CSS Layout

Arrangement and Positioning of the HTML Elements

Au Mau Duong
Technical Trainers

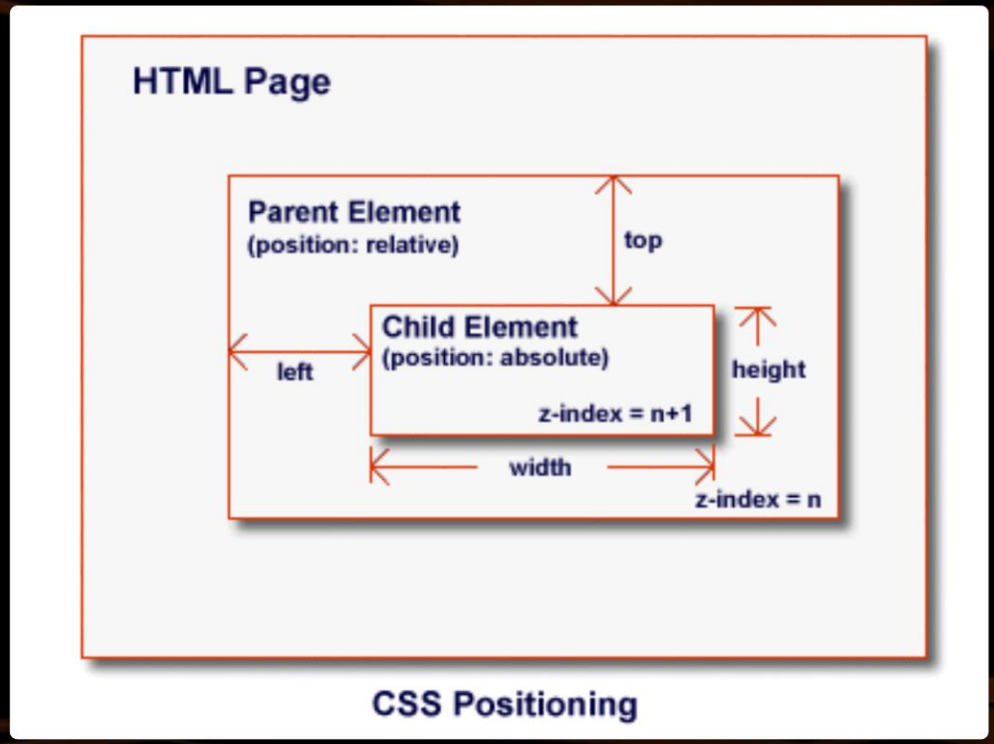


Table of Contents

Width and Height

Width

- **`width`** – defines a value for the width of element, e.g. **`200px`**
- **`width`** applies only for block elements
  - Their with is 100% by default (the width of their parent)
  - The width of inline elements is the width of their content
- **`min-width`** – defines the minimal width
  - **`min-width`** overrides width if (**`width < min-width`**)
- **`max-width`** – defines the maximal width
  - **`max-width`** overrides width if (**`width > max-width`**)

Width Values

- The values of the **width** / **height** properties are numerical:
  - Pixels (**px**)
  - Centimeters (**cm**)
  - Or percentages (**%**)
    - A percent of the available width (of the parent container)
    - Set the **max-width** to 100% to scale-down images (for tablets)

```
img {
    max-width: 100%;
}
```

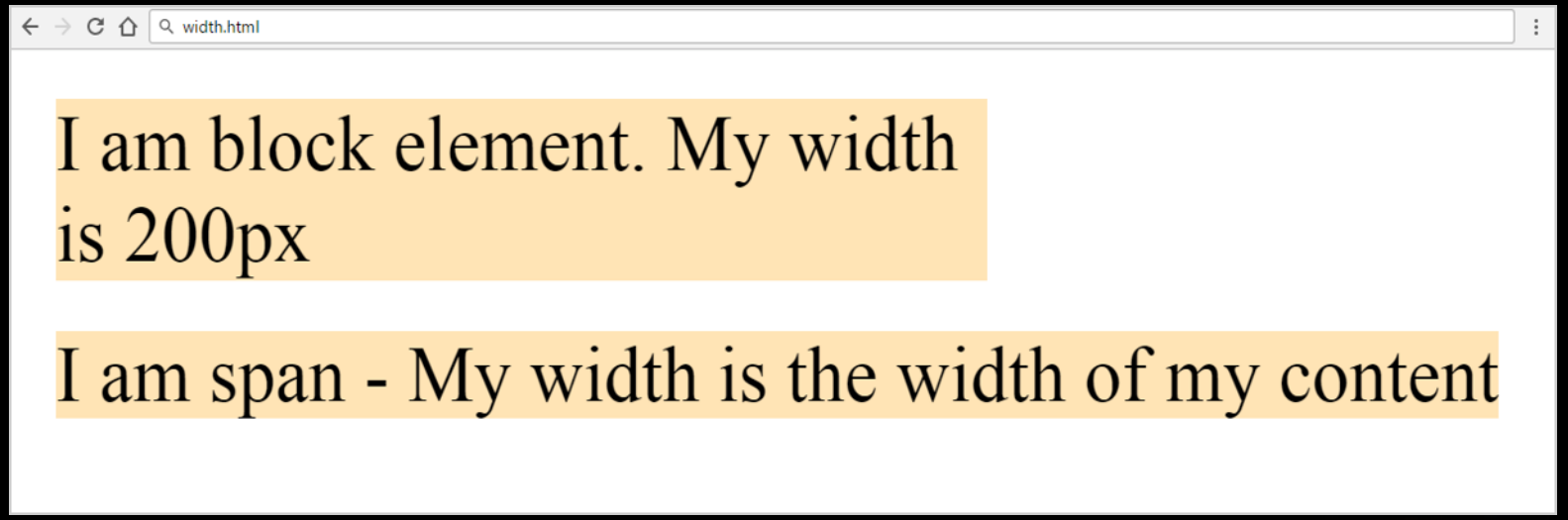
Width, Max-Width, Min-Width

- Block elements have width / height
- Inline elements don't have

```
div, span { width: 200px; }
```

```
p {
  width: 50%;
  min-width: 120px;
  max-width: 220px;
}
```

```
<div>
 I am block …
</div>
<span>
 I am span …
</span>
```



Height

- `height` – defines the height of element, e.g. `100px`
  - `height` applies only on block elements
  - The height of inline elements is the height of their content
- `min-height` – defines the minimal height
  - `min-height` overrides `height`
- `max-height` – defines the maximal height
  - `max-height` overrides `height`

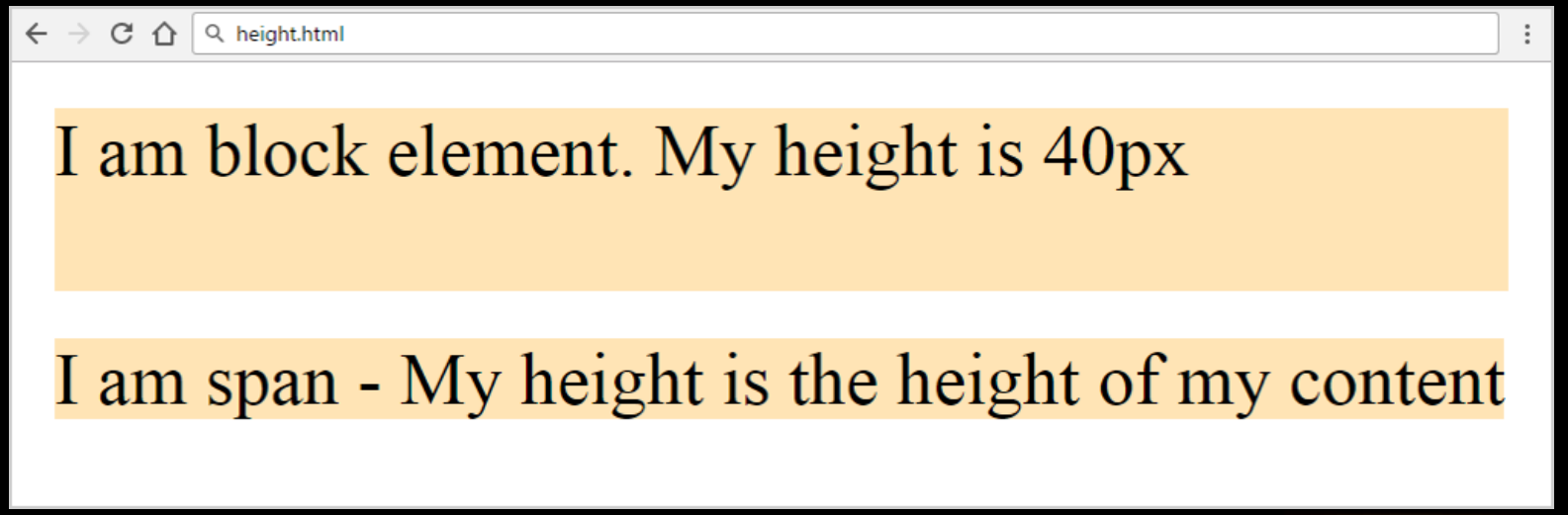
Height, Min-Height, Max-Height

- Block elements have width / height
- Inline elements don't have

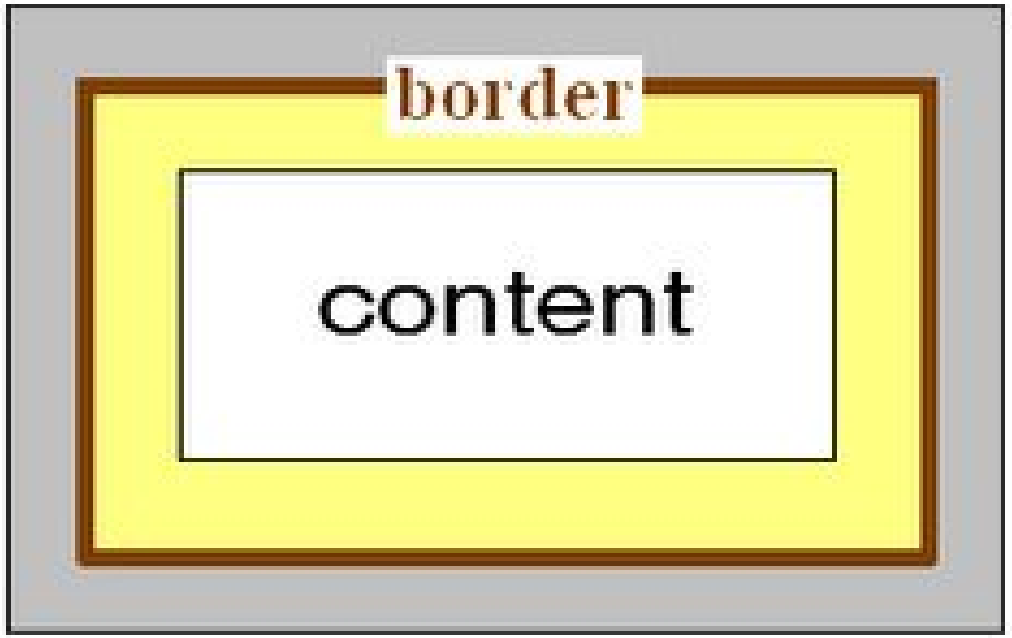
```
div, span { height: 40px; }
```

```
p {
  height: 50%;
  min-height: 50px;
  max-height: 100px;
}
```

```
<div>
 I am block …
</div>
<span>
 I am span …
</span>
```



Margins and Paddings



- margin

- padding

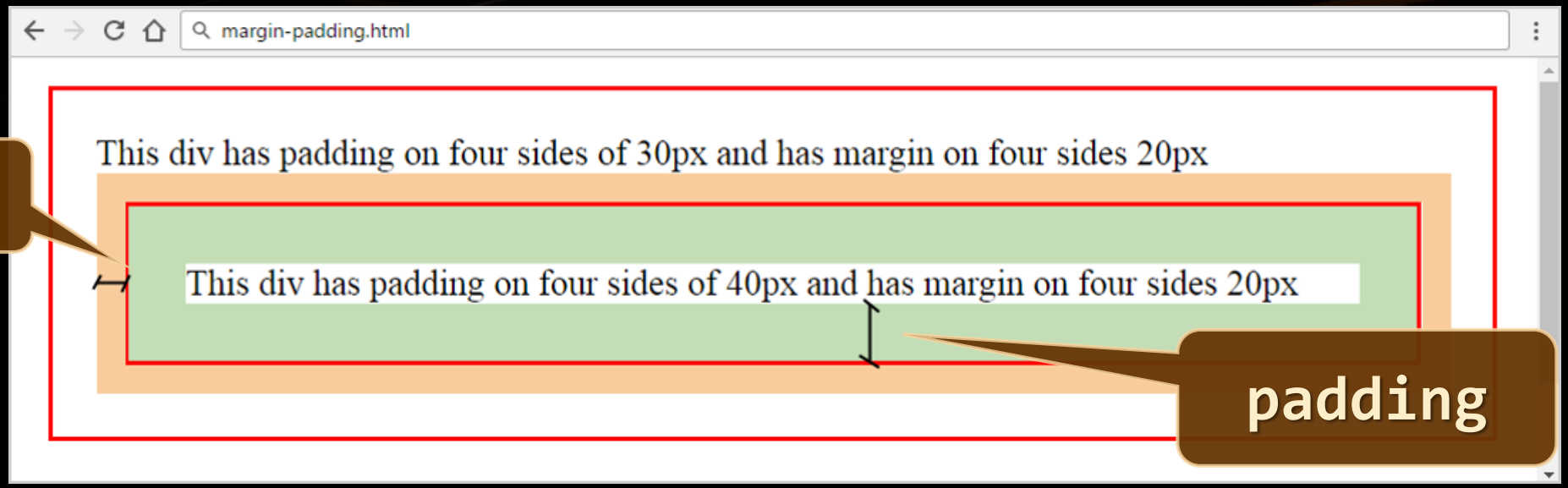
Margin and Padding

- `margin` and `padding` define the spacing around the element
  - `margin` is the spacing outside of the border
  - `padding` is the spacing between the border and the content
- Numerical value, e.g. `10px` or `-5px`
- Can be defined for each of the four sides separately:
  - `margin-top`, `padding-left`, …
- Sometimes vertical margins of adjacent elements may collapse
  - E.g. adjacent paragraphs / parent-child elements / empty blocks

