GDSC Web Session

What is CSS?

CSS (Cascading Style Sheets) is a stylesheet language used in web development to control the visual presentation of web pages. CSS operates by selecting HTML elements and applying rules to modify their appearance. Order of precedence for conflicting styles.

How to add CSS to an HTML file

CSS can be added by three ways

- Inline CSS
- Internal CSS
- External CSS

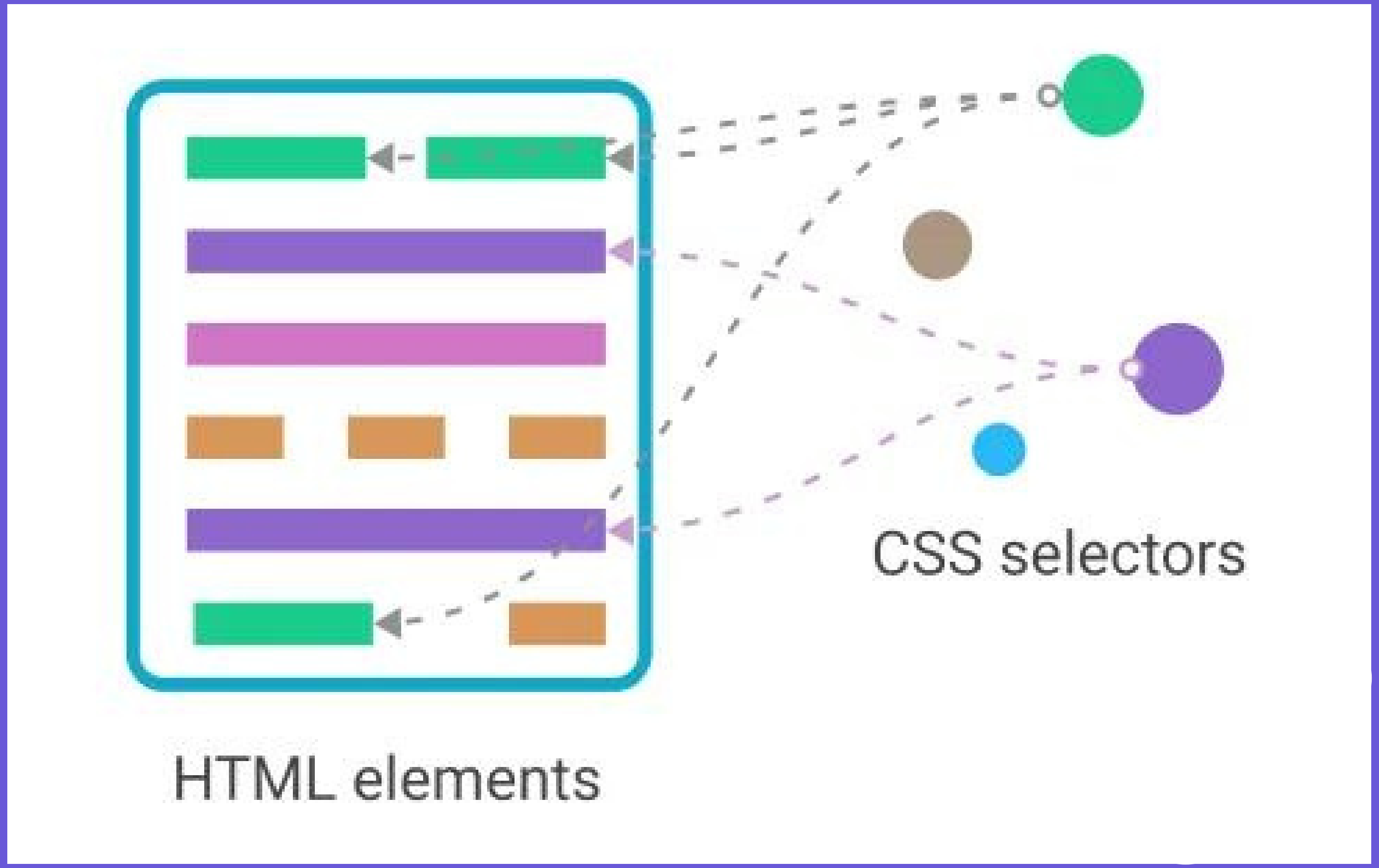
Tip: Install live-server from VS code extensions for better experience.

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

There are two commonly used selectors

- **Simple Selectors**
- **Combinator Selectors**



Simple selectors

- **Element Selector**

The element selector selects HTML elements based on the element name.

div,p,form etc

- **Class Selector**

The class selector selects HTML elements with a specific class attribute.

.object, .item, .(any element class name)

- **Id Selector**

The class selector selects HTML elements with a unique id attribute.

#object, #item, #(any element class name)

Combinator Selectors

- **Decendent Selector (space)**

The descendant selector matches all elements that are descendants of a specified element

```
div p {
     background-color: yellow;
 }
```

- **Child Selector (>)**

The child selector selects all elements that are the children of a specified element.

```
div > p {
     background-color: yellow;
 }
```

- **Adjacent Sibling Selector (+)**

The adjacent sibling selector is used to select an element that is directly after another specific element.

```
div + p {
    background-color: yellow;
 }
```

- **General Sibling Selector (~)**

The general sibling selector selects all elements that are next siblings of a specified element.

```
div ~ p {
    background-color: yellow;
 }
```

Advanced CSS Selectors

1) Psuedo-class Selectors

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

Some commonly used psuedo classes are

- :link
- :visited
- :hover
- :first-child
- :nth-child(number)

2) Psuedo-element Selector

A CSS pseudo-element is used to style specified parts of an element.
For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Some commonly used psuedo elements are

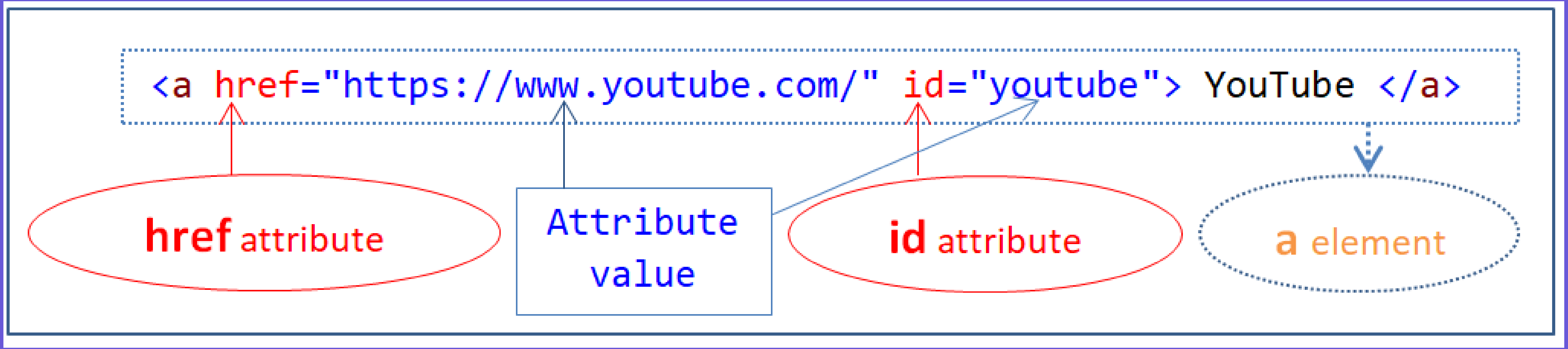
- ::before
- ::after
- ::first-line
- ::first-letter

3) Attribute Selector

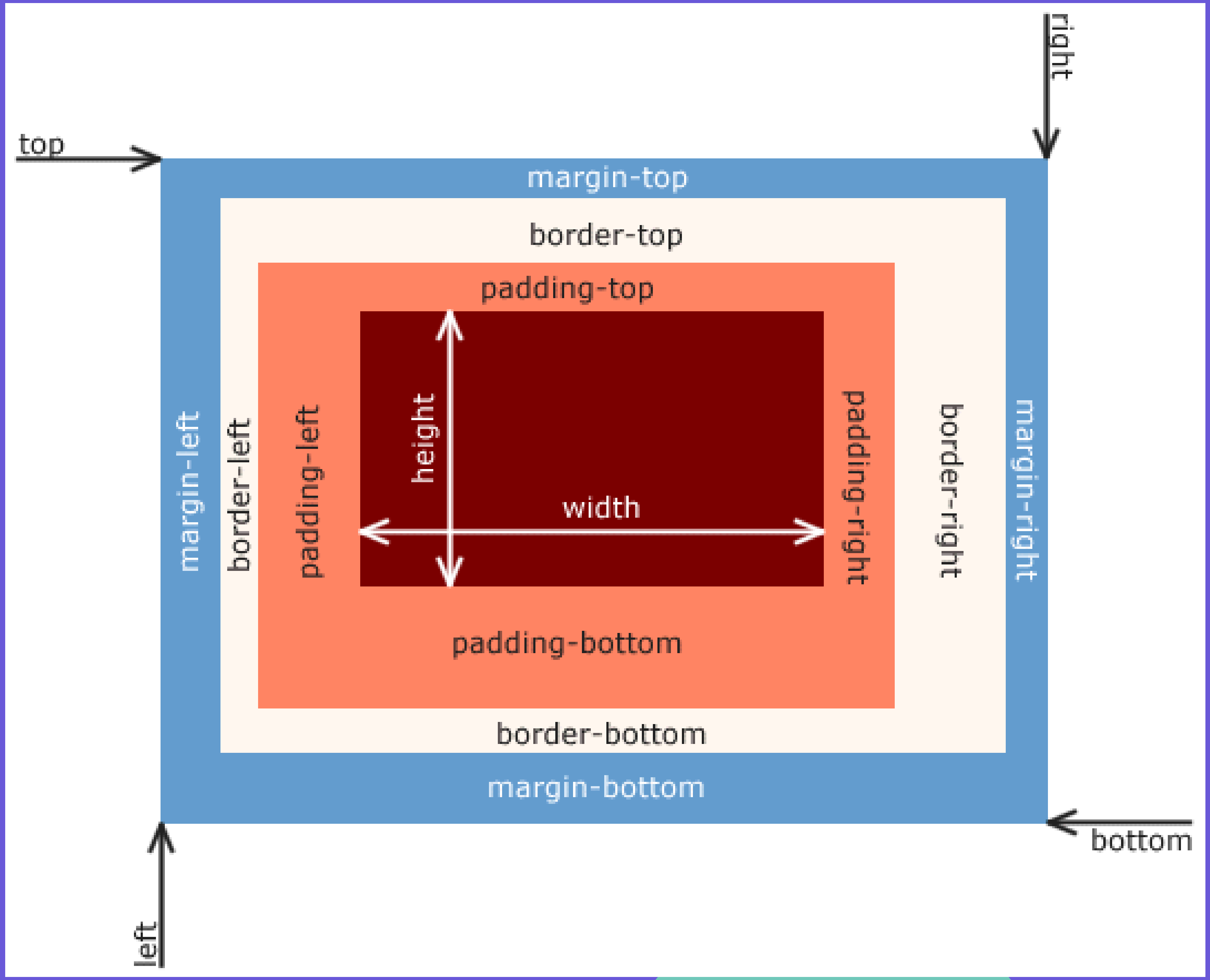
These type of selectors are used to select elements that have specific attributes or attribute values.

