Cascading Style Sheet(CSS)

Styling your Web Page

What is CSS

- CSS is acronym for Cascading Style Sheet
- It is a language that describes the style of an HTML document.
- It is about how elements of HTML should be displayed.
- CSS **saves a lot of work**.
- It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**

An Example of CSS

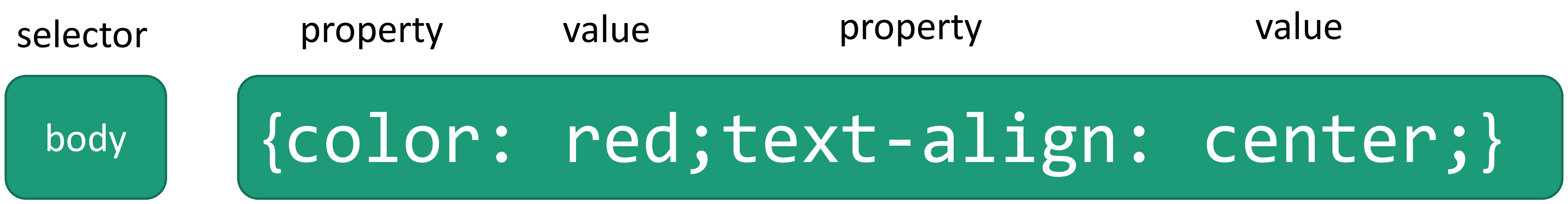
```css
body {
  background-color: red;
}

h1 {
  color: blue;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

All HTML body elements will be RED in colour

All h1 elements will be BLUE in colour and centre aligned

All <p> elements will have this STYLE

CSS Syntax



```
body {color: red;text-align: center;}
```

- The Selector identifies the element or block where the style needs to be applied.
- Each CCS style is written as ‘**Key=Value**’ entry, separated by ‘**colone**’.
- The key is a predefined property.
- The value is the style to be applied.

CSS Selectors

- Simple selectors
  - select elements based on name, id, class
- Combinator selectors
  - select elements based on a specific relationship between them
- Pseudo-class selectors
  - select elements based on a certain state
- Pseudo-elements selectors
  - select and style a part of an element
- Attribute selectors
  - select elements based on an attribute or attribute value

The CSS element Selector

```css
p {
  text-align: center;
  color: blue;
}
```

All <p> elements on the page will be center-aligned, with a blue text color

Id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#note {
  text-align: left;
  color: blue;
}
```

```
<p id=“note">This is a note</p>
```

Class Selector

- The class selector selects the class attribute in HTML elements.
- To select elements with a specific class, we write a period (.) character, followed by name of the class.

```
.right {
  text-align: right;
  color: blue;
}
```

```
<h1 class=“right">Blue heading aligned right</h1>
<p class=“right"> Blue paragraph aligned right</p>
```

The class selector can use more than one class

```
<p class=“right large">This paragraph refers to two classes.</p>
```

Universal Selector

- The universal selector (*) selects all HTML elements on the page.

```
* {
  text-align: center;
  color: blue;
}
```

We will see a demo on this

Grouping Selector

```css
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

```css
h1, h2, p {
  text-align: center;
  color: red;
}
```

```html
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
```

Individual elements will get their own style

How to Insert or add CSS?

- There are three ways of inserting a style sheet to your HTML document:
  - External CSS
  - Internal CSS
  - Inline CSS

Cascading Order

- All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style
    1. inside an HTML element
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

Some important CSS Style settings

- Colours
- Background
- Margins
- Paddings
- Text/Font
- Height/Width
- Position
- Tables
- Overflow
- Float
- Display
- Lists
- Outline
- Align

CSS

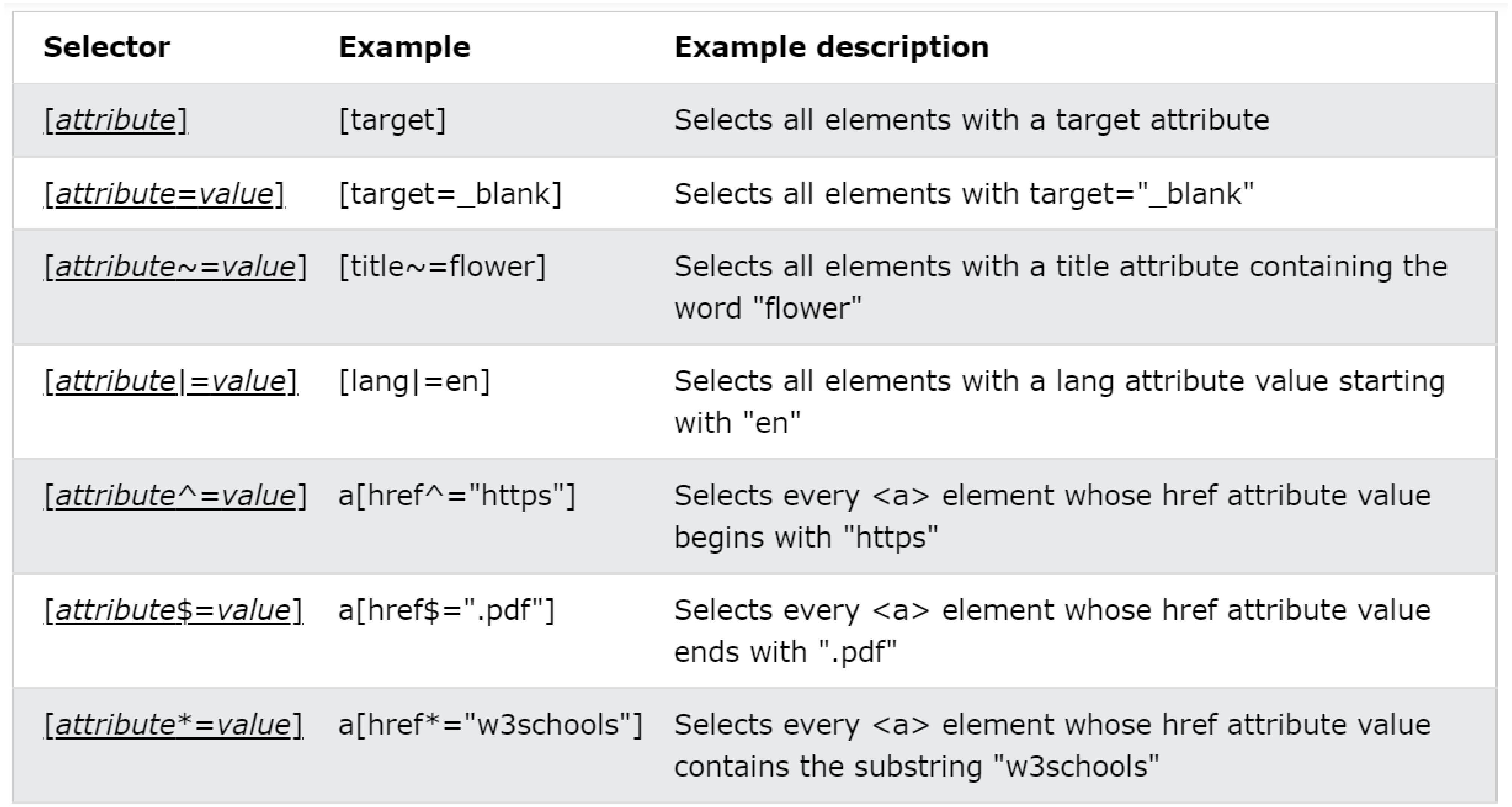
Attribute selectors

CSS: attribute selectors

- The CSS attribute selector matches elements based on the presence or value of a given attribute.