Margin and Padding: Short Rules

- `margin: 5px;`
  - Sets all four sides to have margin of 5px;
- `margin: 10px 20px;`
  - top and bottom to 10px, left and right to 20px;
- `margin: 5px 3px 8px;`
  - top 5px, left/right 3px, bottom 8px
- `margin: 1px 3px 5px 7px;`
  - top, right, bottom, left (clockwise from top)
- Same for `padding`

Margins and Paddings



margin

padding

margin-top

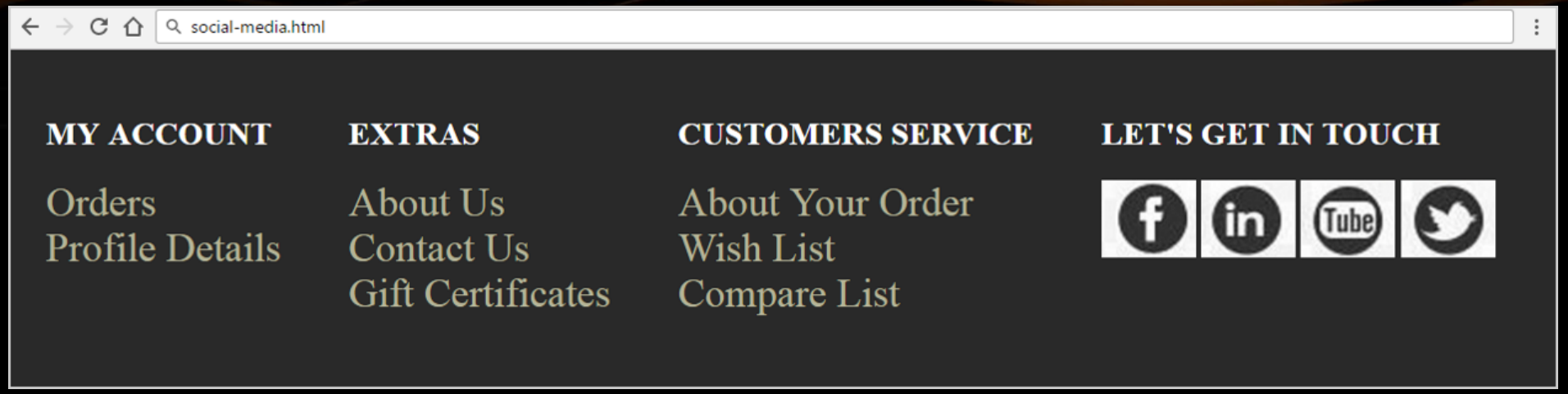
margin-right

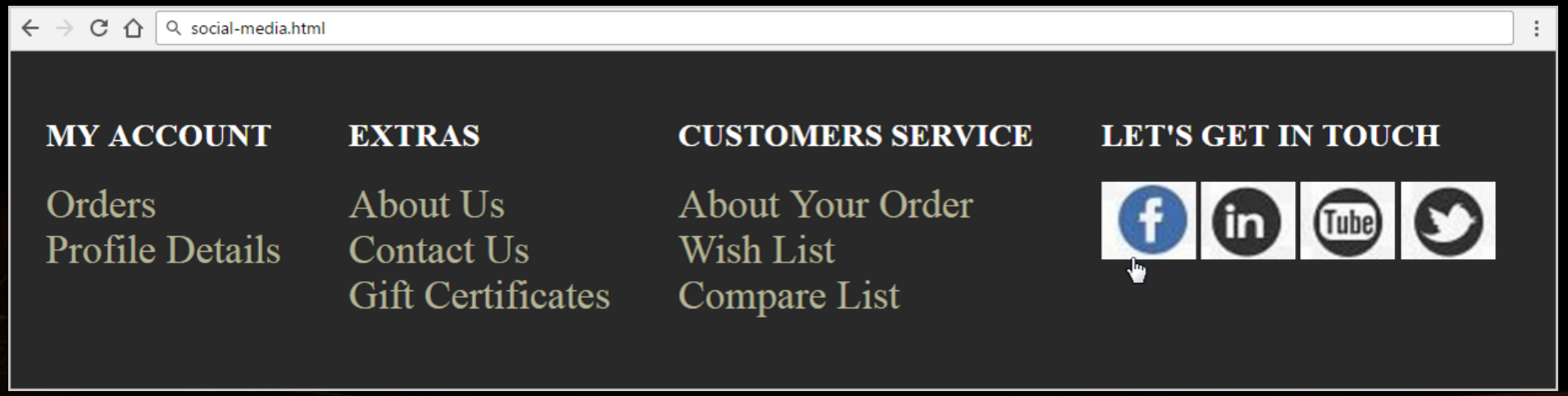
margin-bottom

margin-left

```
div { margin: 20px 20px 20px 20px
      padding: 30px 30px 30px 30px }
```

Problem: Social Media





Solution: Social Media – HTML (1)

```
<footer>
  <section>
    <h2>my account</h2>
    <ul>
      <li><a href="#">orders</a></li>
      <li><a href="#">profile details</a></li>
    </ul>
  </section>
  <!-- TODO: put the rest <section> here -->
    <!-- TODO: put the rest <ul> here -->
      <!-- TODO: put the rest <li> here -->
```

Solution: Social Media – HTML (2)

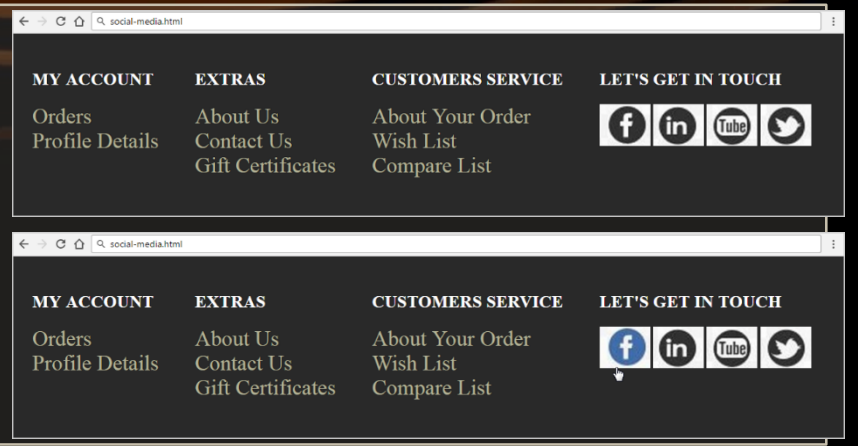
```
  <section>
    <h2>let's get in touch</h2>
      <div>
        <div class="facebook"><a href="#"></a></div>
        <div class="linkedin"><a href="#"></a></div>
        <div class="youtube"><a href="#"></a></div>
        <div class="twitter"><a href="#"></a></div>
      </div>
  </section>
</footer>
```

Solution: Social Media – CSS (1)

```
ul, li, div, section, a {
  margin: 0;
  padding: 0; }
body {
 background: rgb(41, 41, 41); }
footer {
  height: 150px;
  margin: 0 auto; }
h2 {
  text-transform: uppercase; }
section {
  float: left;
  margin-left: 10px;
  padding: 20px;
  height: 150px;
  color: #ffffff;
}
ul {
  list-style: none;
}
a {
  text-decoration: none;
  color: rgb(181, 180, 152);
  font-size: 30px;
  text-transform: capitalize;
}
```

Solution: Social Media – CSS (2)

```
footer > section > div div {
  width: 70px;
  height: 58px;
  background: url("images/social.jpg");
  display: inline-block; }
```




```
.twitter { background-position: 772px 126px; }
.linkedin { background-position: 642px 126px; }
.youtube { background-position: 325px 126px; }
.facebook, .twitter, .youtube, .linkedin { cursor: pointer; }
.facebook:hover { background-position: 0 60px; }
.linkedin:hover { background-position: 642px 63px; }
.youtube:hover { background-position: 325px 63px; }
.twitter:hover { background-position: -66px 63px; }
```

Overflow

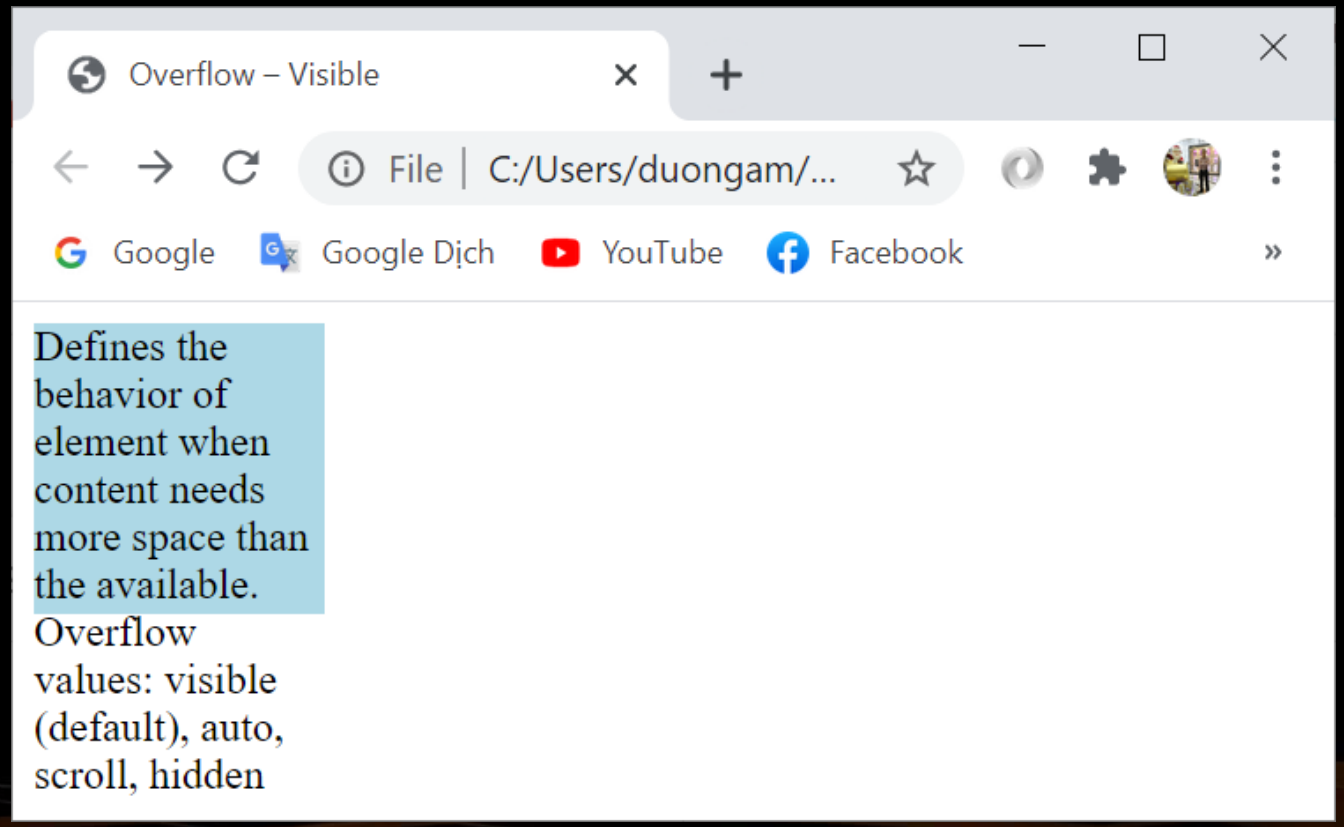
Overflow

- Defines the behavior of element when content needs more space than the available
- `overflow` values:
  - `visible` (default) – content spills out of the element
  - `auto` – show scrollbars if needed
  - `scroll` – always show scrollbars
  - `hidden` – any content that cannot fit is clipped

Overflow – visible

```
<div class="visible"> Defines the behavior of element when content needs more space than the available. Overflow values:  visible (default), auto, scroll, hidden </div>
```

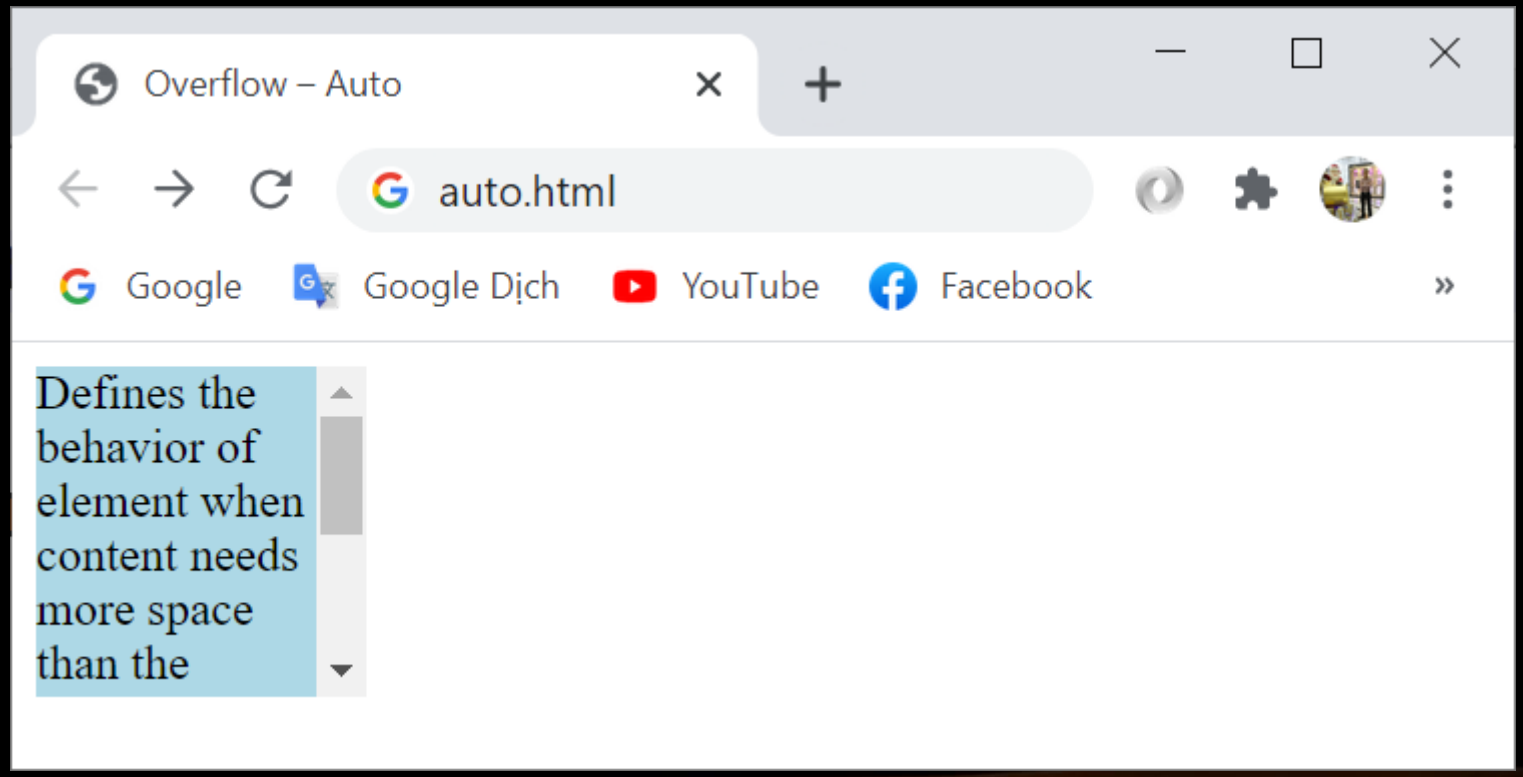
```
<style>
div.visible {
  background-color: lightblue;
  width: 200px;
  height: 110px;
  overflow: visible;
}
</style>
```



