
  - Bootstrap 5 styles HTML headings (<h1> to <h6>) with a bolder font-weight and a responsive font-size.
  - There are six classes from: .h1 to .h6
- **Display Headings**
  - Display headings are used to stand out more than normal headings (larger font-size and lighter font-weight),
  - there are six classes to choose from:

    .display-1 to .display-6:

Responsive Containers

- can also use the **.container-sm|md|lg|xl** classes to determine when the container should be responsive.
- The **max-width of the container will change on different screen sizes/viewports**:



Class	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px	XXL ≥1400px
`.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

Bootstrap 5 Buttons

- The button classes can be used on <a>, <button>, or <input> elements
- Classes – **btn** followed by one of the contextual classes btn-primary, btn-secondary, btn-success, btn-info, btn-warning, btn-danger, btn-dark, btn-light, btn-link

Bootstrap 5 Alerts

- Bootstrap 5 provides an easy way to create predefined alert messages
- Alerts are created with the .alert class, followed by one of the contextual classes .alert-success, .alert-info, .alert-warning, .alert-danger, .alert-primary, .alert-secondary, .alert-light or .alert-dark

Bootstrap 5 Navbars

- **A navigation bar is a navigation header that is placed at the top of the page**
- With Bootstrap, a navigation bar **can extend or collapse**, depending on the screen size.
- A standard navigation bar is created with the **.navbar** class, followed by a responsive collapsing class: **.navbar-expand-xxl|xl|lg|md|sm** (stacks the navbar vertically on xxlarge, extra large, large, medium or small screens).
- To add links inside the navbar, use either an **<ul> element** (or a <div>) with class="**navbar-nav**". Then add <li> elements with a **.nav-item** class followed by an <a> element with a **.nav-link** class:

Advantages of Bootstrap

- Easy to use
- Responsive features
- Mobile-first approach
- Browser compatibility