

# Web Programming

## CSS Part III

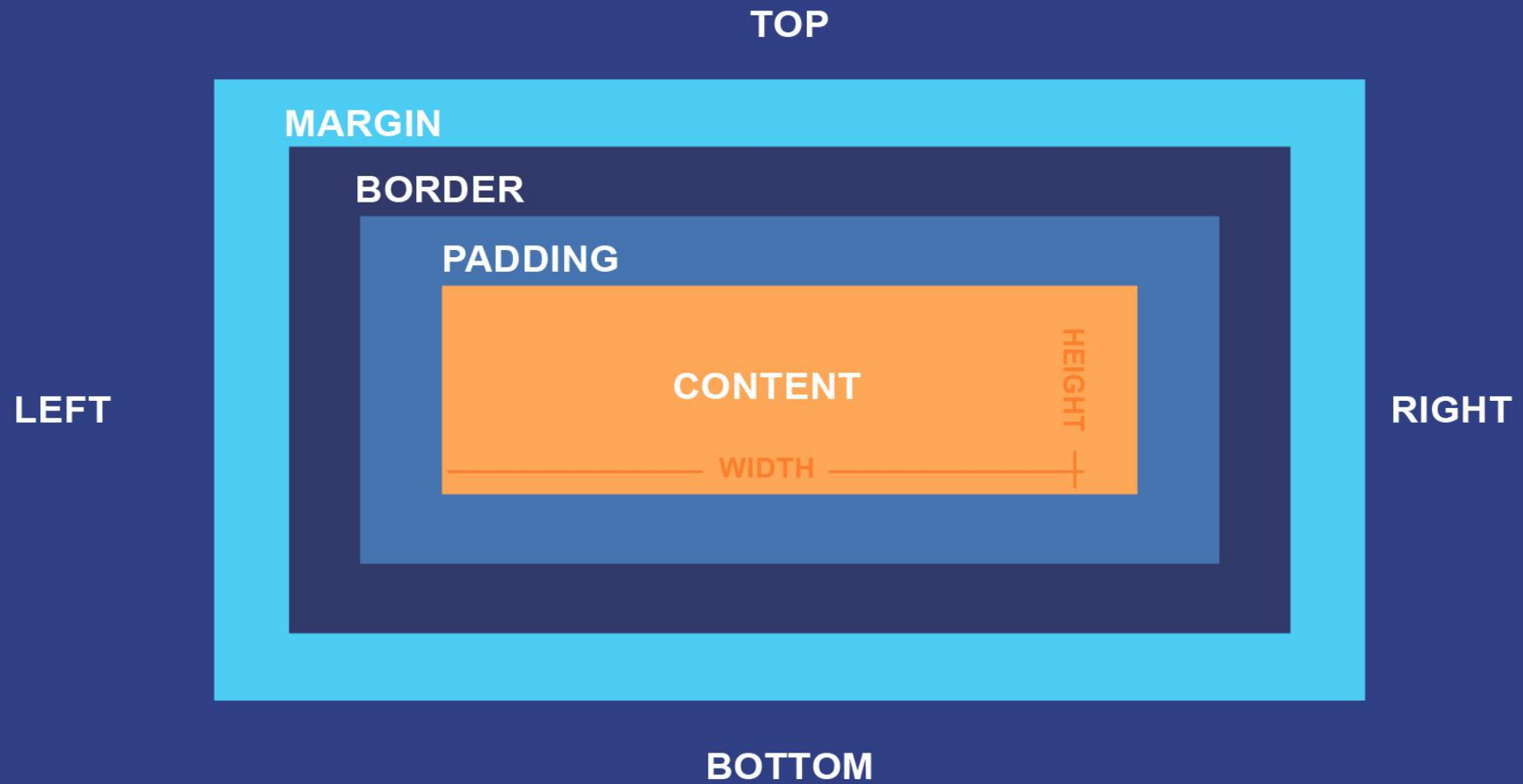


# Part III.

# Positioning

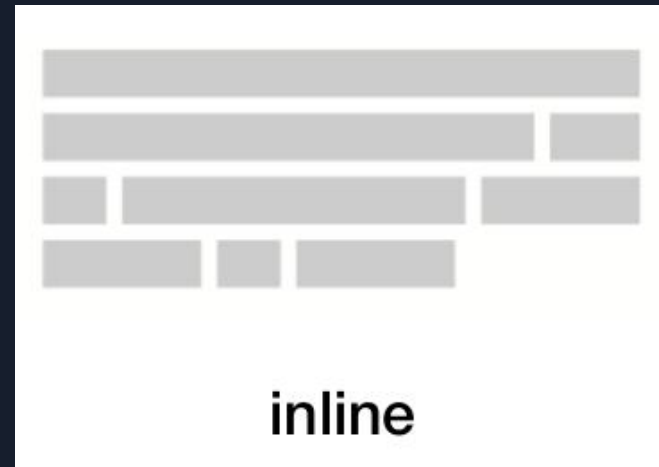
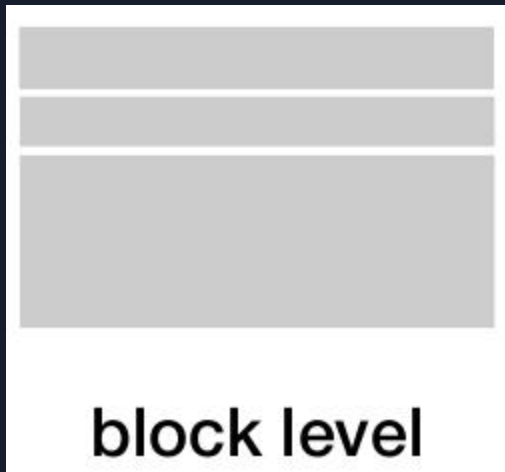
# Boxes

# The Box Model



# Block level vs. inline

- Imagine that there is an invisible box around every HTML element
- Block level elements start on a new line
  - E.g., <h1>, <p>, <ul>, <li>, ...
- Inline elements flow with the text
  - E.g., <a>, <em>, <img>, ...



# width property

- By default, block elements are given a width equally to the parent element's width
- **width** applies only to block elements and to the `<img>` element

# Display type

- display specifies the type of box used for a HTML element
- Values:
  - **inline** block-level element acts like an inline element
  - **block** inline element acts like a block element
  - **inline-block** block-level element flows like an inline element, but retains other features of a block-level element
- **none** element is hidden from the page

# Example

## HTML

```
<ul>  
  <li> Home </li>  
  <li> About </li>  
  <li> News </li>  
  <li> Partners </li>  
  <li> Contact </li>  
</ul>
```

- Home
- About
- News
- Partners
- Contact



# Example: inline

css

```
li {  
  display: inline;  
  padding: 3px;  
  border: 1px solid grey;  
  width: 5em;  
}
```

← has no effect (width of inline elements is ignored)

</> inline.html U

Preview inline.html X

css styles2.css U

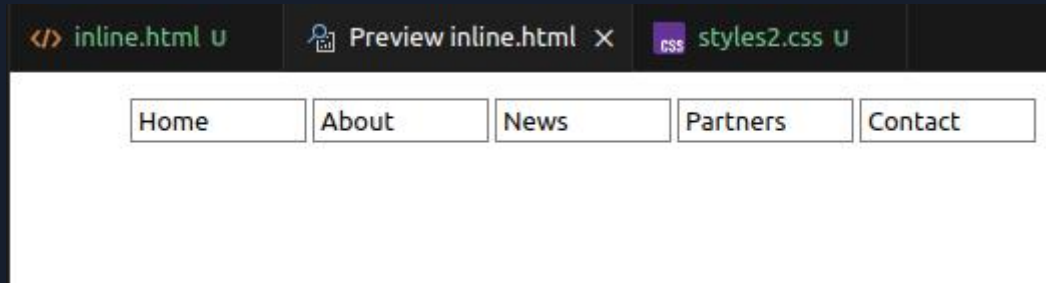
Home About News Partners Contact

# Example: inline

css

```
li {  
  display: inline-block;  
  padding: 3px;  
  border: 1px solid grey;  
  width: 5em;  
}
```

← has effect



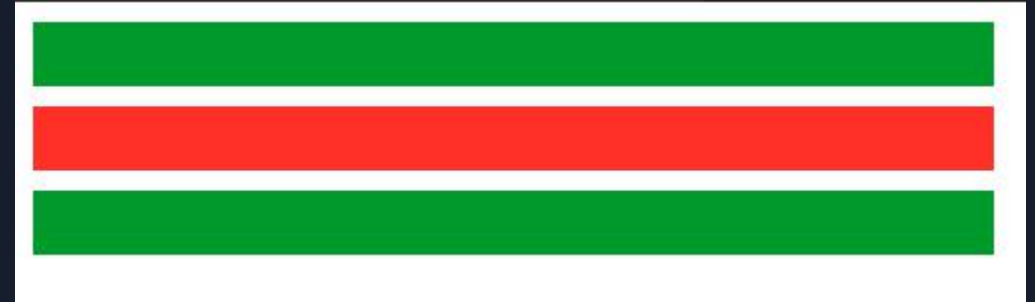
# Visibility

- **visibility** specifies whether an element is visible
- Values
  - **visible** the element is visible (default)
  - **hidden** the element is hidden (but it still takes up space!)
- Note: an element that is set to invisible will still takes up the space on the page
  - (Use **display: none;** for hiding it completely)