Overflow – auto

```
<div class="auto"> Defines the behavior of element when content needs more space than the available. Overflow values:  visible (default), auto, scroll, hidden </div>
```

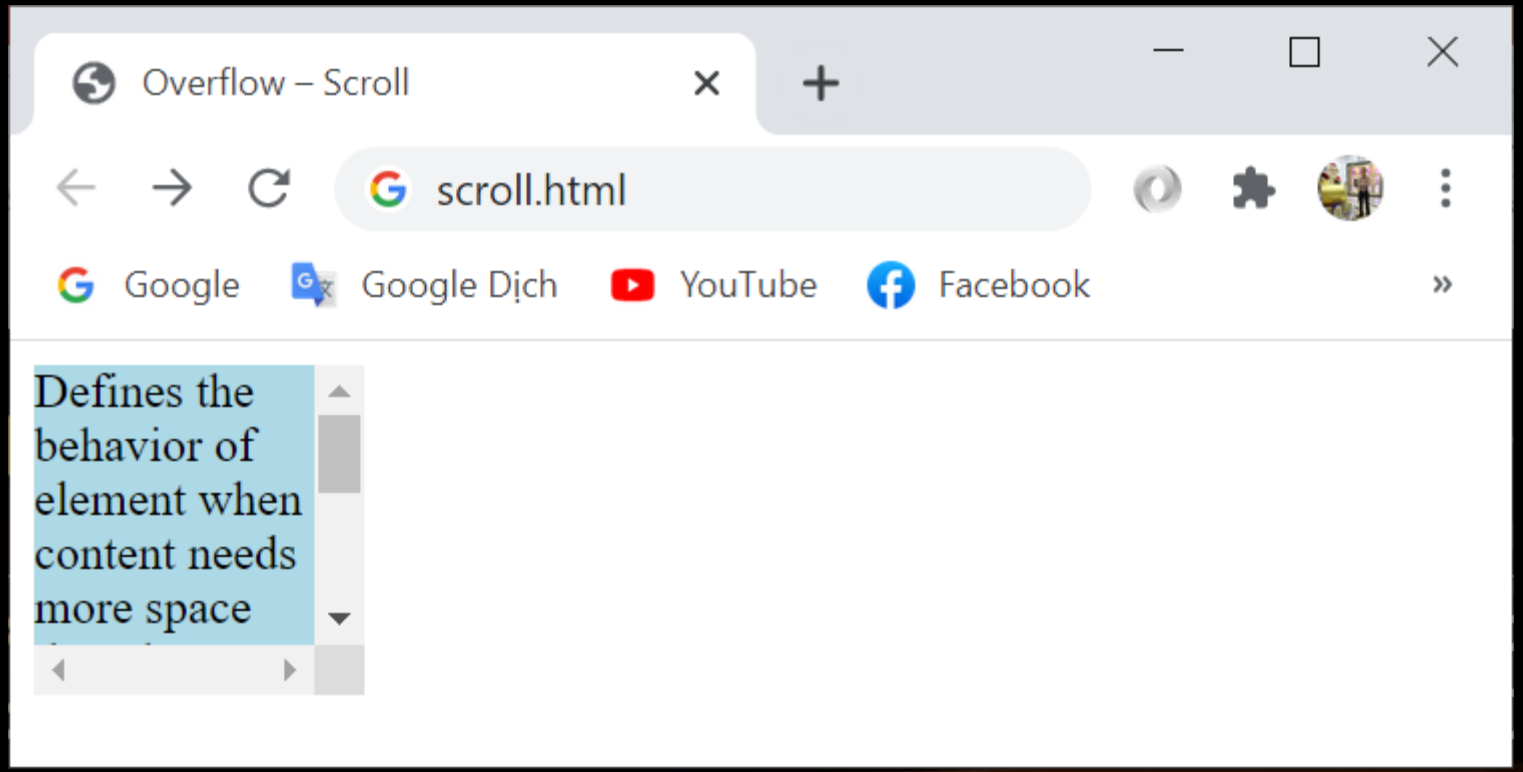
```
<style>
div.auto {
  background-color: lightblue;
  width: 110px;
  height: 110px;
  overflow: auto;
</style>
```



Overflow – scroll

```
<div class="scroll"> Defines the behavior of element when content needs more space than the available. Overflow values:  visible (default), auto, scroll, hidden </div>
```

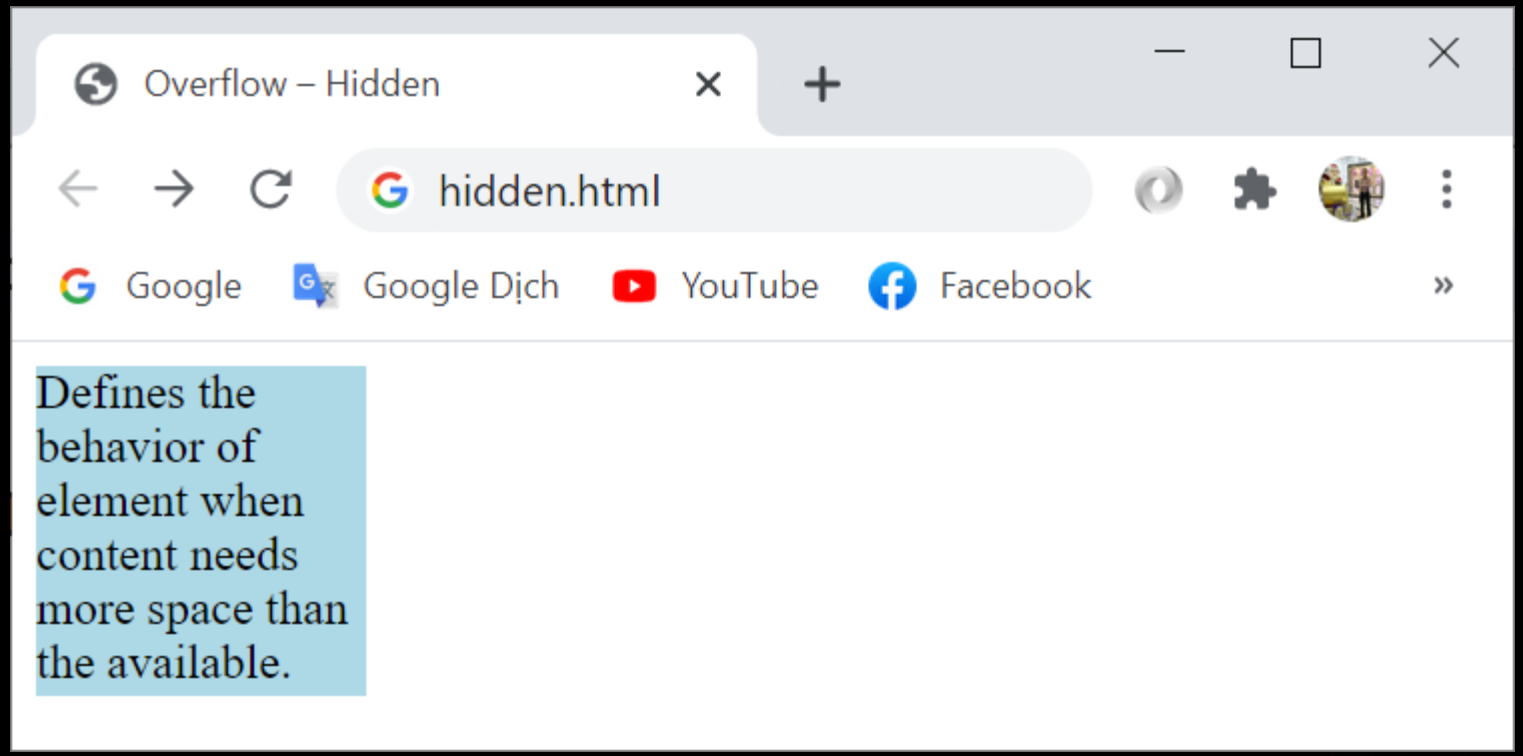
```
<style>
div.scroll {
  background-color: lightblue;
  width: 110px;
  height: 110px;
  overflow: scroll;
</style>
```



Overflow – hidden

```
<div class="hidden"> Defines the behavior of element when content needs more space than the available. Overflow values:  visible (default), auto, scroll, hidden </div>
```

```
<style>
div.hidden {
  background-color: lightblue;
  width: 110px;
  height: 110px;
  overflow: hidden;
</style>
```



Display

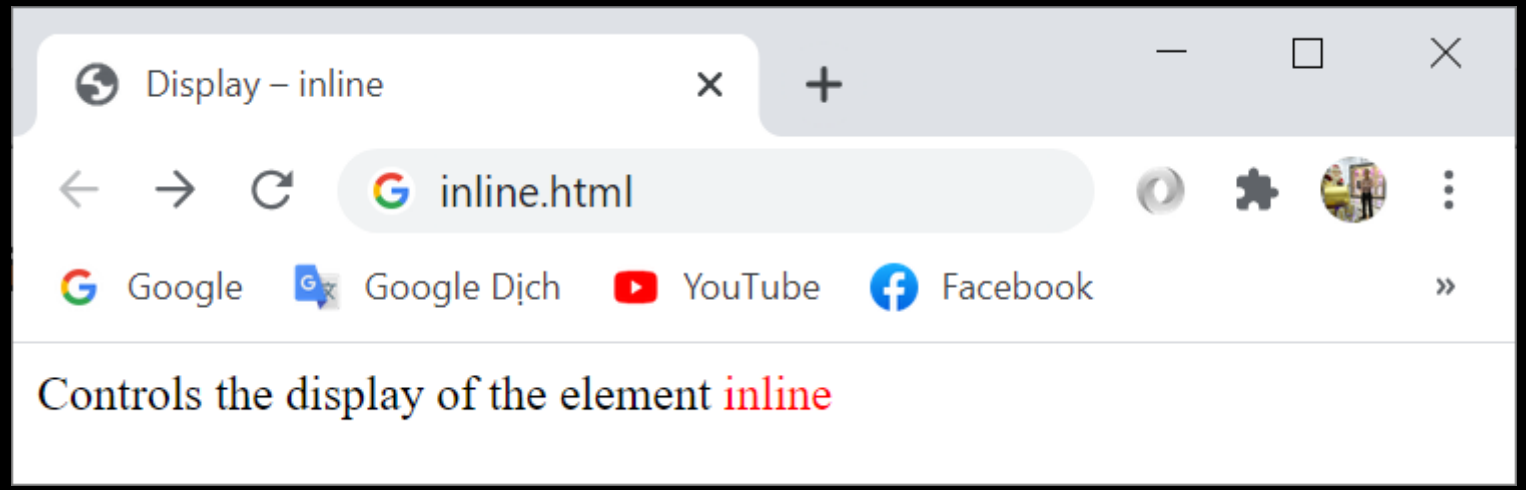
Display

- Controls the display of the element
  - Defines the way it is rendered and if breaks should be placed before and after the element
- **`inline`**: no breaks are placed before and after
  - Like typical **`<span>`** is element
  - The element's height and width depend on the size of the content
- **`block`**: breaks are placed before AND after the element
  - Like typical **`<div>`** is a block element
  - **`Height`** and **`width`** may not depend on the size of the content

Display – inline

```
<div> Controls the display of the element
      <p class="inline"> inline </p>
 </div>
```

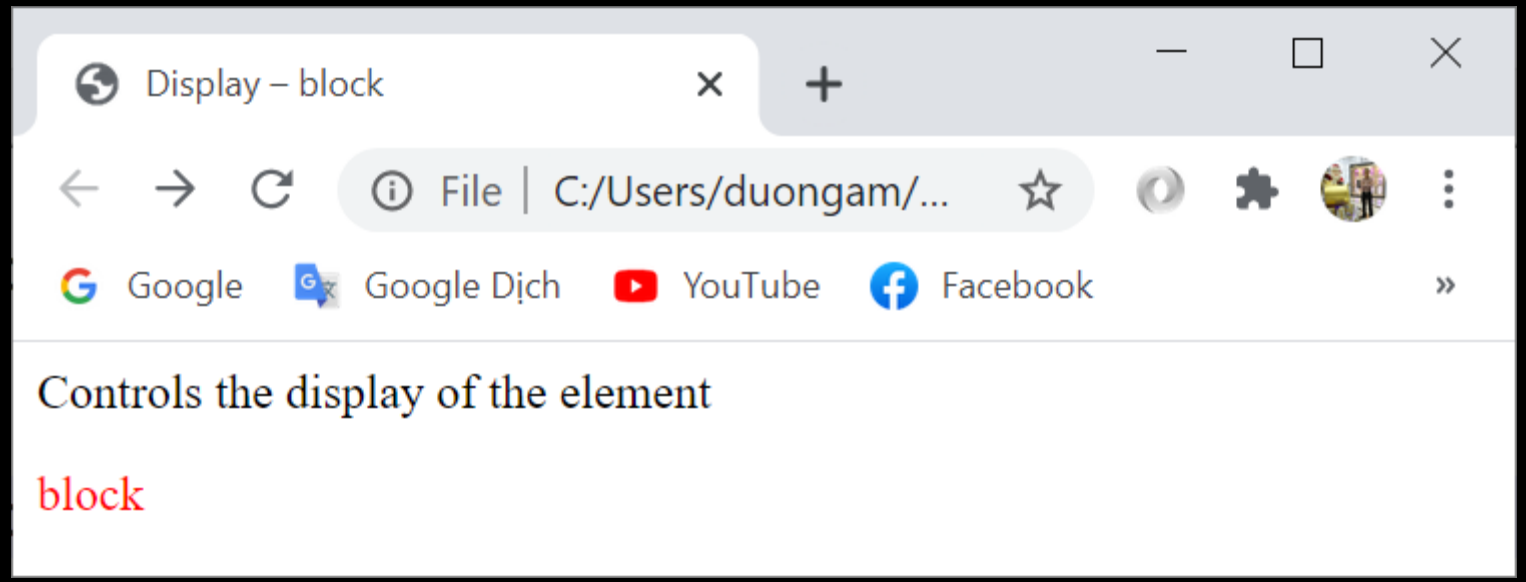
```
<style>
p { color: red; }
p.inline {
display: inline; }
</style>
```



Display – block

```
<div> Controls the display of the element
      <p class="block"> block </p>
 </div>
```

```
<style>
p { color: red; }
p.block {
display: block; }
</style>
```



Display

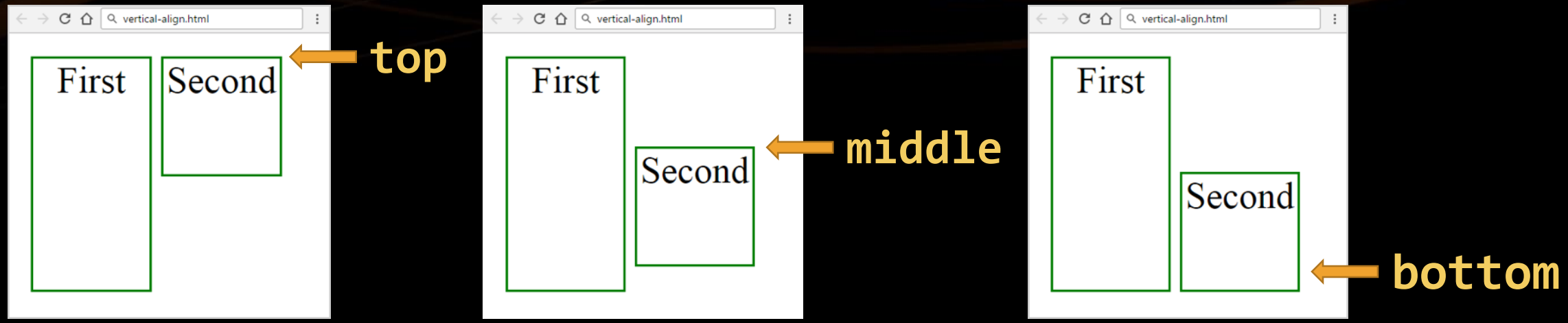
- `inline-block`
  - No breaks are placed before and after (like inline)
  - `Height` and `width` can be applied (like in block elements)
- `table`, `table-row`, `table-cell`
  - The elements are arranged in a table-like layout
- `none`
  - The element is hidden and its dimensions are not used to calculate the surrounding elements rendering
  - Differs from `visibility: hidden`

Display – inline-block

```
<div> Controls the display of the element
      <p class="inline-block"> block </p>
 </div>
```

```
<style>
p { color: red; }
p.inline-block {
display: inline-
block;
width: 100px;
height: 50px }
</style>
```

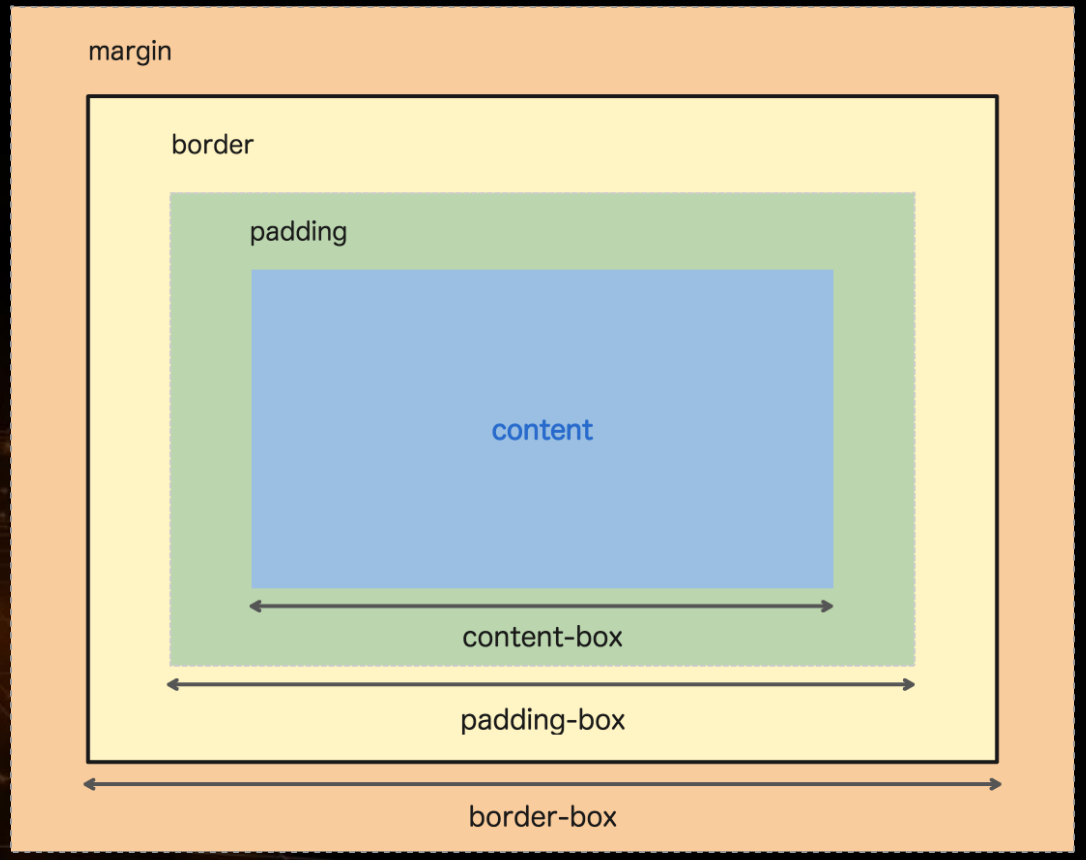
Inline-Block Elements: Vertical-Align

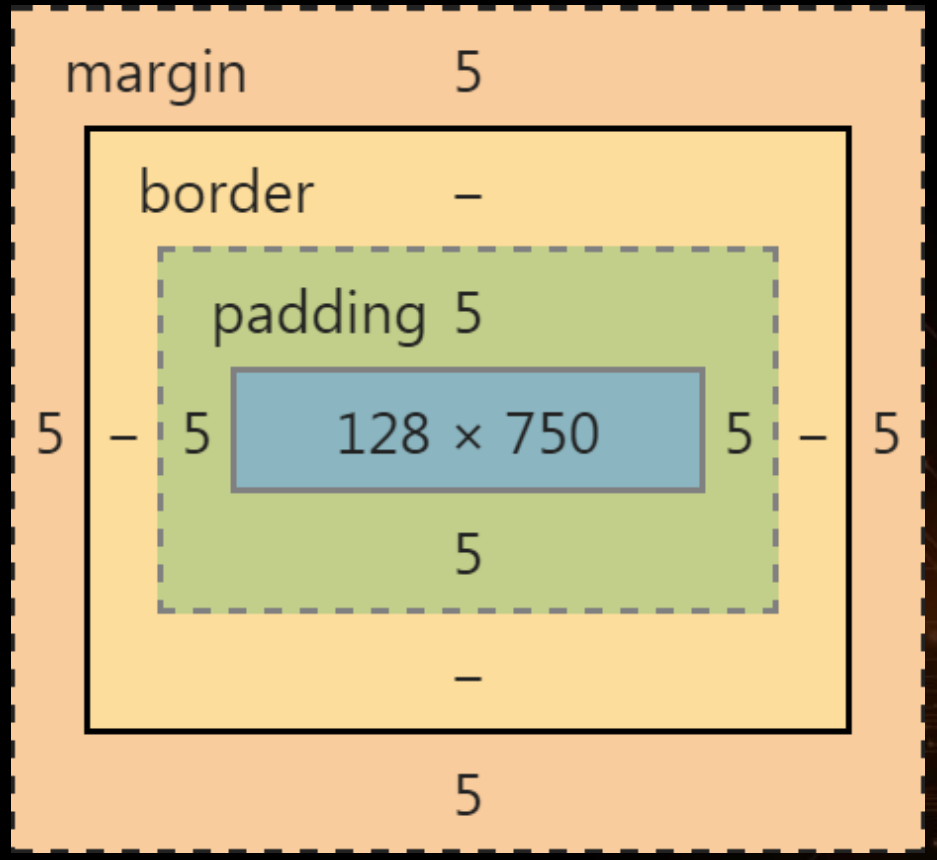



```
.boxes li {
  display: inline-block;
  text-align: center;
  width: 50px;
  border: 1px solid green;
}
```

```
.boxes li:nth-child(1) {
  height: 100px;
  vertical-align: top;
}
.boxes li:nth-child(2) {
  height: 50px; }
```

The Box Model
Border Box and Content Box

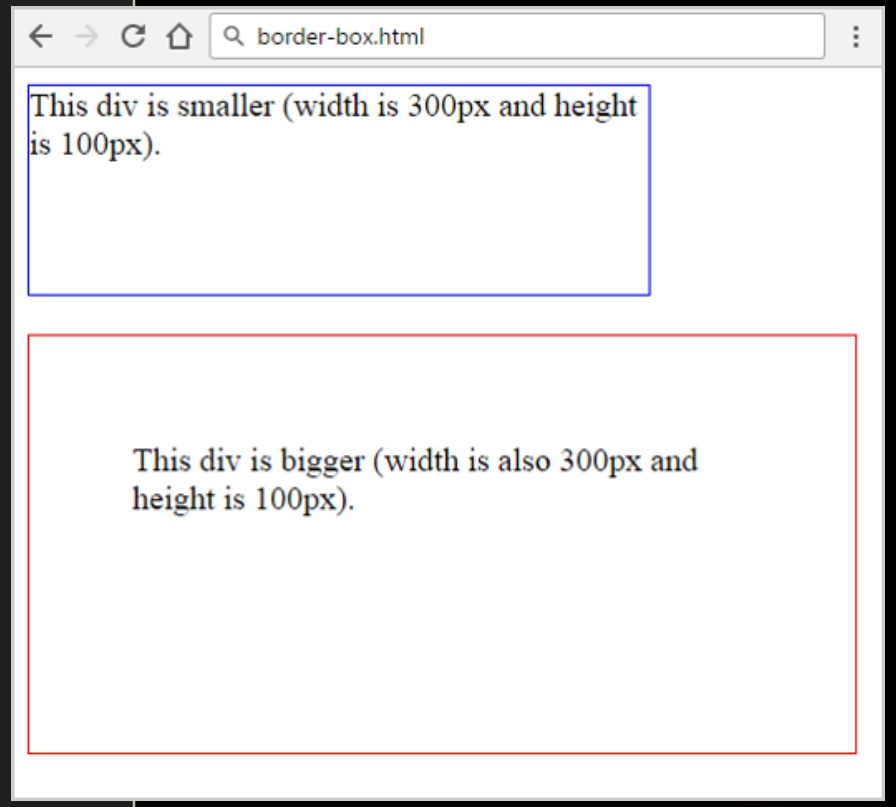




CSS Box-Sizing: Content-Box

```
<div class="div1">This div is smaller…</div>
<div class="div2">This div is bigger…</div>
```

```
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue; }
.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid blue;
}
```



CSS3 Box-Sizing

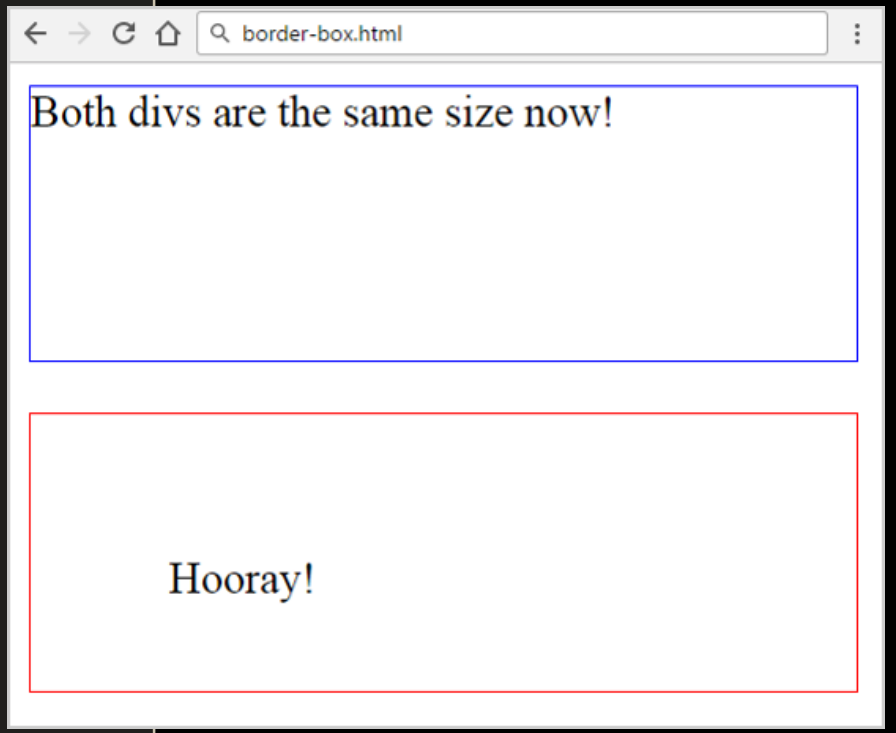
- Determine whether you want an element to render it's borders and padding within its specified width, or outside of it.
- Possible values:
  - **`box-sizing: content-box`** (default)
    box width: 288 pixels + 10 pixels padding and 1 pixel border on each side = 300 pixels
  - **`box-sizing: border-box`**
    box width: 300 pixels, including padding and borders

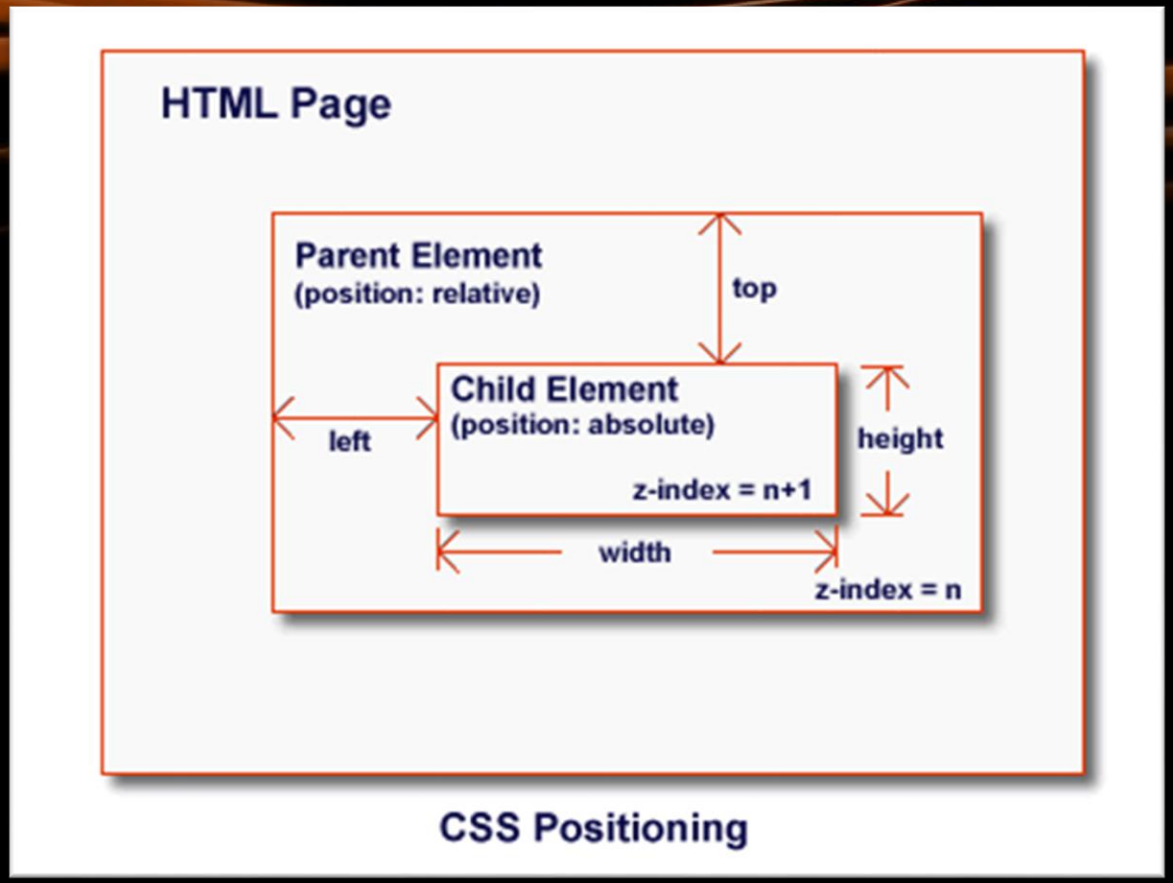
CSS Box-Sizing: Border-Box

```
<div class="div1">Both divs…</div>
```

```
<div class="div2">Hooray!</div>
```

```
* { box-sizing: border-box; }
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue; }
.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid blue; }
```





CSS Positioning

Positioning of the HTML Elements

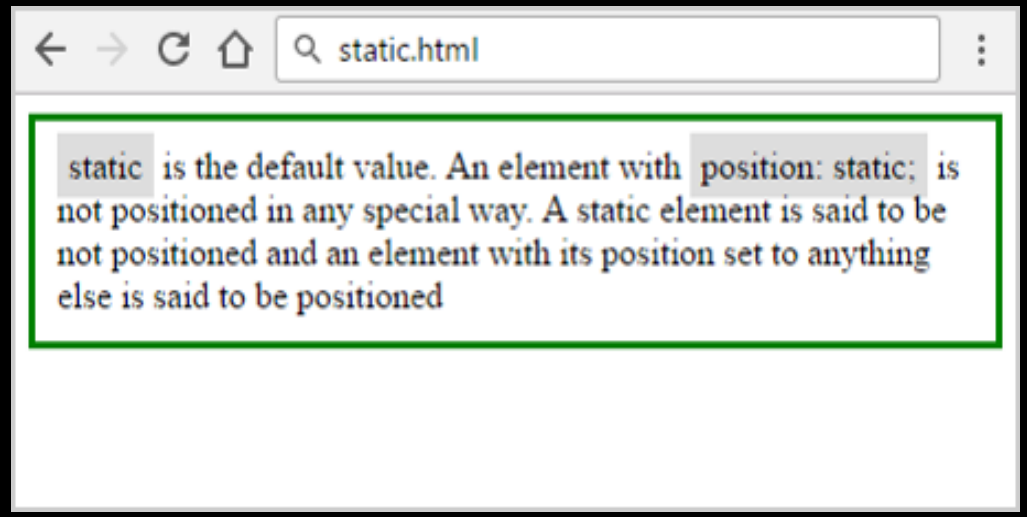
Positioning

- **`position`**: defines the positioning of the element in the page content flow
- The value is one of:
  - **`static`** (default)
  - **`relative`** – relative position according to where the element would appear with static position
  - **`absolute`** – position according to the innermost positioned parent element
  - **`fixed`** – same as absolute, but ignores page scrolling

Positioning: Static

```
<div class="static">
  static is the default value…
</div>
```

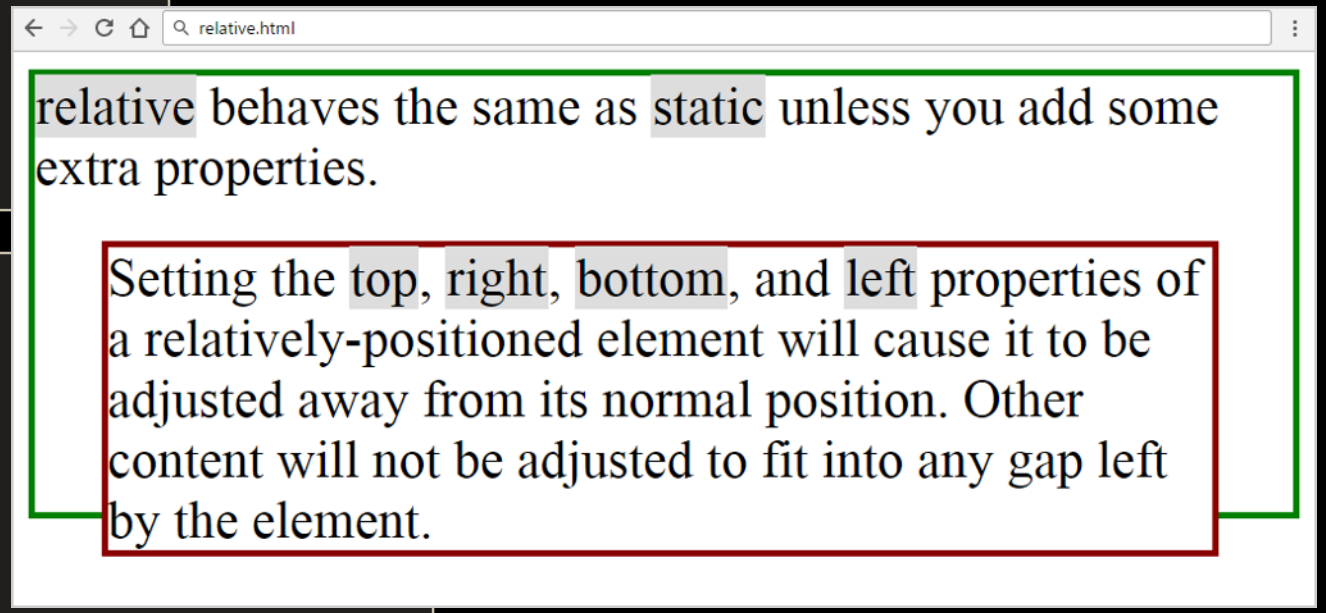
```
.static {
  position: static;
  border: 3px solid green;
  padding: 10px;
}
```



Positioning: Relative

```
<div class="relative1">
  relative behaves …
  <div class="relative2">
    Setting the top…
  </div>
</div>
```

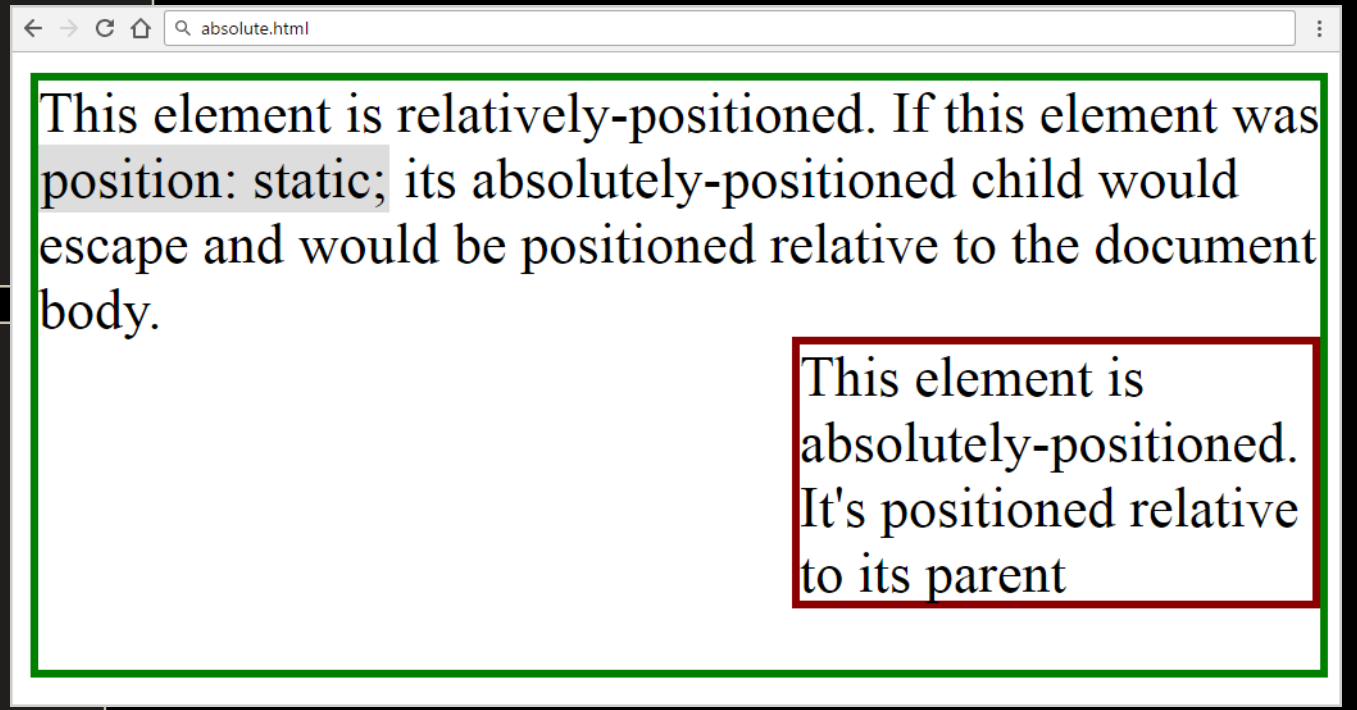
```
.relative1 {
  position: relative;
}
.relative2 {
  position: relative;
  top: 20px; left: 30px; }
```



Positioning: Absolute

```
<div class="relative">
  This element is relatively…
  <div class="absolute">
    This element is…
  </div>
</div>
```

```
.relative {
  position: relative;
}
.absolute {
  position: absolute;
  top: 100px; right: 0; }
```



Positioning: Fixed

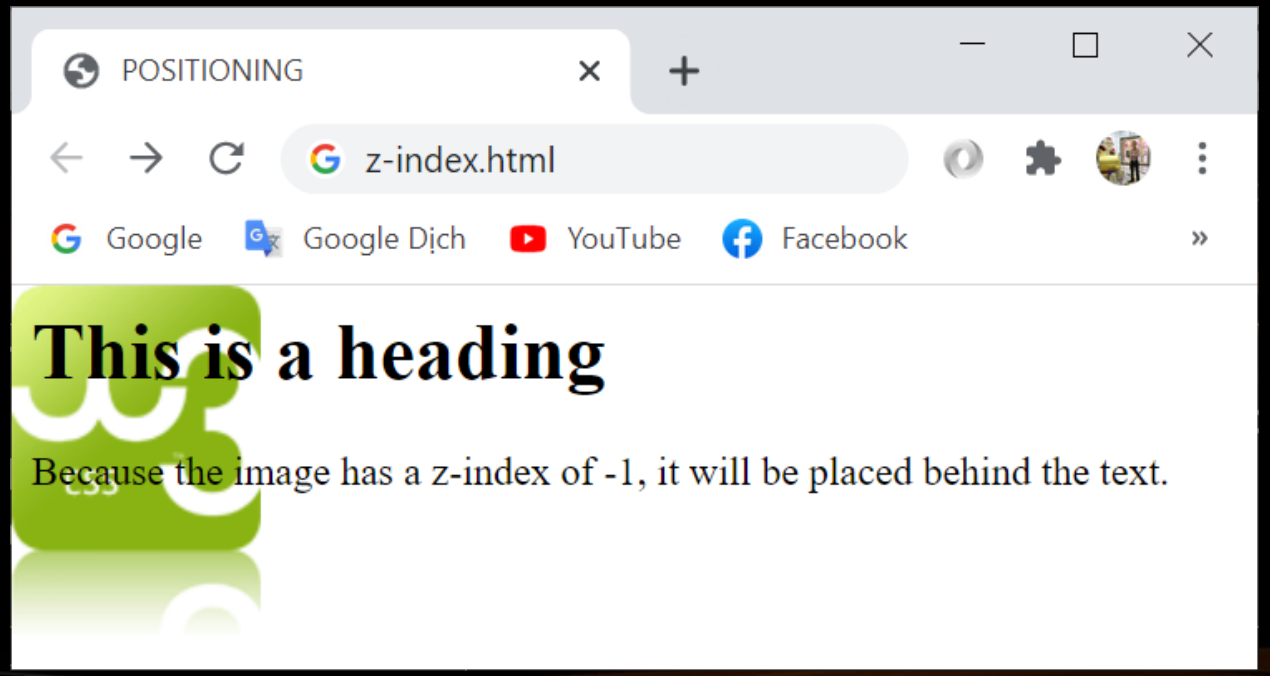
```
.fixed {
  position: fixed;
  top: 40px;
  background: white;
  width: 150px;
  border: 3px solid #8b603d; }
.div {
  background: #CCCCCC;
  opacity: 0.8;
  width: 150px;
  height: 50px;
  border: 3px solid #0000FF;
}
```


```
<div class="fixed">
A fixed element is positioned…
</div>
<div class="div"></div>
```

Positioning: z-index

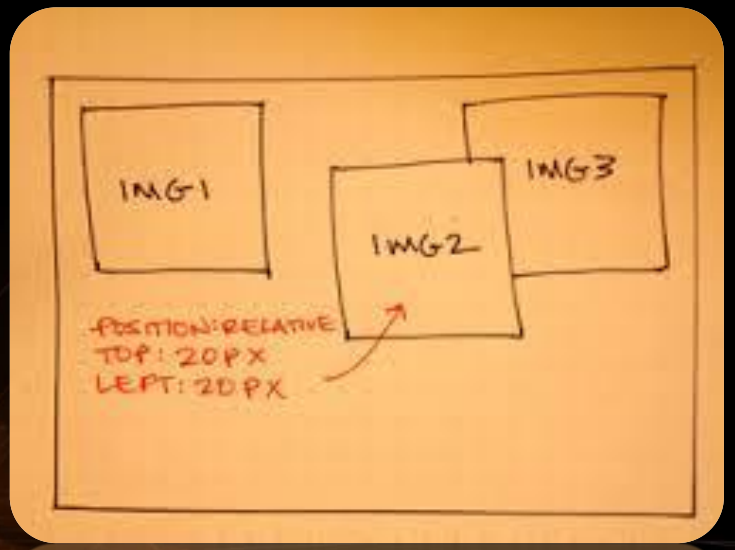
```
<h1>This is a heading</h1>
<img src="w3css.gif" width="100" height="140">
<p>Because the image has a z-index of -1, it will be placed behind the
text.</p>
```

```
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
</style>
```

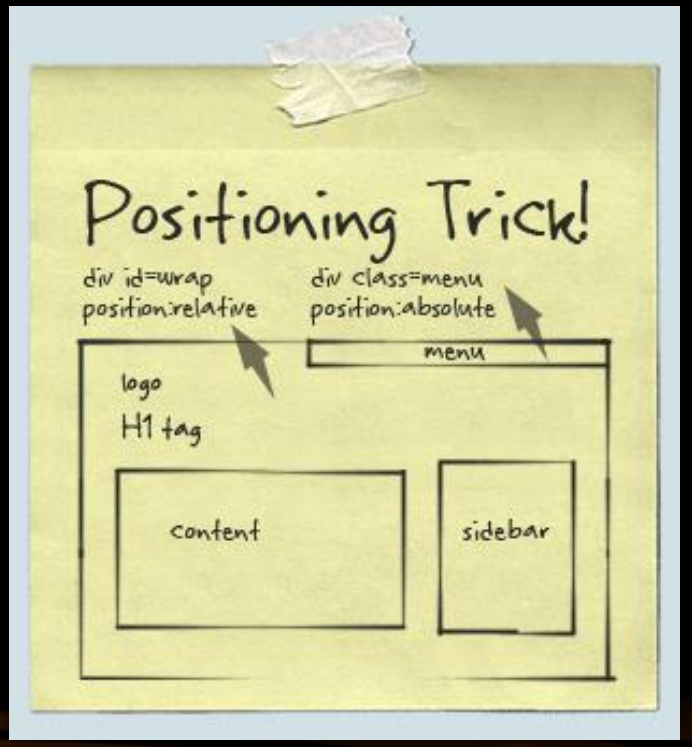


Positioning

- top, left, bottom, right: specifies offset of absolute/fixed/relative positioned element as numerical values
- z-index : specifies the stack level of positioned elements
  - Understanding stacking context



Each positioned element creates a stacking context.
Elements in different stacking contexts are overlapped according to the stacking order of their containers. For example, there is no way for #A1 and #A2 (children of #A) to be placed over #B without increasing the z-index of #A.

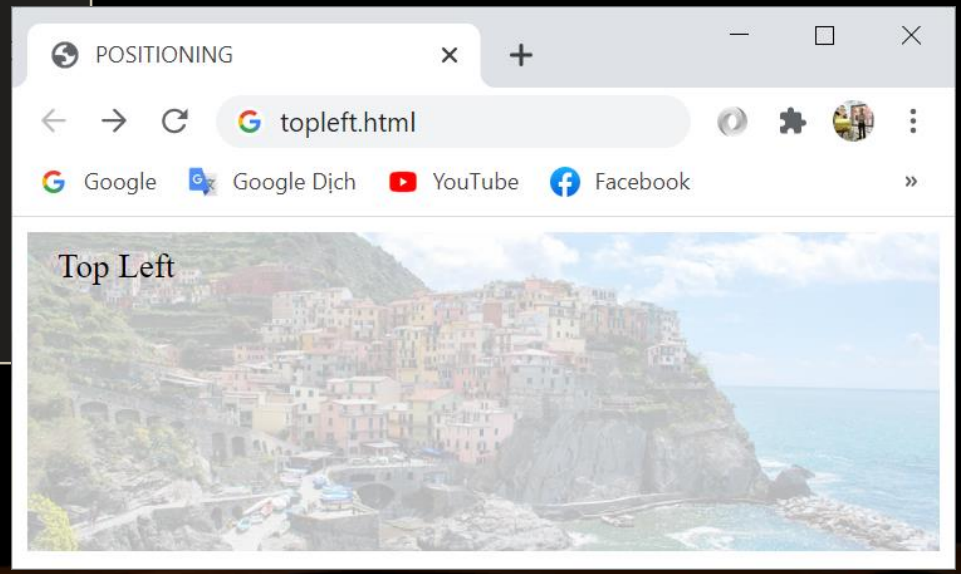


Positioning: top-left

```
<div id="container">
  <img src="img_5terre_wide.jpg" alt="Cinque Terre" width="1000"
height="300">
  <div class="topleft">Top Left</div>
</div>
```

```
<style>
img {
  width: 100%;
  height: auto;
  opacity: 0.3;
}
.container {
  position: relative;
}
```

```
.topleft {
  position: absolute;
  top: 8px;
  left: 16px;
  font-size: 18px;
}
</style>
```

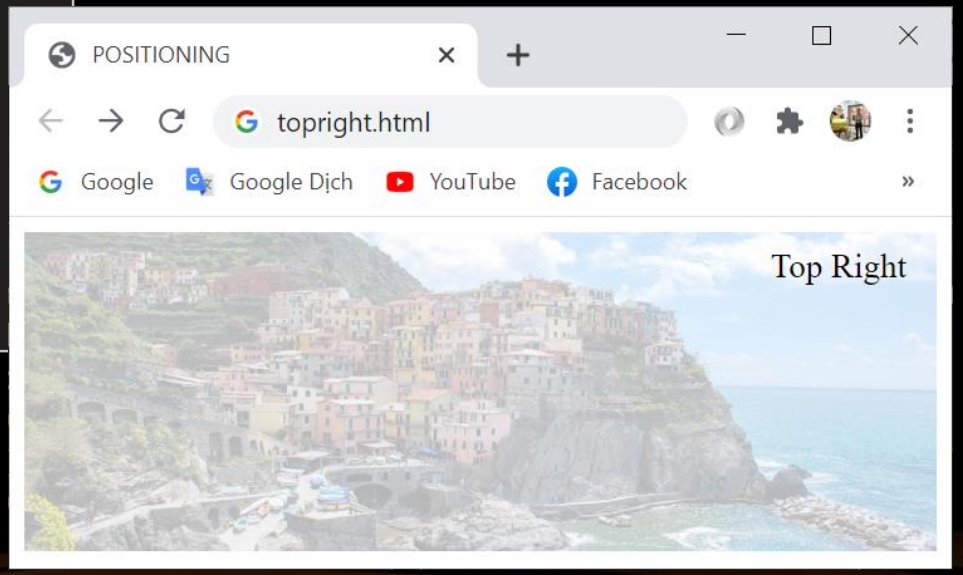


Positioning: top-right

```
<div id="container">
  <img src="img_5terre_wide.jpg" alt="Cinque Terre" width="1000"
height="300">
  <div class="topright">Top Right</div>
</div>
```

```
<style>
img {
  width: 100%;
  height: auto;
  opacity: 0.3;
}
.container {
  position: relative;
}
```

```
.topright {
  position: absolute;
  top: 8px;
  right: 16px;
  font-size: 18px;
}
</style>
```

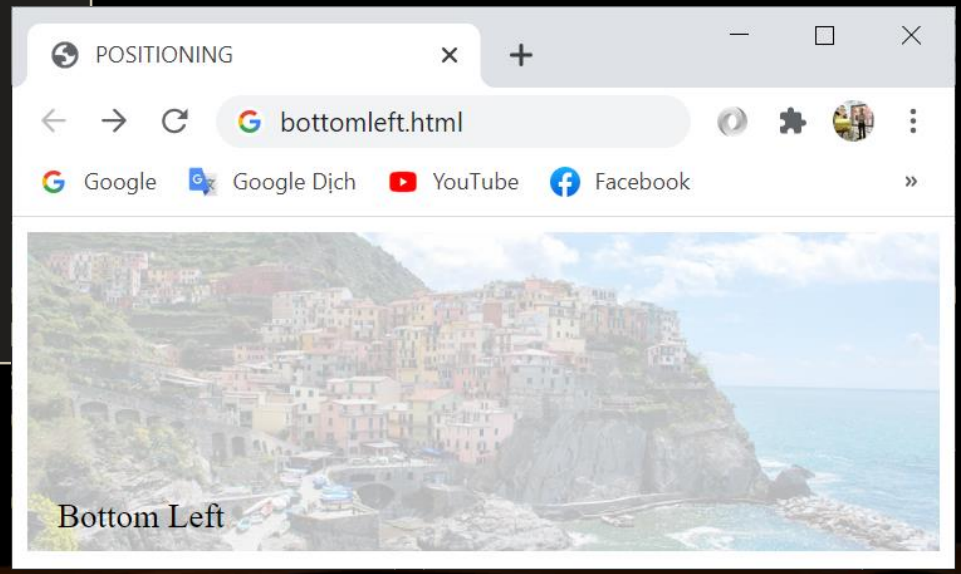


Positioning: bottom-left

```
<div id="container">
  <img src="img_5terre_wide.jpg" alt="Cinque Terre" width="1000"
height="300">
  <div class="bottomleft">Bottom Left</div>
</div>
```

```
<style>
img {
  width: 100%;
  height: auto;
  opacity: 0.3;
}
.container {
  position: relative;
}
```

```
. bottomleft {
  position: absolute;
  bottom: 8px;
  left: 16px;
  font-size: 18px;
}
</style>
```

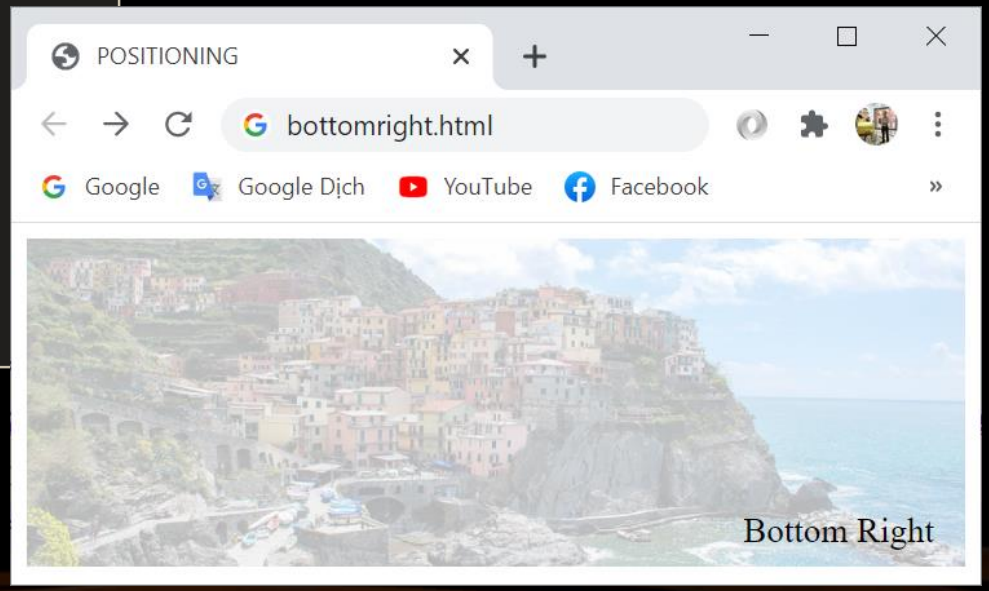


Positioning: bottom-right

```
<div id="container">
  <img src="img_5terre_wide.jpg" alt="Cinque Terre" width="1000"
height="300">
  <div class="bottomright">Bottom Right</div>
</div>
```

```
<style>
img {
  width: 100%;
  height: auto;
  opacity: 0.3;
}
.container {
  position: relative;
}
```

```
.bottomright {
  position: absolute;
  bottom: 8px;
  right: 16px;
  font-size: 18px;
}
</style>
```