```
/* <a> elements with a title attribute */
 a[title] {
     color: purple;
 }


 /* <a> elements with an href matching "https://example.org" */
 a[href="https://example.org"] {
     color: green;
 }
```

CSS attribute selectors



Selector	Example	Example description
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[*attribute$=value*]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[*attribute*=value*]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"

Image courtesy: https://www.w3schools.com/css/css_attribute_selectors.asp

CSS

Pseudo classes

What are Pseudo classes?

- A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s).
- For example, :hover can be used to change a button's color when the user's pointer hovers over it.

```
  /* Any button over which the user's pointer is hovering */

button:hover {
  color: blue;
}
```

Pseudo Classes

Dynamic Pseudo classes

- :link
- :visited
- :hover
- :active
- :focus

The UI element states pseudo-classes

- :enabled
- :disabled
- :checked

target Pseudo classes

- :target

Language Pseudo Classes

- :lang

Negation Pseudo Classes

- :not

Structural pseudo-classes

- :root
- :nth-child
- :nth-last-child
- :nth-of-type
- :nth-last-of-type
- :first-child
- :last-child
- :first-of-type
- :last-of-type
- :only-child
- :only-of-type
- :empty

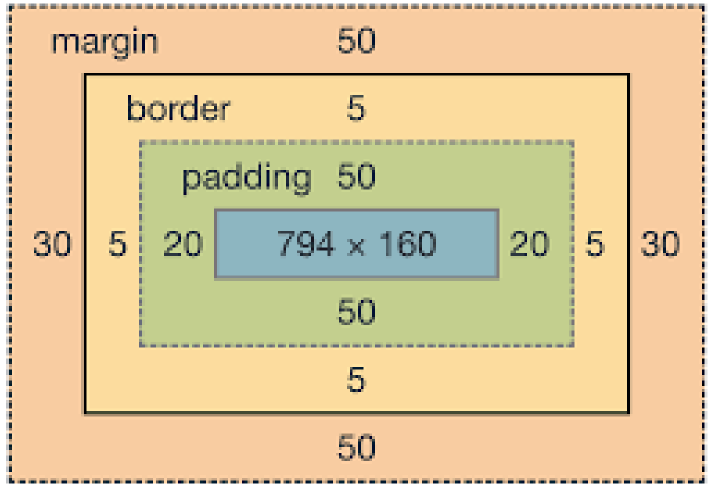
CSS

box-sizing

box-sizing

- The box-sizing CSS property sets how the total width and height of an element is calculated.
- generally we create elements with a fixed width and height and then add border and padding to it which increases the set size of the element box.
- box-sizing helps maintain the size even after padding and border are set

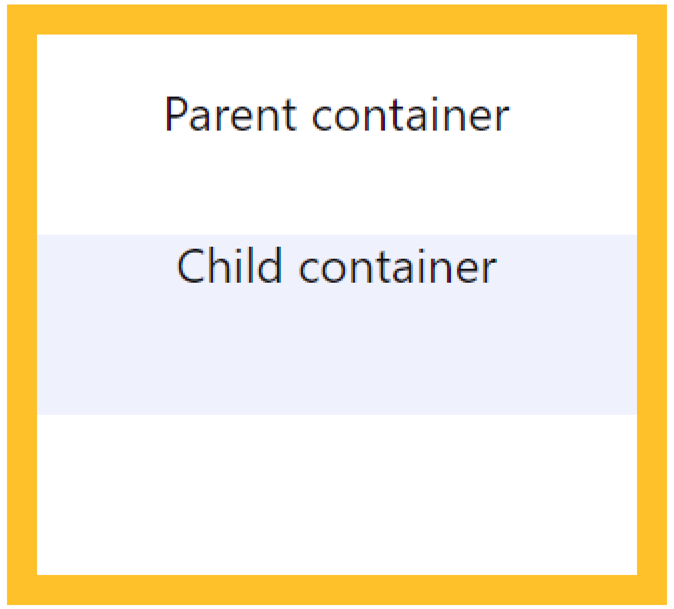
box-sizing: border-box;
box-sizing: content-box;



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

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

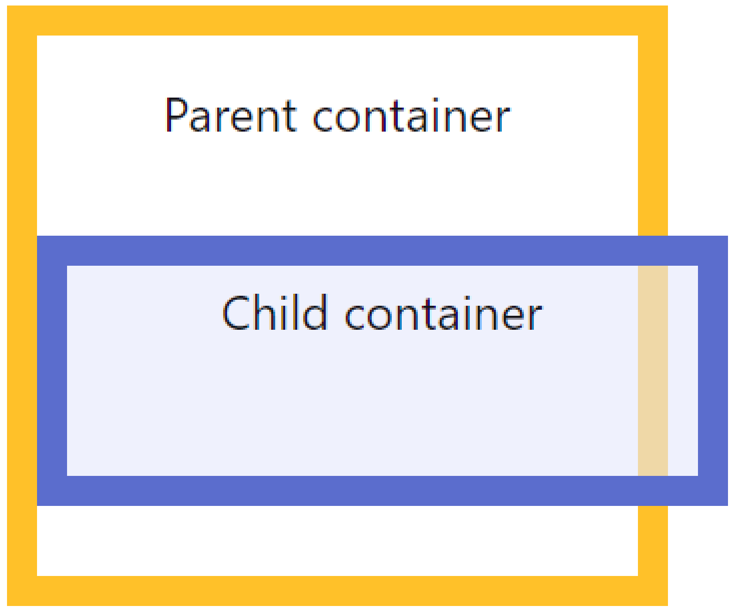
```
box-sizing: border-box;
width: 100%;
border: solid #5B6DCD 10px;
padding: 5px;
```



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

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

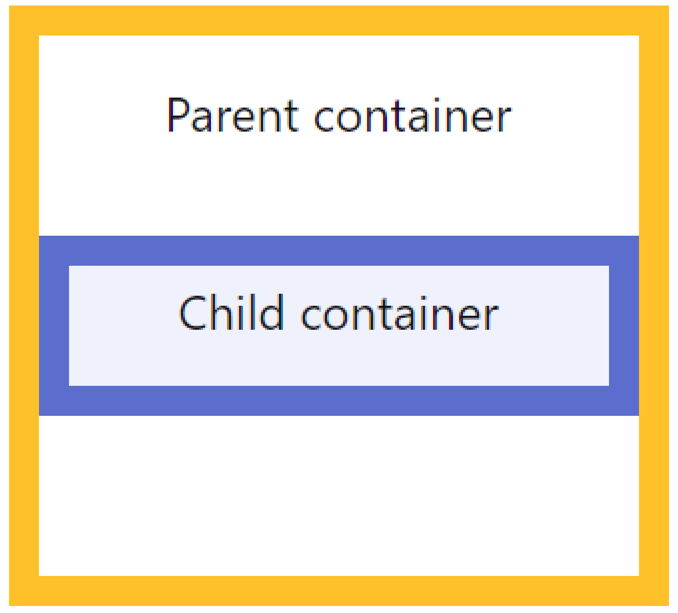
```
box-sizing: border-box;
width: 100%;
border: solid #5B6DCD 10px;
padding: 5px;
```