Some commonly used elements are

- [attribute] (Ex: a[target])
- [attribute="value"] (Ex: a[target="_blank"])
- [attribute~="value"] (Ex: [title~="flower"])


```
<a href="https://www.youtube.com/" id="youtube"> YouTube </a>
```

CSS Box Model



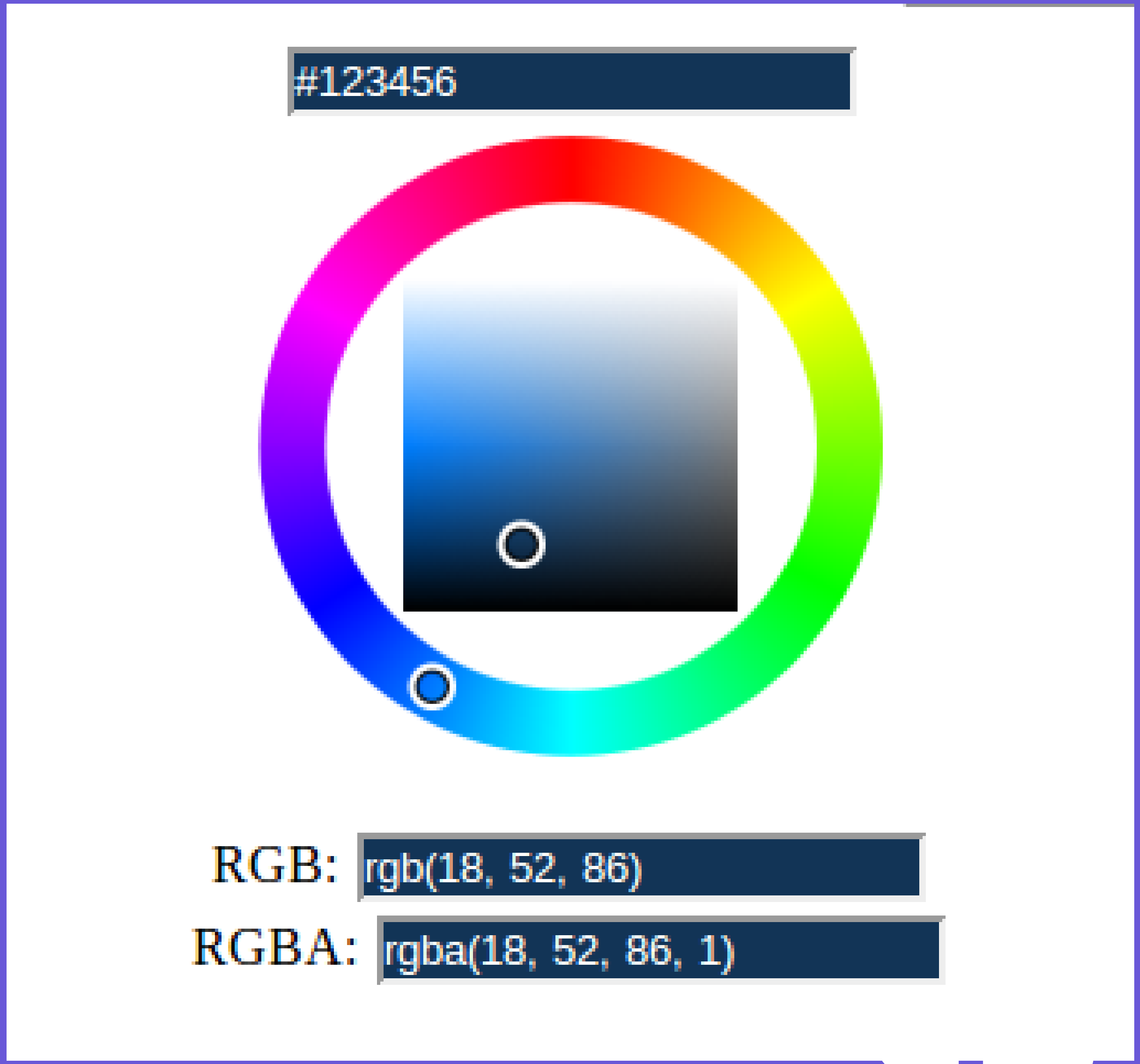
Text Styling and Formatting in CSS

- Text Align
- Text Color
- Text Decoration
- Font-size
- Font-family
- Line height
- Letter Spacing
- Font-weight

CSS Color Properties:

CSS offers various ways to define and manipulate colors but does the same work.

- RGB
- RGBA
- Hexadecimal Color Codes

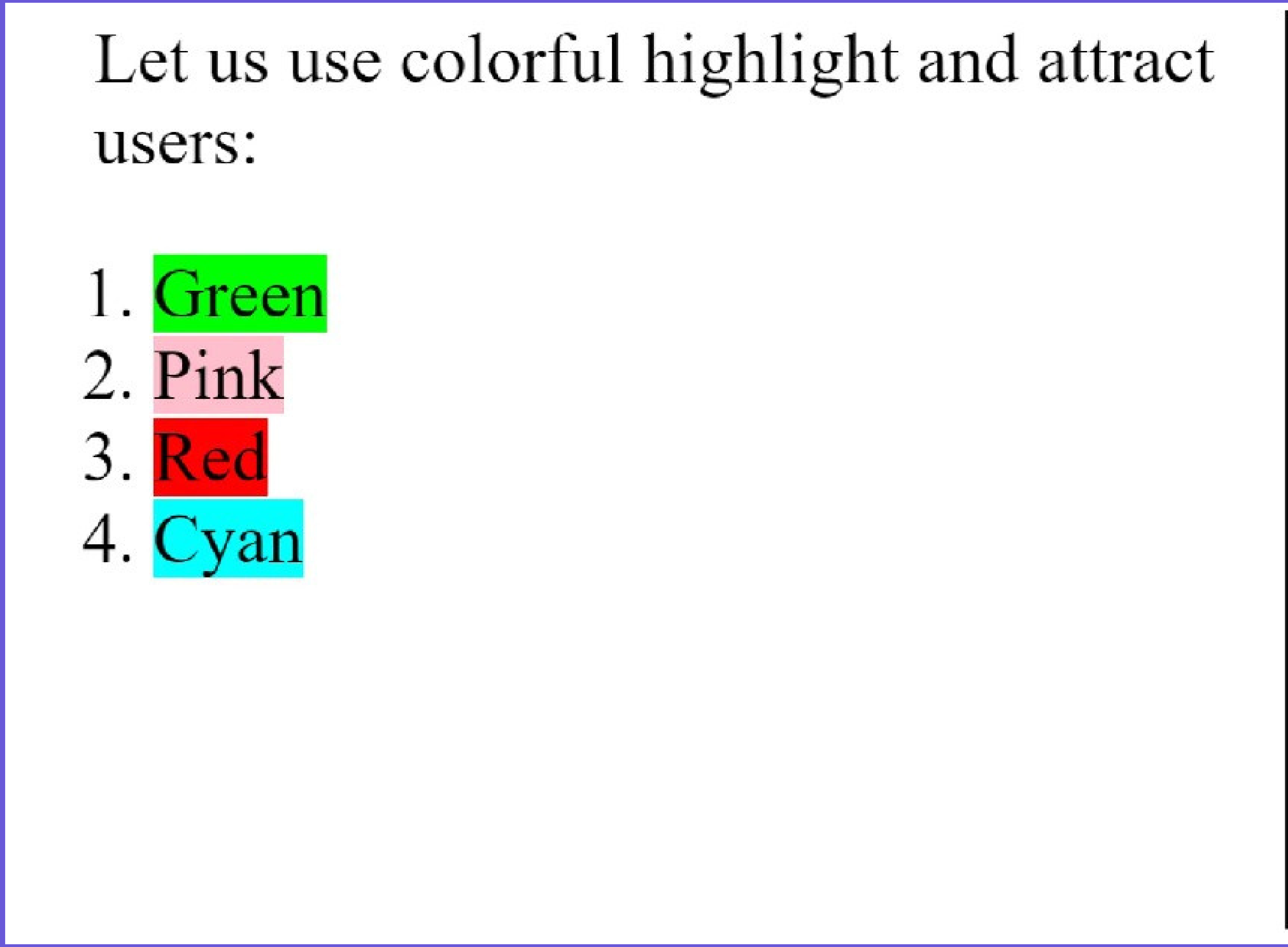


#123456

RGB: rgb(18, 52, 86)

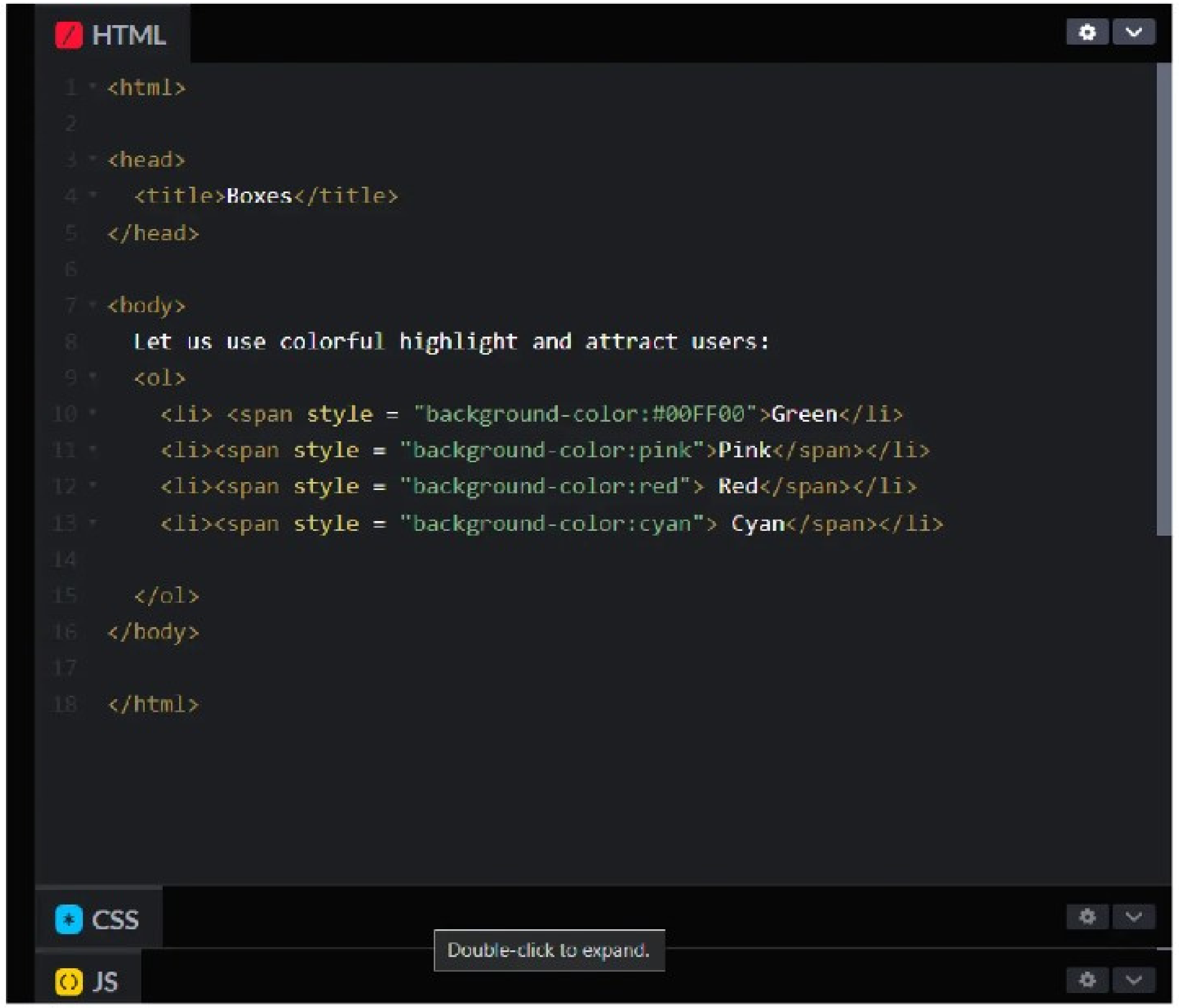
RGBA: rgba(18, 52, 86, 1)

CSS Color Properties:



Let us use colorful highlight and attract users:

1. Green
2. Pink
3. Red
4. Cyan



```
<html>

<head>
  <title>Boxes</title>
</head>

<body>
  Let us use colorful highlight and attract users:
  <ol>
    <li> <span style = "background-color:#00FF00">Green</li>
    <li><span style = "background-color:pink">Pink</span></li>
    <li><span style = "background-color:red"> Red</span></li>
    <li><span style = "background-color:cyan"> Cyan</span></li>

  </ol>
</body>

</html>
```

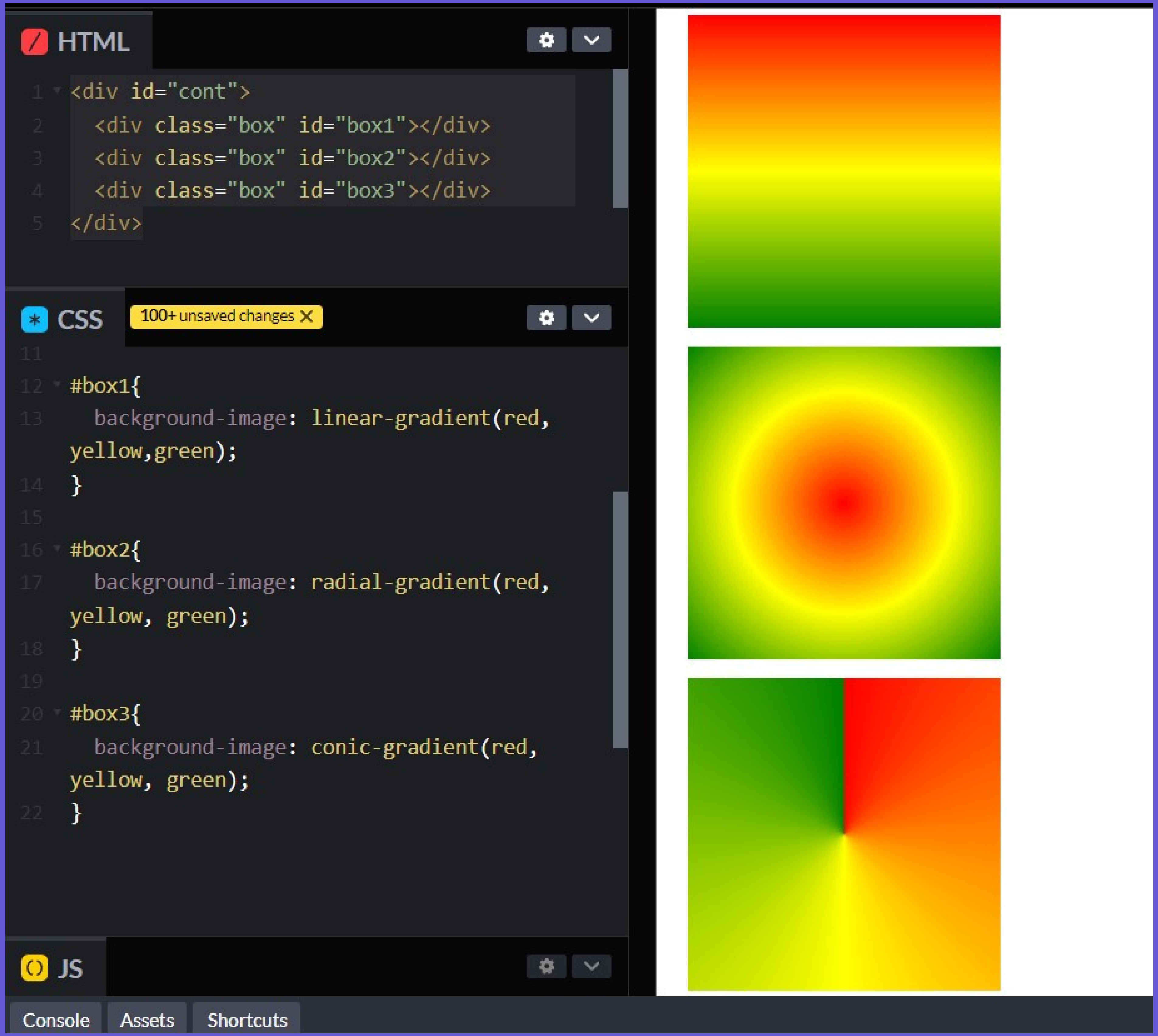
CSS Gradients:

CSS gradients helps you to design more cooler web pages by creating smooth transitions between two or more colors, adding depth and dimension to your designs.

There are 3 types of gradients:

- Linear Gradient
- Radial Gradient
- Conic Gradient

CSS Gradient Example:



```html
<div id="cont">
  <div class="box" id="box1"></div>
  <div class="box" id="box2"></div>
  <div class="box" id="box3"></div>
</div>
```

```css

#box1{
  background-image: linear-gradient(red,
yellow,green);
}

#box2{
  background-image: radial-gradient(red,
yellow, green);
}

#box3{
  background-image: conic-gradient(red,
yellow, green);
}
```

Text Shadow:

Specify the shadow's color, offset, blur radius, and spread distance to create unique text effects.

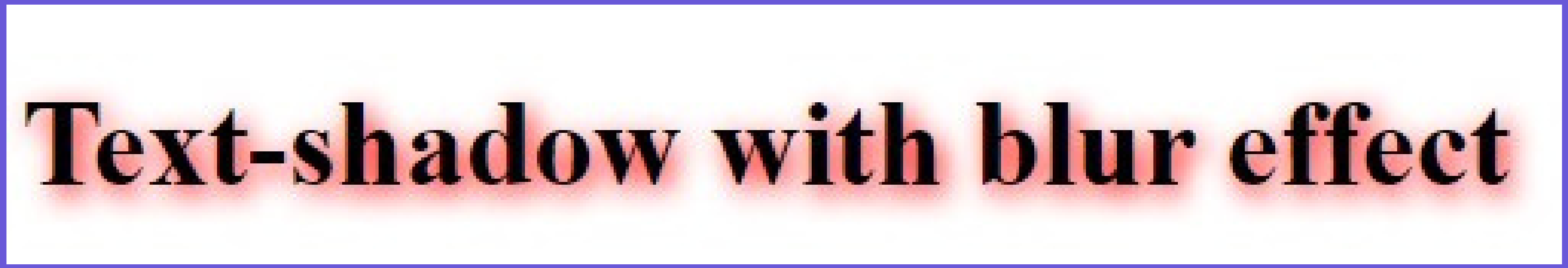
Syntax :

```
text-shadow : h-shadow v-shadow blur-radius color|none|initial|inherit;
```

Code :

```
h1 {
  text-shadow: 2px 2px 8px #FF0000;
}
```

Output :



Box Shadow:

This property enables you to cast shadows around elements like divs. .This is useful for creating card-like or raised element effects.

Syntax :

```
box-shadow: none|h-offset v-offset blur spread color |inset|initial|inherit;
```

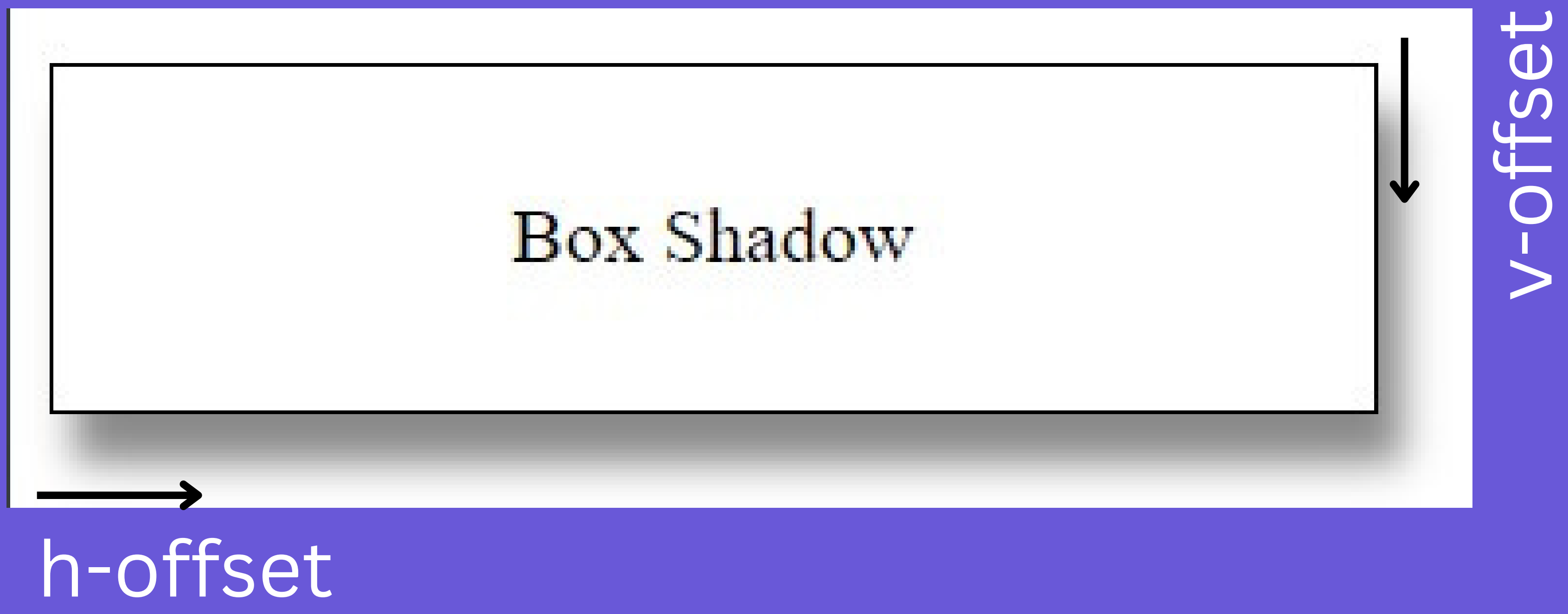
Code :