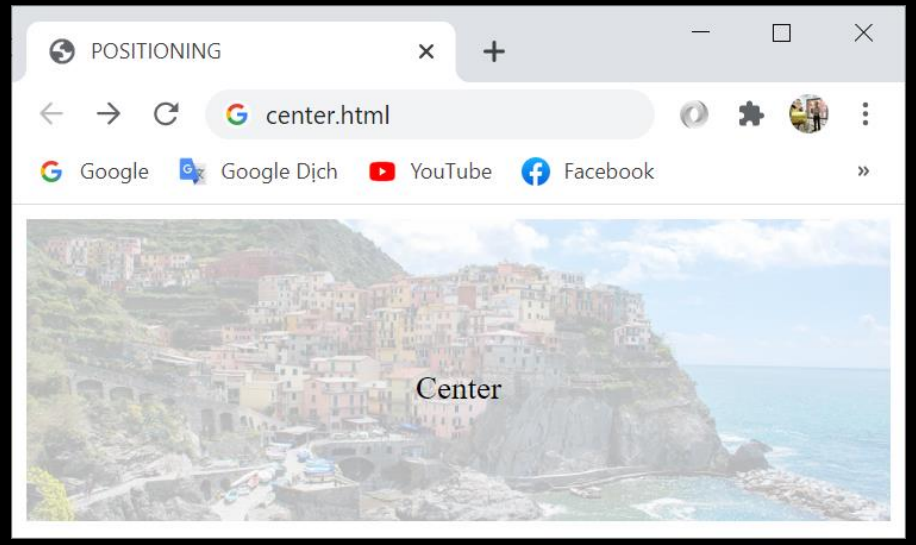


Positioning: center

```
<div id="container">
  <img src="img_5terre_wide.jpg" alt="Cinque Terre" width="1000"
height="300">
  <div class="center">Center</div>
</div>
```

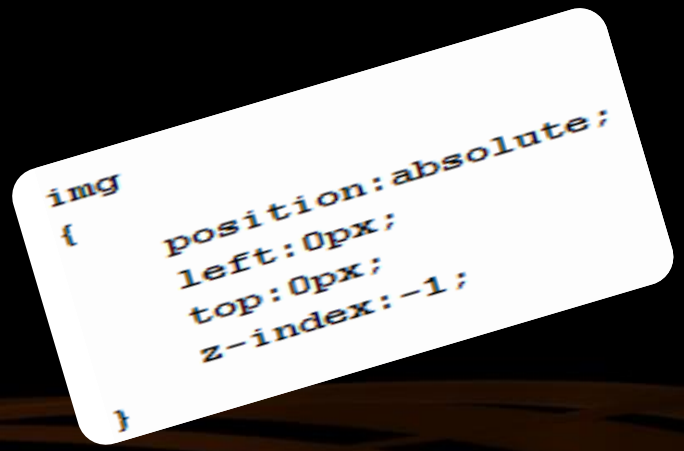
```
<style>
img {
  width: 100%;
  height: auto;
  opacity: 0.3;
}
.container {
  position: relative;
}
```

```
.center {
  position: absolute;
  top: 50%;
  width: 100%;
  text-align: center;
  font-size: 18px;
}
</style>
```



Vertical Align

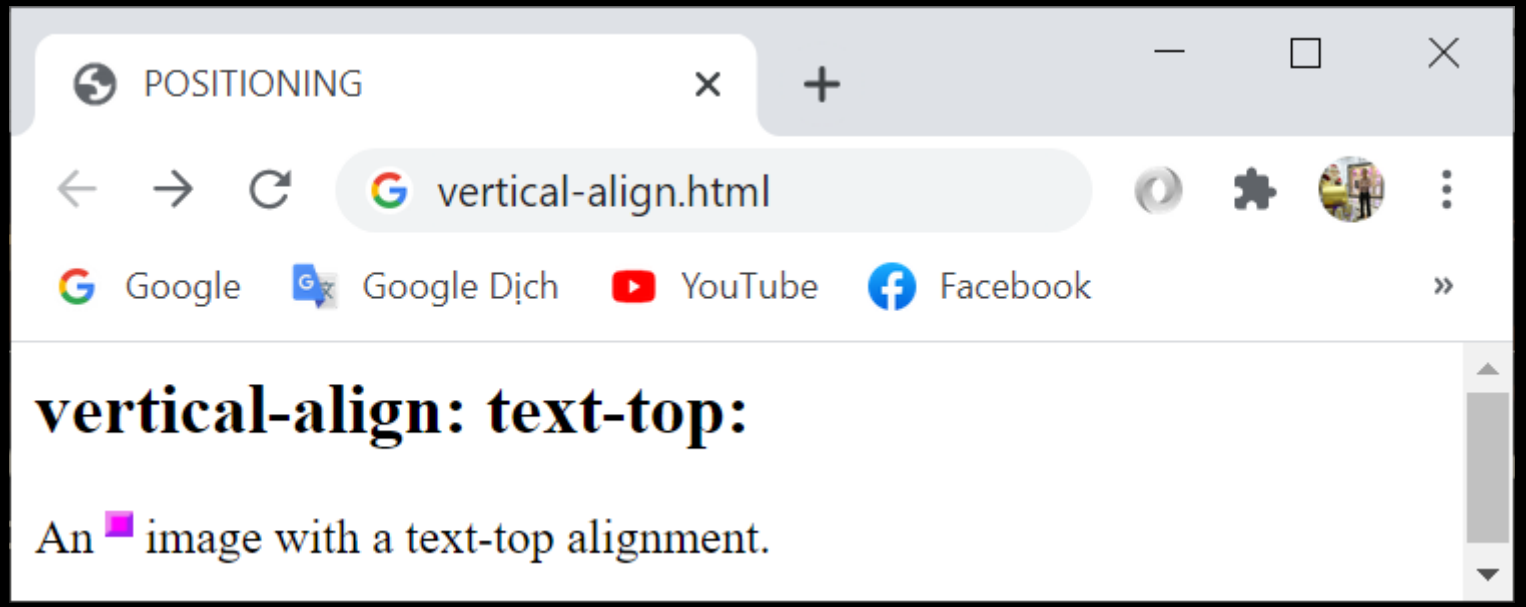
- **What it actually does?**
  - `vertical-align` only applies to inline or inline-block elements
  - Affects the alignment of the element itself, not its contents
  - For table cells, the alignment affects the cell contents, not the cell
  - Sets the vertical-alignment of an inline element, according to the line height
- **What it does NOT do?**
  - All the elements inside the vertically aligned element change their vertical position
- Values: `baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom` or numeric
- http://www.impressivewebs.com/css-vertical-align/



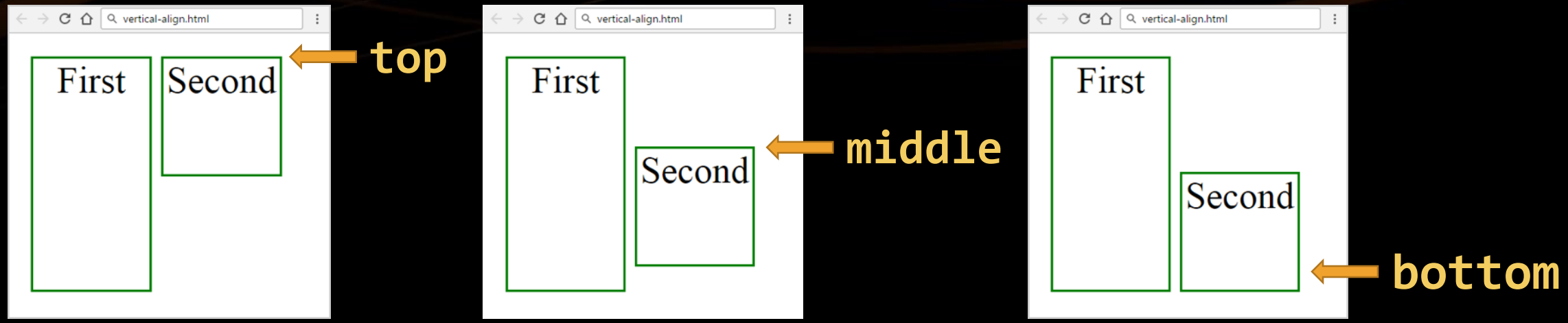
Vertical Align: text-top

```
<p>
    An <img class="b" src="sqpurple.gif" width="9" height="9">
    image with a text-top alignment.
</p>
```

```
<style>
img.b {
  vertical-align:
text-top;
}
</style>
```



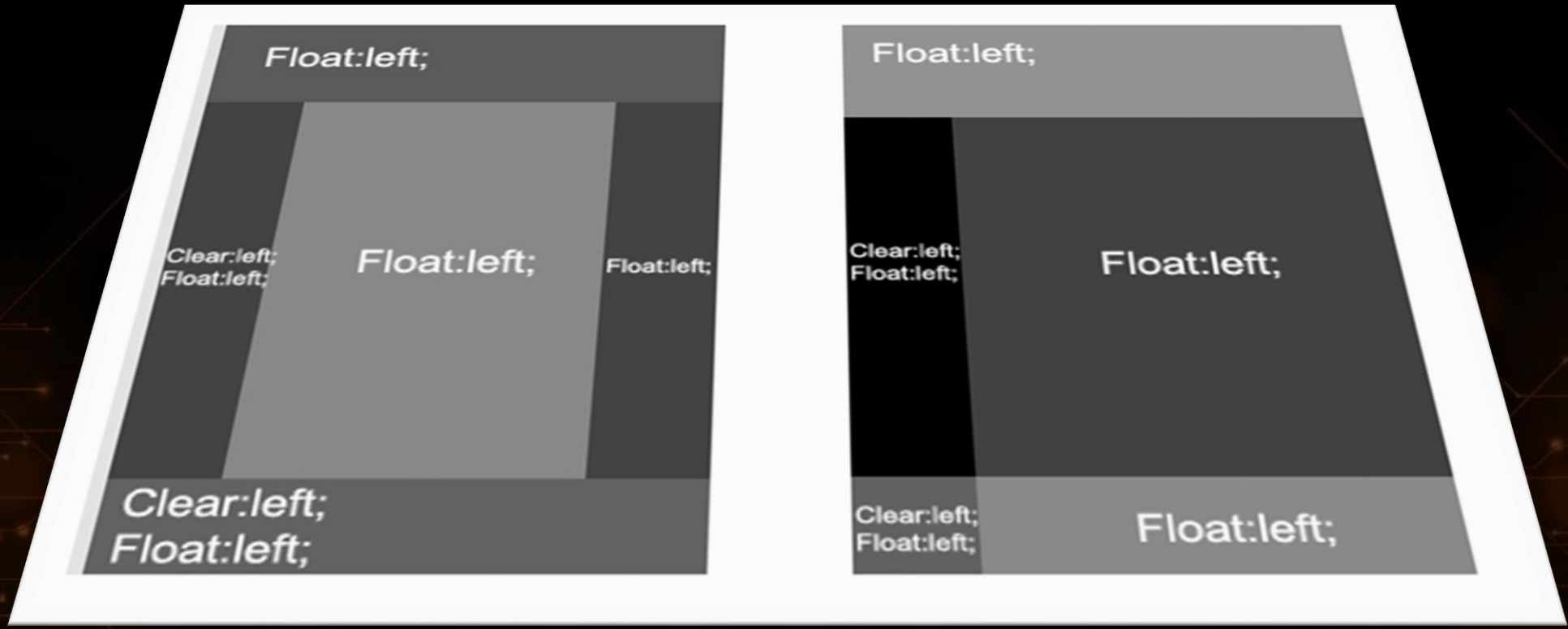
Inline-Block Elements: Vertical-Align




```
.boxes li {
  display: inline-block;
  text-align: center;
  width: 50px;
  border: 1px solid green;
}
```

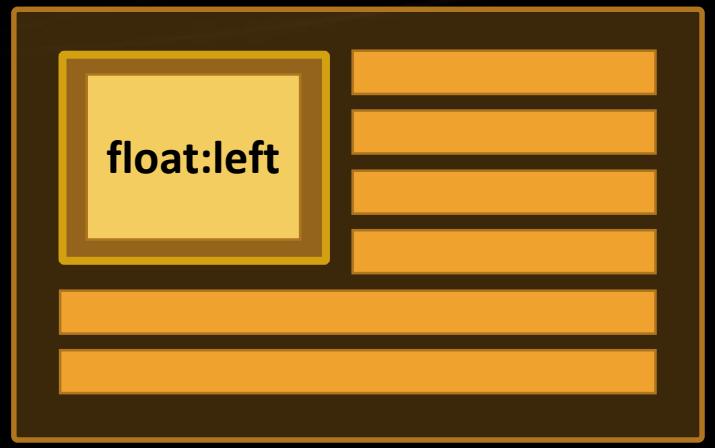
```
.boxes li:nth-child(1) {
  height: 100px;
  vertical-align: top;
}
.boxes li:nth-child(2) {
  height: 50px; }
```

Floating Elements



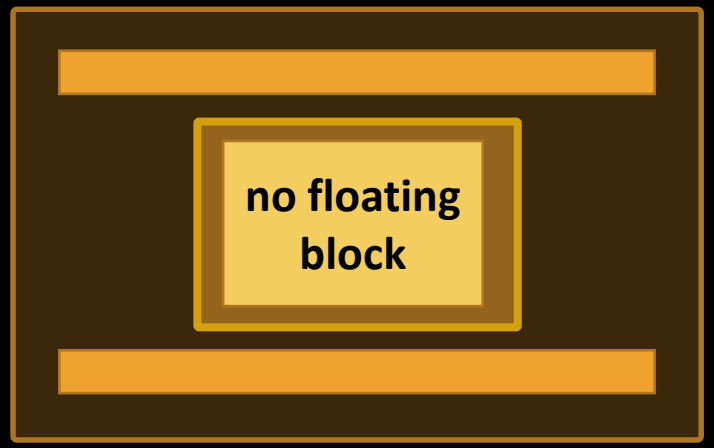
Floating Elements

- HTML elements can float left and right



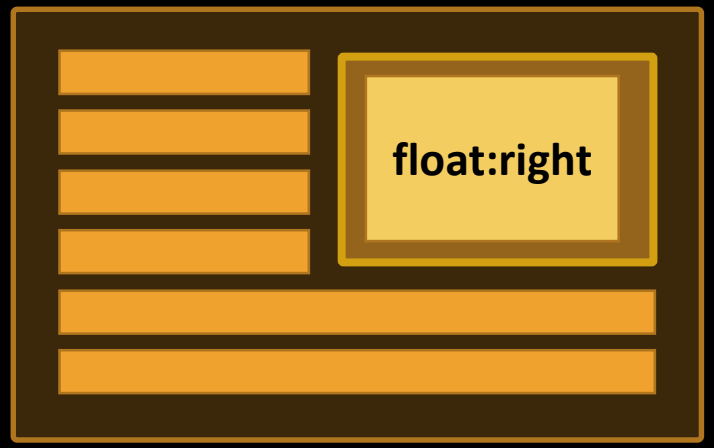
Some text around …

```
<div style=
"float:left">
 float:left
</div>
```



Some text before …

```
<div style=
"margin:0 auto">
 no floating block
</div>
```



