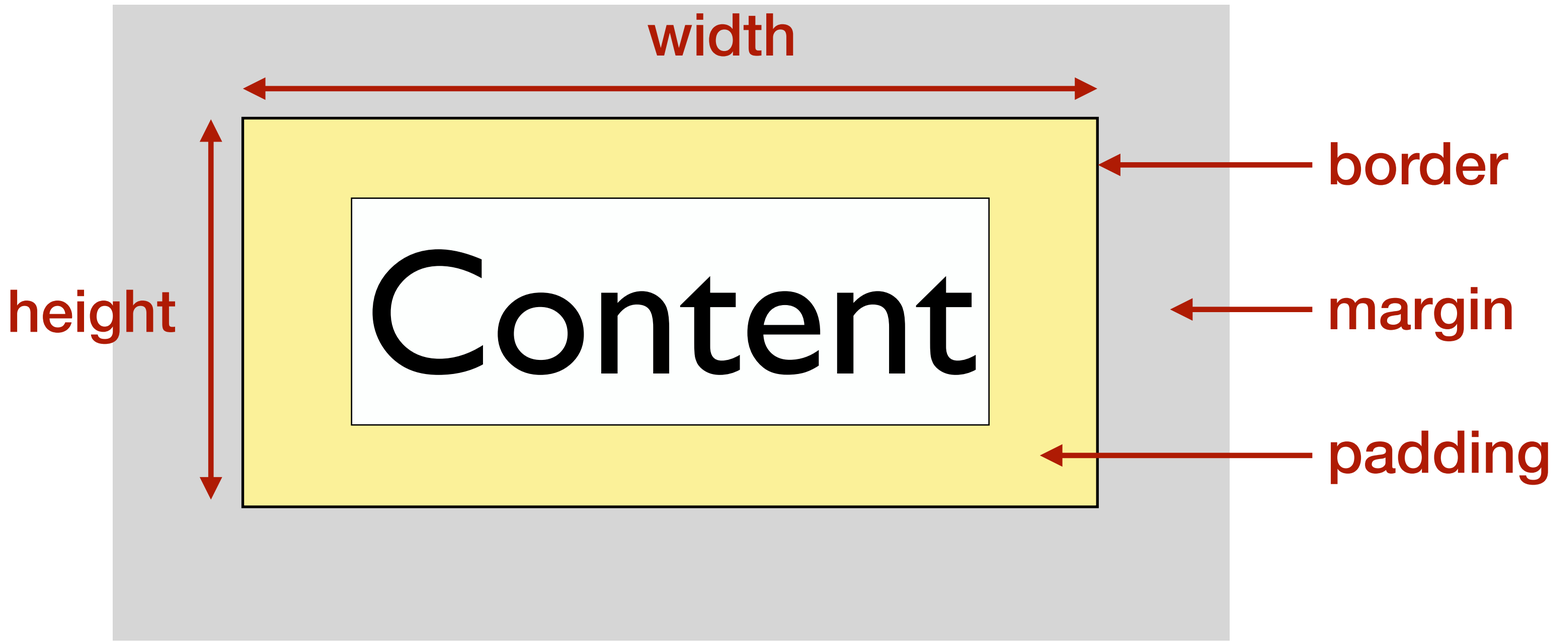
Web Programming
CSS Part III.

Krisztian Balog | University of Stavanger

Part III

Positioning

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>, …`**

block level

inline

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

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

Example: inline

CSS

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

has no effect (width of inline elements is ignored)

Example: inline-block

CSS

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

has effect

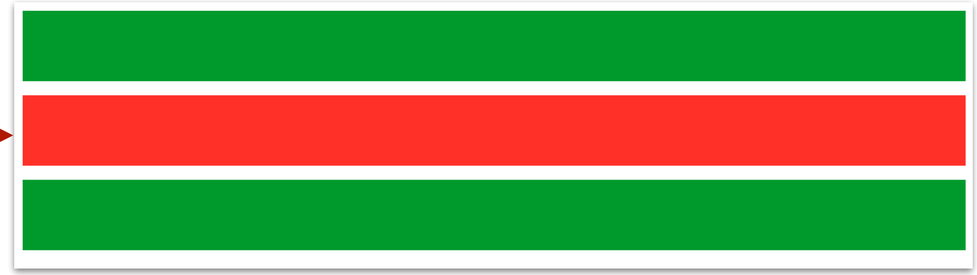
Visibility

- **`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

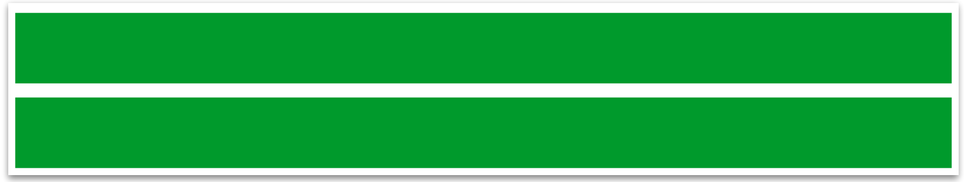
HTML

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



CSS

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



CSS

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

Exercise #1

https://github.com/kbalog/web-programming/tree/master/
exercises/css/positioning

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

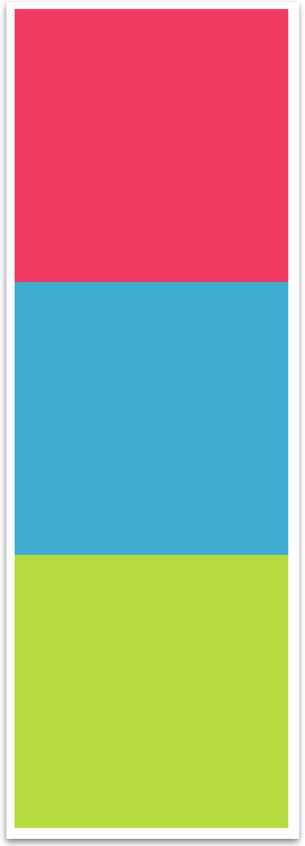
Example: normal flow

HTML

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

CSS

```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.

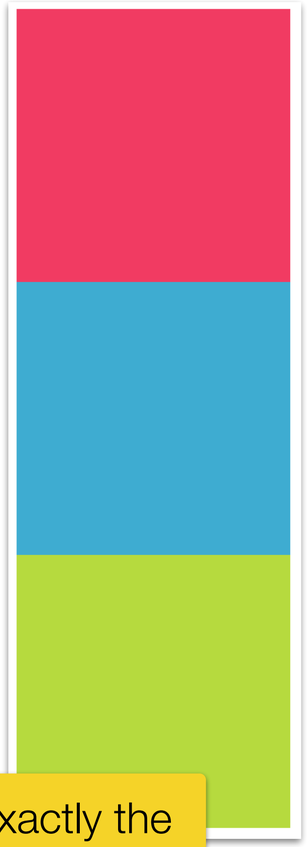
Example

HTML

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

CSS

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



No offset defined, so far it behaves exactly the same way as statically positioned elements.

Example

examples/css/positioning/position_relative.html

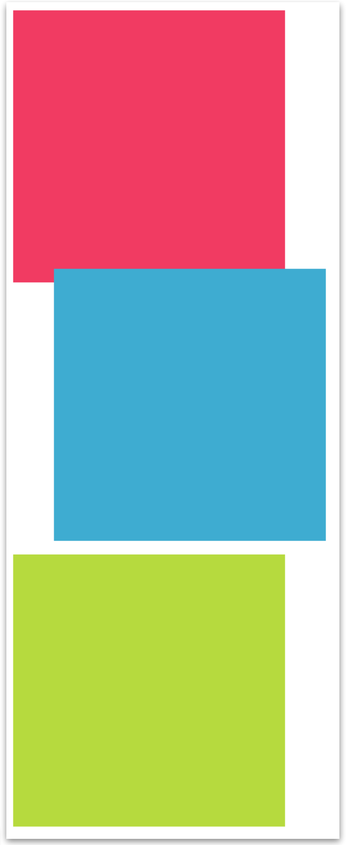
HTML

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

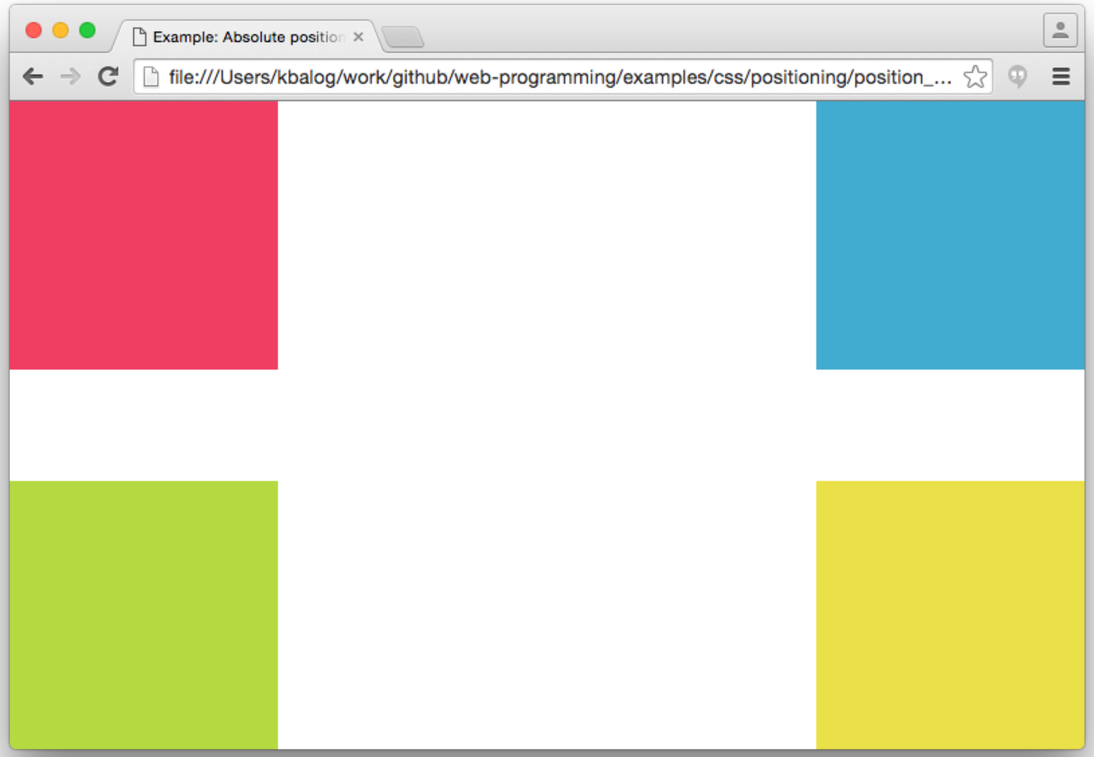
- **`position: absolute`**
- Element's position is set with respect to its containing element
  - That is the first parent element with a position other than static
- Set **`top, bottom, left,`** or **`right`**
  - in pixels, percentages, or em
- Element is taken out of the normal flow (no longer affects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

Example

examples/css/positioning/position_absolute.html

CSS