h-offset blur

```
div {
  box-shadow: 5px 10px 8px #888888;
}
```

v-offset color

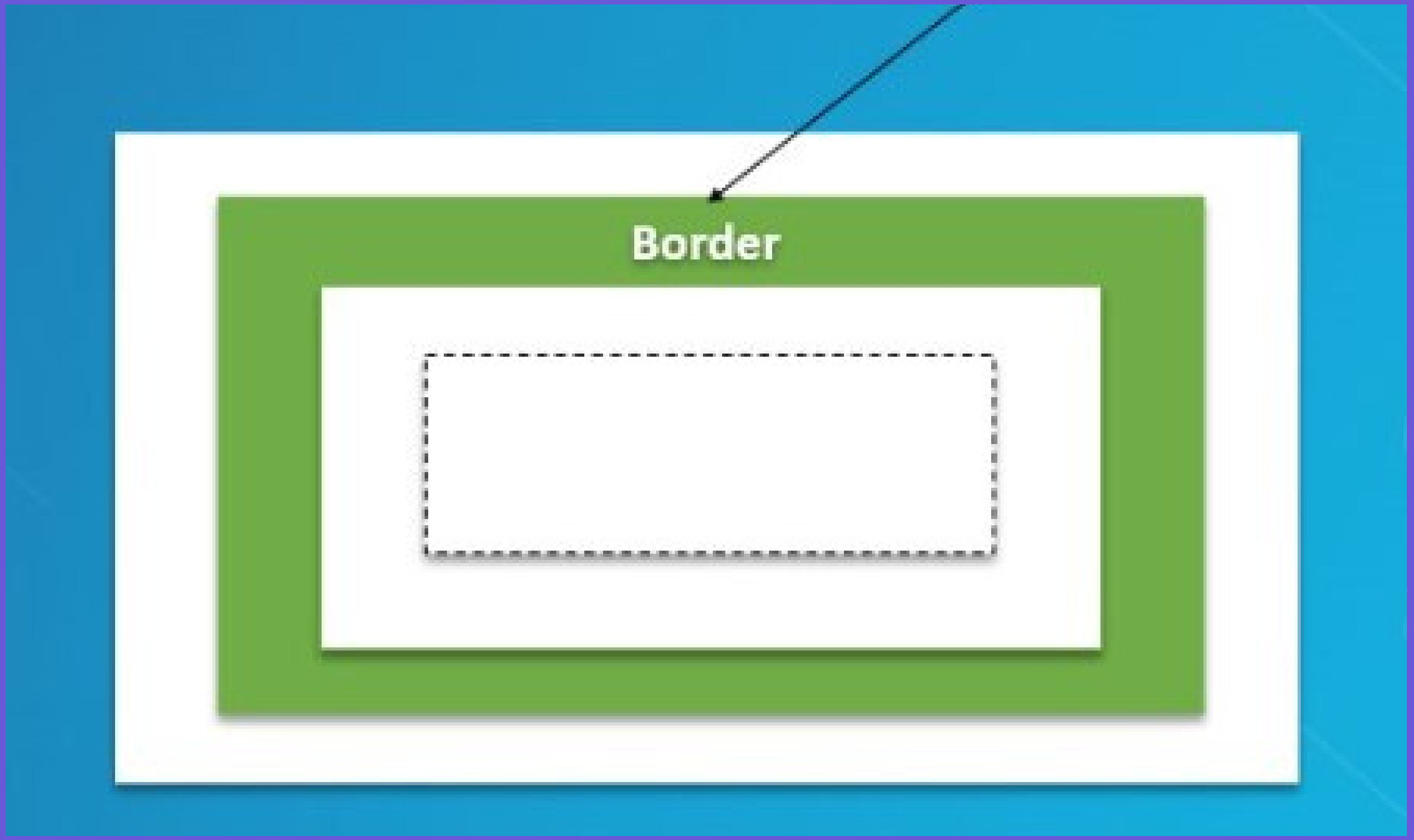
Output :



CSS Border Properties:

- Borders are an essential part of element styling in CSS, providing visual separation and distinction.

- There are various ways to manipulate borders:
- **border-width**
- **border-color**
- **border-style**
- **border-radius**



- **border-width**

This property allows you to control the thickness or width of the border. You can set it in pixels, em, or other units.

- **border-color**

You can specify the color of the border using this property, just like you would with text or background colors.

- **border-style**

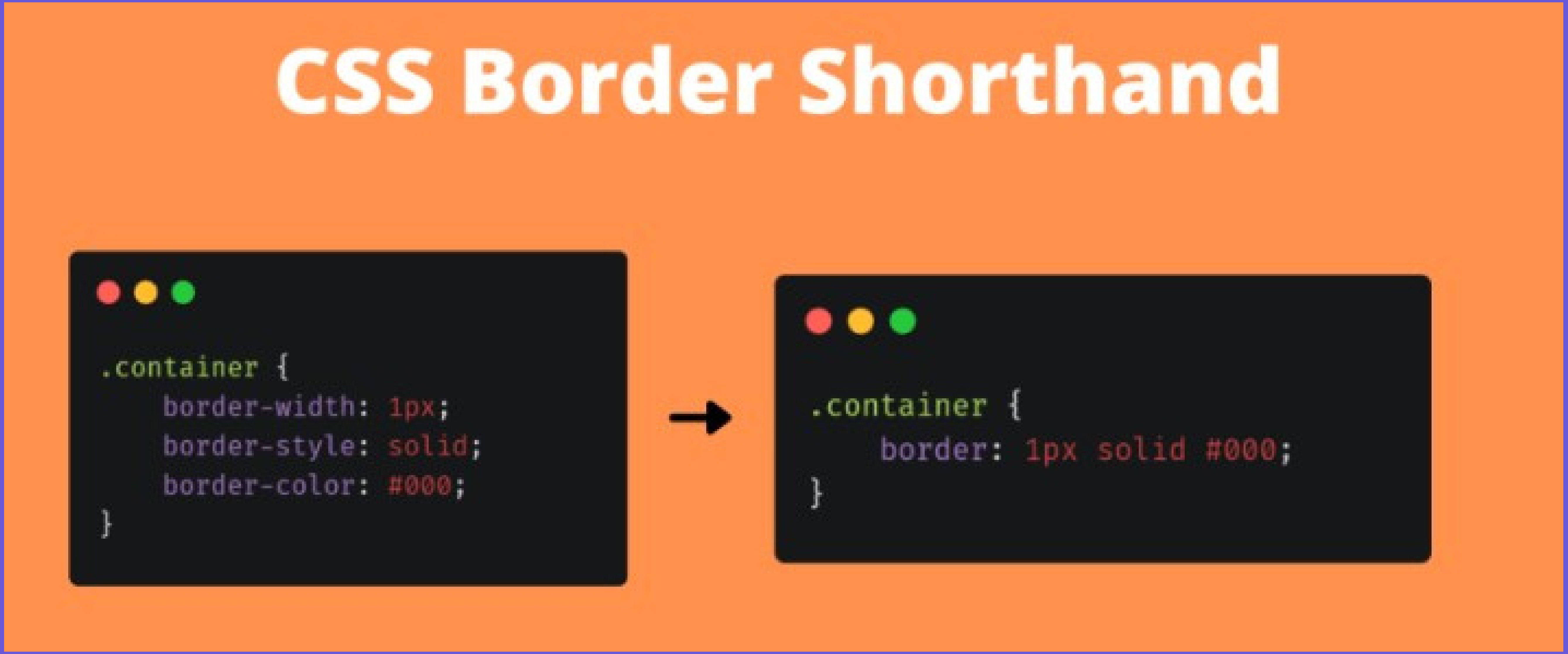
Border style defines how the border appears. Types of styles includes: solid, dashed,dotted,groove,outset,inset etc.

- **border-radius**

To create rounded corners, use border-radius. It softens the sharp edges of an element.

```
div{
    border:11px solid red;
    border-radius:25px;
 }
```

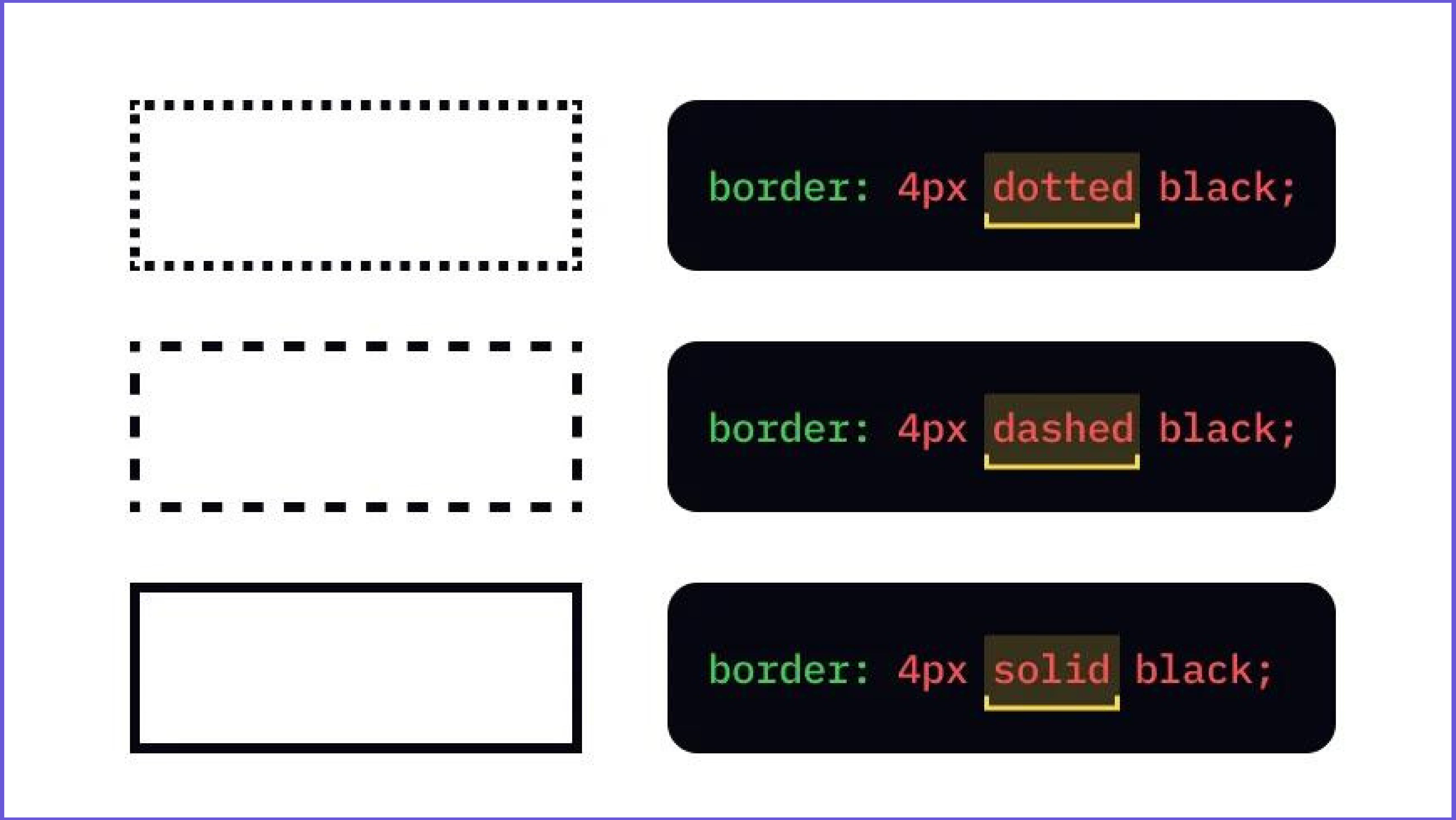
Border Shorthand Properties:



```css
.container {
    border-width: 1px;
    border-style: solid;
    border-color: #000;
}
```

```css
.container {
    border: 1px solid #000;
}
```

Border Example:



```
border: 4px dotted black;
```

```
border: 4px dashed black;
```

```
border: 4px solid black;
```

CSS Background Properties:

- background-color

Used to set color in the background.

```css
div{
    background-color:yellow;
}
```