```
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;
```



Float and Clear

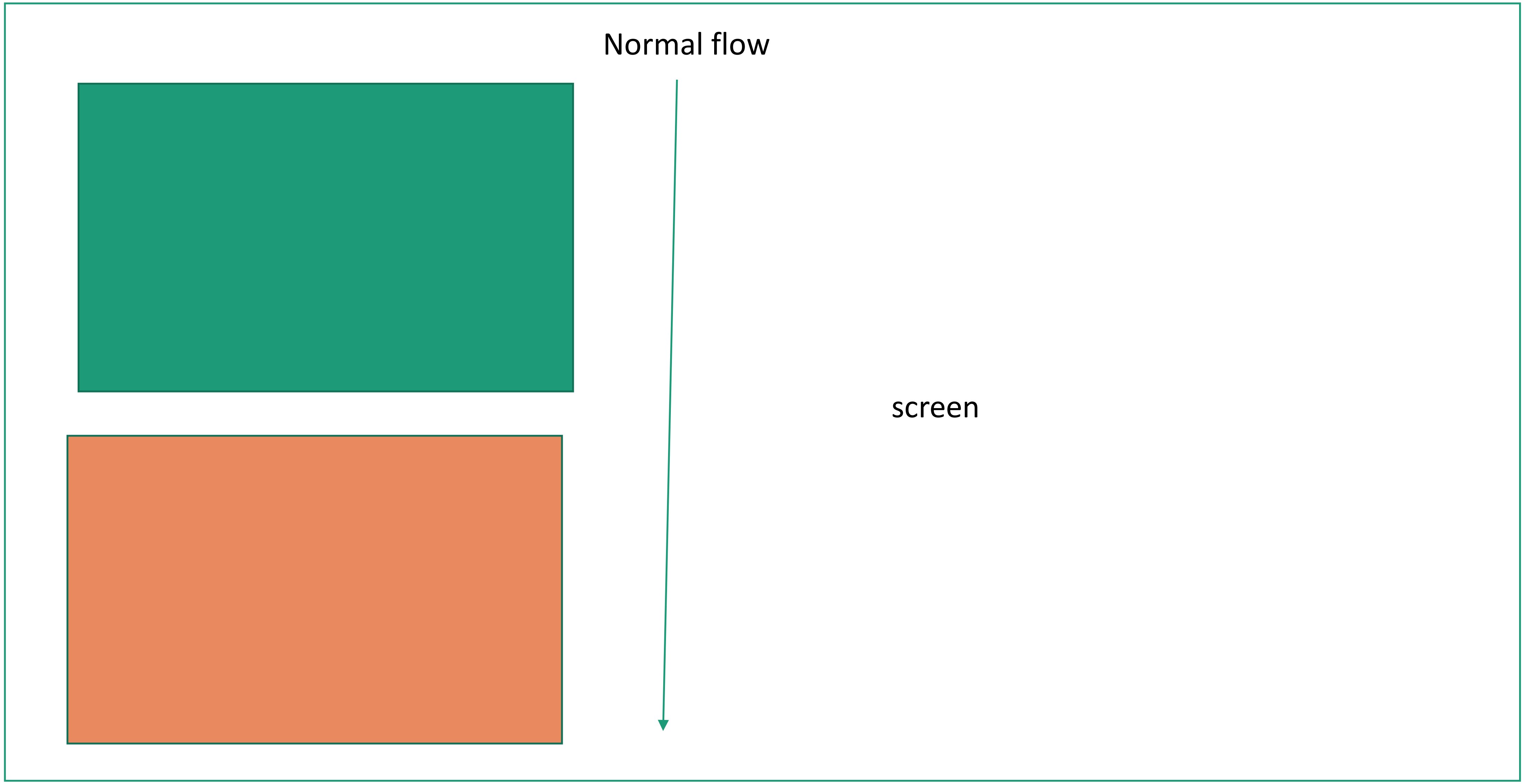
CSS Float

- 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

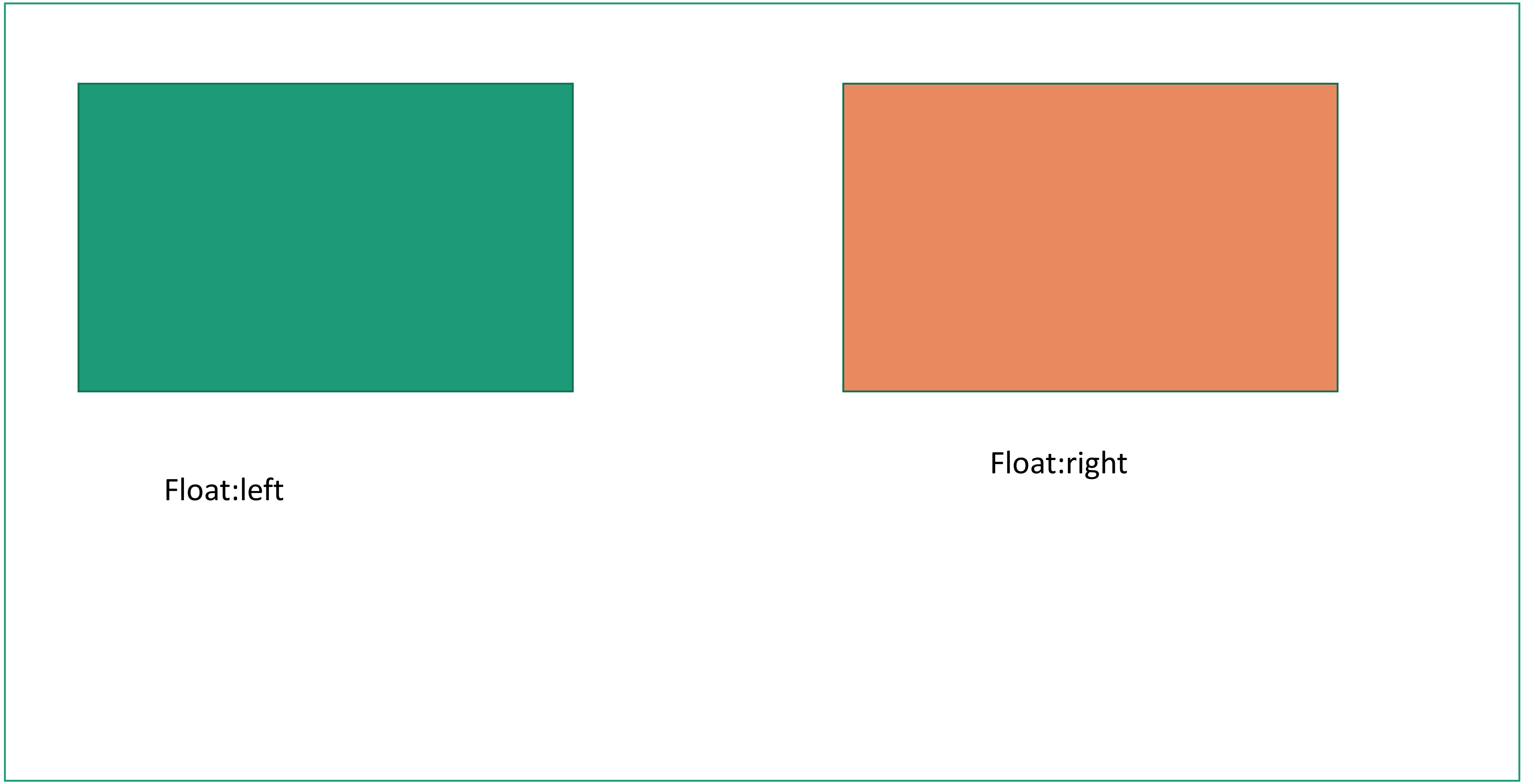
Float Usage