# Display vs. visibility

## HTML

```
<div> </div>  
<div id="mydiv"> </div>  
<div> </div>
```



## CSS

```
#mydiv {  
  display: none;  
}
```



## CSS

```
#mydiv {  
  visibility: hidden;;  
}
```



# Positioning

- Property: **position**
- Values:
  - **static** default positioning
  - **relative** position relative to where it would normally appear
  - **absolute** position
  - **fixed** position
  - **inherit** inherit from parent element

# Static positioning

- `position: static`
- Normal flow
- This is the default setting, no need to specify it
  - Unless needed to overwrite a positioning that had been previously set

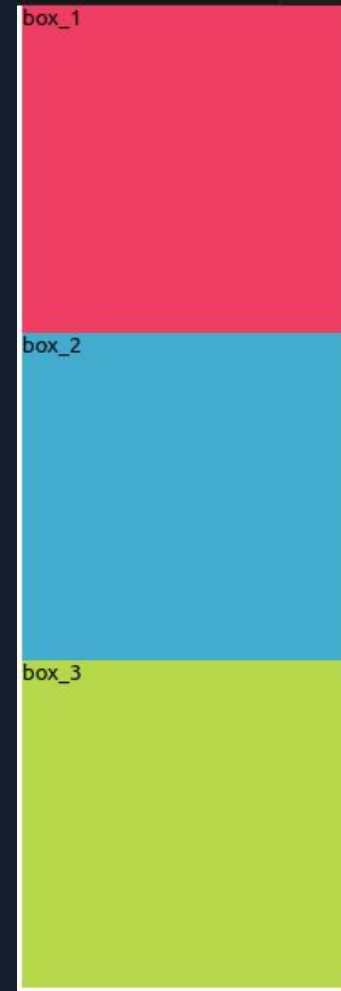
# Example: normal flow

## HTML

```
<div id="box_1">box_1</div>  
<div id="box_2">box_2</div>  
<div id="box_3">box_3</div>
```

## CSS

```
div {  
  width: 200px;  
  height: 200px;  
}  
#box_1 {  
  background: #ee3e64;  
}  
#box_2 {  
  background: #44accf;  
}  
#box_3 {  
  background: #b7d84b;  
}
```



# Relative positioning

- position: *relative*
- Move it *relatively* to where it would have been in the normal flow using *top* or *bottom*, and *left* or *right*
- Unit: px, %, em, etc.



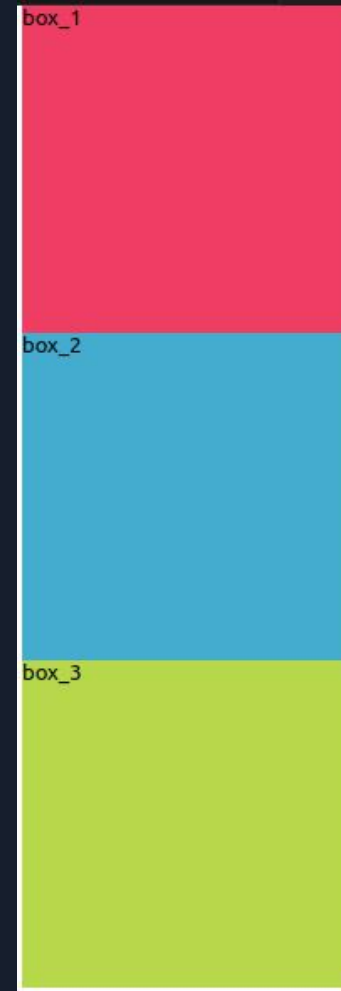
# Example: Position Relative

## HTML

```
<div id="box_1">box_1</div>  
<div id="box_2">box_2</div>  
<div id="box_3">box_3</div>
```

## CSS

```
div {  
  width: 200px;  
  height: 200px;  
}  
#box_1 {  
  background: #ee3e64;  
}  
#box_2 {  
  background: #44accf;  
  position: relative;  
}  
#box_3 {  
  background: #b7d84b;  
}
```



# Example: Position Relative

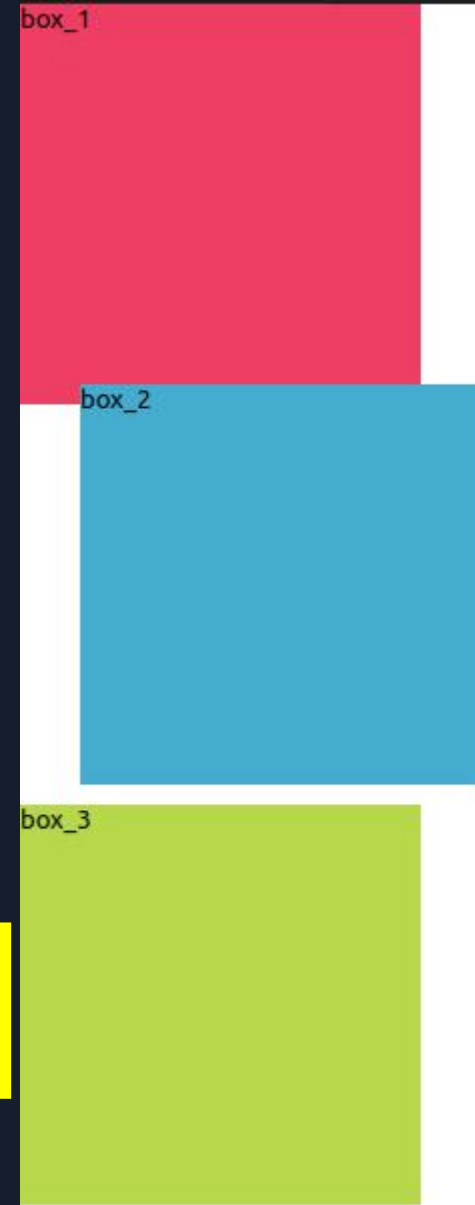
## HTML

```
<div id="box_1">box_1</div>  
<div id="box_2">box_2</div>  
<div id="box_3">box_3</div>
```

## CSS

```
div {  
  width: 200px;  
  height: 200px;  
}  
#box_1 {  
  background: #ee3e64;  
}  
#box_2 {  
  background: #44accf;  
  position: relative;  
  left: 30px;  
  bottom: 10px;  
}  
#box_3 {  
  background: #b7d84b;  
}
```

Pushed 30px from the left and  
10px from the bottom.



# Absolute positioning

- position: absolute
- Element's position is set with respect to its containing element
  - That is the first parent element with a position **other than static**
- Set top, bottom, left, or right
  - in pixels, percentages, or em
- Element is taken out of the normal flow (no longer affects the position of other elements)