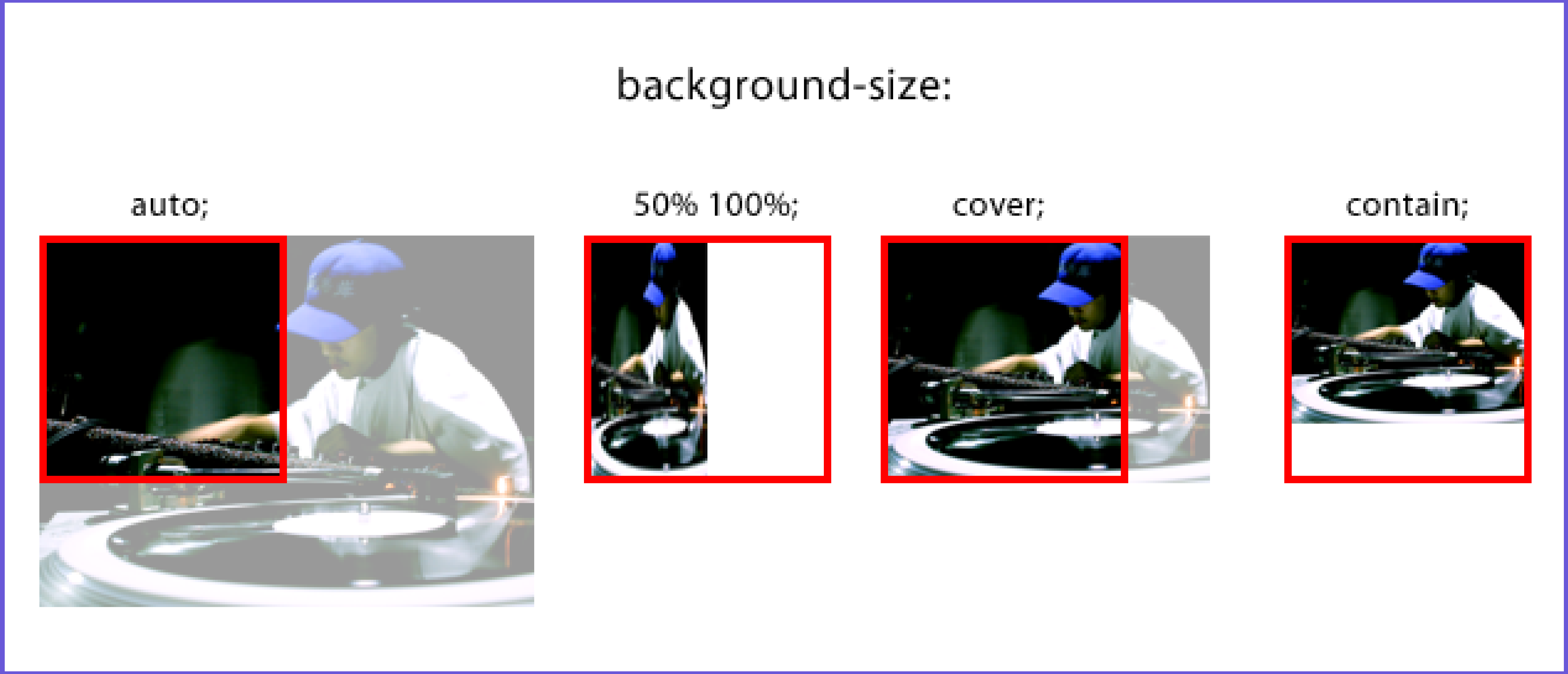
- background-image

Used to display image in the background.

```css
div{
  background-image:url(image.jpg);
}
```

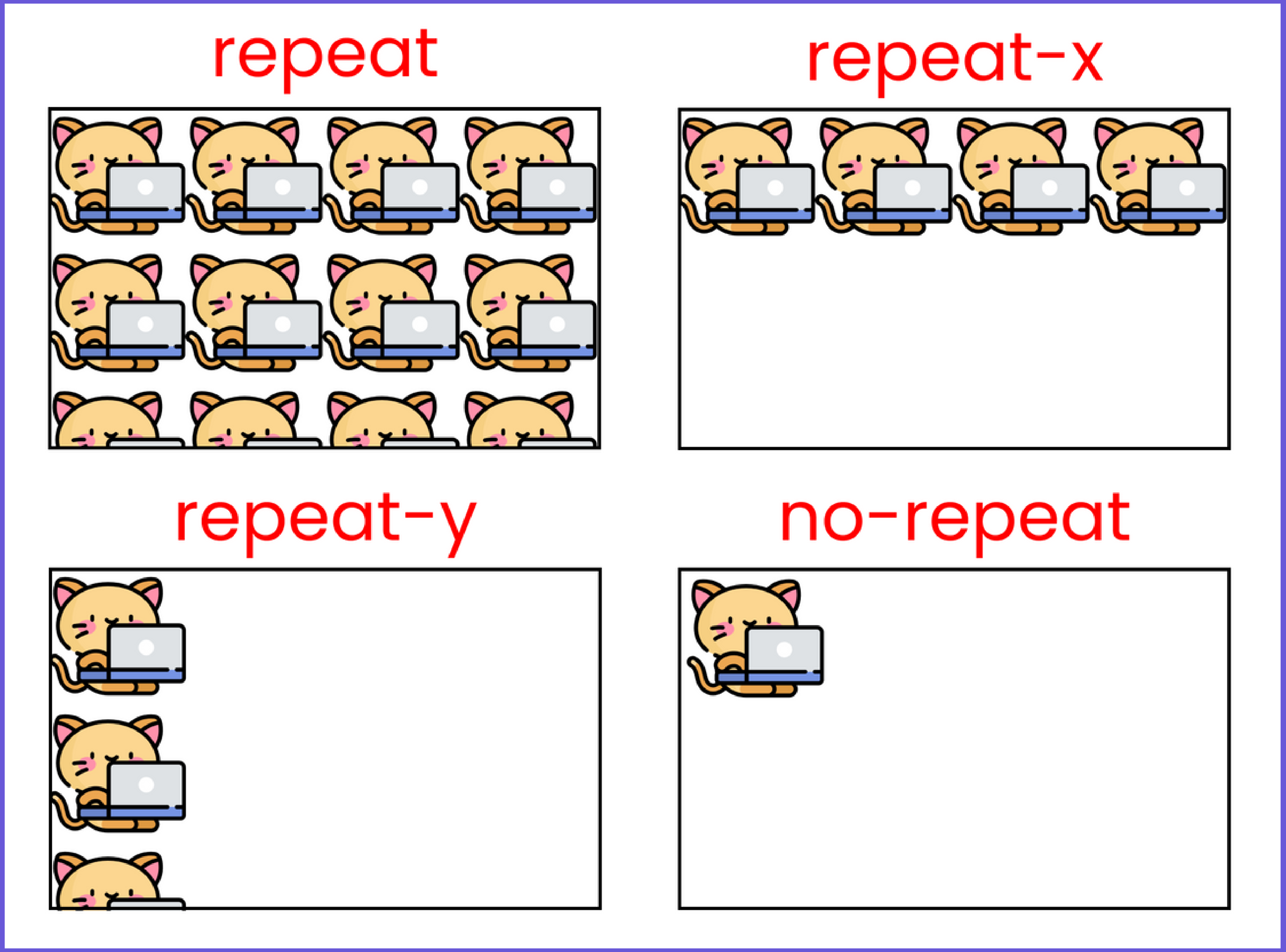
- **background-size**

This property specifies the size of the background image. We can set image size to cover/contain/auto or manually specify the size of background image



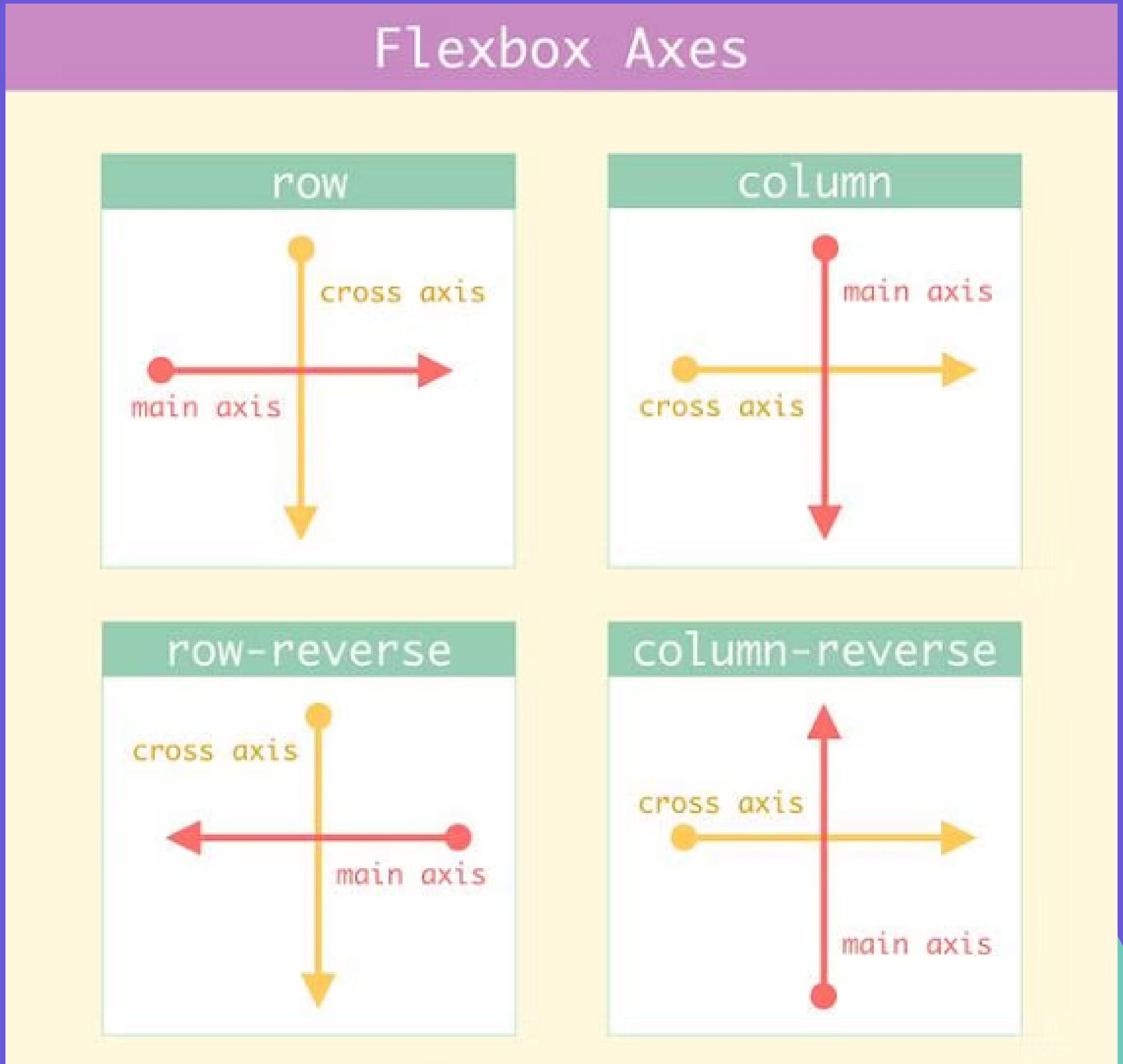
- background-repeat

This property determines how the background image repeats. We can set image size to repeat | repeat-x | repeat-y | no-repeat | initial (default).



CSS Flexbox

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.



Basics of Flex

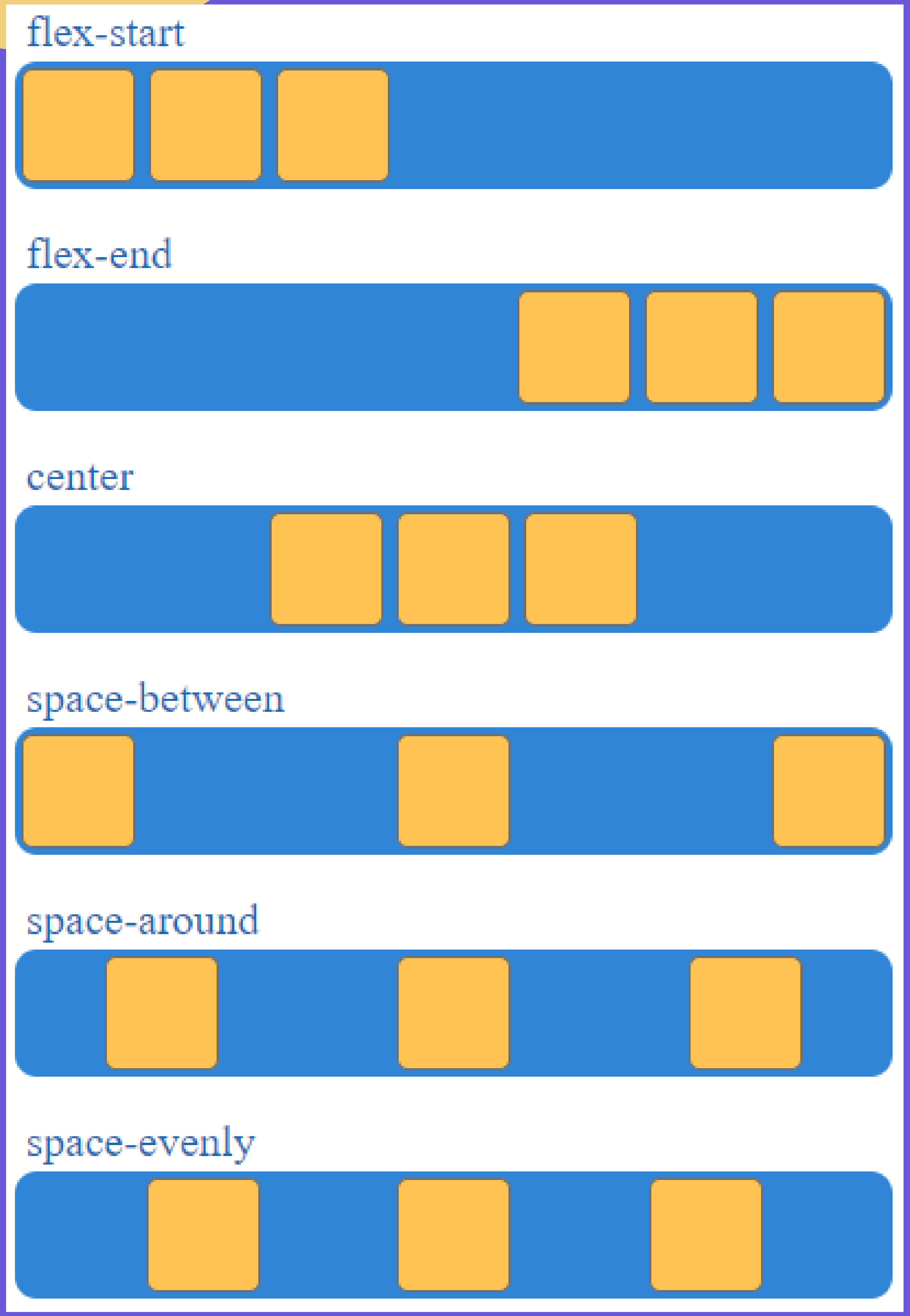
How to make items flex?

```
display : flex;
```