```
float: left;
float: right;
float: none;
float: inline-start;
float: inline-end;
```

CSS Float Property



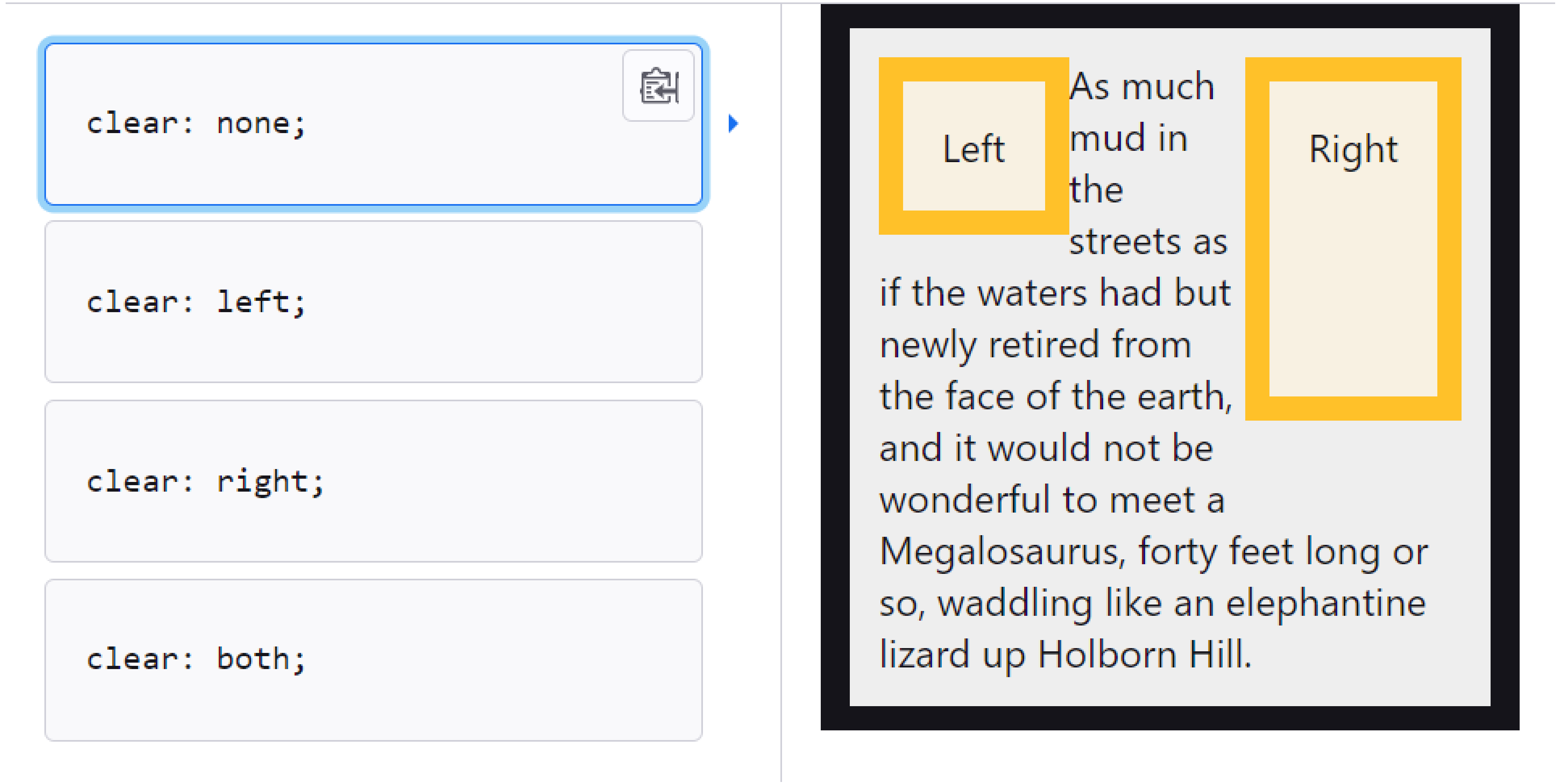
CSS Float Property



clear

- The clear CSS property sets whether an element must be moved below (cleared) floating elements that precede it.
- The clear property applies to floating and non-floating elements.

```
clear: none;
clear: left;
clear: right;
clear: both;
clear: inline-start;
clear: inline-end;
```



```
clear: none;
```

```
clear: left;
```

```
clear: right;
```

```
clear: both;
```

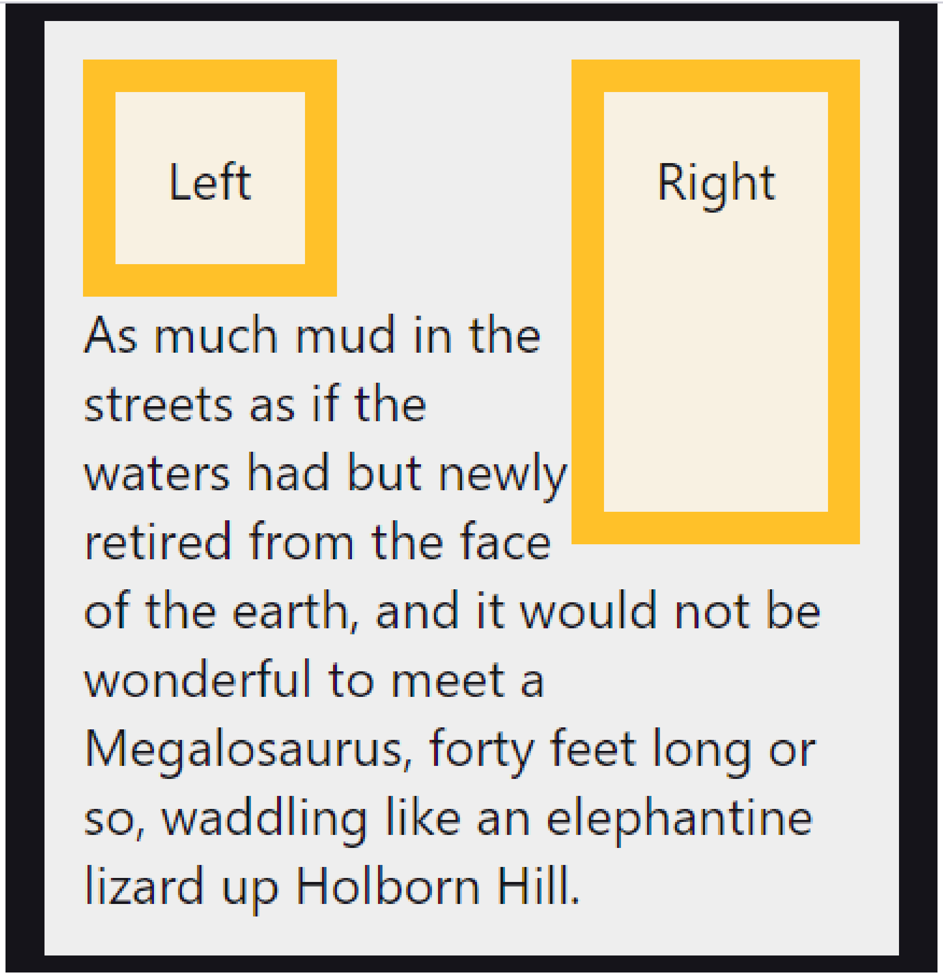
As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

```
clear: none;
```

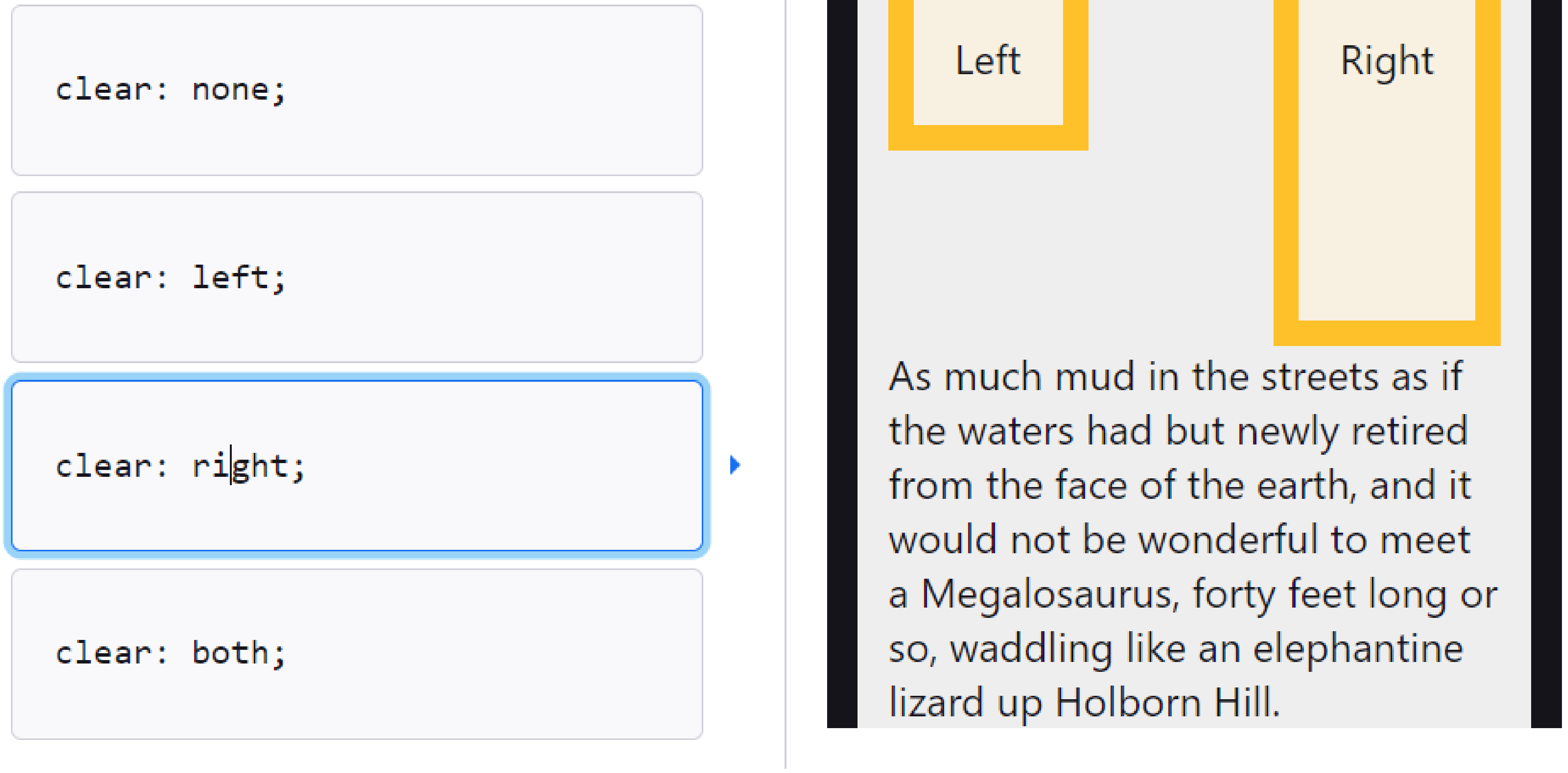
```
clear: left;
```

```
clear: right;
```

```
clear: both;
```



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.



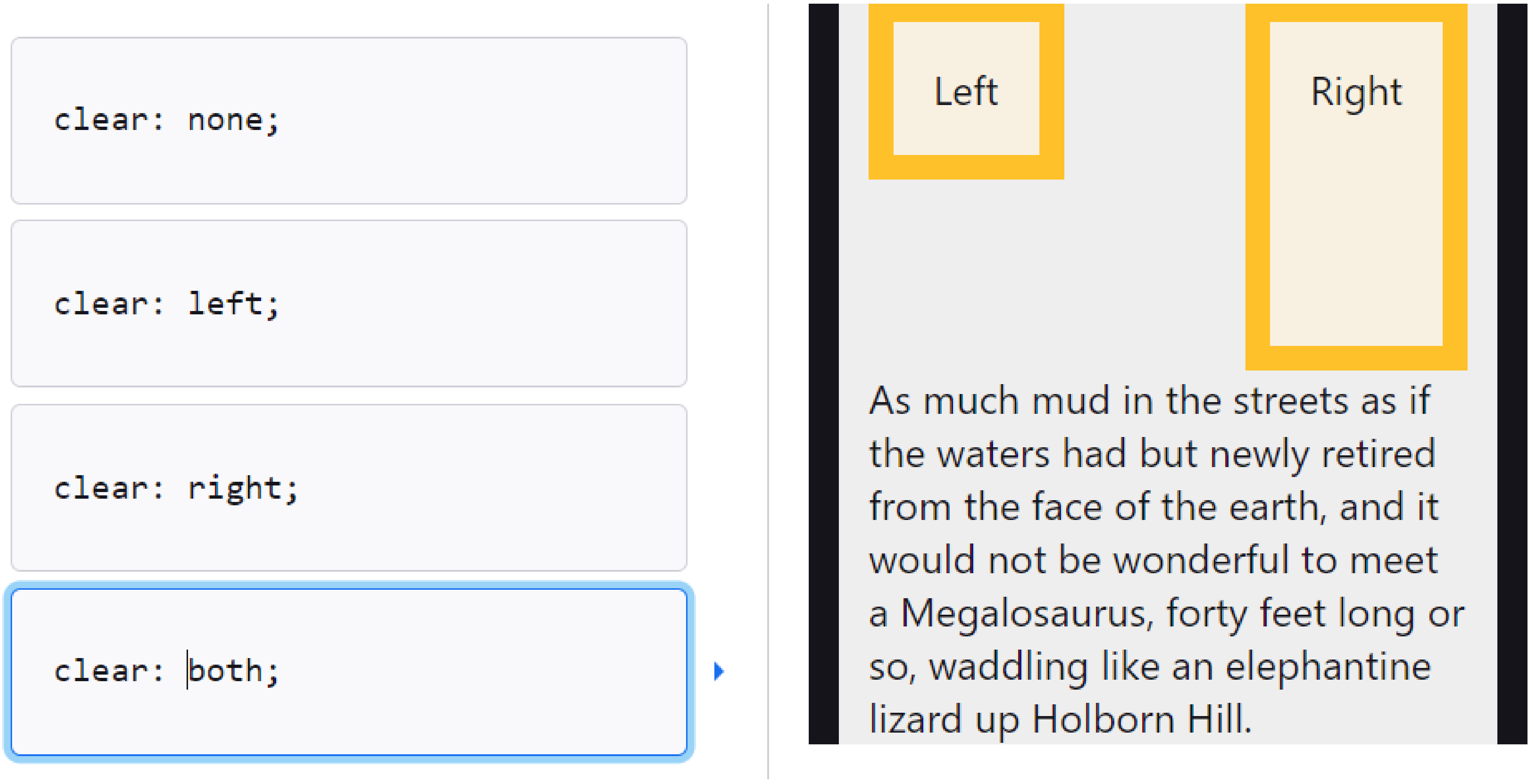
```
clear: none;
```

```
clear: left;
```

```
clear: right;
```

```
clear: both;
```

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.



```
clear: none;
```

```
clear: left;
```

```
clear: right;
```

```
clear: both;
```

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.

CSS

position

position property

- 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.

```
position: static;
position: relative;
position: absolute;
position: fixed;
position: sticky;
```

Associated Properties

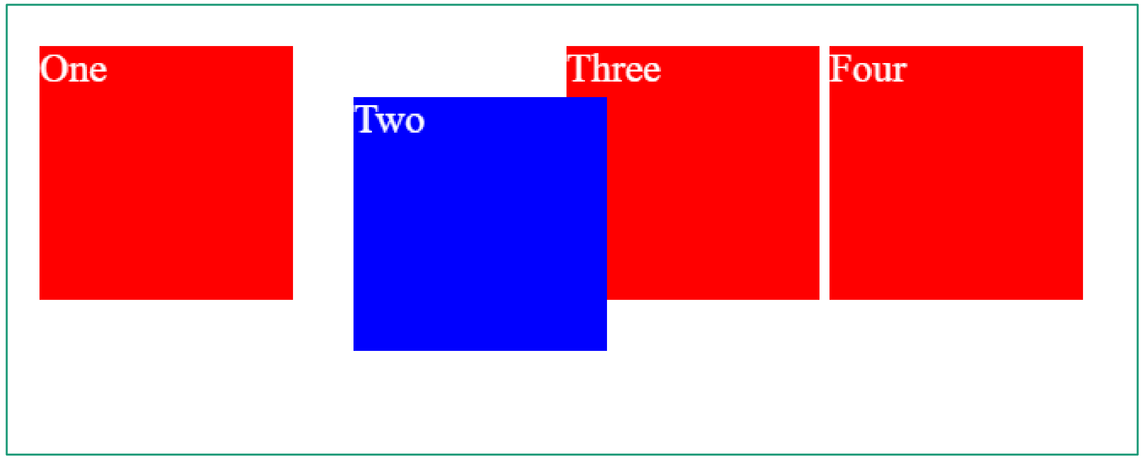
left
top

right
bottom

position: An example

```
* {
  box-sizing: border-box;
}

.box {
  display: inline-block;
  width: 100px;
  height: 100px;
  background: red;
  color: white;
}

#two {
  position: relative;
  top: 20px;
  left: 20px;
  background: blue;
}
```

```
<div class="box" id="one">One</div>
<div class="box" id="two">Two</div>
<div class="box" id="three">Three</div>
<div class="box" id="four">Four</div>
```



CSS

z-index

z-index

- The z-index CSS property sets the z-order of a positioned element and its descendants or flex items.
- Overlapping elements with a larger z-index cover those with a smaller one.

CSS

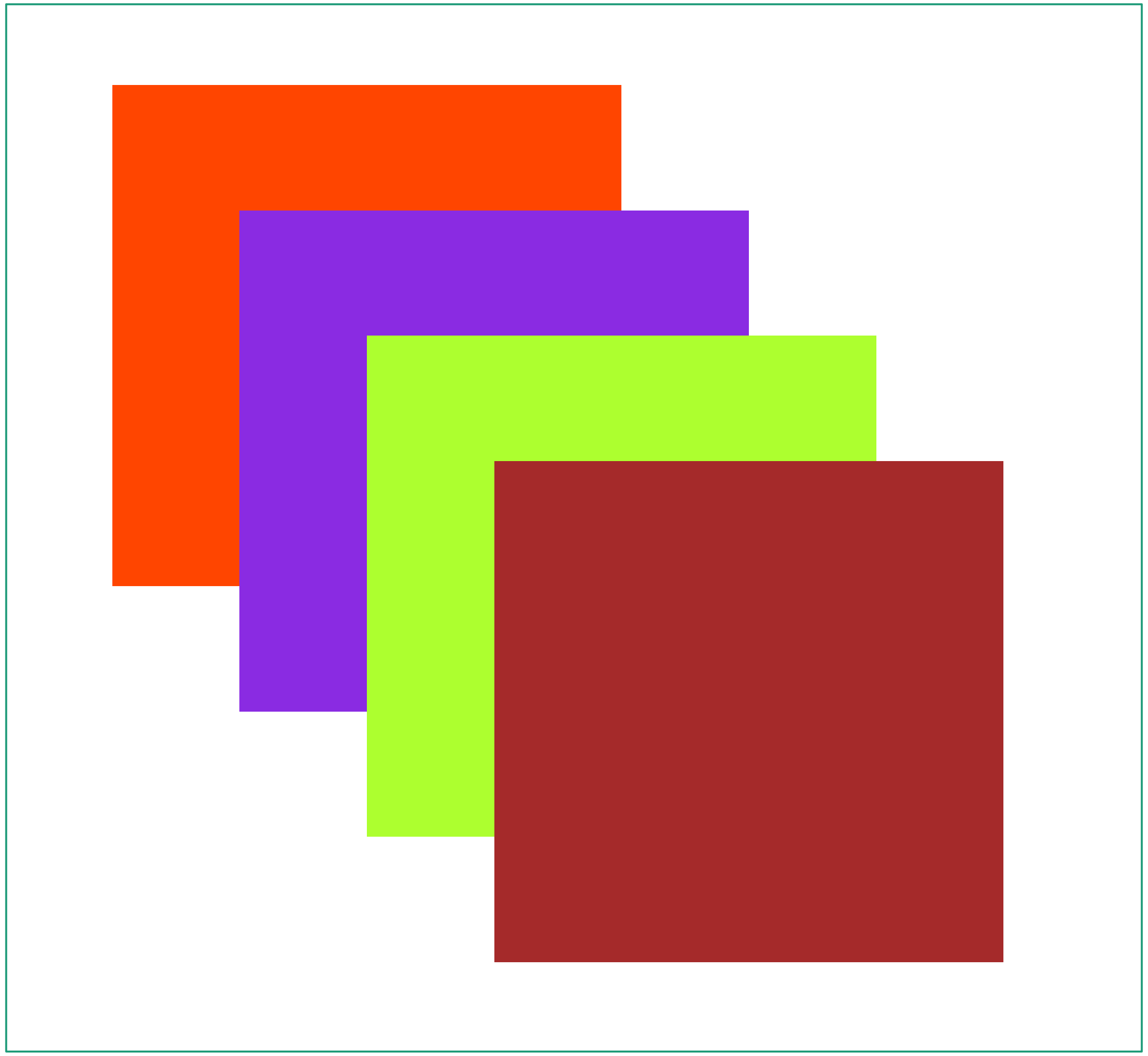
```
.box{
          width: 200px;
          height: 200px;
      }

      #box1{
         position: absolute;
          background-color: orangered;
          margin-top: 50px;
          margin-left: 50px;
          z-index: 1;
      }
      #box2{
         position: absolute;
          background-color: blueviolet;
          margin-top: 100px;
          margin-left: 100px;
          z-index: 2;
      }
```

```
#box3{
          position: absolute;
          background-color: greenyellow;
          margin-top: 150px;
          margin-left: 150px;
          z-index: 3;
    }
    #box4{
          position: absolute;
          background-color: brown;
          margin-top: 200px;
          margin-left: 200px;
          z-index: 4;
    }
```

HTML