```css
#box_1 {
    background: #ee3e64;
    position: absolute;
    top: 0;
    left: 0;
}
#box_2 {
    background: #44accf;
    position: absolute;
    top: 0;
    right: 0;
}
#box_3 {
    background: #b7d84b;
    position: absolute;
    bottom: 0;
    left: 0;
}
#box_4 {…}
```



Example #2

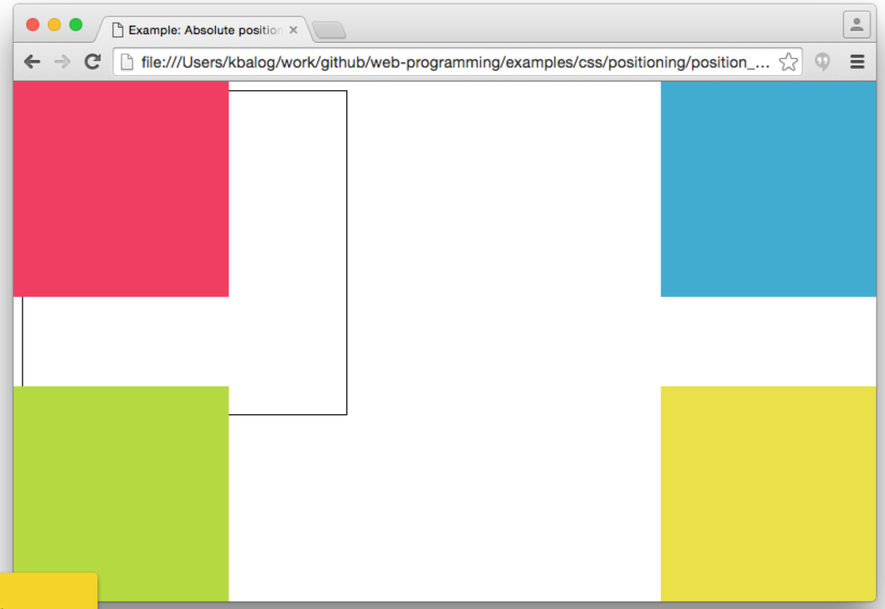
- Absolute positioning is specifies relation with respect to the parent element (first element with non-static position)!

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;
}
```



No position defined for #container (i.e., static).

Example #2

examples/css/positioning/position_absolute2.html

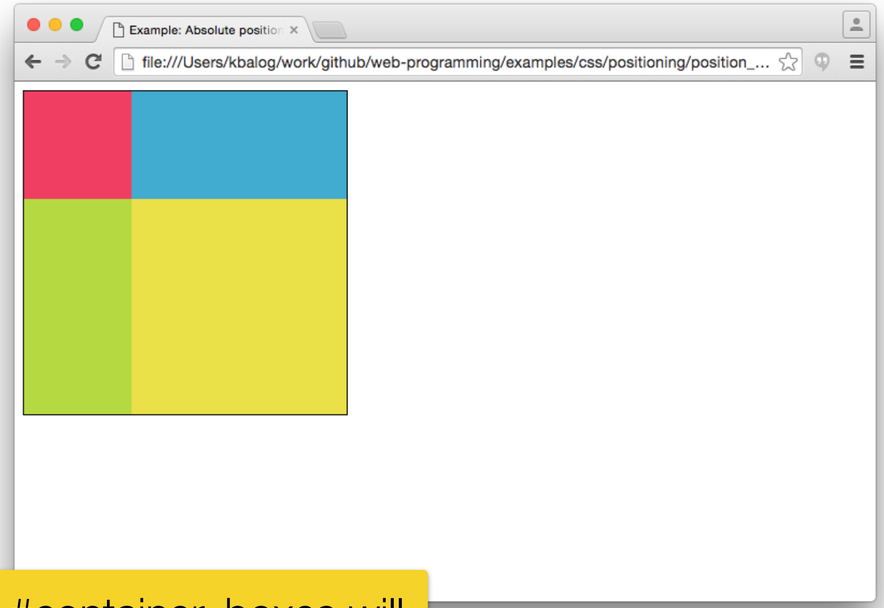
- Absolute positioning is specifies relation with respect to the parent element (first element with non-static position)!

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;
}
```



Non-static position for #container, boxes will be positioned with respect to this.

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

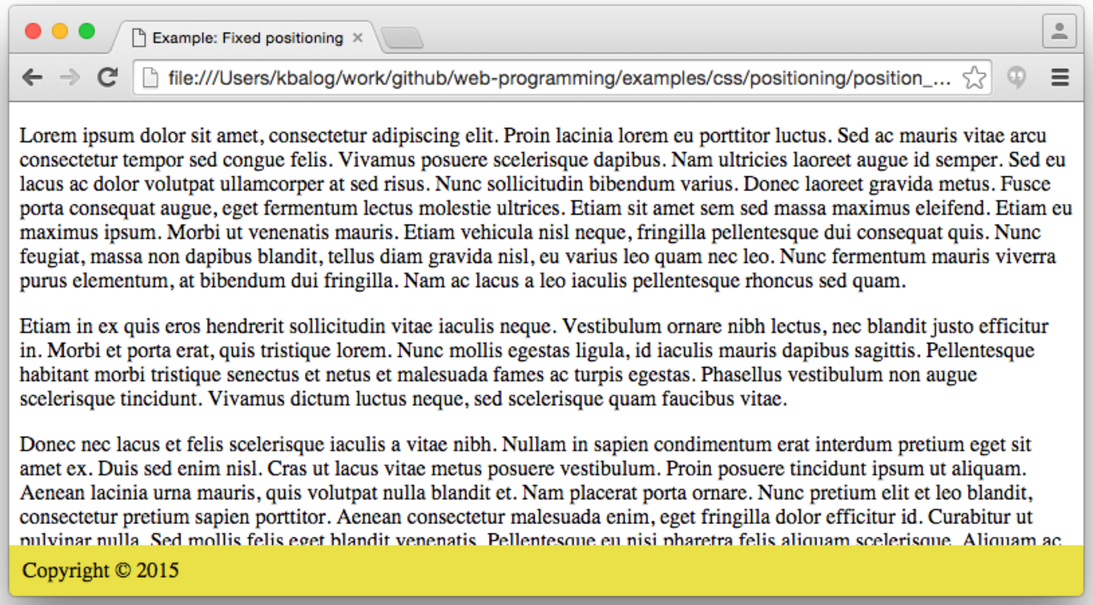
examples/css/positioning/position_fixed.html

HTML

```
<p> … </p>
<div id="footer">Copyright
&copy; 2015</div>
```

CSS

```
#footer {
    background: #ebde52;
    position: fixed;
    left: 0;
    bottom: 0;
    padding: 10px;
    width: 100%;
}
```



Exercise #2

https://github.com/kbalog/web-programming/tree/master/
exercises/css/positioning

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

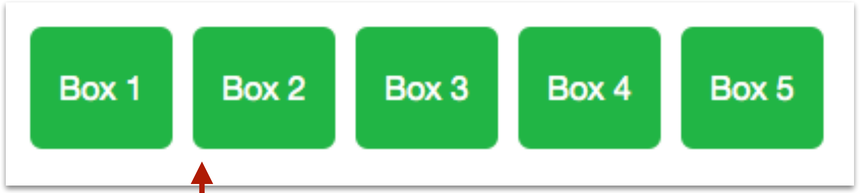
Example

HTML

```
<div class="box-set">
    <div class="box">Box 1</div>
    …
    <div class="box">Box 5</div>
</div>
```

CSS

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



The box-set container is supposed to have a colored background?!

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

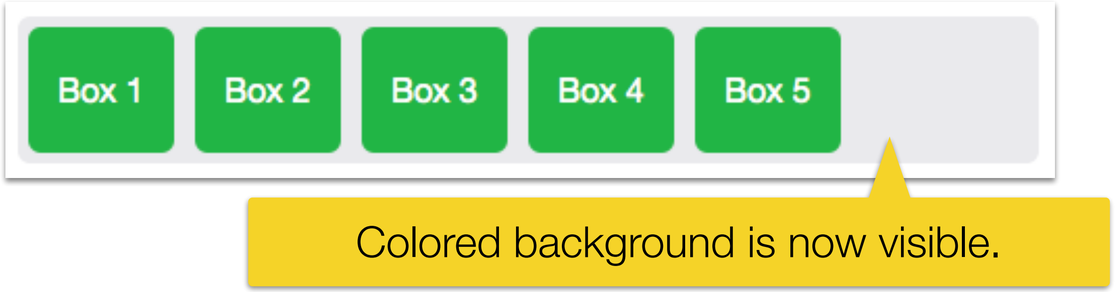
Example

HTML

```
<div class="box-set">
    <div class="box">Box 1</div>
    …
    <div class="box">Box 5</div>
</div>
```

CSS

```
.box-set {
    background: #eaeaed;
    overflow: auto;
    width: 100%;
}
```



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

Clearing floats

- Disallows floating elements from overlapping other elements
- Property: **`clear`**
- Values
  - **`none`** — elements can touch either side (default)
  - **`left`** — no floating elements allowed on the left side
    - I.e., left-hand side of the box should not touch any other elements appearing in the same containing element
  - **`right`** — no floating elements allowed on the right side
  - **`both`** — no floating elements allowed on either the left or the right side

Example

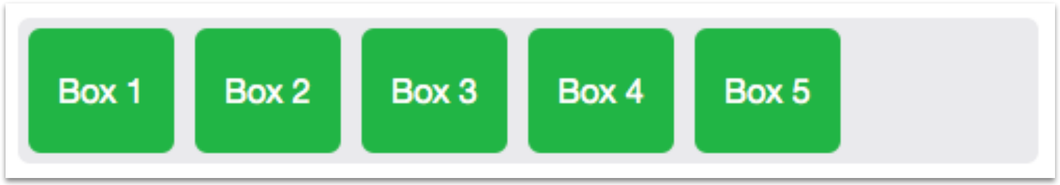
clear: none

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: none;
}
```



Example

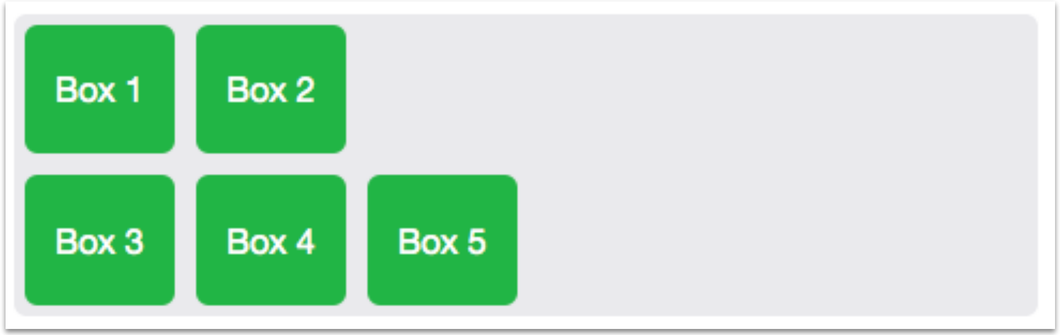
clear: left

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!

Stacking elements

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

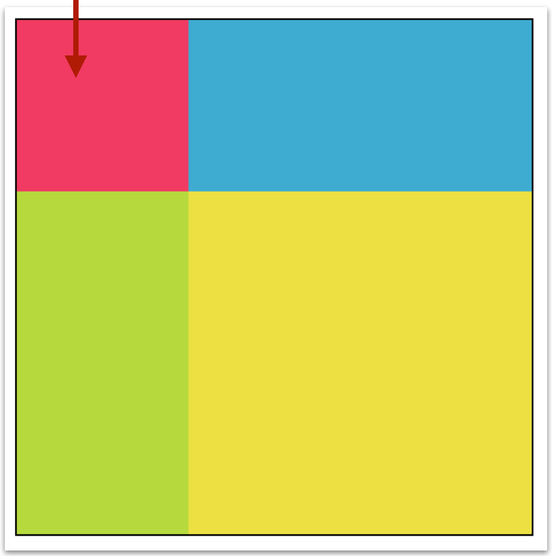
```
z-index: 3;
```

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

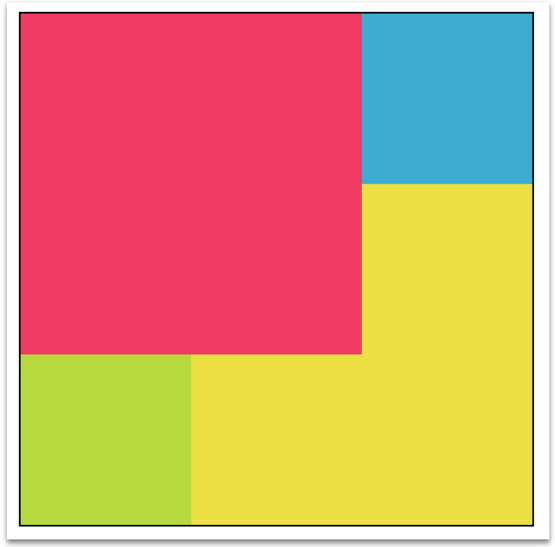
Example

examples/css/positioning/z_index.html

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



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



Exercise #3

https://github.com/kbalog/web-programming/tree/master/
exercises/css/positioning

Some common issues

Center align block element

examples/css/positioning/center_horizontal.html

- To horizontally center a block element (like ** **), use

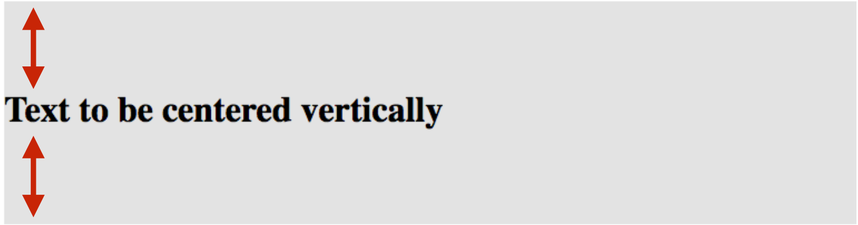
```
margin: auto;
```

- 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

Vertical centering of text

examples/css/positioning/center_vertical.html

- 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

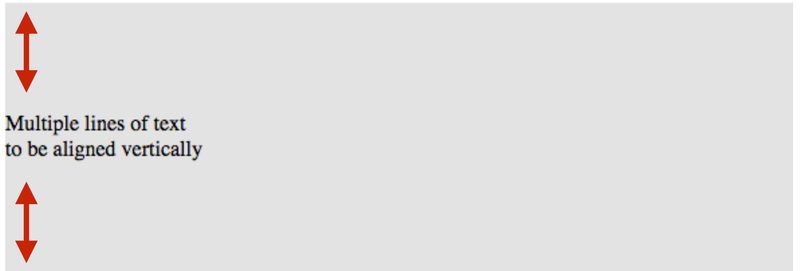
```
div {
    height: 200px;
}

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

Vertical centering of text

examples/css/positioning/center_vertical.html

- 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

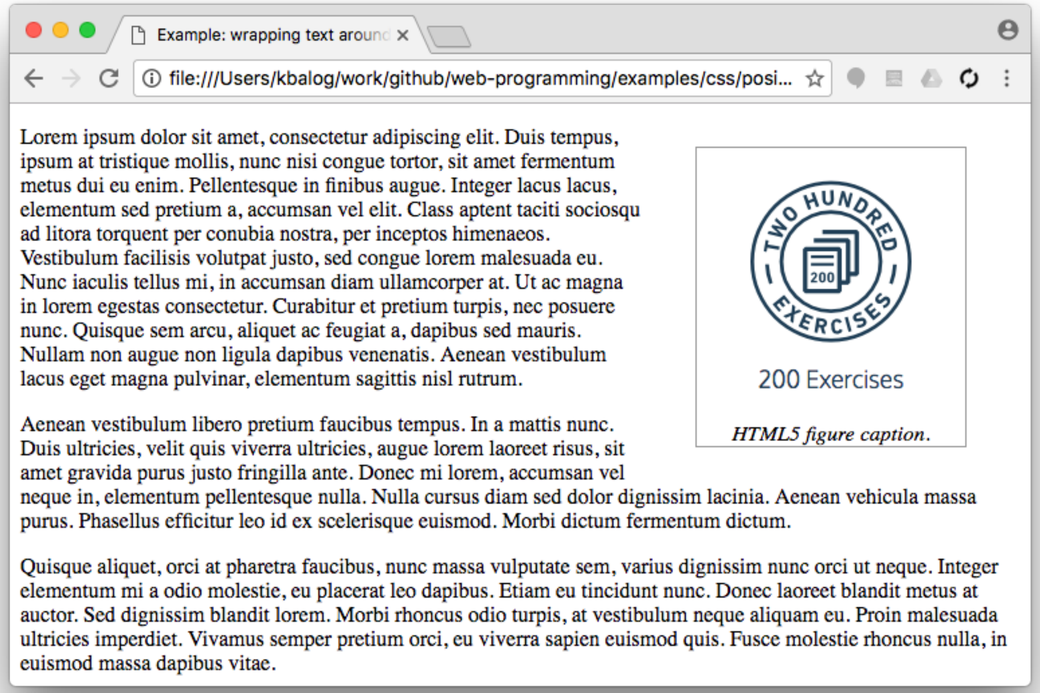
```
div {
    height: 200px;
}

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

Wrap text around image

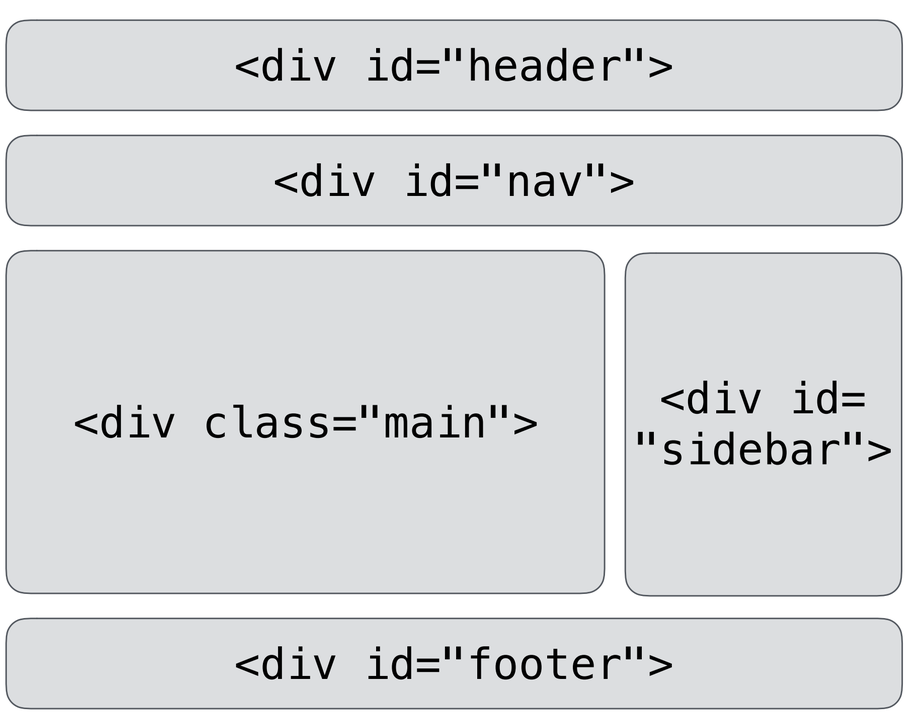
examples/css/positioning/wrap_image.html

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

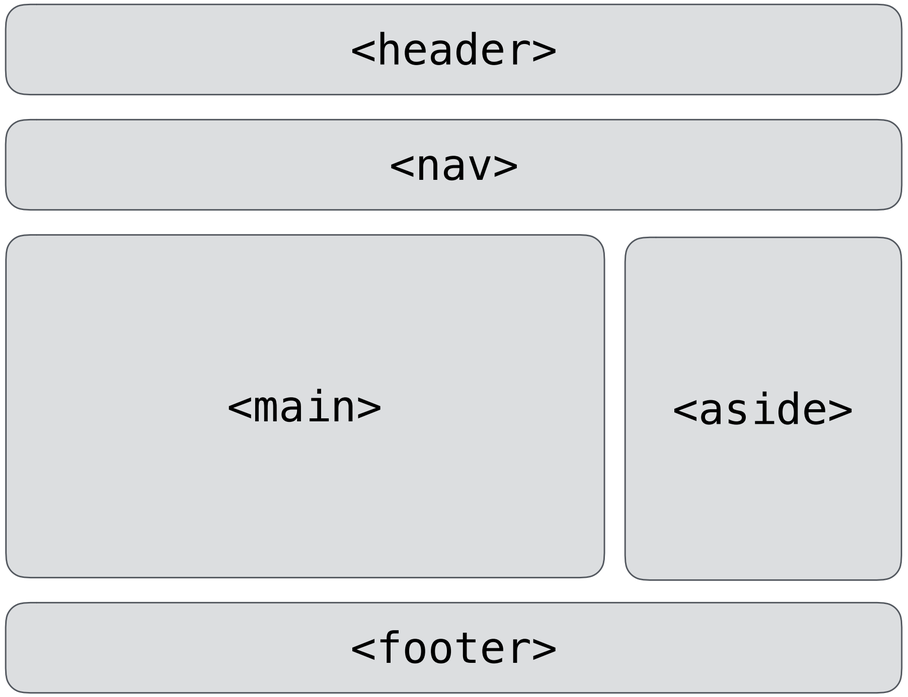


Layouts

Page sections

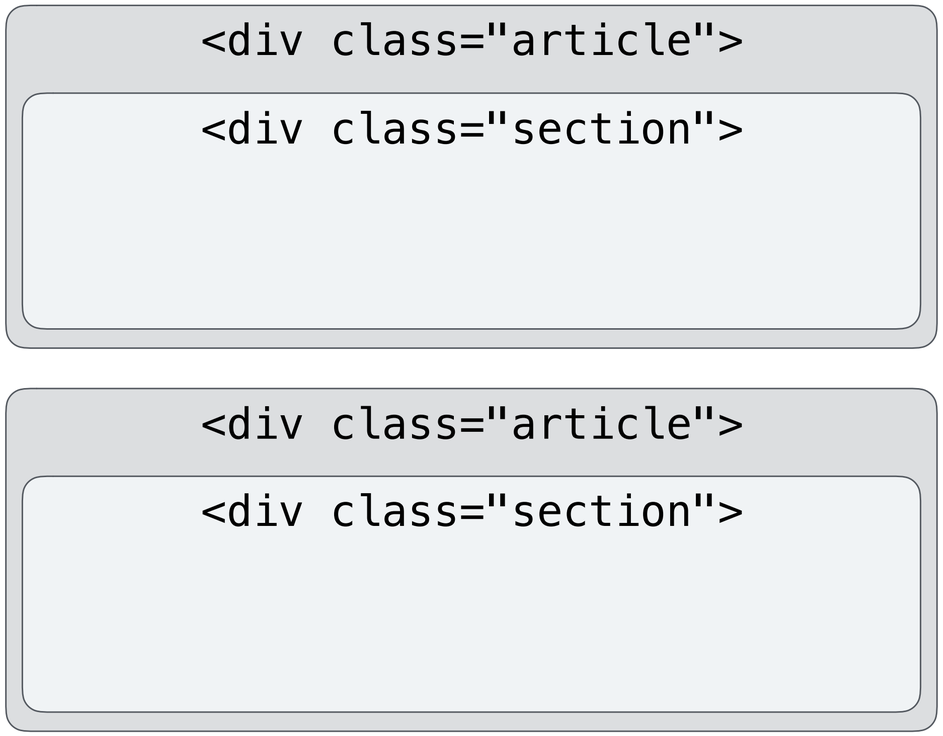


Classic HTML

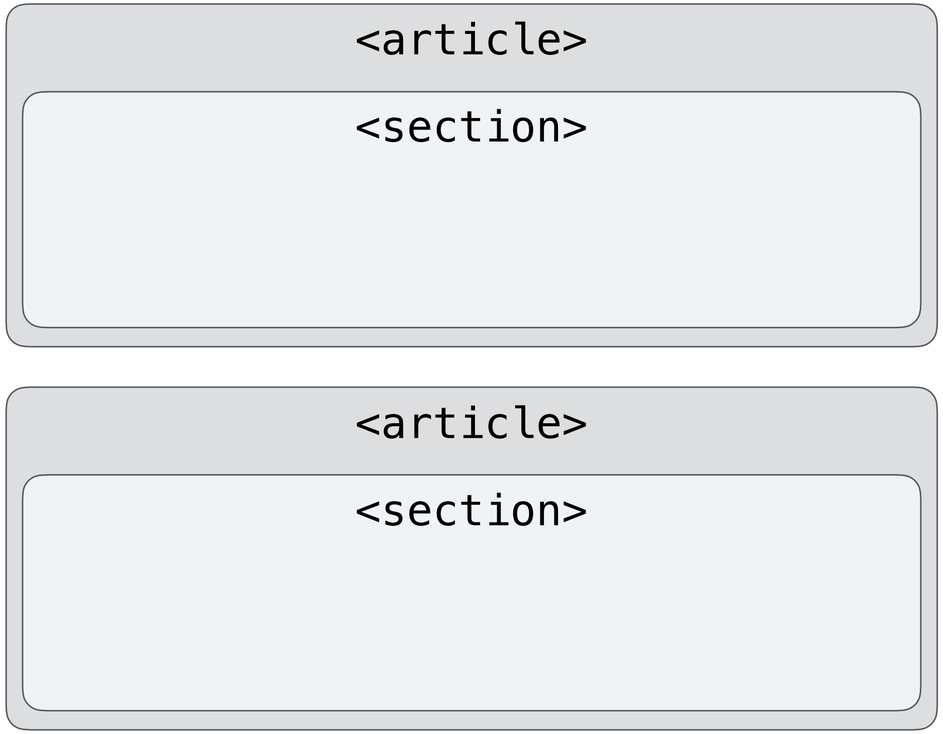


HTML5

Page sections



Classic HTML



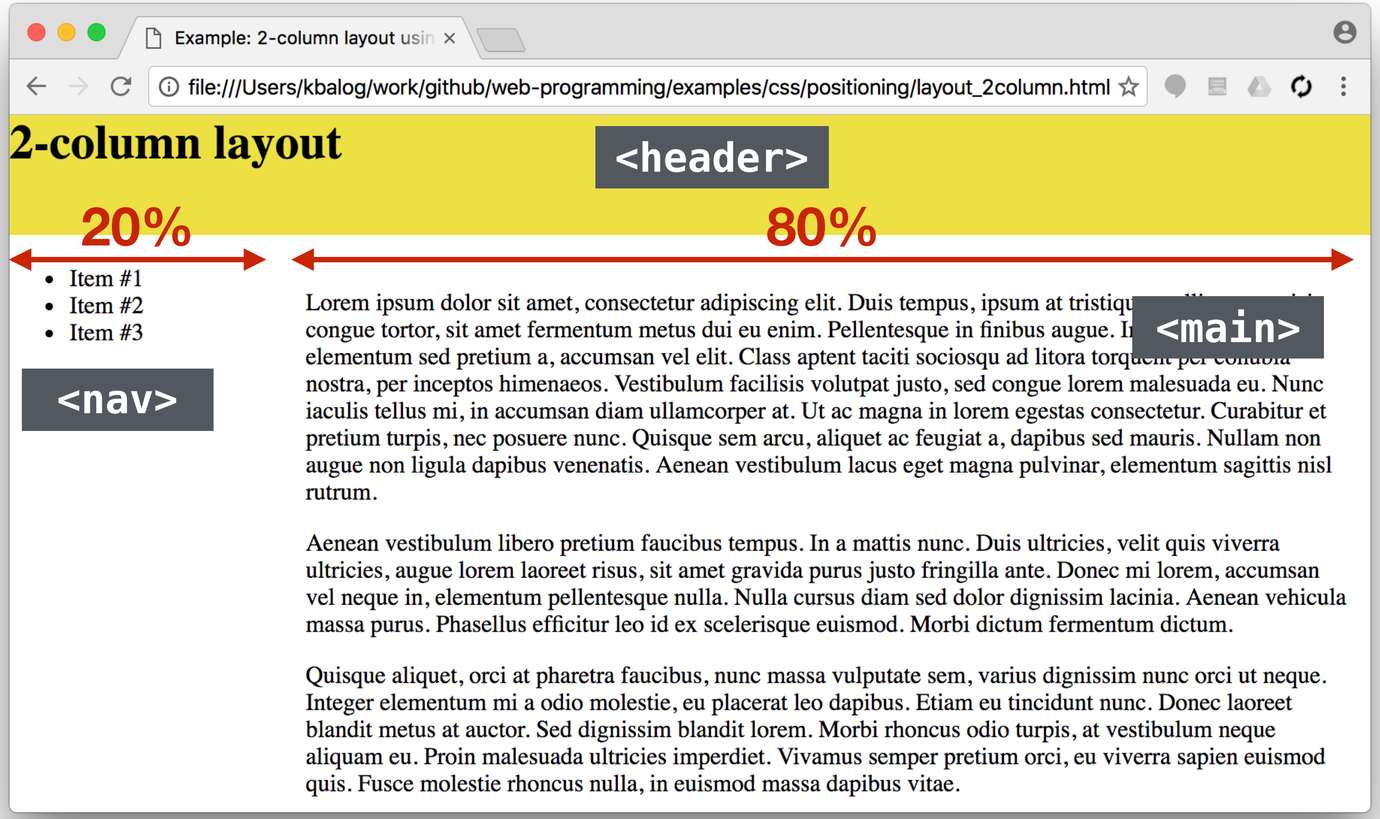
HTML5

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

examples/css/positioning/layout_2column.html



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;
    }
}
```

Change the background-color if the document is smaller than 300 pixels wide.

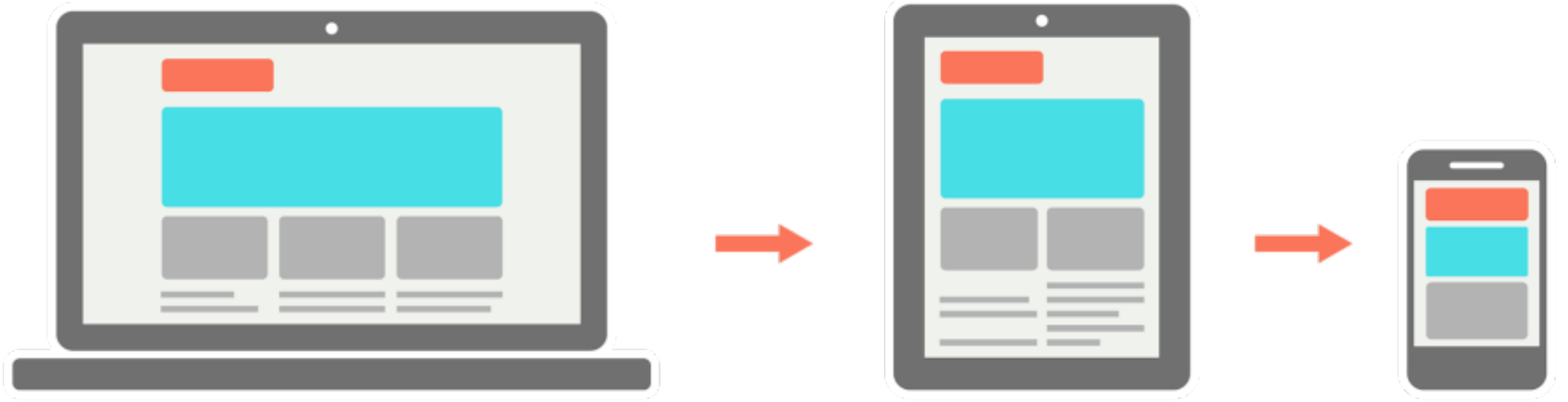
- Also possible to have different style files for different media

```
<link rel="stylesheet" media="mediatype and|not|only (media
feature)" href="mystylesheet.css">
```

- See http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

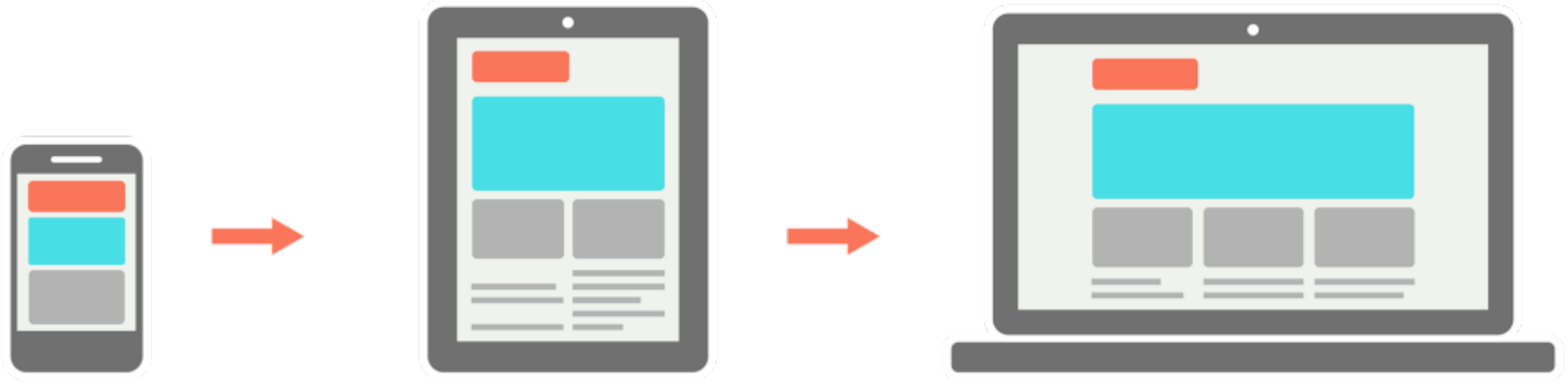
Mobile first

- 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

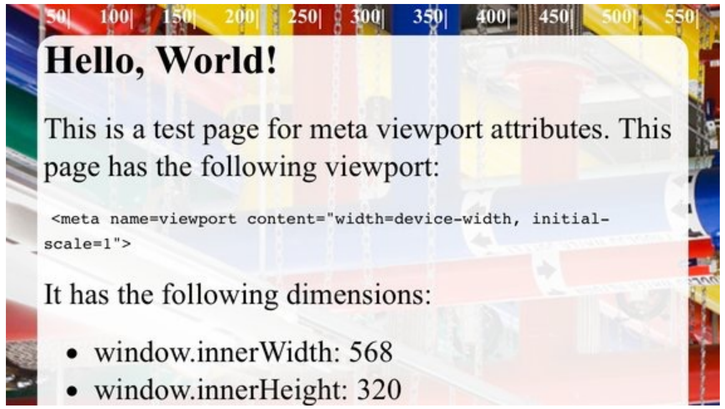
- Pages optimized to display well on mobile devices should include a meta viewport in the head of the document
- Gives the browser instructions on how to control the page's dimensions and scaling
  - Fixed-width or responsive
  - Zoom level

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



Exercise #4

https://github.com/kbalog/web-programming/tree/master/
exercises/css/positioning

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/