Basic commands in flex:

- flex-direction
- justify-content
- align-items

justify-content

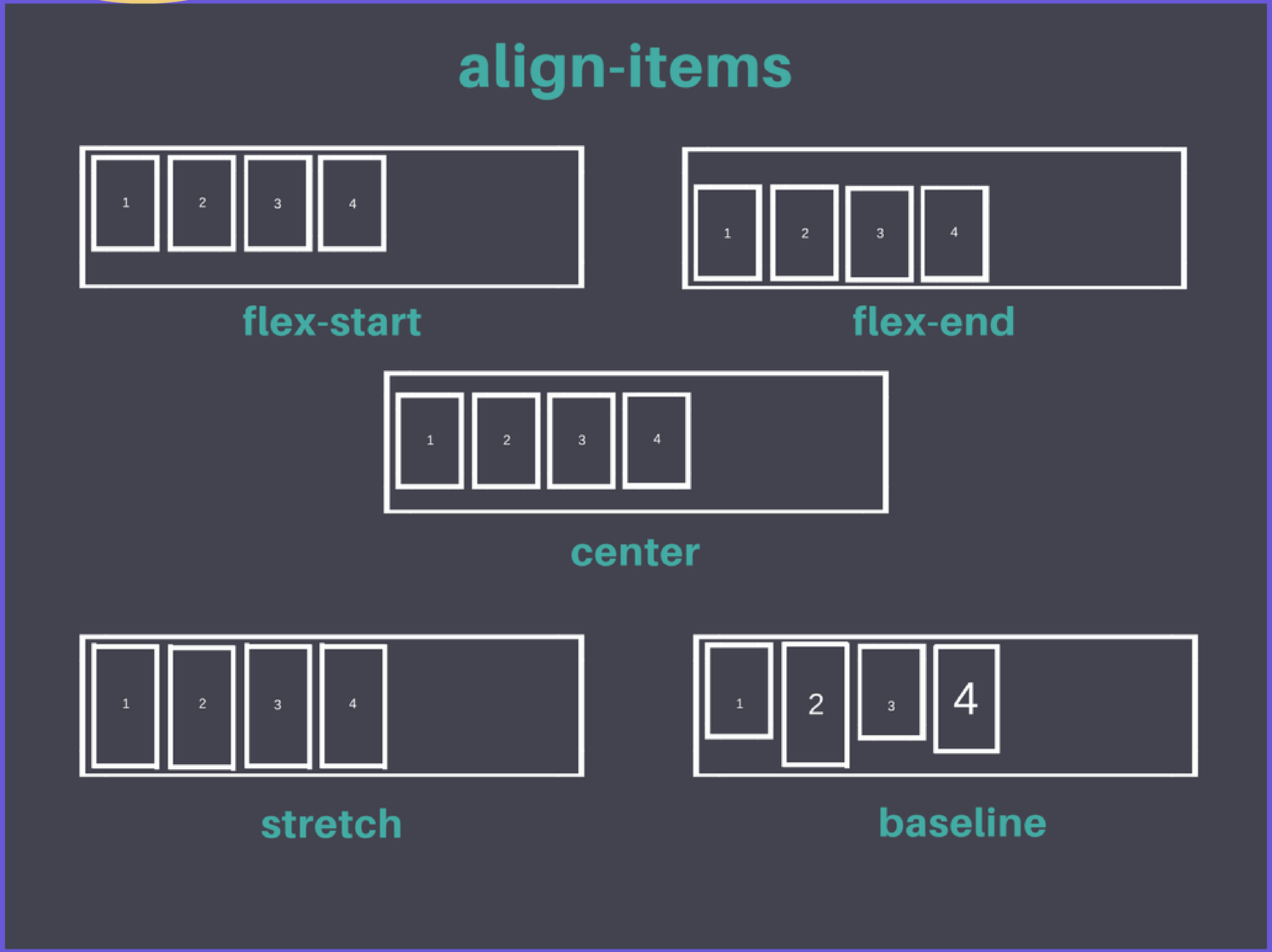


--> justify-content command moves the items that are across the main axis

1) flex-start
2) flex-end
3) center

a) space-between
b) space-around
c) space-evenly

align-items



--> align-items command moves the items that are across the cross axis

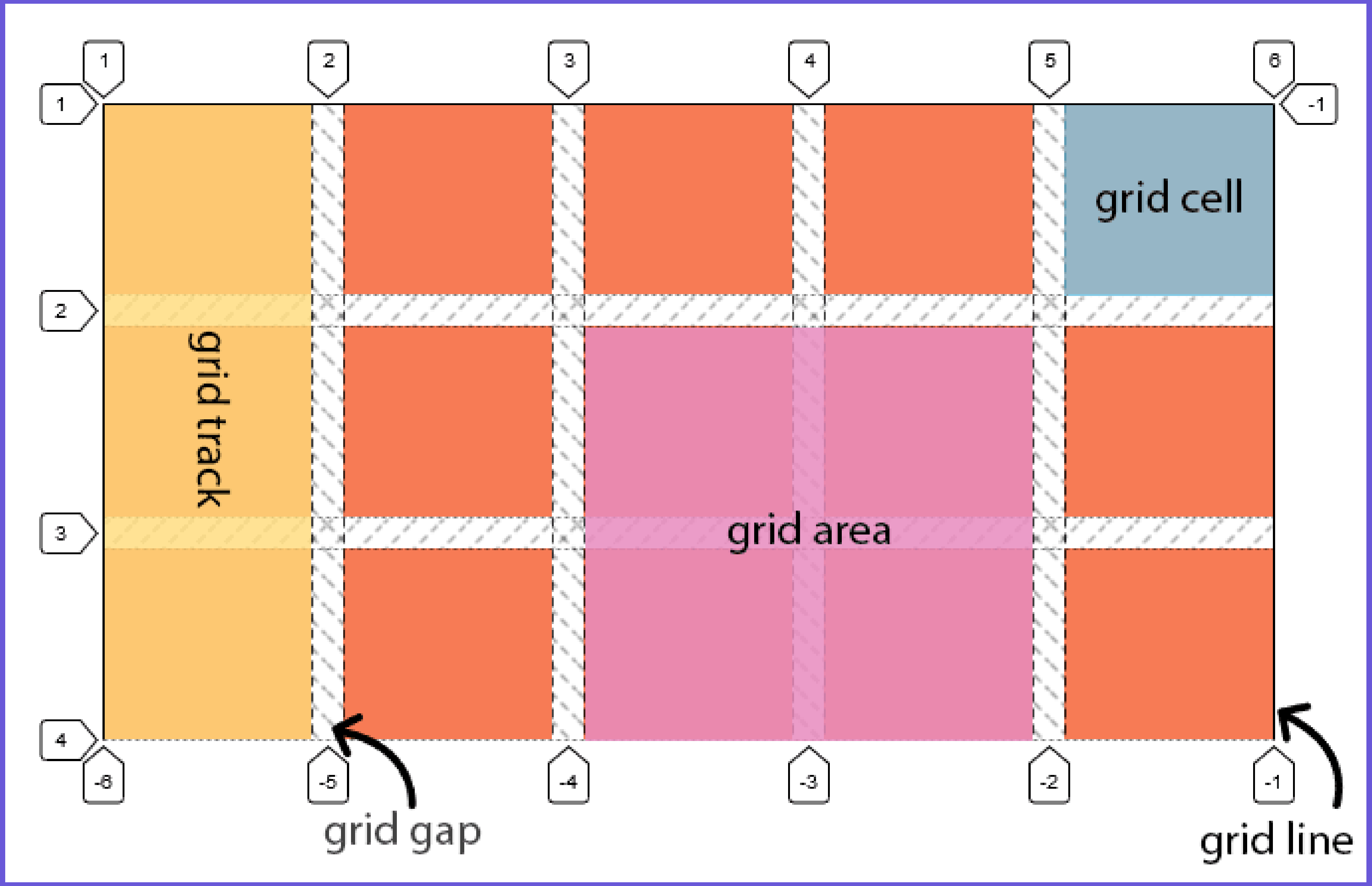
1) flex-start
2) flex-end
3) center

a) stretch
b) baseline

CSS Grid:

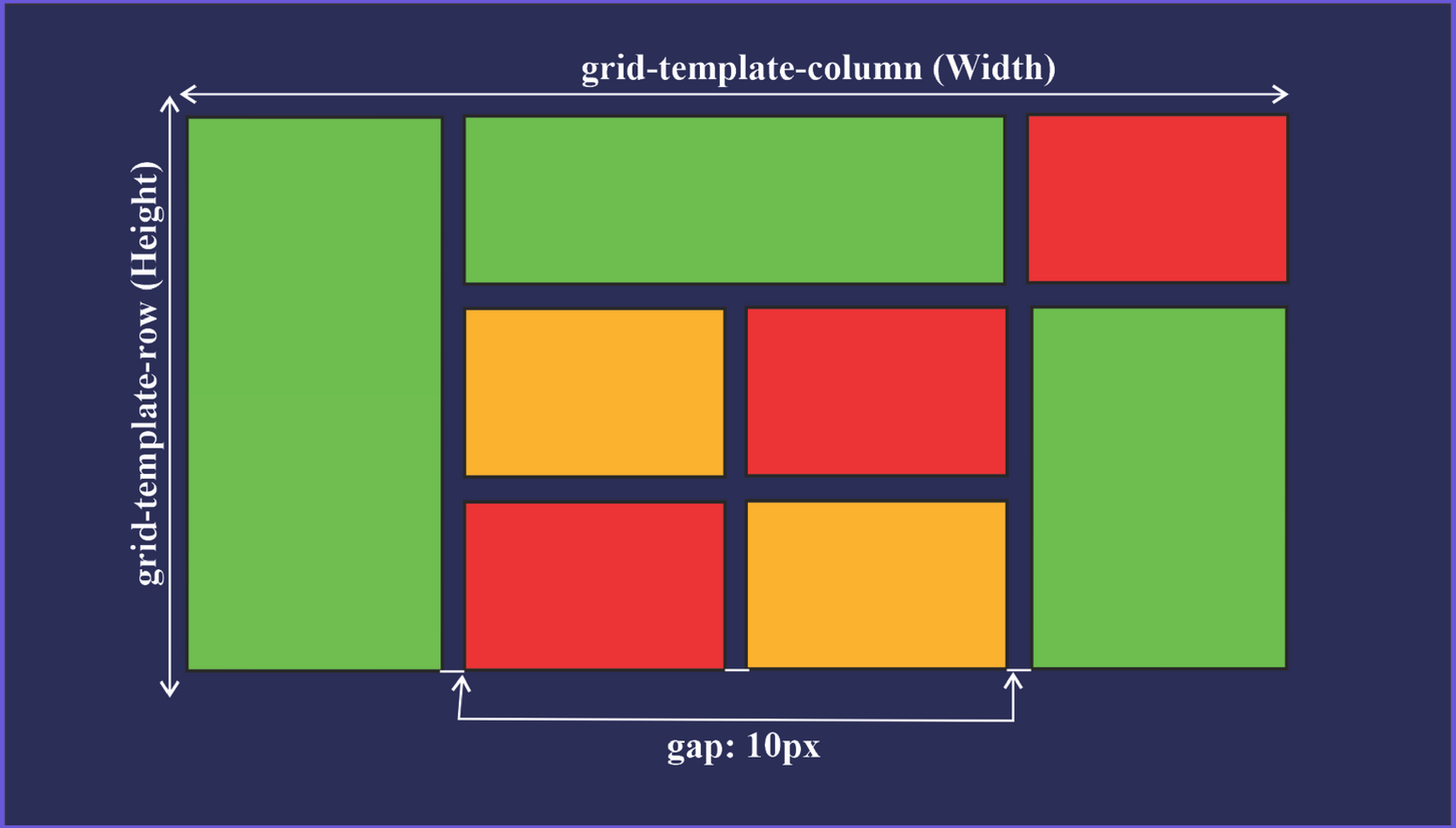
CSS Grid Layout is a powerful two-dimensional grid system for creating complex layouts with precision and flexibility.

Define Grid : To create a grid container, use `display: grid;` in CSS. This transforms the container's children into grid items.

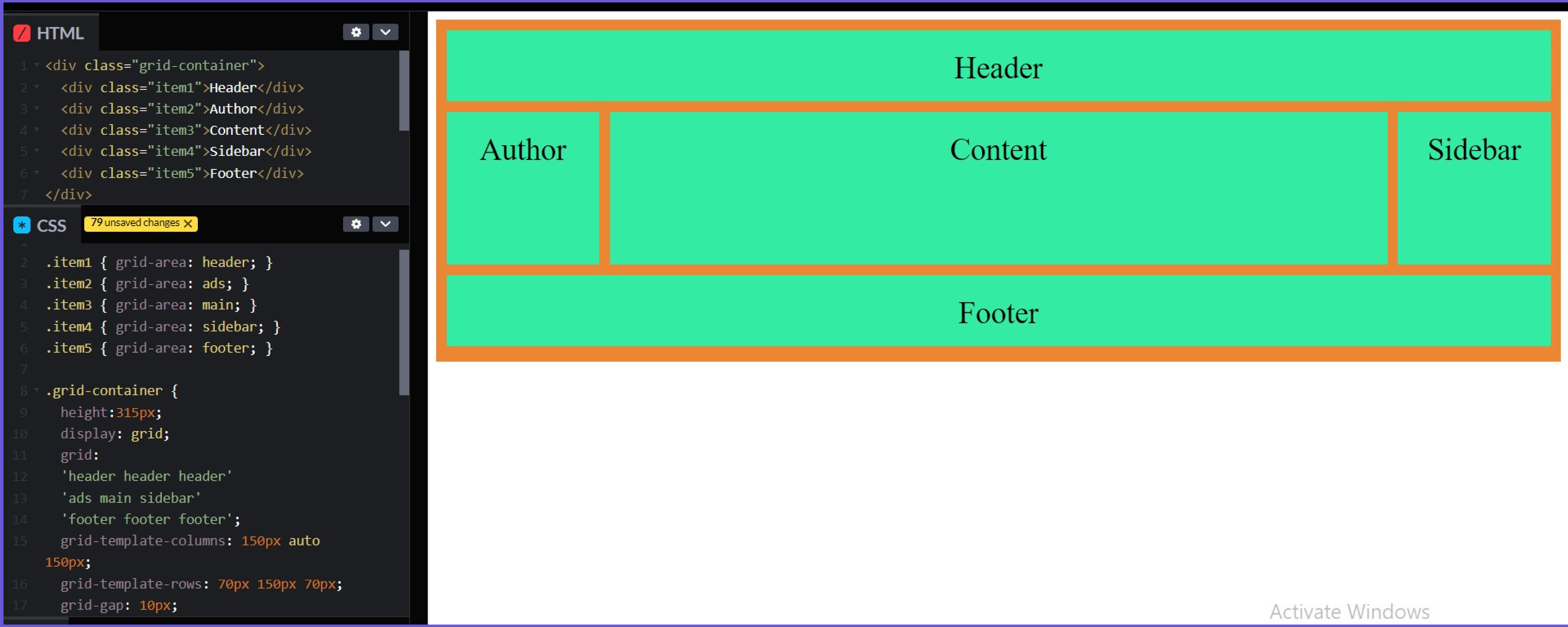


Grid Rows/Columns & Grid Gaps :

Utilize **grid-template-rows** and **grid-template-columns** to define the size and structure of your grid's rows and columns. Set the gap between grid items with **grid-gap**. It's a shorthand for specifying both row and column gaps.



Grid Example :



```html
<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Author</div>
  <div class="item3">Content</div>
  <div class="item4">Sidebar</div>
  <div class="item5">Footer</div>
</div>
```

```css
.item1 { grid-area: header; }
.item2 { grid-area: ads; }
.item3 { grid-area: main; }
.item4 { grid-area: sidebar; }
.item5 { grid-area: footer; }

.grid-container {
  height:315px;
  display: grid;
  grid:
  'header header header'
  'ads main sidebar'
  'footer footer footer';
  grid-template-columns: 150px auto
150px;
  grid-template-rows: 70px 150px 70px;
  grid-gap: 10px;
```

Display Property

As the name suggests, the display property of any HTML element decides how that element will be displayed in the web browser. There are following display properties in CSS:

- block
- inline
- inline-block
- none
- hidden
- flex
- grid

Block display

This value makes an element a block-level element, causing it to take up the full width of its parent container and start on a new line.
HTML elements like <div>, <h1>...<h6>, <section>, <footer>, <header>, <p> are block-level by default but we can override this default value.

Syntax:

```
element{
    display: block;
}
```

Inline display

This property makes an element an inline-level element. An inline-level element does not take the full width of its parent element and neither does it start on a new line. The height and width of an inline element cannot be set in CSS

Elements like <span>,<a> and <img> are inline-level by default but we can override this value.

Syntax:

```
element{
    display: inline;
}
```

Inline-block display

A fusion of Block and Inline properties. Inline-block elements takes the flow of the parent container and do not start on a new line (just like inline elements). But we can set the height and width of inline-block elements in CSS.

Syntax:

```
element{
    display: inline-block;
}
```

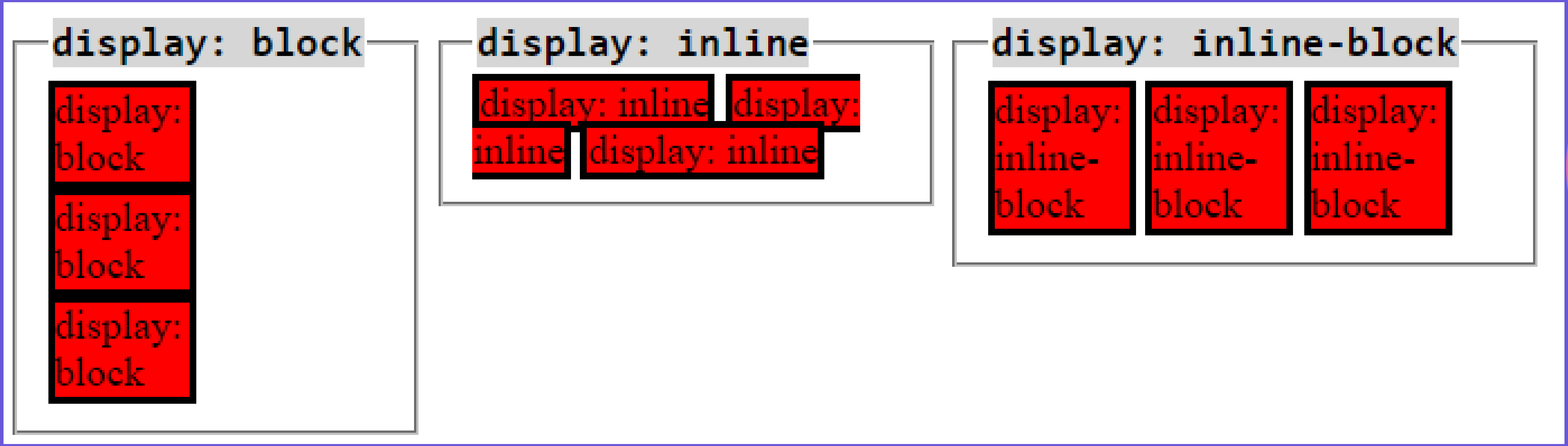
None display

This value hides the element completely, making it invisible and not taking up any space on the page layout.

Syntax:

```
element{
    display: none;
}
```

We must not confuse display: none with visibility: hidden.


display: block
display: block
display: block
display: block
display: inline
display: inline
display: inline
display: inline
display: inline-block
display: inline-block
display: inline-block
display: inline-block

Position property

The “position” property of CSS is used to specify the position of an element on the browser.
There are different positioning values:

- Static
- Relative
- Absolute
- Fixed

Let us learn more about them

Static positioning

All HTML elements are static by default. This is the normal flow of the HTML elements by default
Properties like top, bottom, left and right have no effect on static elements

Syntax:

```
element{
    position: static;
}
```

Relative positioning

Elements with relative positioning are positioned relative to their normal position in the document flow. We can use properties like ‘top’, ‘bottom’, ‘right’ and ‘left’ to decide the offset of the element relative to its normal position.

<u>Syntax:</u>

```
element:{
   position: relative;
   top: 100px;
   left: 10px;
}
```

Absolute positioning

An element with ‘position: absolute’ is positioned relative to its nearest ancestor element (an ancestor with a ‘position’ value other than ‘static’). If there is no such ancestor element, then it is positioned relative to the document body.

Syntax:

```
element{
  position: absolute;
  top: 50px;
  right: 60px;
}
```

Fixed positioning

Elements with ‘position: fixed’ are positioned relative to the viewport (browser window) and remain in the same position even when the page is scrolled.