```
<body>
    <div class="box" id="box1"></div>
    <div class="box" id="box2"></div>
    <div class="box" id="box3"></div>
    <div class="box" id="box4"></div>
</body>
```

OUTPUT



CSS

display

display

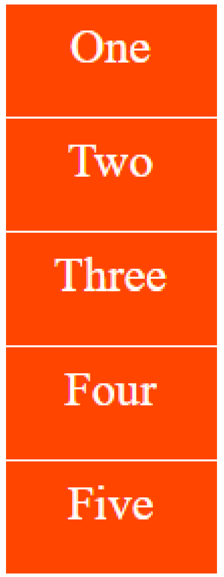
- The display CSS property sets whether an element is treated as a block or inline element and the layout used for its children, such as flow layout, grid or flex.

legacy values

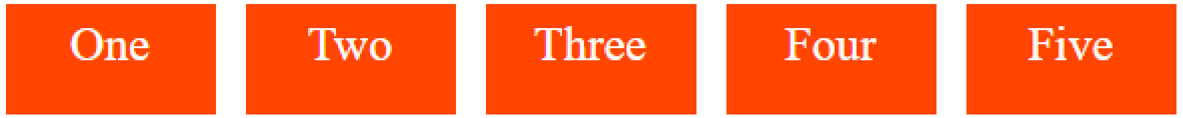
```
display: block;
display: inline;
display: inline-block;
display: flex;
display: inline-flex;
display: grid;
display: inline-grid;
display: flow-root;
```

```
ul li{
        width: 60px;
        height: 30px;
        margin: 1px 3px;
        padding:5px 5px 2px 5px;
        background-color: orangered;
        text-align: center;
        color: white;
    }
```

`{display:block }`



`display: inline-block;`



CSS

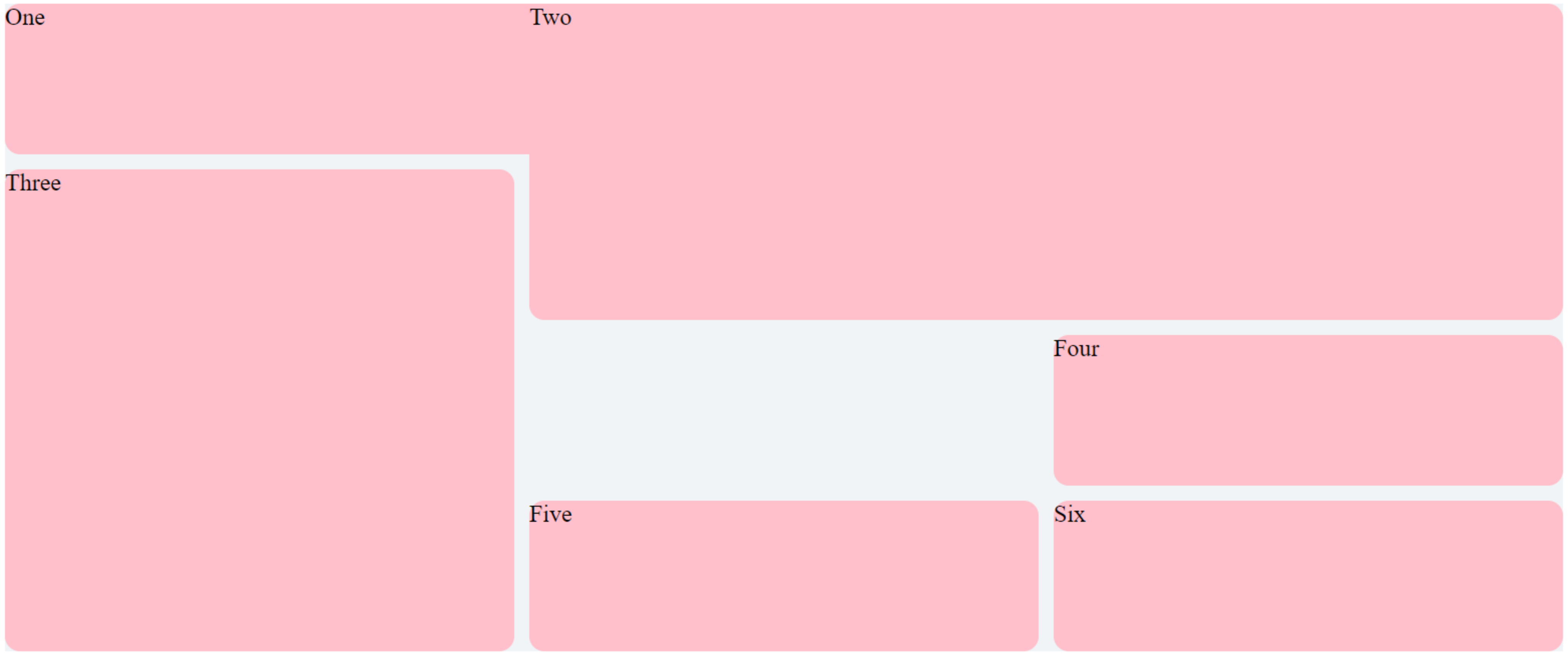
Grid

CSS Grid Layout

- CSS Grid Layout excels at dividing a page into major regions or defining the relationship in terms of size, position, and layer, between parts of a control built from HTML primitives.

- Like tables, grid layout enables an author to align elements into columns and rows.
- However, many more layouts are either possible or easier with CSS grid than they were with tables.
  - a grid container's child elements could position themselves so they actually overlap and layer, similar to CSS positioned elements.

CSS Grid Sample



CSS Grid: How to Use

Flexbox Layout

Flexbox

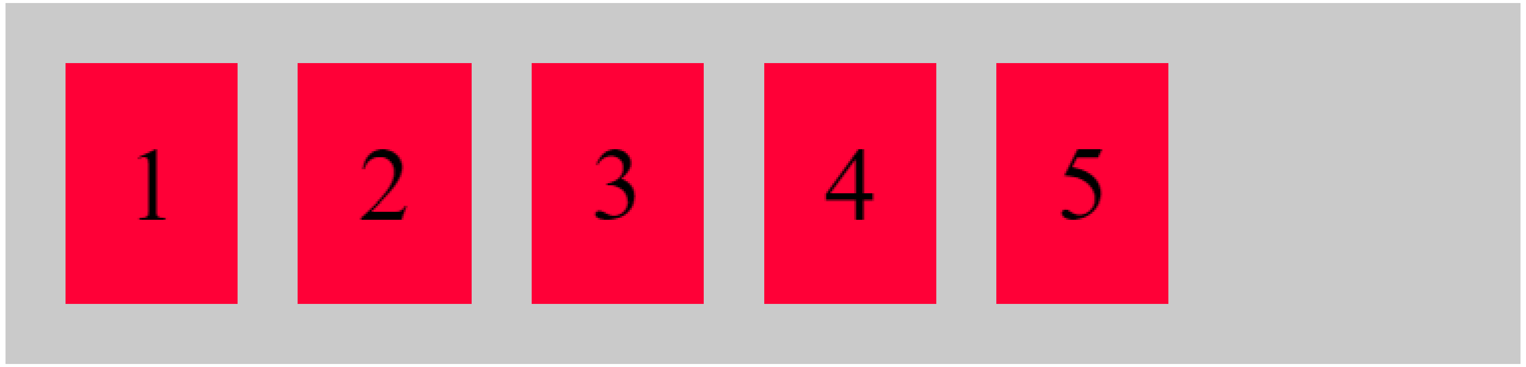
Flexbox Layout helps create Flexible layout easily.
Flexbox enables flexible responsive layout structure without using float or positioning.
To use flexbox model you need to define a flex container:

```
<div class="container">
       <div class="items-1 item">1</div>
       <div class="items-2 item">2</div>
       <div class="items-3 item">3</div>
       <div class="items-4 item">4</div>
       <div class="items-5 item">5</div>
   </div>
```

Flexbox

```
.container{
background-color: #CACACA;
padding: 10px;
margin: 50px;
display: flex;
flex-direction: row;
}

.item{
background-color: #ff0037;
font-size: 35px;
padding: 20px;
margin: 10px;
}
```

```
<div class="container">
        <div class="items-1 item">1</div>
        <div class="items-2 item">2</div>
        <div class="items-3 item">3</div>
        <div class="items-4 item">4</div>
        <div class="items-5 item">5</div>
    </div>
```



Flexbox

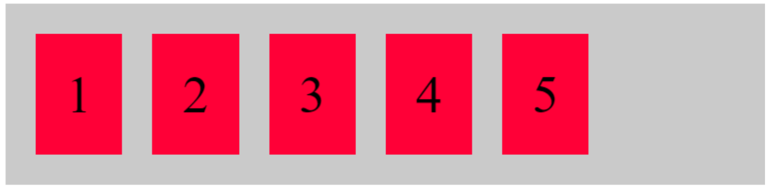
flex container properties

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

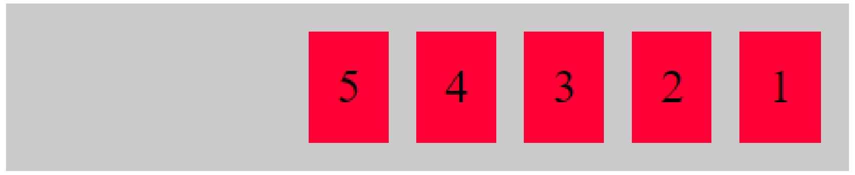
Property: flex-direction

- Defines the direction in which the container wants to stack the flex items
- flex-direction property values are :

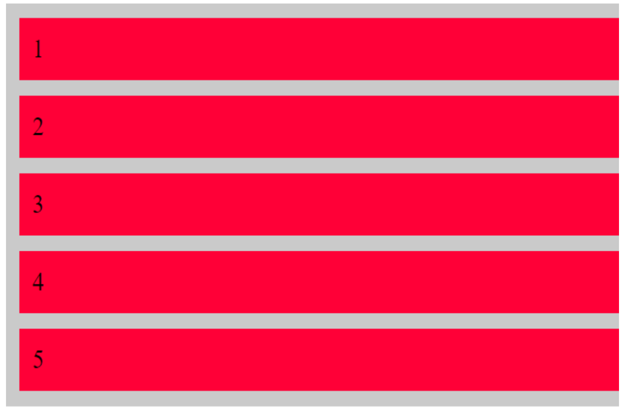
row



row-reverse

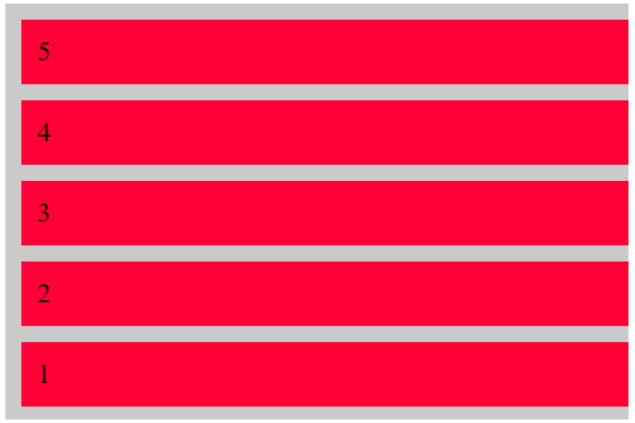


column



inherit
initial

column-reverse



Property: flex-wrap

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

- The flex-wrap property specifies whether the flex items should wrap or not.
- The *wrap* value specifies that the flex items will wrap if necessary.
- The *nowrap* (**default**) value specifies that the flex items will not wrap.
- The wrap-reverse value specifies that the flexible items will wrap if necessary, in reverse order.

Property: **flex-flow**

Using CSS transitions

CSS Transitions

- **CSS transitions** provide a way to control animation speed when changing CSS properties.
  - Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time.
- If you change the color of an element from white to black, usually the change is instantaneous.
- With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized.

Transition Properties

- The properties used in transition are given below

transition-delay
transition-duration
transition-property
transition-timing-function

Transition: An example

```
.target {
                font-size: 14px;
                transition-property: font-size,color;
                transition-duration: 4s;
            }

            .target:hover {
                font-size: 36px;
                color: rgb(61, 4, 61);
            }
```

```
<a class="target">Hover over me</a>
```

CSS

Vendor Prefixes

What are vendor Prefixes (in CSS)

- These are prefixes used by Browser Vendor to use some experimental features supported by the specific browser but not yet release or implemented by other browsers.
- Vendor prefixes allow developers to use experimental features before they are standardized/released
- Developers can experiment with new ideas while—in theory—preventing their experiments from being relied upon and then breaking web developers' code during the standardization process.
- Developers should wait to include the unprefixed property until browser behavior is standardized.

Are they bad!

- Not really!
- They allow the developers experiment with new features.

CSS Prefixes

Vendor Prefix	Used by
-webkit-	Chrome, Safari, newer versions of Opera, almost all iOS browsers including Firefox for iOS; basically, any WebKit based browser
-moz-	Firefox
-o-	old pre-WebKit versions of Opera
-ms-	Internet Explorer and Microsoft Edge

CSS Vendor **Prefix** Example

Non standardized ***transition*** property

```
.myClass {
   -webkit-transition: all 1s linear;
   -moz-transition: all 1s linear;
   -ms-transition: all 1s linear;
   -o-transition: all 1s linear;
   transition: all 1s linear;
 }
```

standardized ***transition*** property

```
.myClass {
    transition: all 1s linear;
  }
```

CSS Vendor Prefixes: Do we need them?

- Since Opera and Edge are Chromium based, -o- and -ms- will probably go soon out of fashion.
- Hence in future vendor prefixes as a whole are going out of fashion, too.

- Writing prefixes is hard, mostly because of uncertainty.
- Do you actually need a prefix for one property?

CSS

Media Queries

Media queris

- Media queries are useful when you want to modify your site or app depending on a device's general type (such as print vs. screen) or specific characteristics and parameters (such as screen resolution or browser viewport width).

- Media queries are used for the following:

    - To conditionally apply styles with the CSS **@media** and @import at-rules.
    - To target specific media for the <style>, <link>, <source>, and other HTML elements with the media= attribute.
    - To test and monitor media states using the Window.matchMedia() and MediaQueryList.addListener() JavaScript methods.

Media query Syntax

- 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
not|only mediatype and (mediafeature and|or|not mediafeature)
 {
  CSS-Code;
}
```

- `Media Types: all, print, screen, speech, defaults to ‘all’`
- `Media Features: refer to Document`

Media query used in HTML

```
<link rel="stylesheet" media="screen and (min-width: 900px)"
href="widescreen.css">

<link rel="stylesheet" media="screen and (max-width: 600px)"
href="smallscreen.css">
```

Media query: Example

```css
body {
  background-color: lightblue;
}

@media screen and (min-width: 400px) {
  body {
    background-color: lightgreen;
  }
}

@media screen and (min-width: 800px) {
  body {
    background-color: lavender;
  }
}
```