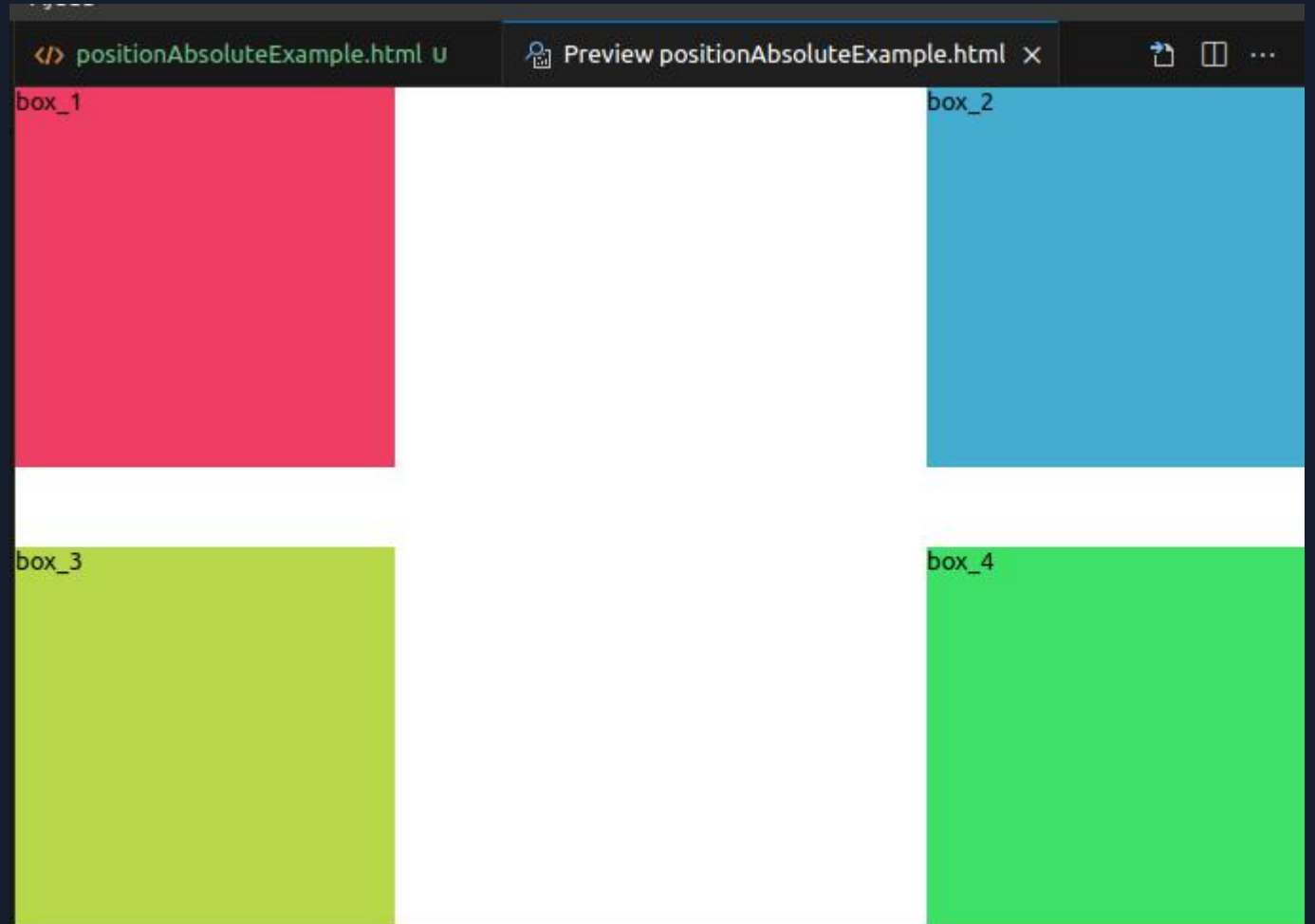
# Example: absolute

## HTML

```
<div id="box_1">box_1</div>
<div id="box_2">box_2</div>
<div id="box_3">box_3</div>
<div id="box_4">box_4</div>
```

## CSS

```
div { width: 200px; height: 200px; }
#box_1 {
  background: #ee3e64;
  position: absolute;
  top: 0px; left: 0px;
}
#box_2 {
  background: #44accf;
  position: absolute;
  top: 0px; right: 0px;
}
#box_3 {
  background: #b7d84b;
  position: absolute;
  bottom: 0px; left: 0px;
}
#box_4 {
  background: #3ee067;
  position: absolute;
  bottom: 0px; right: 0px;
}
```



# Example #2

## HTML

```
<div id="container">  
  <div id="box_1"> </div>  
  <div id="box_2"> </div>  
  <div id="box_3"> </div>  
  <div id="box_4"> </div>  
</div>
```

## CSS

```
#container {  
  border: 1px solid black;  
  width: 300px;  
  height: 300px;  
  position: relative;  
}
```



[https://github.com/weder96/programming\\_web/blob/main/learning-15/positionRelativeExample.html](https://github.com/weder96/programming_web/blob/main/learning-15/positionRelativeExample.html)

# Fixed positioning

- `position: fixed`
- Element's position is set with respect to the browser window
  - Remains there even when the user scrolls
- Set top, bottom, left, or right
  - in pixels, percentages, or em
- Element is taken out of the normal flow (no longer affects the position of other elements)

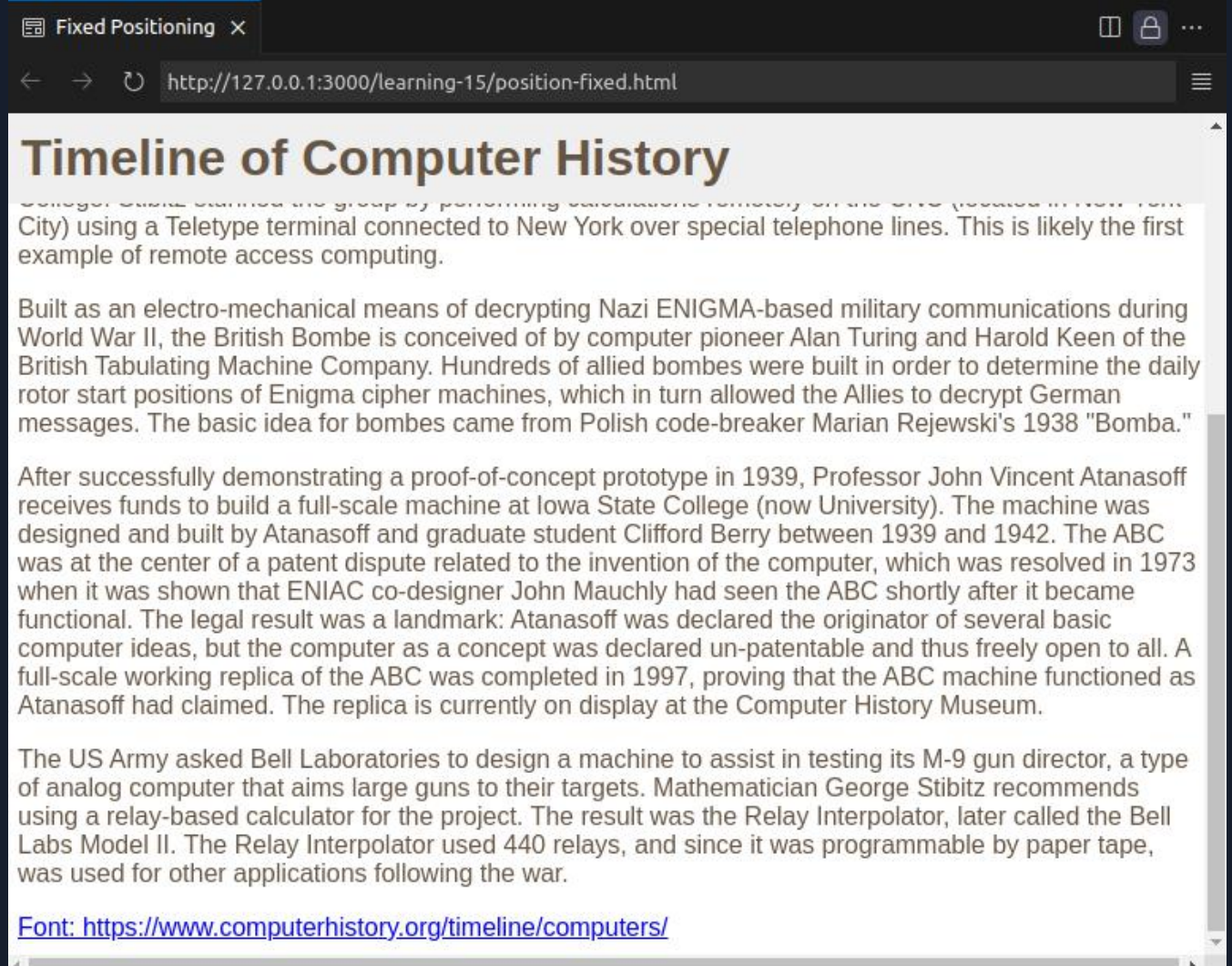
# Example #Fixed

## HTML

```
body >
<h1>Timeline of Computer History</h1>
....
</body>
```

## CSS

```
body {
  width: 750px;
  font-family: Arial, Verdana, sans-serif;
  color: #665544;}
h1 {
  position: fixed;
  top: 0px;
  left: 0px;
  padding: 10px;
  margin: 0px;
  width: 100%;
  background-color: #efefef;
}
p.example {
  margin-top: 100px;
}
```



[https://github.com/weder96/programming\\_web/blob/main/learning-15/position-fixed.html](https://github.com/weder96/programming_web/blob/main/learning-15/position-fixed.html)

# Floating elements

- Allow elements to appear next to each other
- `float: left` or `float: right`
- Element is taken out of the normal flow and placed as far to the **left** or **right** of the containing (block) element as possible
- Also set the width property (otherwise it'll take up the full width of the containing element)
- If you want a bit distance from the edge, set the **margin** on the floating element



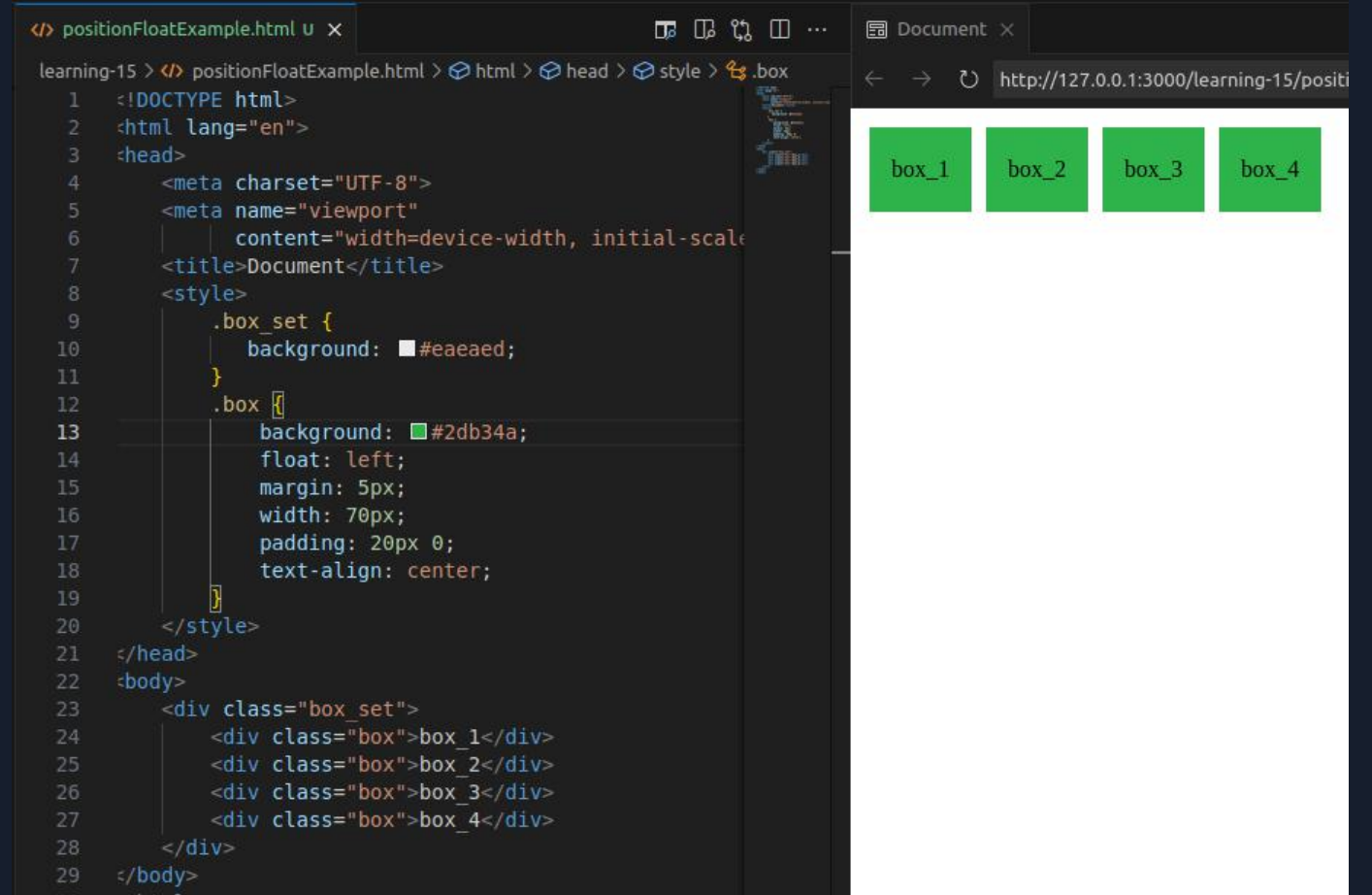
# Example

## HTML

```
<div class="box_set">
  <div class="box">box_1</div>
  <div class="box">box_2</div>
  <div class="box">box_3</div>
  <div class="box">box_4</div>
</div>
```

## CSS

```
.box_set {
  background: #eaeaed;
}
.box {
  background: #2db34a;
  float: left;
  margin: 5px;
  width: 70px;
  padding: 20px 0;
  text-align: center;
}
```



```
positionFloatExample.html u x
learning-15 > positionFloatExample.html > html > head > style > .box
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width, initial-scale=1">
7   <title>Document</title>
8   <style>
9     .box_set {
10       background: #eaeaed;
11     }
12     .box {
13       background: #2db34a;
14       float: left;
15       margin: 5px;
16       width: 70px;
17       padding: 20px 0;
18       text-align: center;
19     }
20   </style>
21 </head>
22 <body>
23   <div class="box_set">
24     <div class="box">box_1</div>
25     <div class="box">box_2</div>
26     <div class="box">box_3</div>
27     <div class="box">box_4</div>
28   </div>
29 </body>
```

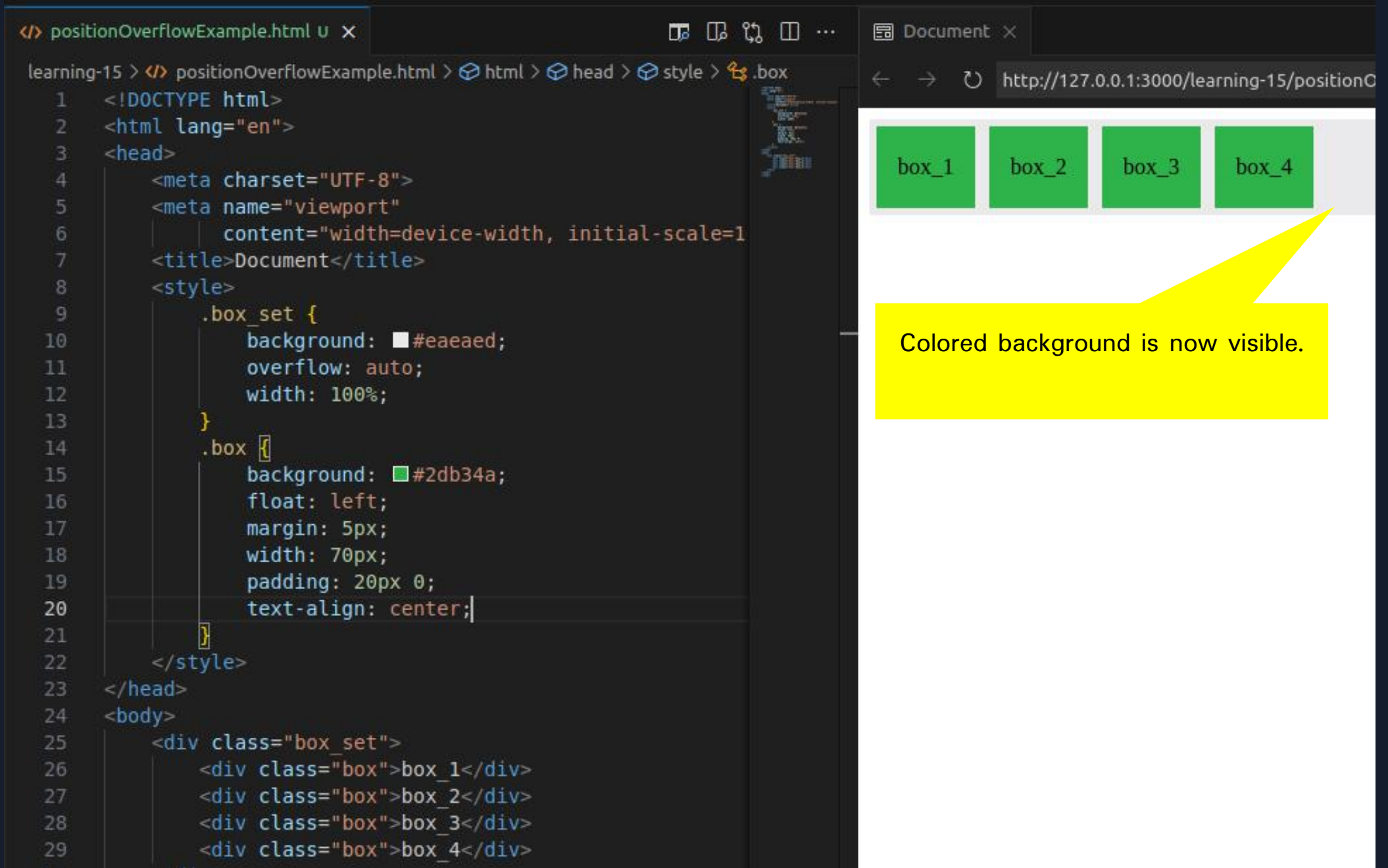
# Parents of floated elements

- If a containing element contains only floating elements, some browsers will treat it 0 pixels tall
- Solution: "overflow" technique
- Set for parent element:

```
overflow: auto;  
width: 100%;
```

- width is required because of older browsers (doesn't have to be 100%)
- Parent element will have an actual height this way
- **Alternative solution: "clearfix" technique**
- See references slide or google it

# Example



The image shows a web browser window displaying a page with four green boxes labeled box\_1, box\_2, box\_3, and box\_4. A yellow callout points to the boxes with the text "Colored background is now visible." The browser's address bar shows the URL <http://127.0.0.1:3000/learning-15/positionC>. The browser's developer tools are open, showing the HTML and CSS for the page.

```
positionOverflowExample.html u x
learning-15 > positionOverflowExample.html > html > head > style > .box
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width, initial-scale=1
7   <title>Document</title>
8   <style>
9     .box_set {
10       background: #eaeaed;
11       overflow: auto;
12       width: 100%;
13     }
14     .box {
15       background: #2db34a;
16       float: left;
17       margin: 5px;
18       width: 70px;
19       padding: 20px 0;
20       text-align: center;
21     }
22   </style>
23 </head>
24 <body>
25   <div class="box_set">
26     <div class="box">box_1</div>
27     <div class="box">box_2</div>
28     <div class="box">box_3</div>
29     <div class="box">box_4</div>
```

# Overflow

- The overflow property specifies what happens if content overflows an element's box
- **Values:**
  - **visible** content renders outside the element's box (default)
  - **hidden** the overflow is clipped, the rest of the content is visible
  - **scroll** the overflow is clipped, but a scrollbar is added to see the rest
  - **auto** if overflow is clipped, a scrollbar is added

# Example

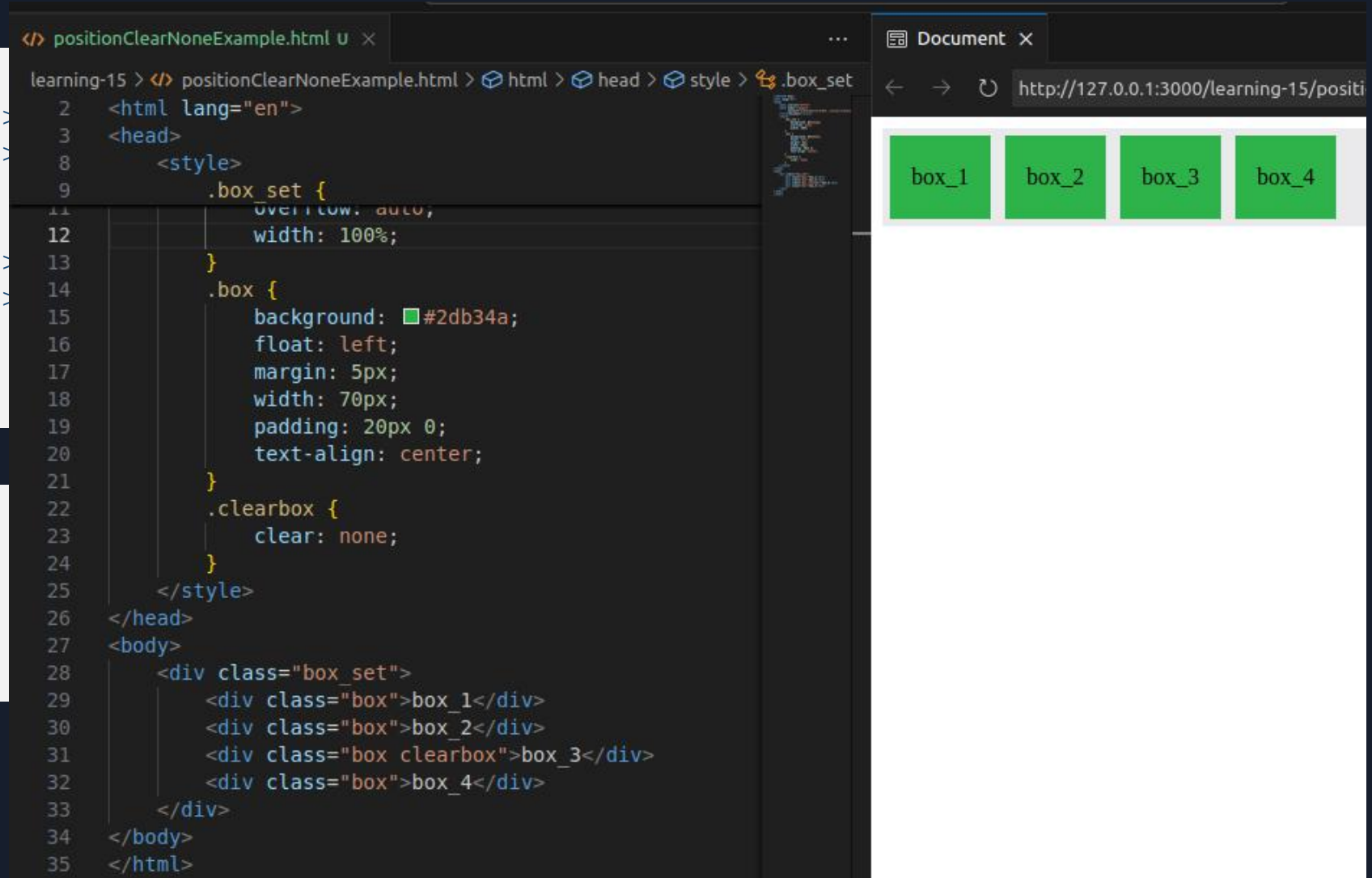
clear: none

## HTML

```
<div class="box-set">
  <div class="box">Box 1</div>
  <div class="box">Box 2</div>
  <div class="box clearfix">
    Box3</div>
  <div class="box">Box 4</div>
  <div class="box">Box 5</div>
</div>
```

## CSS

```
.clearfix {
  clear: none;
}
```



# Example

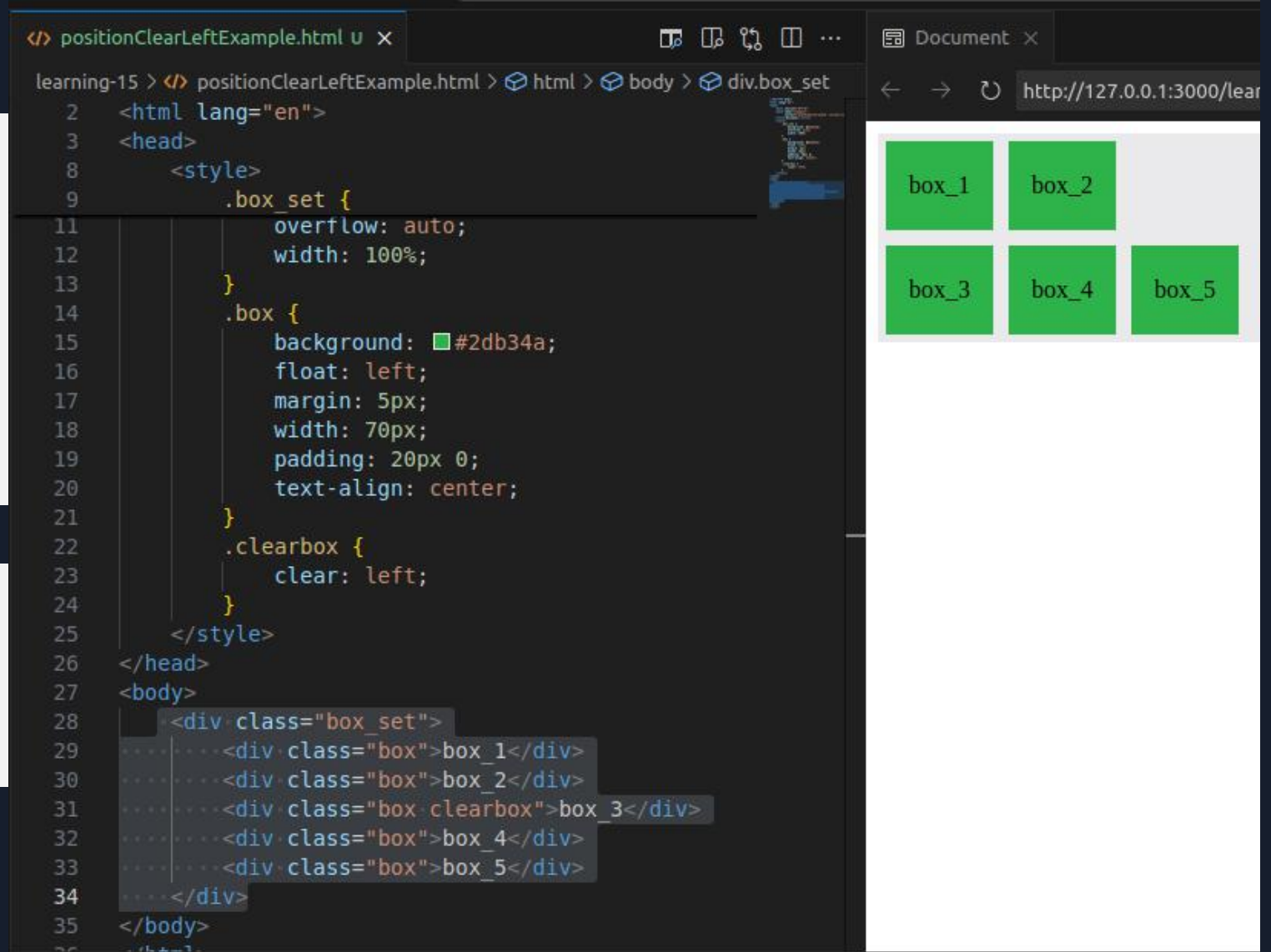
clear: left

## HTML

```
<div class="box_set">
  <div class="box">box_1</div>
  <div class="box">box_2</div>
  <div class="box
clearbox">box_3</div>
  <div class="box">box_4</div>
  <div class="box">box_5</div>
</div>
```

## CSS

```
.clearbox {
  clear: left;
}
```





# Example


clear: right

## HTML

```
<div class="box_set">
  <div class="box">box_1</div>
  <div class="box">box_2</div>
  <div class="box
clearbox">box_3</div>
  <div class="box">box_4</div>
  <div class="box">box_5</div>
</div>
```

## CSS

```
.clearbox {
  clear: right;
}
```



Why is Box 4 not in a new row?!

Clear only clears the floats preceding the element in the document source!

```
positionClearRightExample.html
learning-15 > </> positionClearRightExample.html > html > head > style > .cl
2 <html lang="en">
3 <head>
8 <style>
9   .box_set {
13   }
14   .box {
15     background: #2db34a;
16     float: left;
17     margin: 5px;
18     width: 70px;
19     padding: 20px 0;
20     text-align: center;
21   }
22   .clearbox {
23     clear: right;
24   }
25 </style>
26 </head>
27 <body>
28   <div class="box_set">
29     <div class="box">box_1</div>
30     <div class="box">box_2</div>
31     <div class="box clearbox">box_3</div>
32     <div class="box">box_4</div>
33     <div class="box">box_5</div>
34   </div>
35 </body>
36 </html>
```

# Stacking elements

- Property: z-index
- Value: stack order of the element

```
z-index: 3;
```

- Z-index only works on positioned elements!
- position:absolute, position:relative, or position:fixed



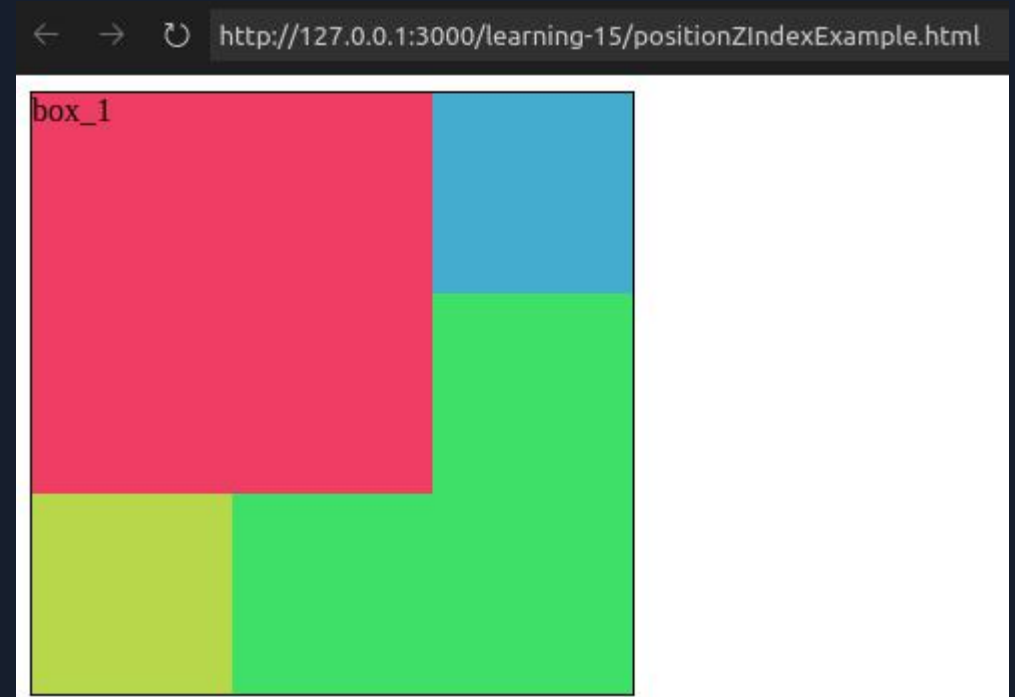
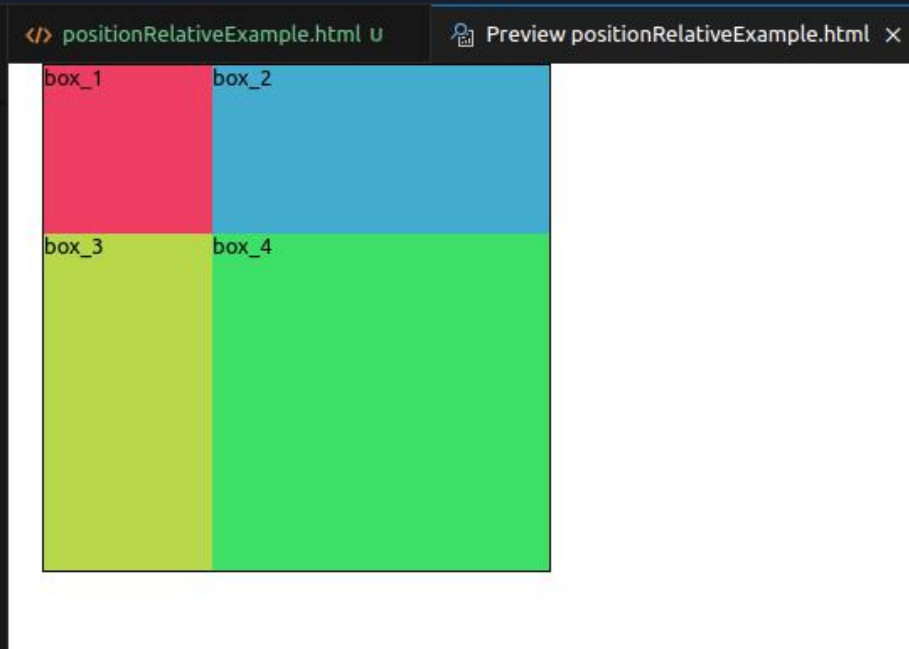
# Example

## CSS

```
#box_1 {  
  background: #ee3e64;  
  position: absolute;  
  top: 0;  
  left: 0;  
}
```

## CSS

```
#box_1 {  
  background: #ee3e64;  
  position: absolute;  
  top: 0;  
  left: 0;  
  z-index: 3;  
}
```



[https://github.com/weder96/programming\\_web/blob/main/learning-15/positionZIndexExample.html](https://github.com/weder96/programming_web/blob/main/learning-15/positionZIndexExample.html)

# Some common issues

# Center align block element

- To horizontally center a block element (like `<div>`), use

```
margin: auto;
```

- Center aligning has no effect if the width property is not set (or set to 100%)
- See also [http://www.w3schools.com/css/css\\_align.asp](http://www.w3schools.com/css/css_align.asp)



# Vertical centering of text

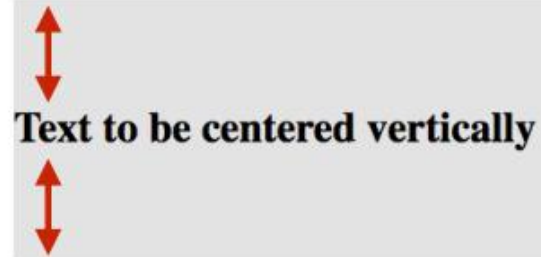
- Line height trick
  - Set line-height to the parent element's height
  - Works only for a single line of text

## HTML

```
<div>  
  <h1>Text to be centered vertically</h1>  
</div>
```

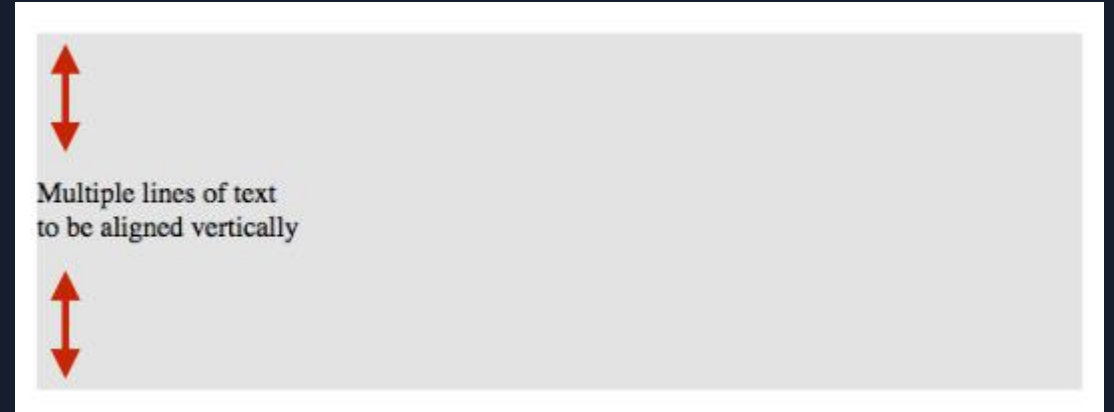
## CSS

```
div {  
  height: 200px;  
}  
h1 {  
  line-height: 200px;  
}
```



# Vertical centering of text #2

- Table cell trick
- Let the element behave like a table cell
- Table cell content can be vertically aligned
- It is important to add the height of the element



## HTML

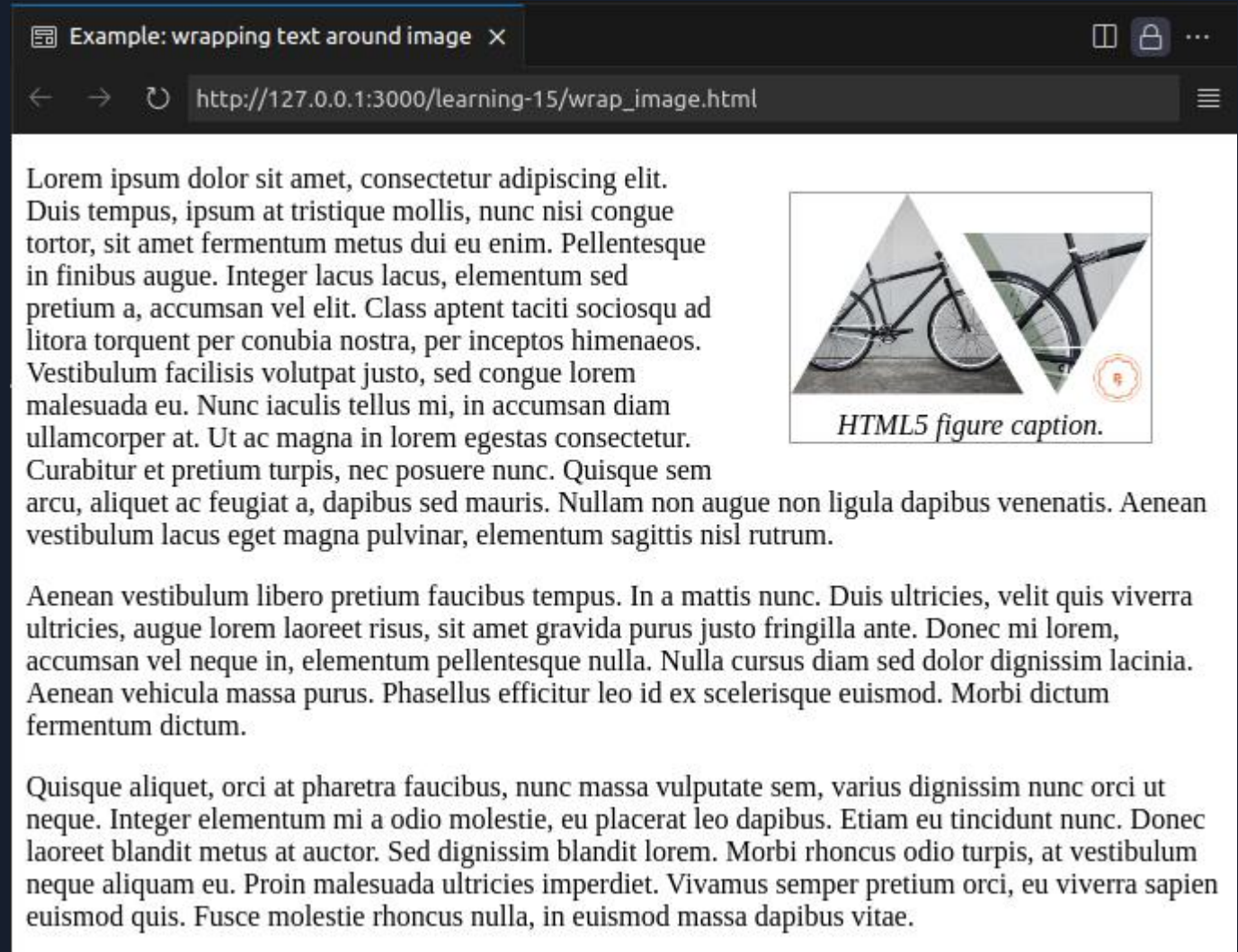
```
<div>  
  <p>Multiple lines of text <br />  
    to be aligned vertically  
  </p>  
</div>
```

## CSS

```
div {  
  height: 200px;  
}  
p {  
  height: 200px;  
  display: table-cell;  
  vertical-align: middle;  
}
```

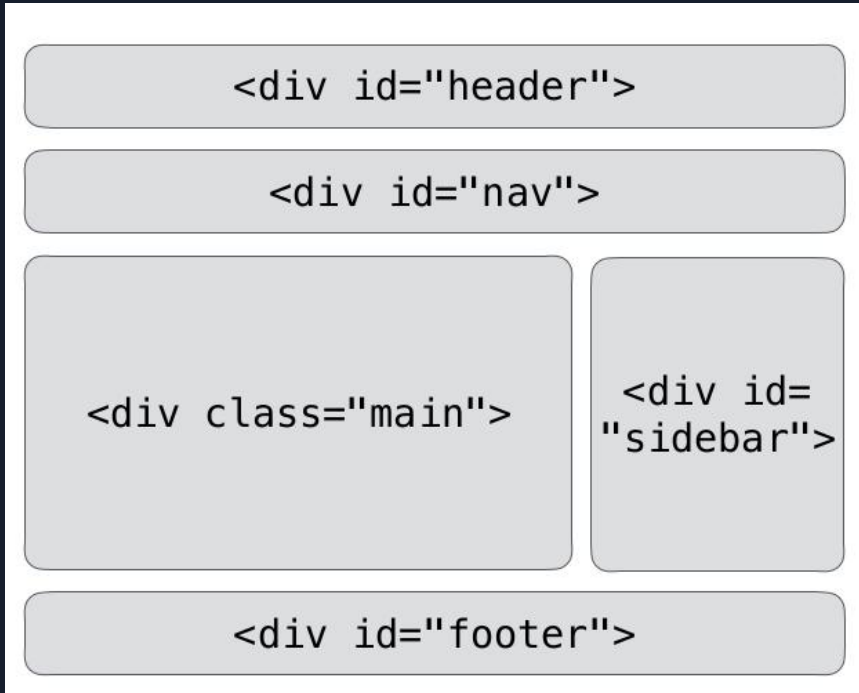
# Wrap text around image

- Float the image (left or right);  
the text will automatically wrap  
around it.

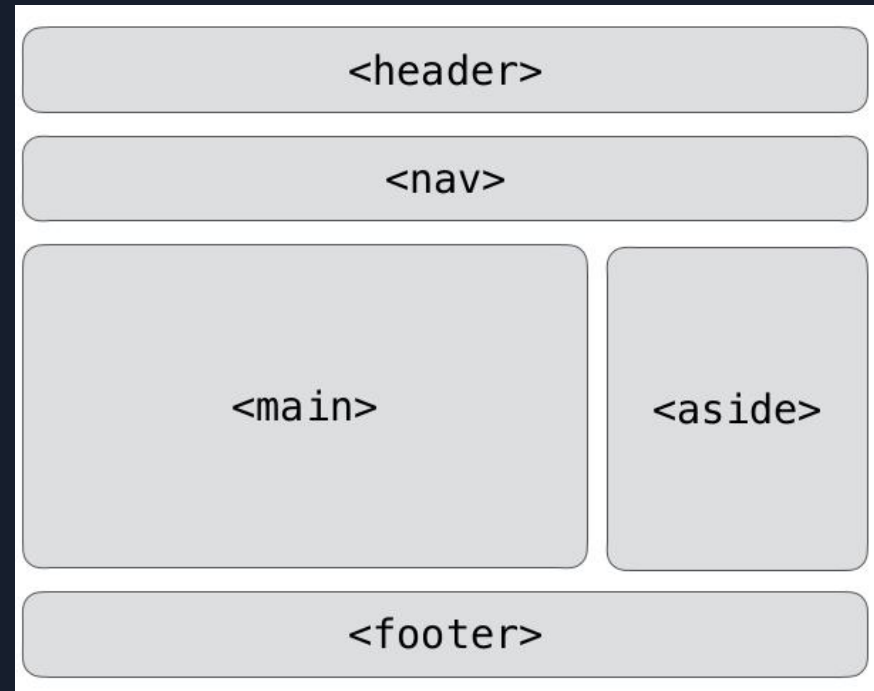


# Layouts

# Page sections



Classic HTML



HTML5



# Page sections

```
<div class="article">
```

```
<div class="section">
```

```
<div class="article">
```

```
<div class="section">
```

Classic HTML

```
<article>
```

```
<section>
```

```
<article>
```

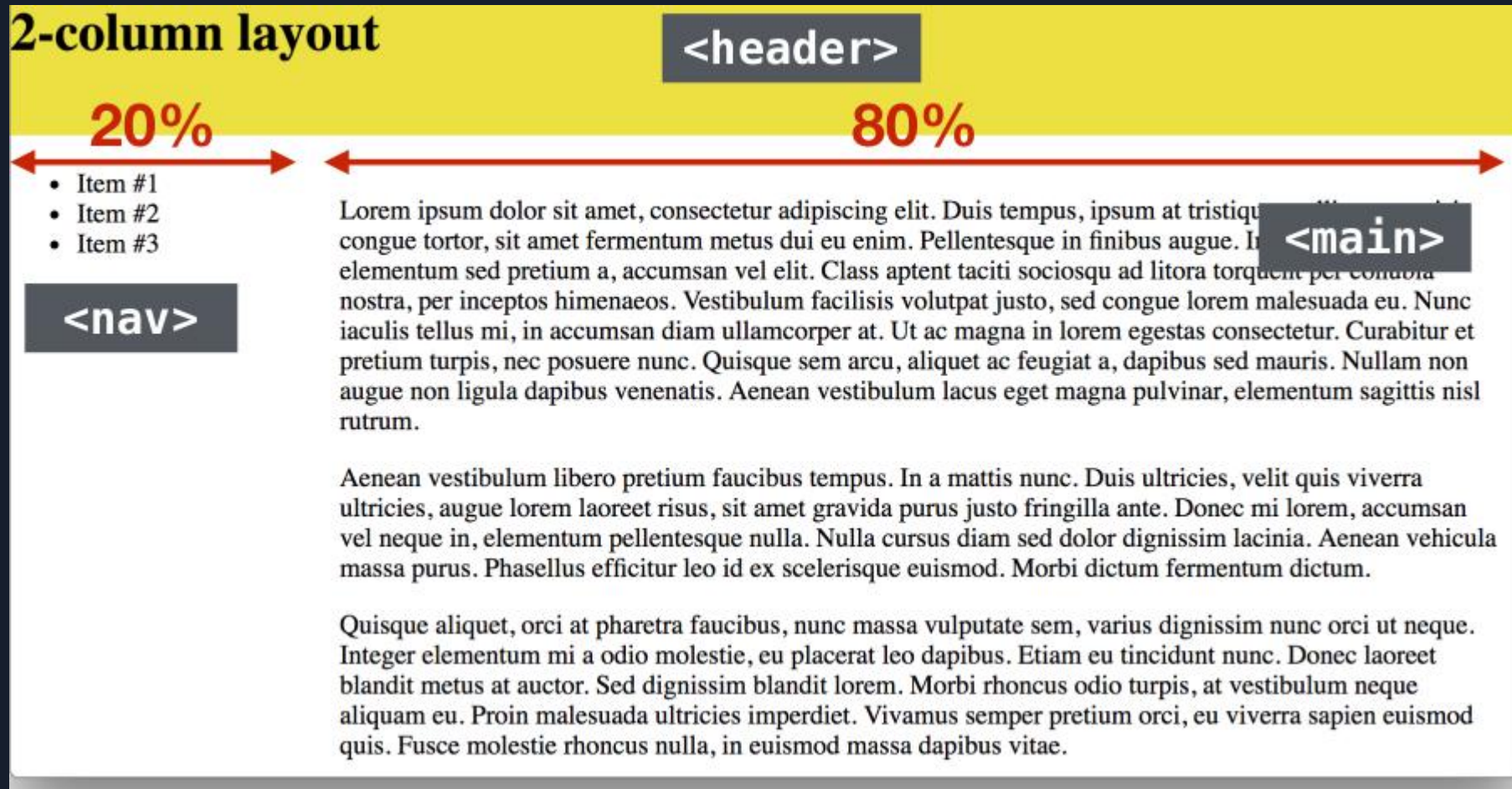
```
<section>
```

HTML5

# Fixed-width vs fluid layouts

- Fixed-width layout
  - Components inside a fixed-width wrapper have either percentage or fixed widths. Typically, grid systems.
- Fluid (or liquid) layout
  - Components have percentage widths (in % or em), thus adjust to the user's screen resolution

# Two-column layout



# Responsive design

- Tailoring layout specifically for the type of screen
- E.g., three column layout for desktops, a two column layout for tablets, and a single column layout on smartphones
- Using a fluid grid and media queries in CSS

# CSS media queries

- CSS technique introduced in CSS3
- Uses the @media rule to include a block of CSS property only if a certain condition is true
- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution
- ...

```
@media mediatype and|not|only (media feature) {...}
```

# CSS media queries (2)

- Possible to write different CSS code for different media types
- For example

```
@media screen and (max-width: 300px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

- Also possible to have different style files for different media

```
<link rel="stylesheet" media="mediatype andnotonly (media feature)"  
href="mystylesheet.css">
```

- See [http://www.w3schools.com/cssref/css3\\_pr\\_mediaquery.asp](http://www.w3schools.com/cssref/css3_pr_mediaquery.asp)

# Mobile first

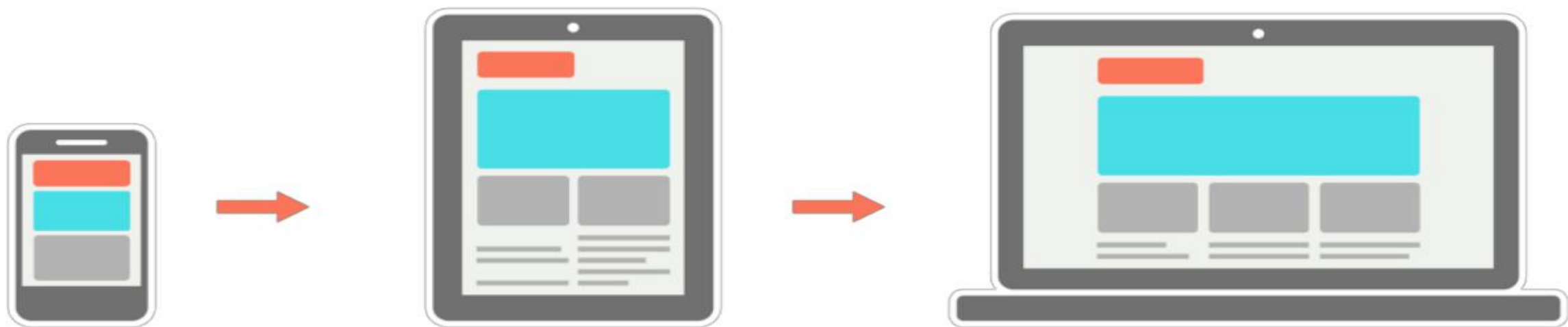
- Both a strategy and a new way of writing code
- Designing an online experience for mobile before designing it for the desktop
- It's easier to translate a mobile design to desktop than the other way around



## Responsive Web Design

---

## Mobile First Web Design





# Meta viewport

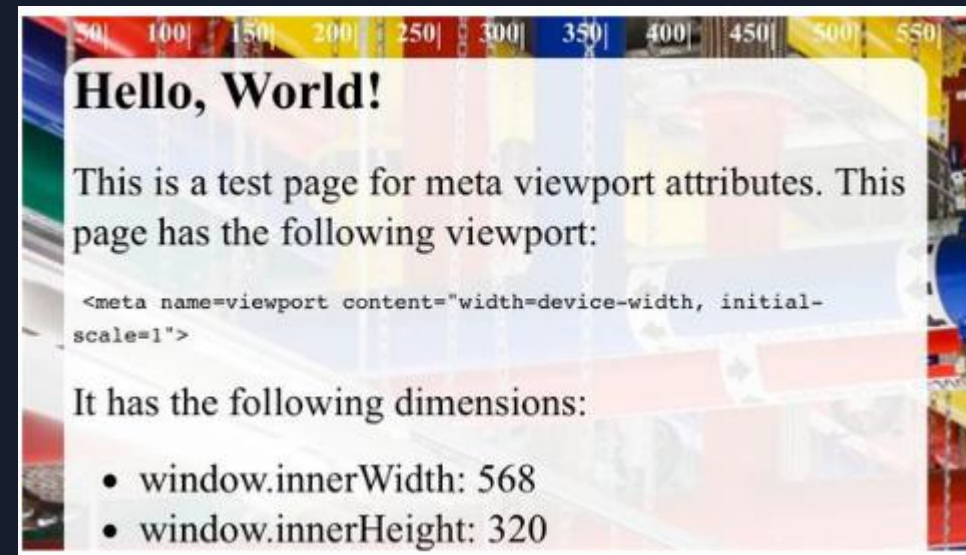
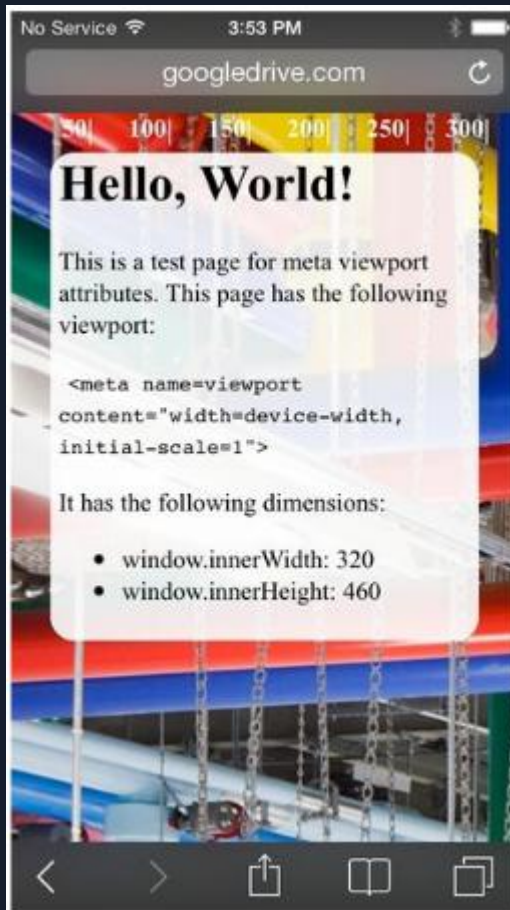
- Pages optimized to display well on mobile devices should include a meta viewport in the head of the document
- Gives the browser instructions on how to control the page's dimensions and scaling
- Fixed-width or responsive
- Zoom level

# Typical setting

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

width of the page follows the screen-width of the device

initial zoom level when the page is first loaded by the browser



# References

- Centering in CSS

- <https://css-tricks.com/centering-css-complete-guide/>

- Floats

- <https://css-tricks.com/all-about-floats/>

- Positioning tutorials

- <http://alistapart.com/article/css-positioning-101>

- <http://learn.shayhowe.com/advanced-html-css/detailed-css-positioning/>