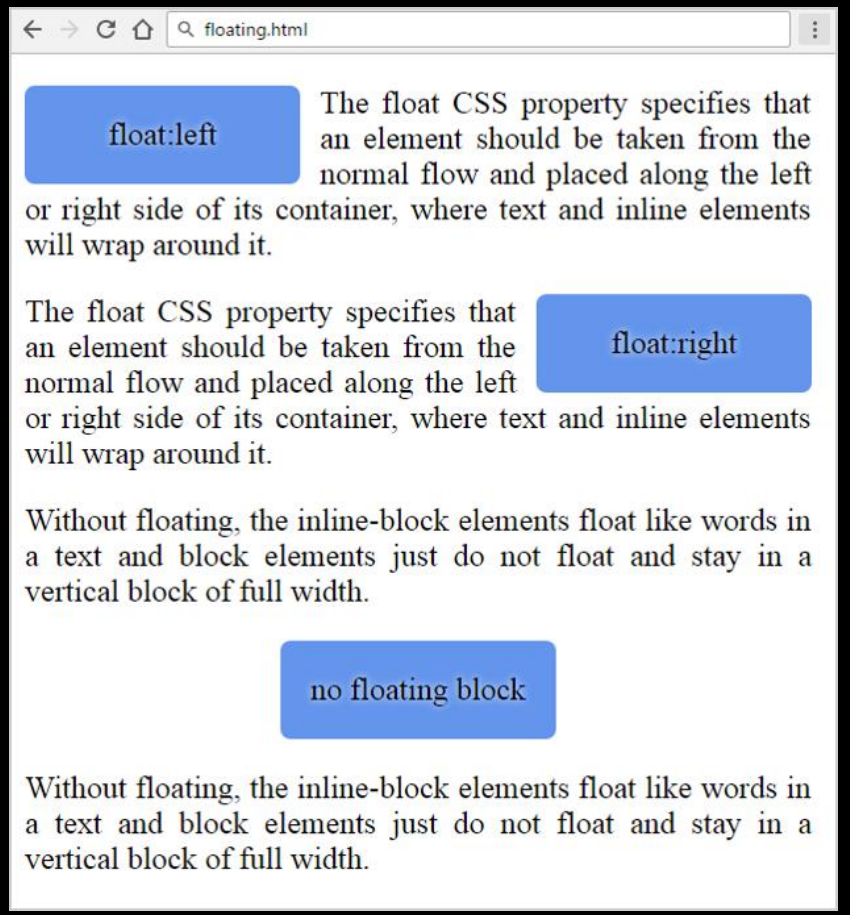
Some text around …

```
<div style=
"float:right">
 float:left
</div>
```

Problem: Floating Elements (HTML)

- Create a Web page (HTML + CSS) like at the screenshot:

```
<main>
  <div class="left">left</div>
  <p>The float CSS property …</p>
  <div class="right">right</div>
  <p>The float CSS property …</p>
  <p>Without floating …</p>
  <div class="center">no float</div>
  <p>Without floating …</p>
</main>
```



Solution: Floating Elements (CSS)

```
main { width: 400px;
  text-align: justify; }
.left, .right, .center {
  width: 120px; height: 30px;
  line-height: 30px;
  text-align: center;
  padding: 10px;
  background: CornflowerBlue;
  border-radius: 5px;
  text-shadow: 0px 0px 5px
white; }
```

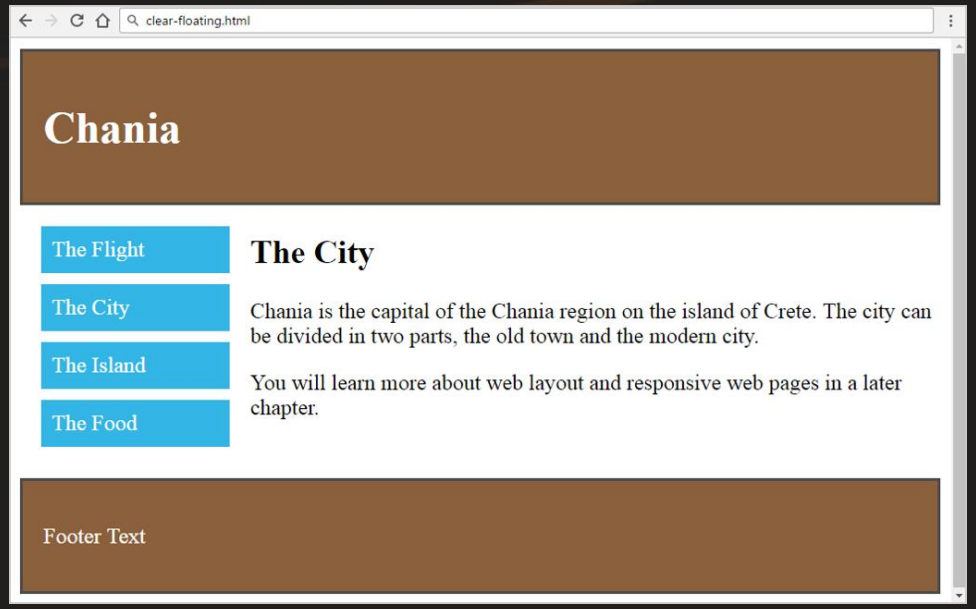
```
.left {
  float: left;
  margin-right: 10px;
}

.right {
  float: right;
  margin-left: 10px;
}

.center {
  margin: 0 auto;
}
```

Problem: Clear Floating

- You are given the HTML for this page. Write the missing CSS. Use left floating, right floating and clearing

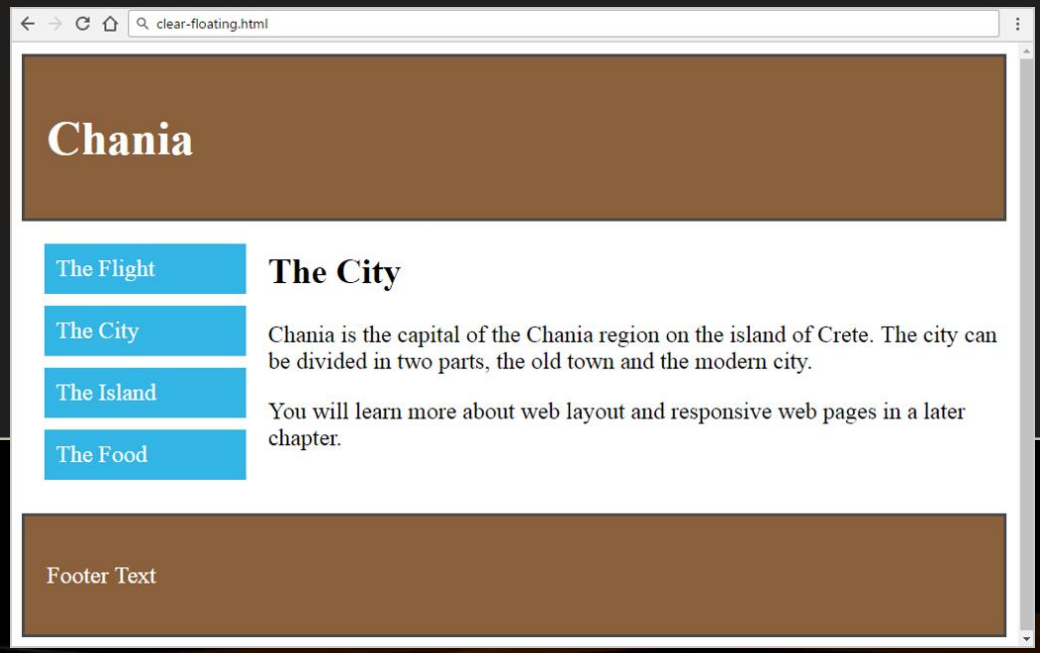
Solution: Clear Floating (HTML)

```
<header>
  <h1>Chania</h1>
</header>
<main class="clearfix">
  <aside class="float-left menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </aside>
```



Solution: Clear Floating (HTML)

```
    <article class="float-right content">
      <h2>The City</h2>
      <p>Chania is the capital of the Chania…</p>
      <p>You will learn more about web layout…</p>
    </article>
</main>
<footer>
    <p>Footer Text</p>
</footer>
```



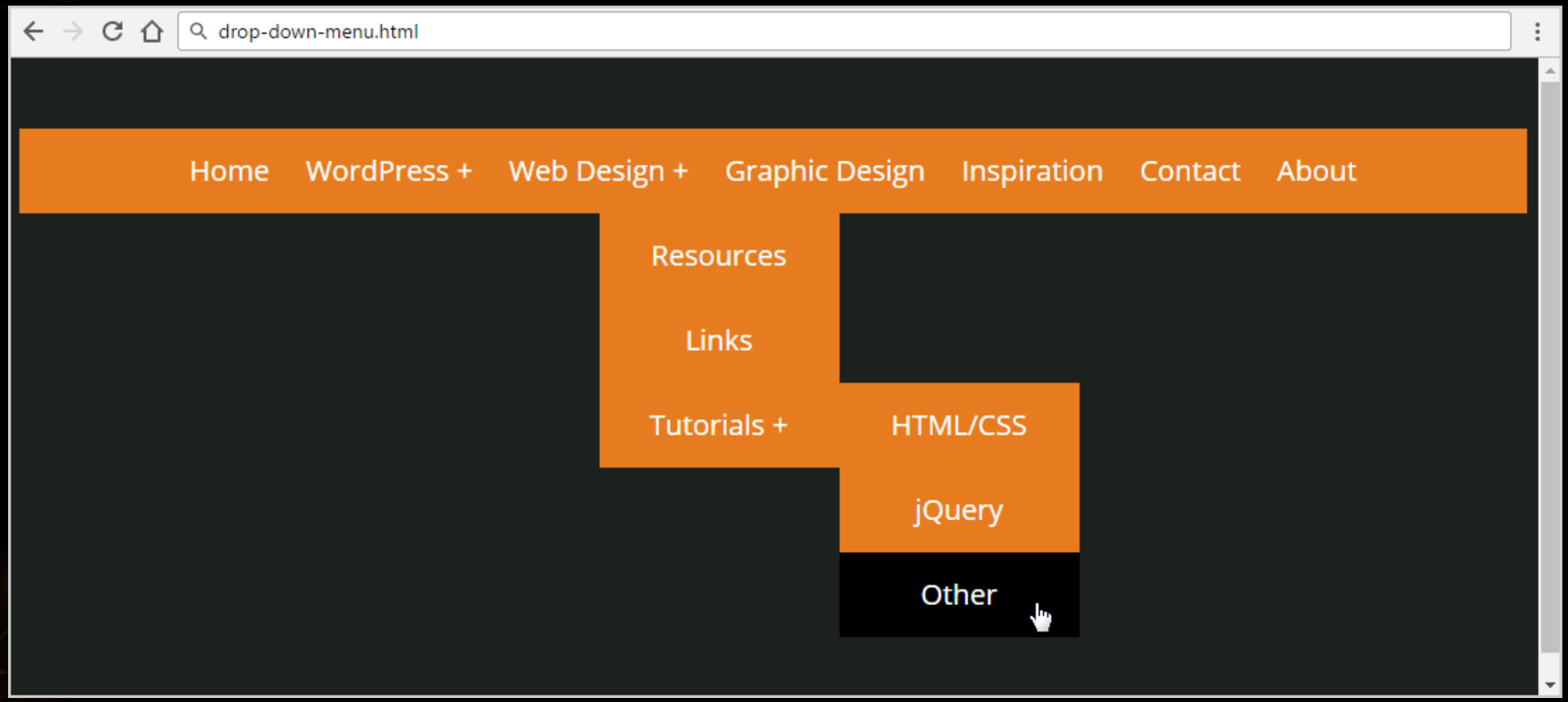
Solution: Clear Floating (CSS)

```
* { box-sizing: border-box; }
header, footer {
  background: #8b603d;
  color: white;
  padding: 15px;
  border: 2px solid #4B4B4B; }
.menu { width: 25%; }
.float-left {
  float: left;
  padding: 15px; }
.float-right { float: right; }
.content { width: 75%; }
```

```
.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0; }
.menu li {
  padding: 8px;
  margin-bottom: 8px;
  background: #33b5e5;
  color: #ffffff; }
.clearfix::after {
  content: "";
  display: block;
  clear: both; height: 0; }
```

Problem: Drop Down Menu

- Create a Web page (HTML + CSS) like at the screenshot:



Solution: Drop Down Menu (HTML)

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <!-- TODO: put the rest list item here-->
    <li><a href="#">Web Design</a>
      <ul>
        <!-- TODO: put the rest list item here-->
        <li><a href="#">Tutorials</a>
          <ul><!-- TODO: put the rest list item here--></ul>
        </li>
      </ul>
    </li><!-- TODO: put the rest list item here-->
  </ul>
</nav>
```

Solution: Drop Down Menu (CSS)

```
@import
url(http://fonts.googleapis.co
m/css?family=Open+Sans);
body {
  font-family: 'Open Sans',
sans-serif;
  background: #1d211e;
  font-size: 22px;
  line-height: 22px;
  color: #ffffff;
  text-align: center; }
a {
  color: #ffffff; }
```

```
nav {
  margin: 50px 0;
  background-color: #e67c1f; }
nav ul {
  padding: 0;
  margin: 0;
  list-style: none;
  position: relative;
}
nav ul li {
  display: inline-block;
  background-color: #e67c1f;
}
```

Solution: Drop Down Menu (More CSS)

```
nav a {
  display: block;
  padding: 0 10px;
  color: #FFF;
  font-size:20px;
  line-height: 60px;
  text-decoration: none; }
nav a:hover {
  background-color: #000000;
}
nav ul ul {
  display: none;
  position: absolute; }
```

```
nav ul li:hover > ul {
  display: inline-block; }
nav ul ul li {
  width: 170px;
  position: relative;
  display: block; }
nav ul ul ul li {
  position: relative;
  top: -60px;
  left: 85px; }
li > a::after { content:' +'; }
li > a:only-child::after {
  content:''; }
```

Summary

- Width and Height - **max/min width** and **height** in numerical values
- Margin and Padding
- Overflow - **visible**, **auto**, **scroll**, **hidden**
- Display - **block**, **inline**, **inline-block**, **table**, **none**
- Visibility - **visible**, **hidden**
- Box model – **content-box**, **border-box**
- Positioning – **absolute**, **relative**, **fixed**
- Float - **left**, **right**, **clear**
- Z-Index

CSS Layout

Questions?