Syntax:

```
element{
  position: fixed;
  top: 20px;
  left: 50px;
}
```

Float Property

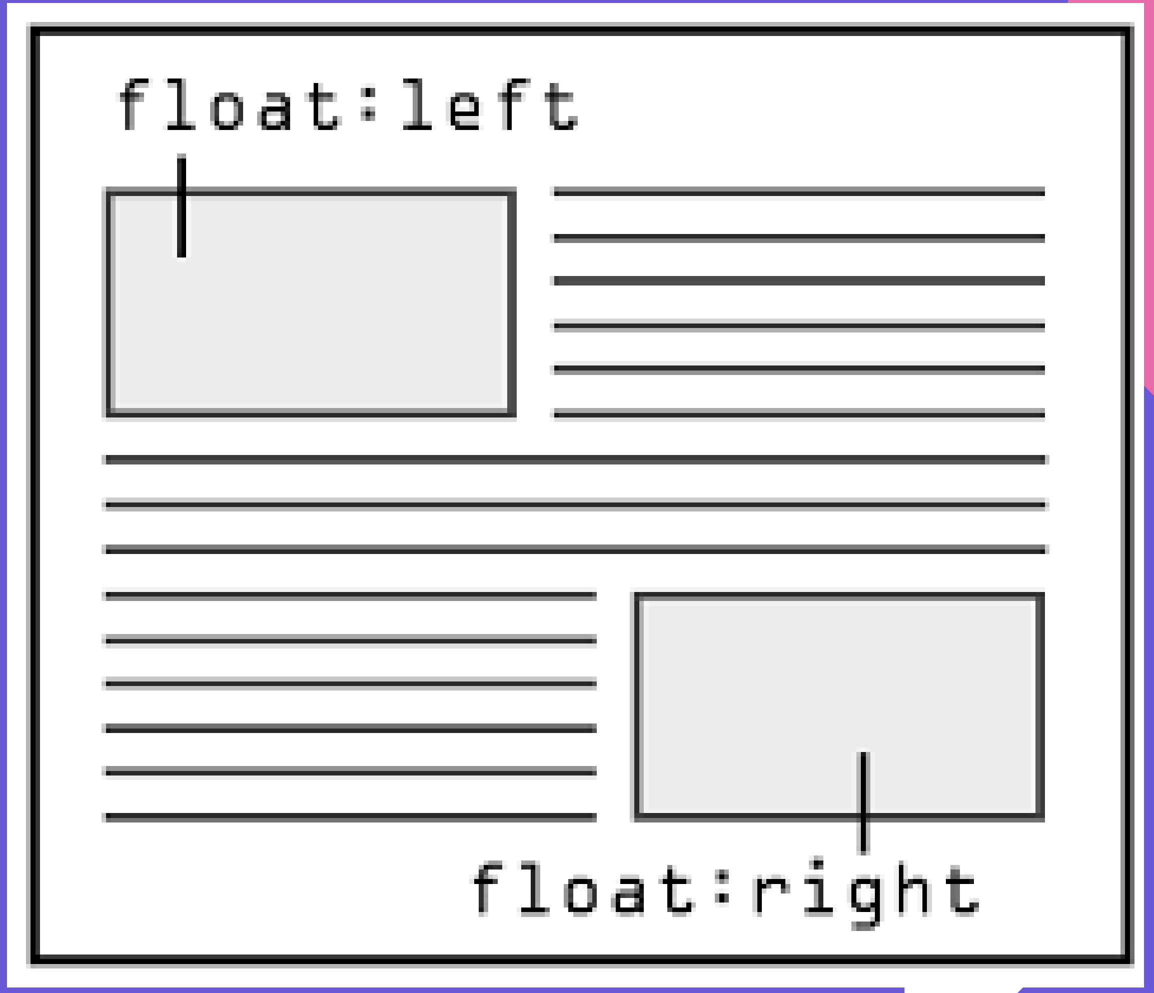
It specifies whether an element to left, right or not at all. Elements with ‘position: absolute’ do not get affected by the float property.

Elements next to a floating element will flow around it. Default value of float property is none

.

float: left element floats to the left of its container

float: right element floats to the right of its container



Transitions

CSS transitions are a way of changing the CSS properties of different CSS elements smoothly. To create a CSS transition, we must specify two things:

- the CSS property we want to get transitioned (like background, height, width, etc.)
- the duration of the transition. If the duration part is not specified, the transition will have no effect, because the default value is 0.
-

Let us learn about different transition properties.

Transition properties

- Transition property: It specifies the name of CSS property the transition is intended for
- Transition delay: It specifies the delay after which the transition will start (in seconds)
- Transition duration: It specifies the time that will be taken for the transition to complete
- Transition timing-function: It specifies the speed curve

Animations

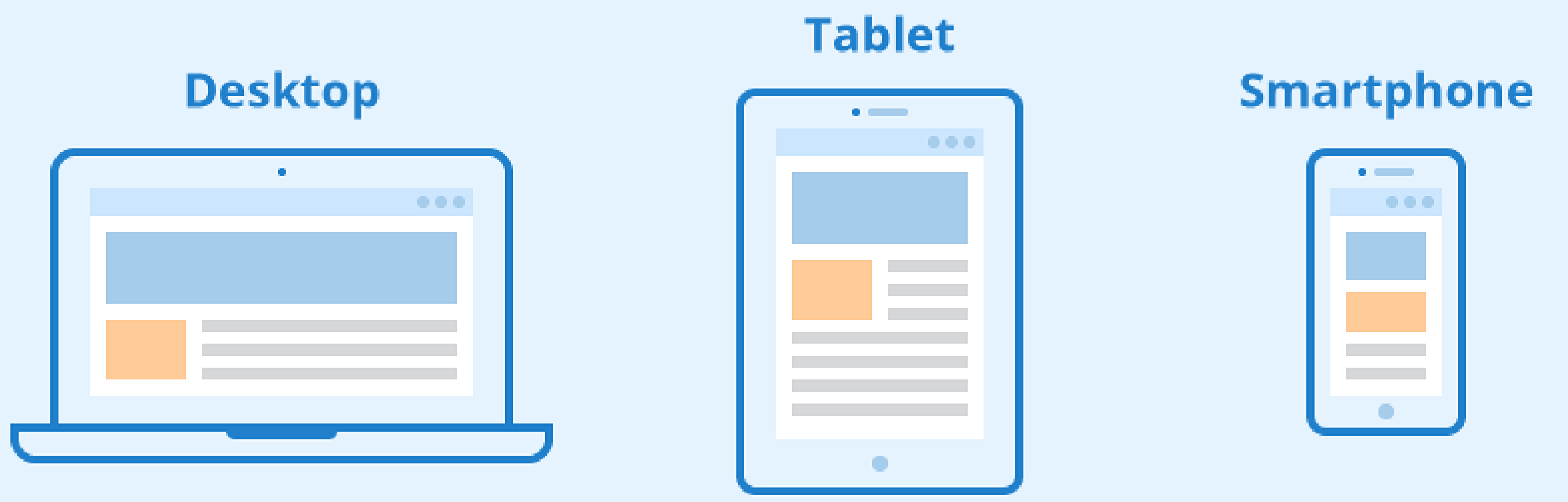
As the name suggests, these are used to add dynamic animations to our web page. To understand animations, we first need to understand keyframes.

<u>Keyframes:</u>
Think of keyframes as a set of instructions for the animation. You tell the browser what an element should look like at different points in time

Animation properties

- animation-name: specifies the name of the animation
- animation-duration: specifies how long the animation takes to complete
- animation-delay: specifies the delay before start of animation
- animation-iteration-count: specifies how many times the animation should run
- animation-timing-function: specifies the speed curve

Media Queries



```
@media screen and
(min-width: 1024px)
{...}
```

```
@media screen and
(min-width: 768px) and
(max-width: 1023px)
{...}
```

```
@media screen and
(max-width: 767px)
{...}
```

--> Media Queries are the methods we use to make the website responsive

--> “Responsive” ?

- A website is known to be responsive if it can be displayed on screens having different sizes

Media Queries

General syntax :

```
@media (){
}
```

--> Media Queries are more like if-else statements

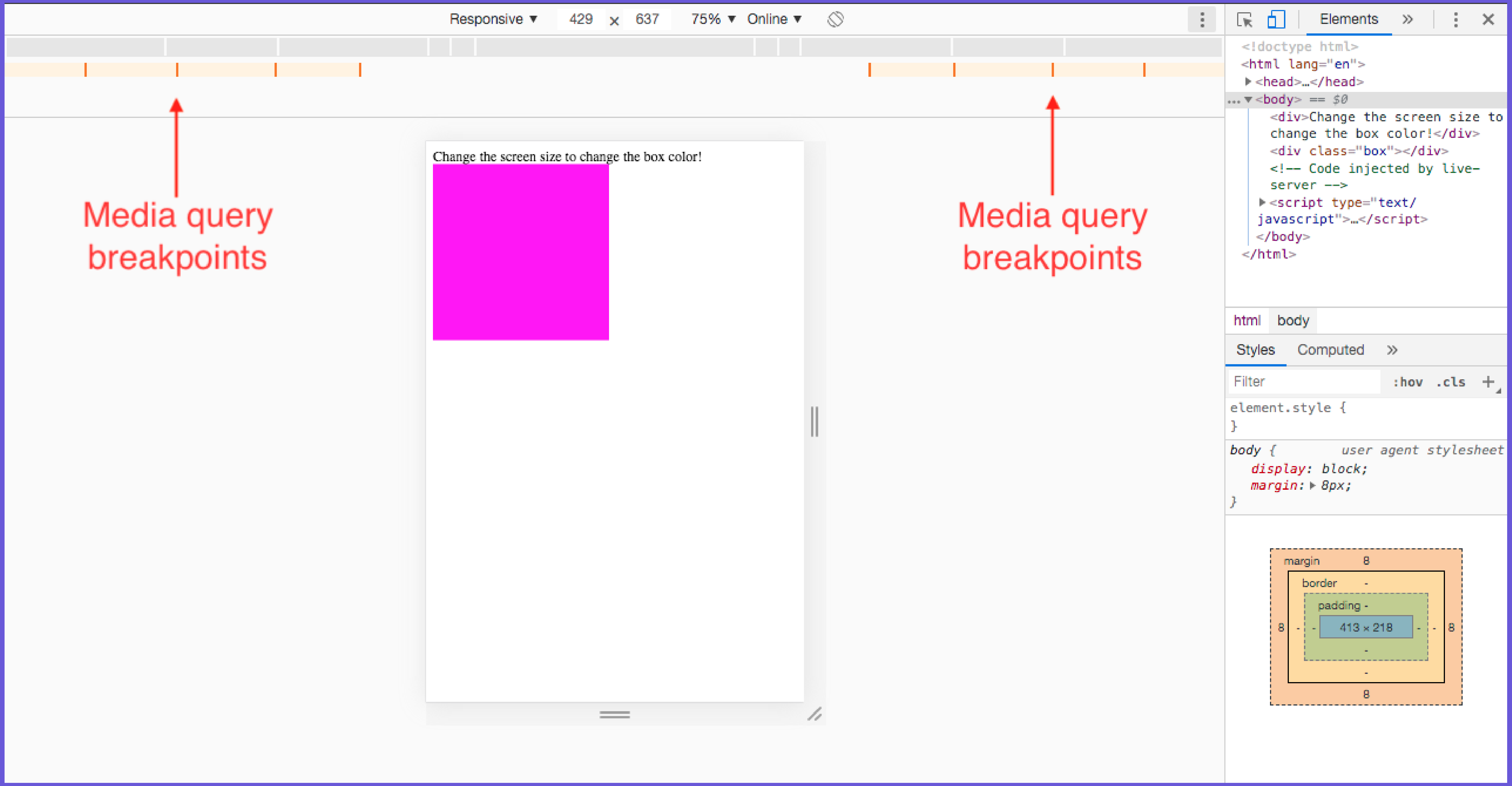
--> For example consider:

```
@media( width <=400){
background-color: green;
```

Working method:

--> In the above example, the background color changes to green only id the website width is less than 400px

Media Queries



Change the screen size to change the box color!

```
<!doctype html>
<html lang="en">
 ▸<head>…</head>
…▾<body> == $0
    <div>Change the screen size to
    change the box color!</div>
    <div class="box"></div>
    <!-- Code injected by live-
    server -->
   ▸<script type="text/
   javascript">…</script>
  </body>
</html>
```

```
element.style {
}
body {                user agent stylesheet
   display: block;
   margin: ▸ 8px;
}
```

--> This is how the media query breakpoints are shown when we check different dimensions of the screen size

Resources for CSS:

- W3 schools: https://www.w3schools.com/
- MDN reference: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference