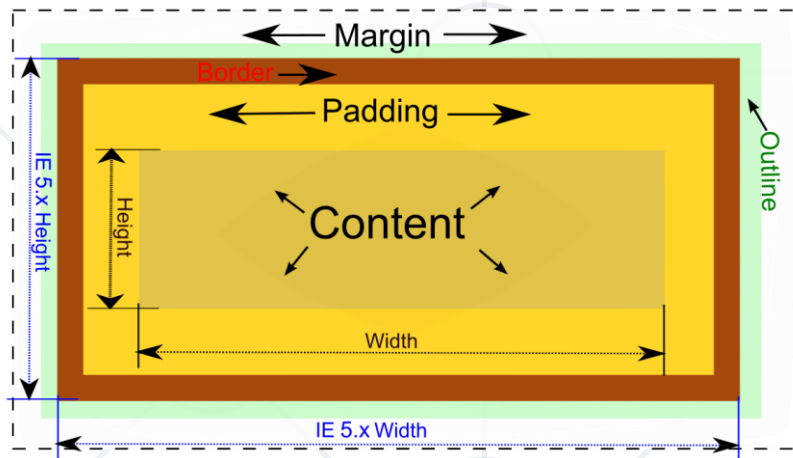


CSS Box Model & Position



SoftUni Team

Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#html-css

Table of Contents

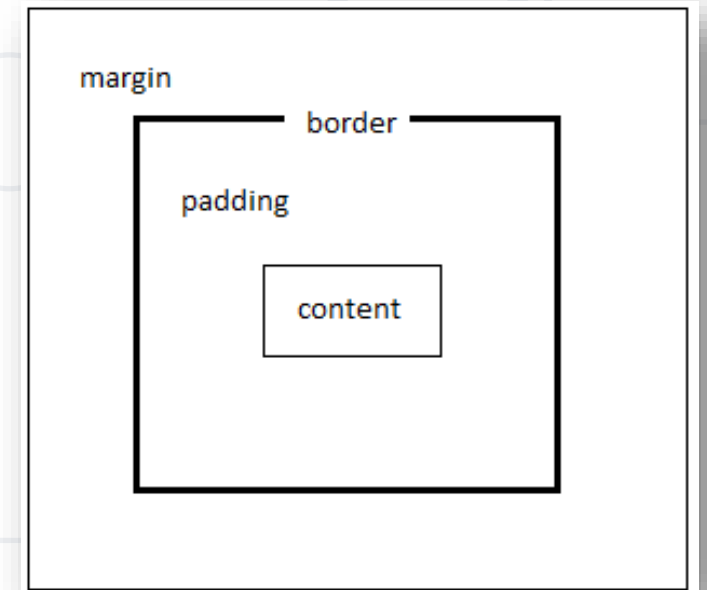
1. What is Box Model?
2. Typography
3. Position



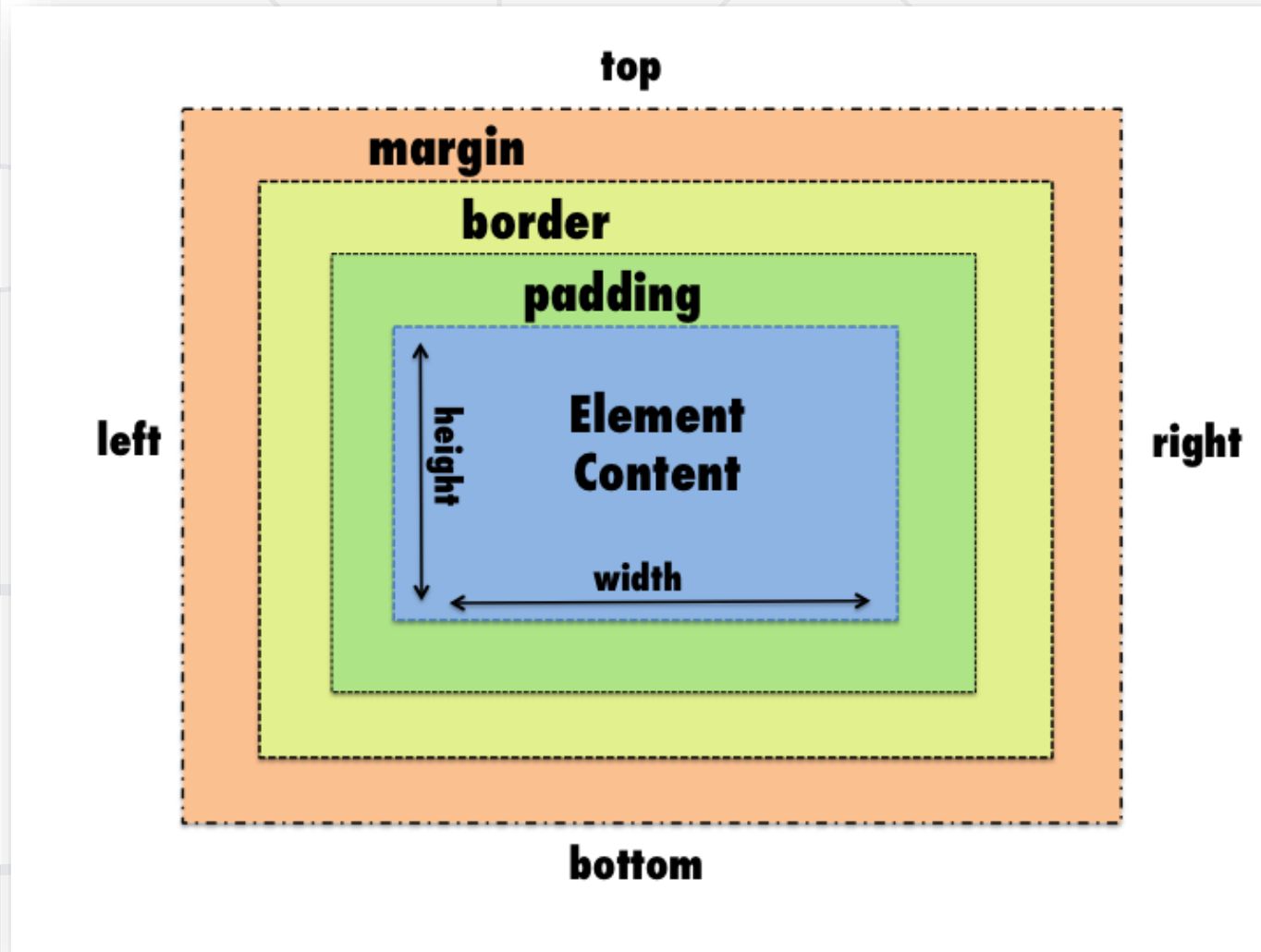


What is Box Model?

- When laying out a document, the browser's rendering engine represents **each element** as a **rectangular box** according to the standard CSS basic box model
- CSS determines the **size**, **position**, and **properties** (color, background, border size, etc.) of these boxes
- [Reference Documentation](#)



Box Model – CSS Properties



- The display CSS property defines the display type of an element, which consists of the two basic qualities of **how an element generates boxes** — the **outer** display type defining how the **box participates in flow layout**, and the **inner** display type defining **how the children of the box are laid out**



“BLOCK BOX”



“INLINE BOXES”

- HTML elements historically were categorized as either "block-level" elements or "inline" elements. By default, a block-level element **occupies the entire space** of its parent element (container), thereby creating a "block"

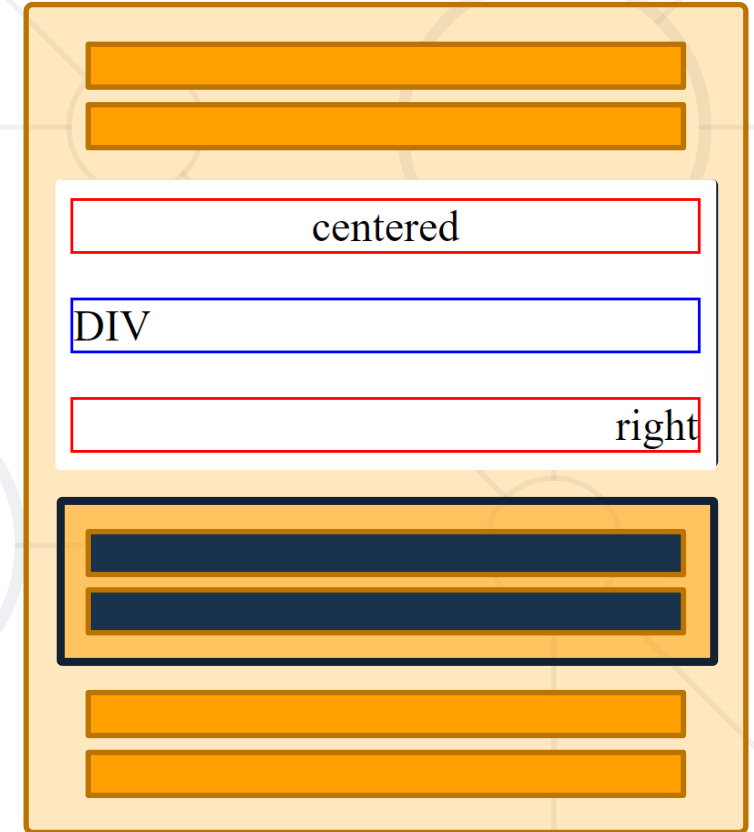
```
.box {  
  display: block;  
}
```

- Browsers typically display the block-level element with a **newline** both **before** and **after** the element. You can visualize them as a stack of boxes
- **main, header, article, section, fieldset, nav, ul, ol, li, form, h1-h6, p, div**

Block Elements – Example

```
<p style="border:1px solid red;  
text-align:center">centered</p>  
<div style="border:1px solid  
blue">DIV</div>  
<p style="border:1px solid red;  
text-align:right">right</p>
```

display: block



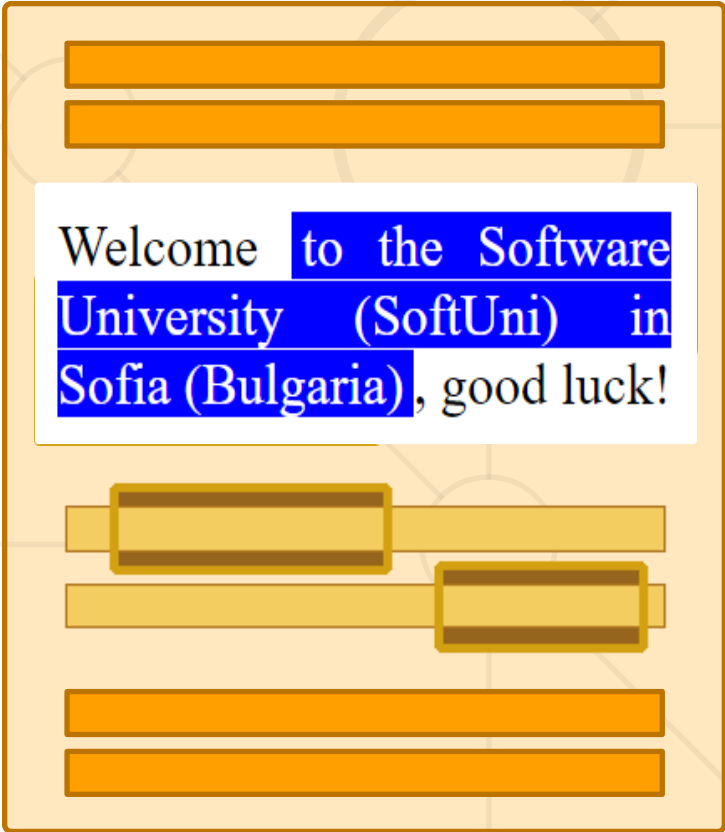
- Inline elements are those which only occupy the space bounded by the tags defining the element, instead of breaking the flow of the content.
- Inline element: **don't start** on a new line. They appear on the same line as the content and tags beside them
- **a, label, map, span, strong, em, i, img, textarea, input, button, select**
- You can add margins and padding just on **right** and **left** sides of any inline element

```
.box {  
    display: inline;  
}
```

Inline Elements – Example

```
<p style="text-align:justify"> Welcome  
<span style="color:white;  
background:blue; padding-right:3px;  
padding-left:3px;">  
to the Software University (SoftUni)  
in Sofia (Bulgaria)</span>, good  
luck!</p>
```

display: inline



Welcome to the Software
University (SoftUni) in
Sofia (Bulgaria), good luck!

- Gives us the ability to use vertical padding and margin on inline elements as well as adding width and height
- One common use for using inline-block is for creating navigation links horizontally

```
.box {  
  display: inline-block;  
}
```

Inline-Block Elements – Example

```
<div style="text-align:justify;">  
  <div style="display:inline-block;  
background:green">green</div>  
  <div style="display:inline-block;  
background:red">red block</div>  
  ...  
</div>
```

display: inline-block



- Defines the width of the element
- The **width CSS property** sets an element's width. By default, it sets the width of the content area, but if box-sizing is set to border-box, it sets the width of the border
- Example

- pixels / em / rem
- Fixed width

```
article {  
  width: 240px;  
  background: #8ce;  
}
```

Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

- percentages
- Relative to container's width

```
article {  
  width: 50%;  
  background: #8ce;  
}
```

Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

- **Default width of block elements**

- If you don't declare a width, and the box has static or relative positioning, the width will remain 100% in width and the padding and border will push inwards instead of outward. But if you explicitly set the width of the box to be 100%, the padding will push the box outward as normal

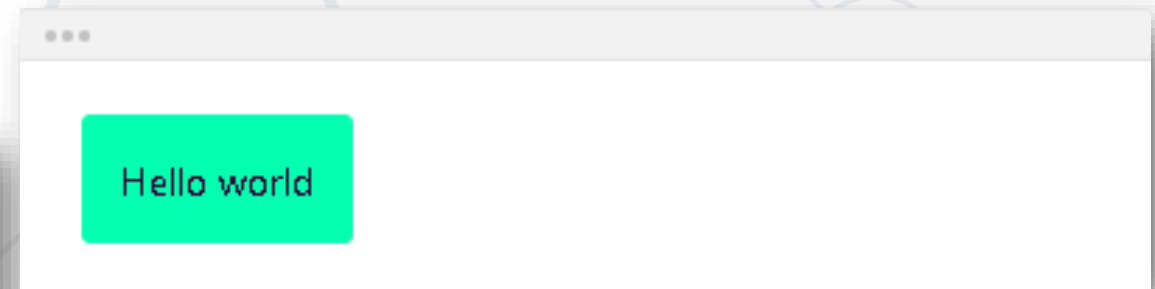
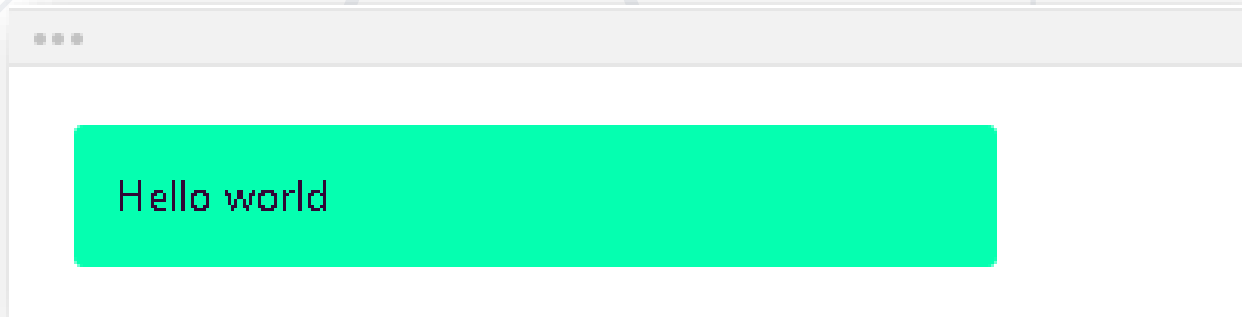
- **auto** (default)
- Auto-calculated width

```
article {  
  width: auto;  
  background: #8ce;  
}
```


Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

- Min-width - defines the **minimum** width the element
 - **min-width: 300px;** - if the **minimum** width is **larger** than the element's **actual** width, the min width will be applied
- If the **minimum** width is **smaller** than the element's **actual** width, the min width has **no effect** - **min-width: 5px;**



- Max-width - defines the **maximum** width the element can be
 - **max-width: none;** - the element has **no limit** in terms of width
 - **max-width: 150px;**
 - **max-width: 2000px;** - you can use numeric values like **pixels**, **(r)em**, **percentages...**
- If the **maximum** width is **larger** than the element's **actual** width, the max width has **no effect**



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Width – Example

```
<body>
  <div>
    I am block element. My width is 200px.
  </div>
  <span>I am span. My width is the width of my content.</span>
</body>
```

```
div, span {
  width: 200px;
  background-color: lightgreen;
}
```

← → ↻ ⓘ Файл | Example.html

I am block element. My width
is 200px.

I am span. My width is the width of my content.

- The height CSS property specifies the height of an element. By default, the property defines the height of the content area. If box-sizing is set to border-box, however, it instead determines the height of the border area.
- Example

```
.box {  
    height: 150px;  
}
```

- **auto** (default)
- Auto-calculated height

```
article {  
  height: auto;  
  background: #8ce;  
}
```

Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

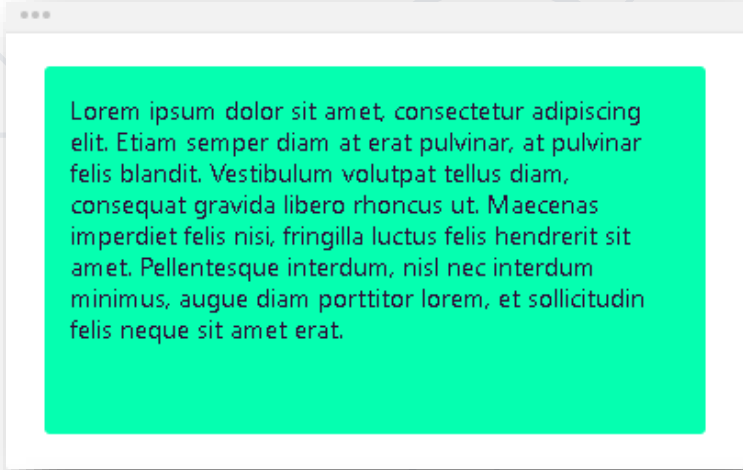
- numeric values like **px** / **pt** / **em** / **rem** / **%**

```
article {  
  height: 100px;  
  background: #8ce;  
}
```

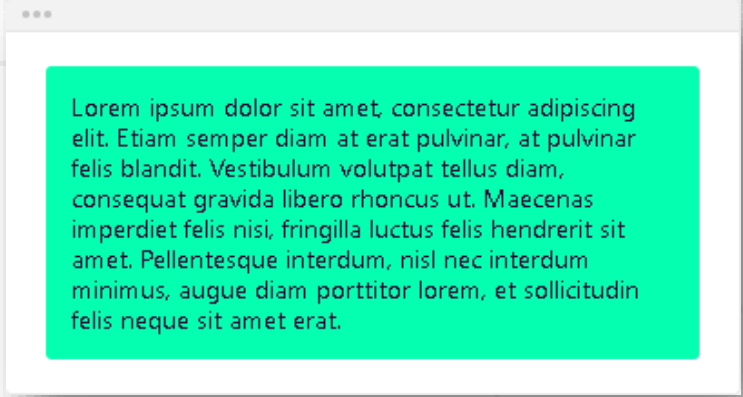
Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

- **Min-height** - defines the minimum height the element
 - **min-height: 200px;** - if the **minimum** height is **larger** than the element's **actual** height, the min height will be applied
 - **min-height: 5px;** - if the **minimum** height is **smaller** than the element's **actual** height, the min height has **no effect**

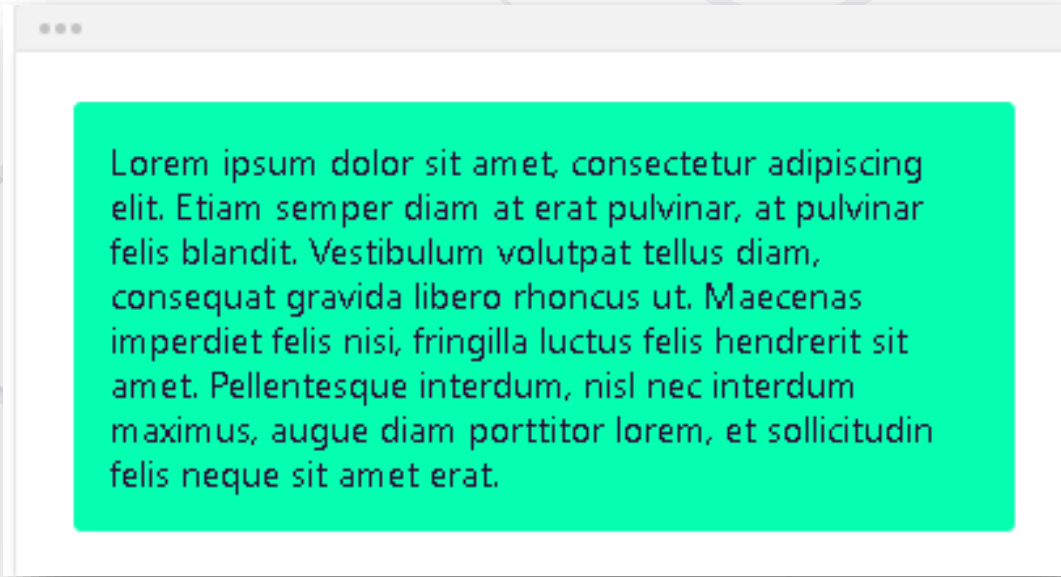


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum minimus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.



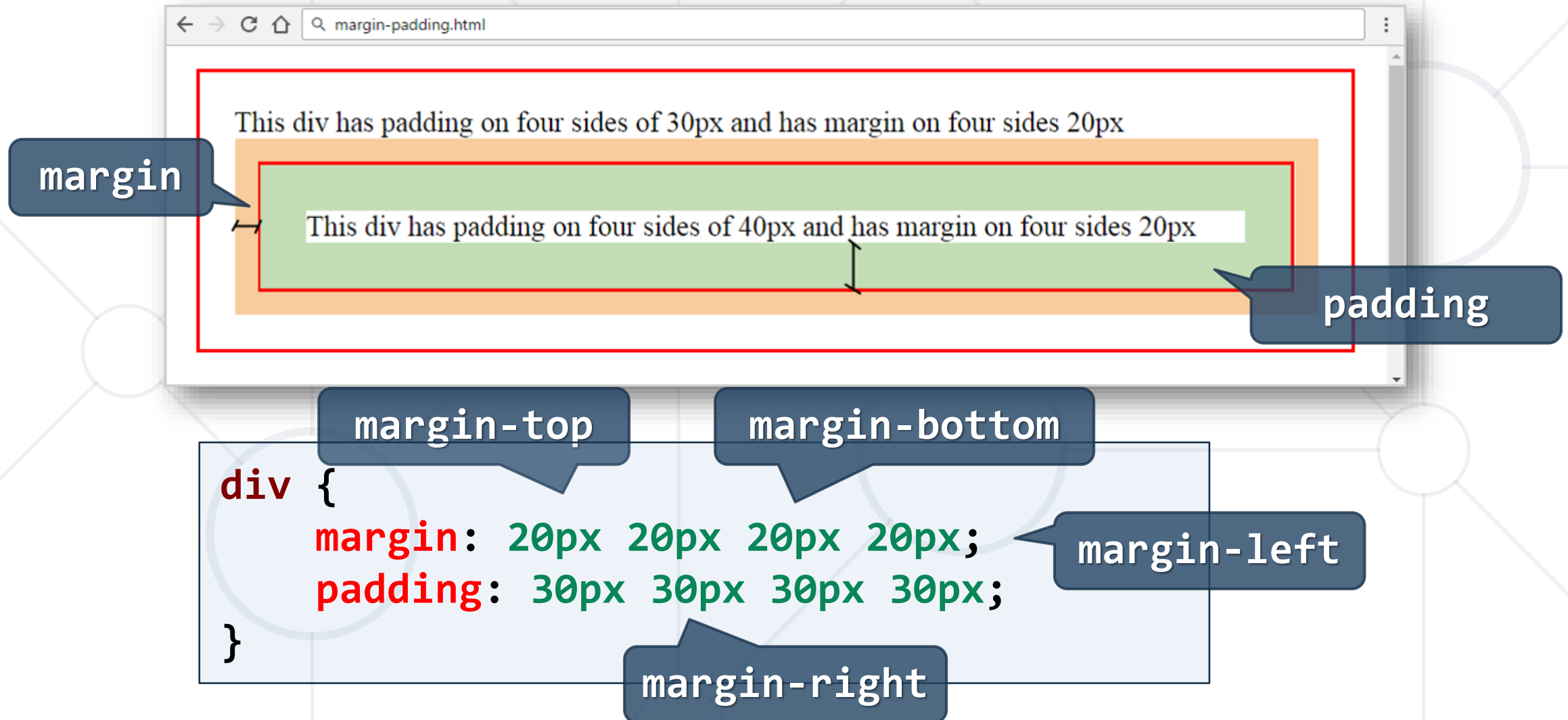
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum minimus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

- **Max-height** - defines the maximum height the element can be
 - **max-height: none;** - the element has **no limit** in terms of height
 - **max-height: 2000px;** - if the **maximum** height is **larger** than the element's **actual** height, the max height has **no effect**



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Margins and Paddings



The diagram illustrates the concepts of margins and padding in CSS. It features a browser window showing a page with two nested divs. The outer div is outlined in red and contains the text "This div has padding on four sides of 30px and has margin on four sides 20px". The inner div is outlined in green and contains the text "This div has padding on four sides of 40px and has margin on four sides 20px". A blue callout box labeled "margin" points to the space between the two divs. Another blue callout box labeled "padding" points to the space between the inner div's content and its border. Below the browser window, a code snippet is shown with four blue callout boxes pointing to specific parts of the code: "margin-top" points to the first value (20px), "margin-bottom" points to the second value (20px), "margin-left" points to the third value (20px), and "margin-right" points to the fourth value (20px).

margin

padding

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

margin-top

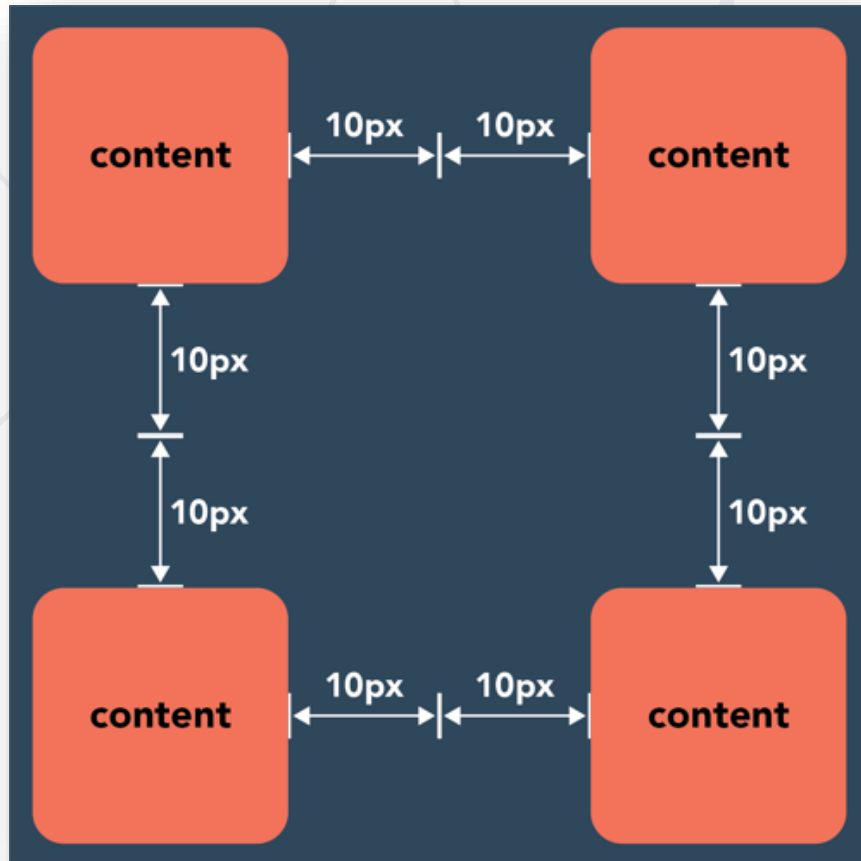
margin-bottom

margin-left

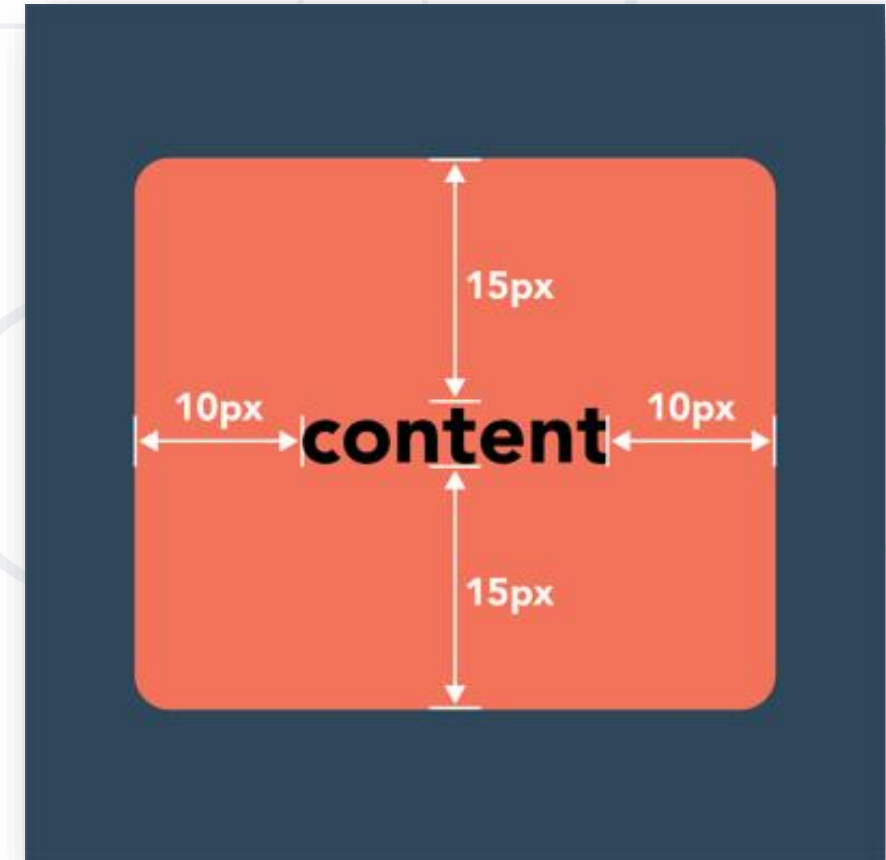
margin-right

Margins and Paddings

- **Margin** – defines the space **outside** the element



- **Padding** – defines the space **inside** the element



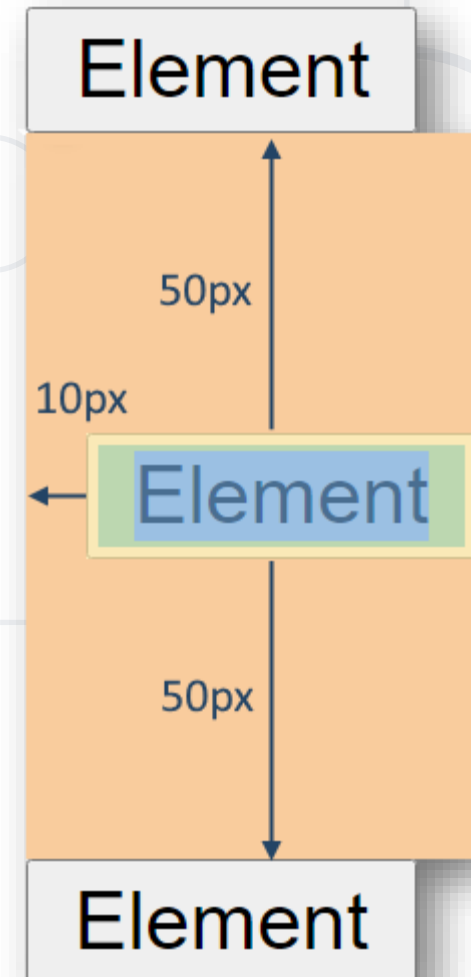
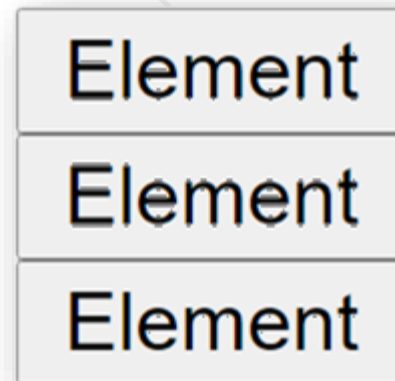
- The margin CSS property sets the margin area on all four sides of an element. It is a shorthand for margin-top, margin-right, margin-bottom, and margin-left
- Example

```
.box {  
    margin: 50px;  
}
```

```
<button class="first">Element</button>
<button class="second">Element</button>
<button class="third">Element</button>
```

```
button {
  display: block;
}

button.second {
  margin-top: 50px;
  margin-left: 10px;
  margin-bottom: 50px;
  margin-right: 0;
}
```



- The padding CSS property sets the padding area on all four sides of an element. It is a shorthand for padding-top, padding-right, padding-bottom, and padding-left
- Example

```
.box {  
    padding: 50px;  
}
```

```
<div>content</div>
```

```
div {  
  background-color: #85c1e9;  
  display: inline-block;  
  text-align: center;  
  
  padding-top: 20px;  
  padding-left: 10px;  
  padding-bottom: 20px;  
  padding-right: 10px;  
}
```

content



content

Shorthand Margin / Padding

- Shorthand margin rules:

```
button.second {  
  margin: 10px 20px 10px 20px;  
}
```

Diagram illustrating the shorthand margin rule for `button.second`. The margin values are 10px (top), 20px (right), 10px (bottom), and 20px (left). The directions are labeled: top, right, bottom, and left.

```
button.second {  
  margin: 20px 10px;  
}
```

Diagram illustrating the shorthand margin rule for `button.second`. The margin values are 20px (top & bottom) and 10px (left & right). The directions are labeled: top & bottom, and left & right.

- Shorthand padding rules:

```
div {  
  padding: 5px 10px 8px 15px;  
}
```

```
li {  
  padding: 5x 10px;  
}
```

- The border CSS property sets an element's border. It's a shorthand for border-width, border-style, and border-color
- Example

```
.box {  
  border: 10px solid #000;  
}
```

- **Border** – define the style of the borders:
 - **width** (e. g. **1px** / **2px** / **3px**)
 - **style** (e. g. **solid** / **dashed** / **dotted**)
 - **color** (e. g. **blue** / **#eee**)

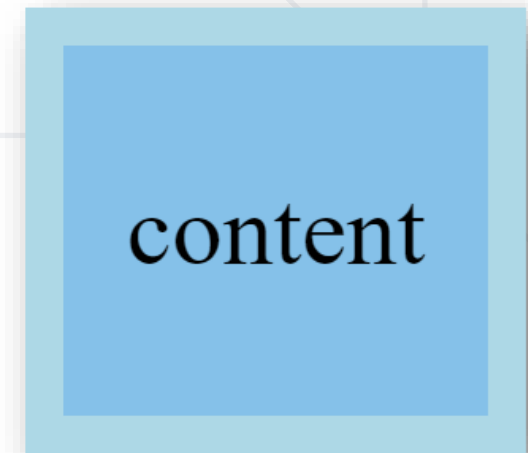
width

style

color

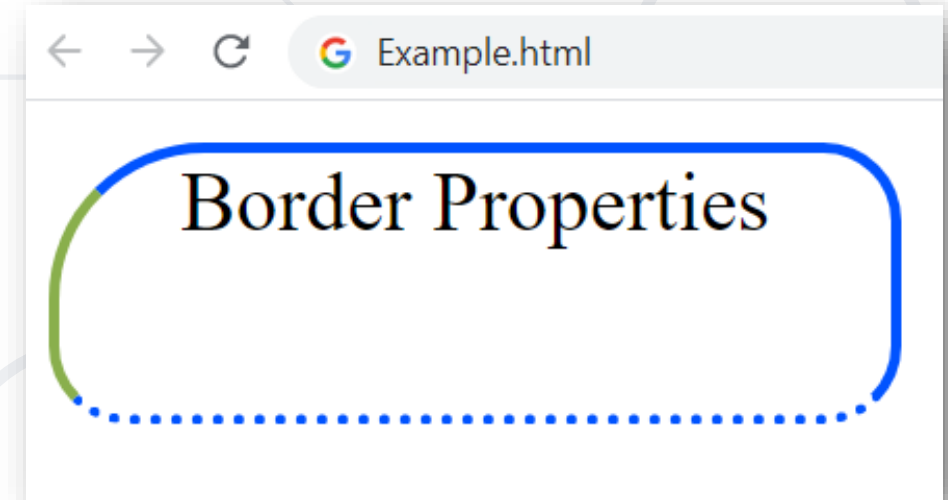
```
border: 4px dashed navy;
```

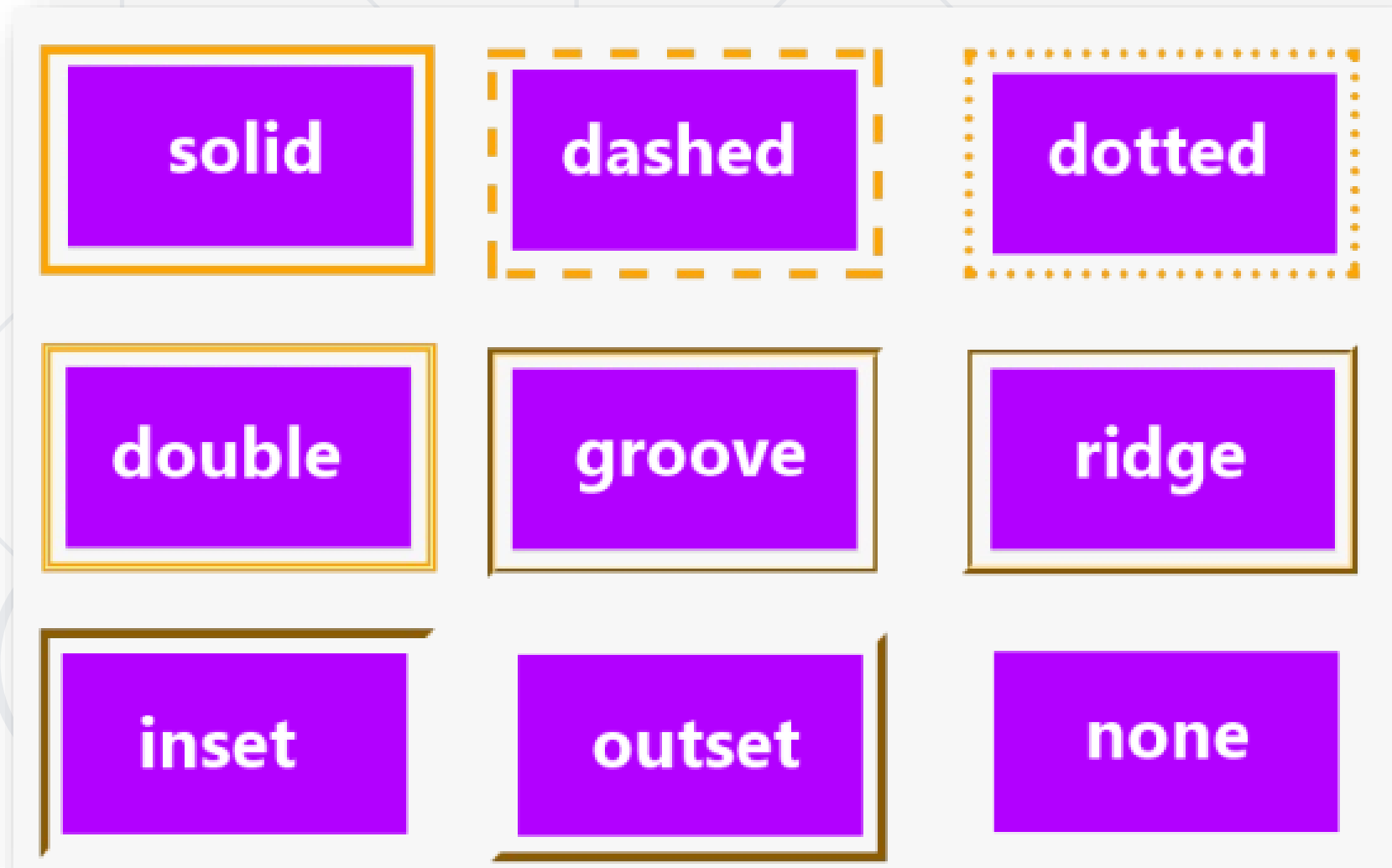
```
border: 6px solid lightblue;
```



Border Properties

```
div {  
  width: 160px;  
  height: 50px;  
  border-width: 2px;  
  border-style: solid;  
  border-color: #0053ff;  
  border-radius: 15px;  
  border-top-left-radius: 30px;  
  border-bottom-style: dotted;  
  border-left-color: #89af4c;  
  text-align: center;  
}
```







The CSS Box Model

- In CSS we have several types of boxes that generally fit into the categories of block boxes and inline boxes. The type refers to how the box behaves in terms of page flow and in relation to other boxes on the page. Boxes have an **inner display type** and **an outer display type**
- In general, you can set various values for the display type using the **display** property, which can have various values



"BLOCK BOX"



"INLINE BOXES"

Outer display type - Block

- If a box has an outer display type of **block**, then:
 - The box will break onto a new line.
 - The **width** and **height** properties are respected.
 - **Padding**, **margin** and **border** will cause other elements to be pushed away from the box.
 - If **width** is not specified, the box will extend in the **inline** direction to fill the space available in its container.
 - Some HTML elements, such as **<h1>** and **<p>**, use **block** as their outer display type by default.



"BLOCK BOX"



"INLINE BOXES"

Outer display type - Inline

- If a box has an outer display type of **inline**, then:
 - The box will not break onto a new line.
 - The **width** and **height** properties will not apply.
 - **Top** and **bottom padding, margins**, and **borders** will apply but will not cause other **inline** boxes to move away from the box.
 - **Left** and **right padding, margins**, and **borders** will apply and will cause other **inline** boxes to move away from the box.
 - Some HTML elements, such as **<a>**, ****, **** and **** use inline as their outer display type by default.

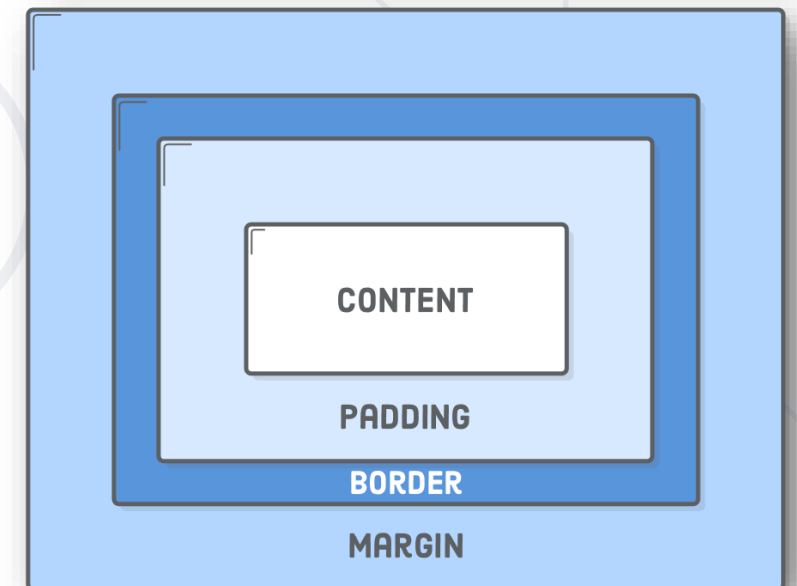


"BLOCK BOX"

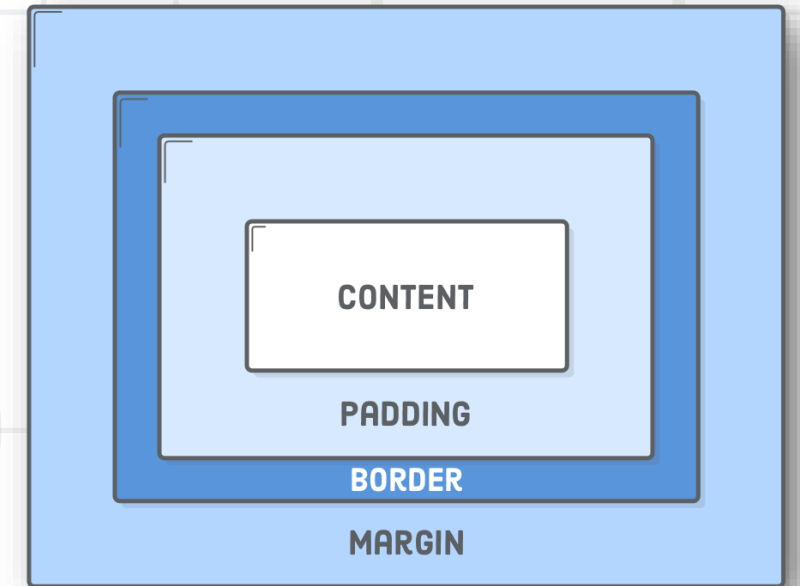


What is the CSS box model?

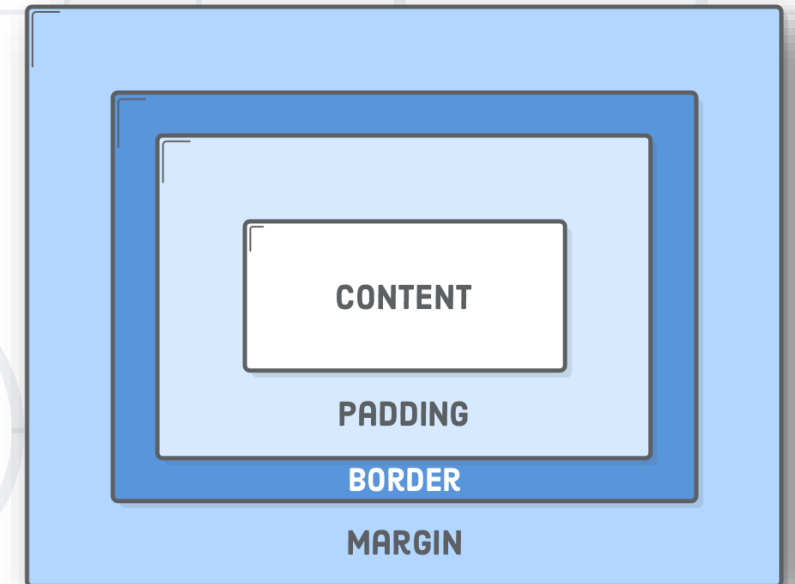
- The CSS box model defines how the different parts of a box — margin, border, padding, and content — work together to create a box that you can see on a page. Inline boxes use just some of the behavior defined in the box model
- To add complexity, there is a standard and an alternate box model. By default, browsers use the standard box model



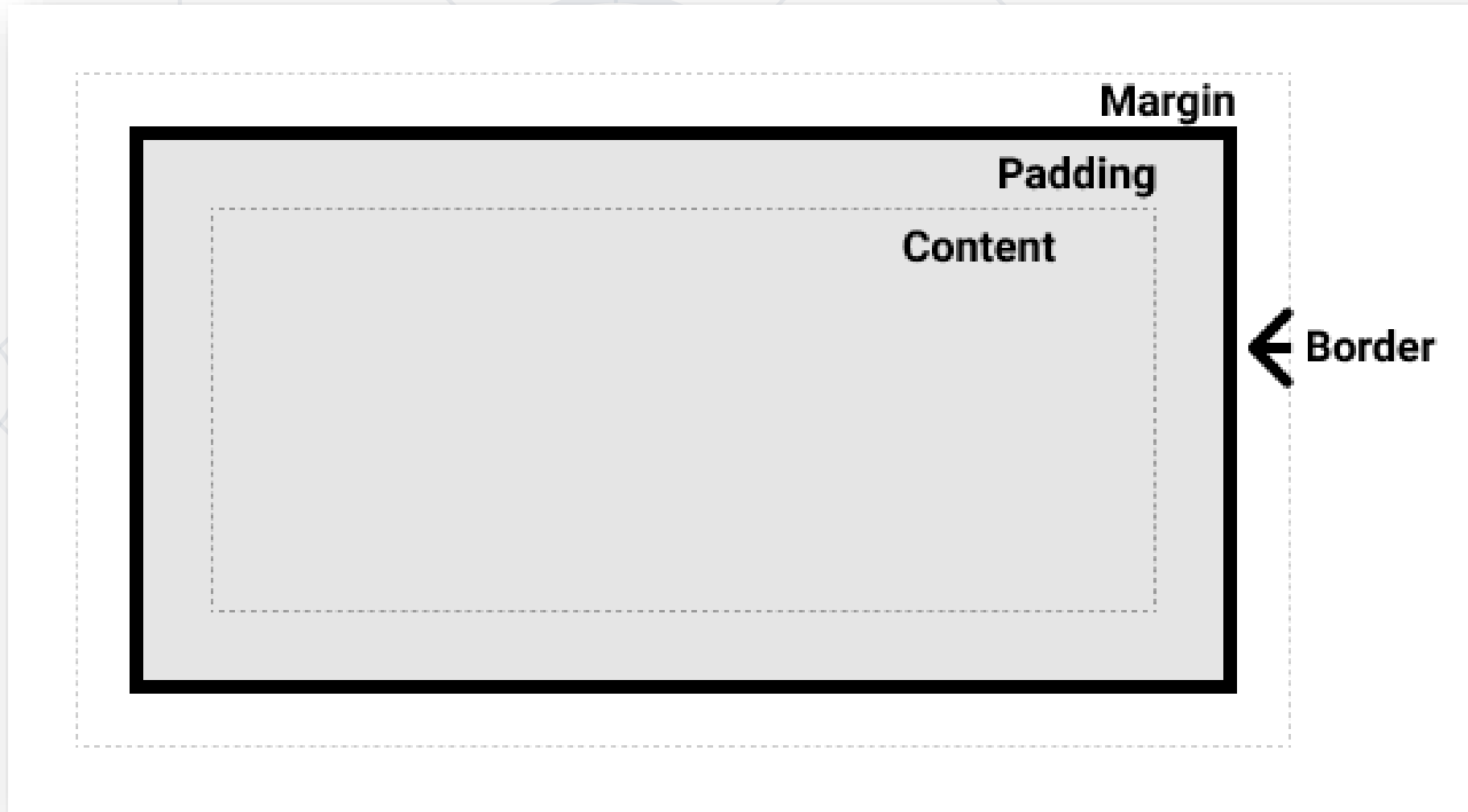
- Making up a block box in CSS we have the:
 - **Content box**: The area where your content is displayed; size it using properties like inline-size and block-size or width and height
 - **Padding box**: The padding sits around the content as white space; size it using padding and related properties



- Making up a block box in CSS we have the:
 - **Border box**: The border box wraps the content and any padding; size it using border and related properties
 - **Margin box**: The margin is the outermost layer, wrapping the content, padding, and border as whitespace between this box and other elements; size it using margin and related properties

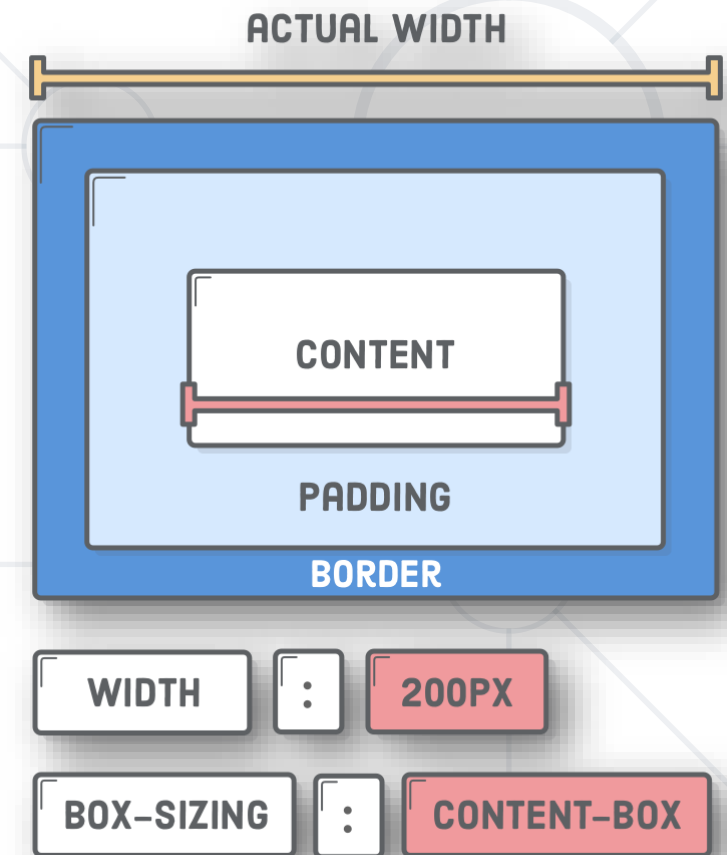


What is the CSS box model?



The standard CSS box model

- In the standard box model, if you give a box an inline-size and a block-size (or width and a height) attributes, this defines the inline-size and block-size (width and height in horizontal languages) of the content box
- Any padding and border is then added to those dimensions to get the total size taken up by the box



The standard CSS box model

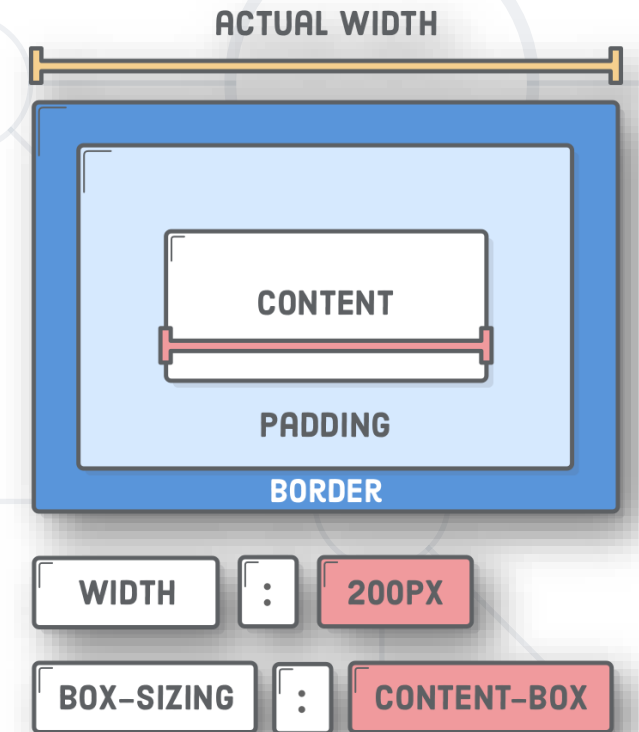
- If we assume that a box has the following CSS:
- The actual space taken up by the box will be 410px wide ($350 + 25 + 25 + 5 + 5$) and 210px high ($150 + 25 + 25 + 5 + 5$)

```
.box {  
  width: 350px;  
  height: 150px;  
  margin: 10px;  
  padding: 25px;  
  border: 5px solid black;  
}
```



The alternative CSS box model

- In the alternative box model, any width is the width of the visible box on the page
- The content area width is that width minus the width for the padding and border (see image below)
- No need to add up the border and padding to get the real size of the box



The alternative CSS box model

- To turn on the alternative model for an element use:

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



The alternative CSS box model

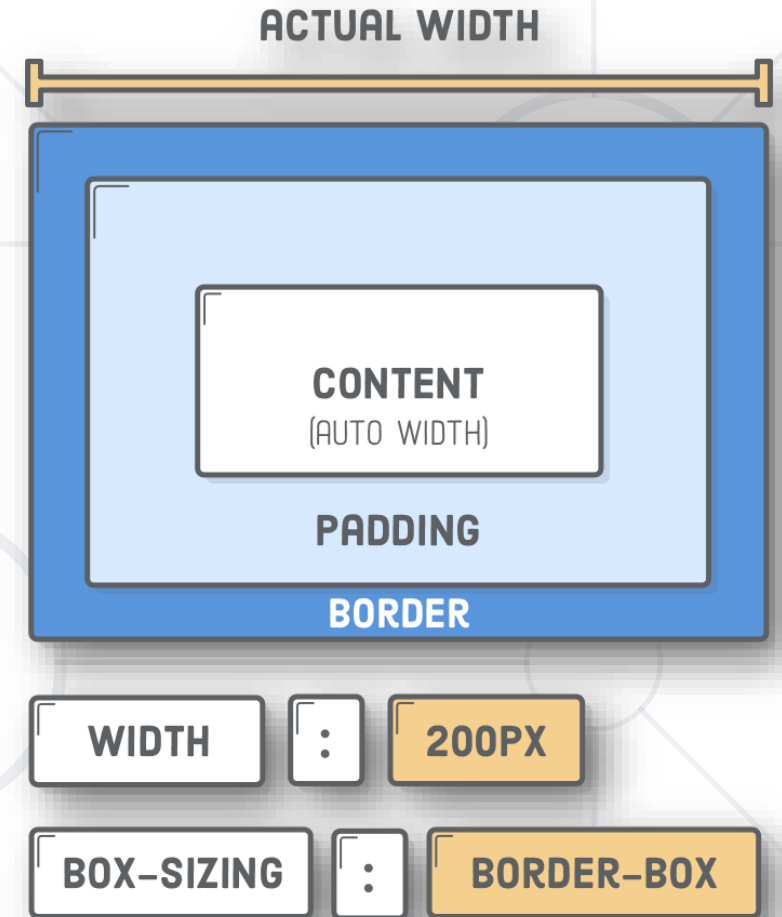
- If we assume the box has the same CSS as above:
- Now, the actual space taken up by the box will be 350px in the inline direction and 150px in the block direction

```
.box {  
  width: 350px;  
  height: 150px;  
  margin: 10px;  
  padding: 25px;  
  border: 5px solid black;  
}
```



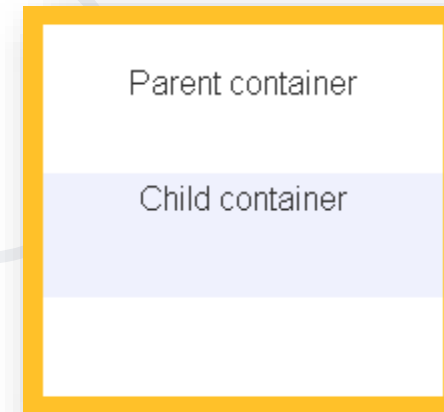
Universal box sizing with inheritance

```
html {  
  box-sizing: border-box;  
}  
  
*, *:before, *:after {  
  box-sizing: inherit;  
}
```



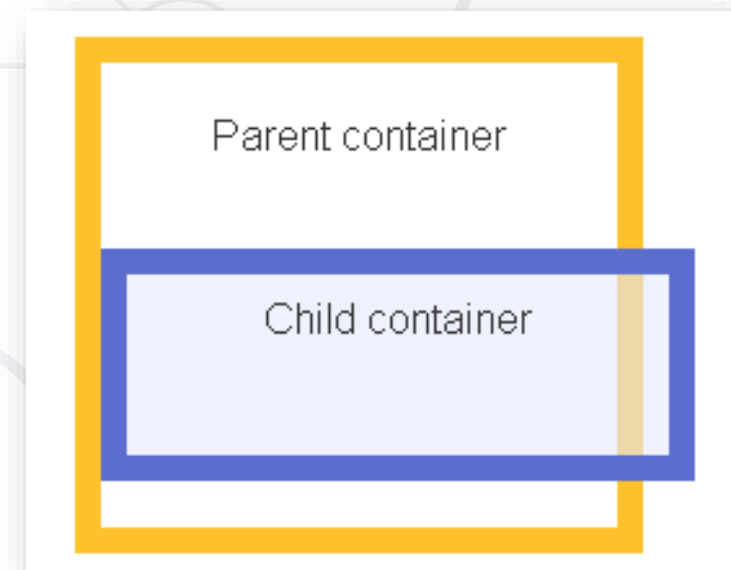
- Sets how the total width and height of an element is calculated
 - **content-box** - initial and default value
 - The **width** and **height** properties include the content
 - They **do NOT include** the padding, border and margin

```
div {  
  box-sizing: content-box;  
  width: 200px;  
}
```



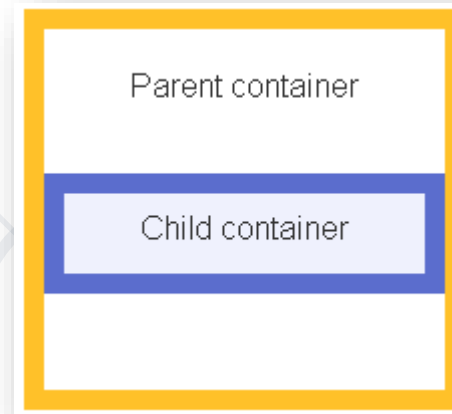
- The full width is: $200\text{px} + 2 \cdot 10\text{px} + 2 \cdot 5\text{px} = 230\text{px}$

```
div {  
  box-sizing: content-box;  
  width: 200px;  
  border: 10px solid #5b6dcd;  
  padding: 5px;  
}
```



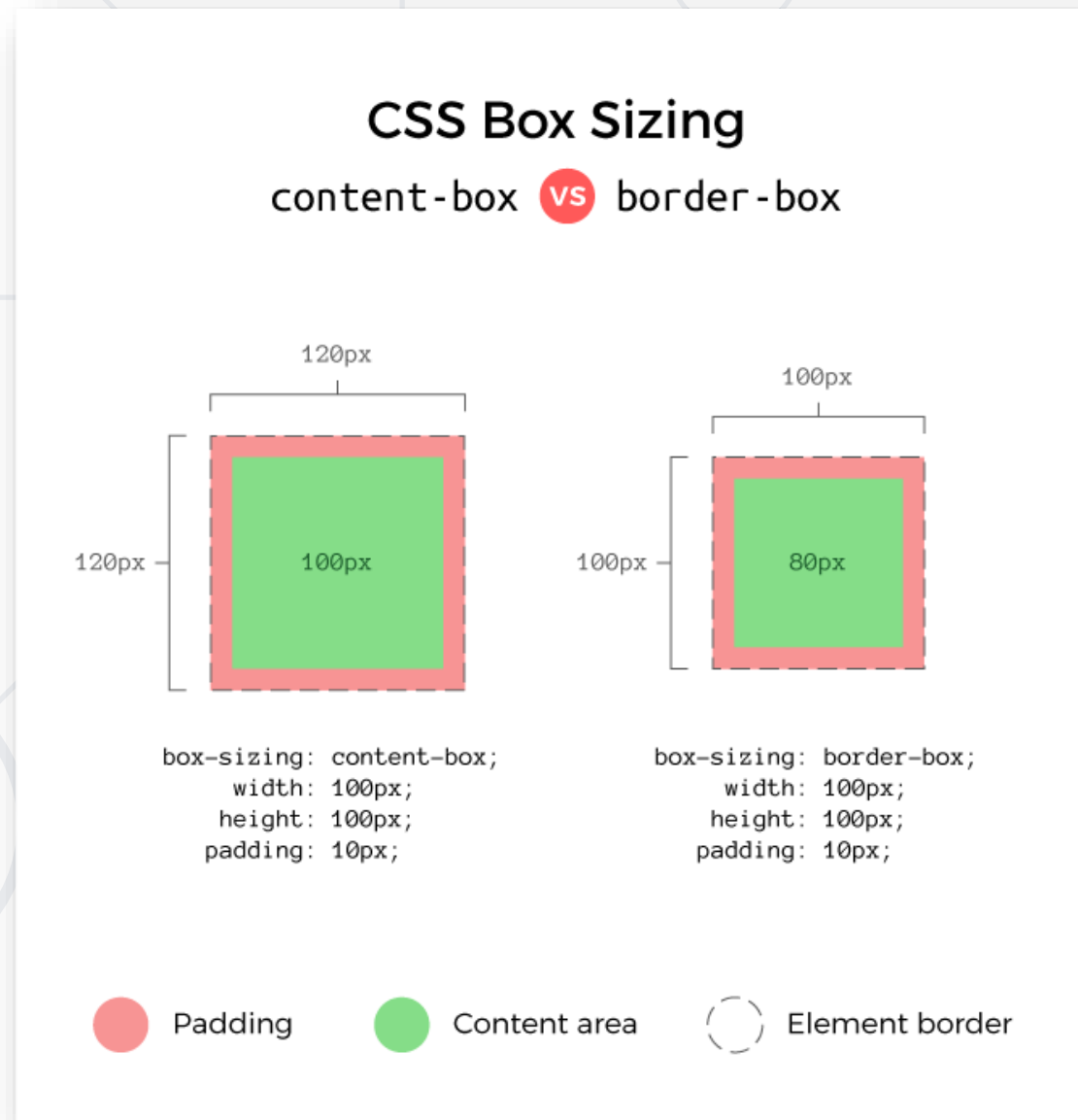
- **border-box** - the **width** and **height** of the element apply to all parts of the element: the **content**, the **padding** and the **borders**

```
box-sizing: border-box;  
width: 200px;  
border: 10px solid #5b6dcd;  
padding: 5px;
```



- The full width is **200px**
- The content width is equal to: **$200\text{px} - 2 * 10\text{px} - 2 * 5\text{px} = 170\text{px}$**

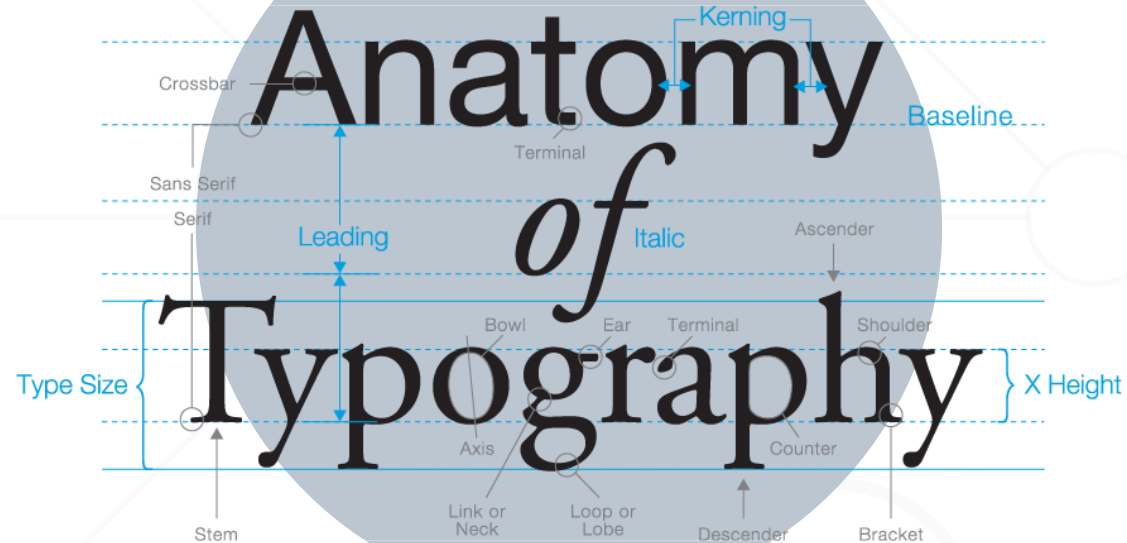
Content-box vs Border-box



- The box-sizing **Reset** takes care of the box-sizing of every element by setting it to border-box using universal CSS selector
- Save your **time** and don't write the same thing **again-and-again**
- Set the "**universal box-sizing**" with inheritance:

```
html {  
    box-sizing: border-box;  
}  
  
*,  
*:before,  
*:after {  
    box-sizing: inherit;  
}
```

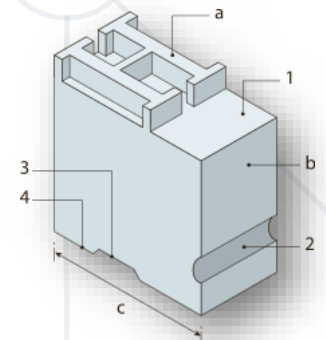
Anatomy *of* Typography



Typography

- Choosing a comfortable **measure**
 - *The **measure** is the number of characters in single line of a column of text*
 - Anything from 45 to 75 characters is widely regarded as a satisfactory length of line for a single-column page set in a serifed text face in a text size
 - The 66-character line (counting both letters and spaces) is widely regarded as ideal. For multiple column work, a better average is 40 to 50 characters

- Choose a basic **leading**
 - **Leading** (pronounced "ledding") is so called because, in mechanical presses, strips of lead are placed between lines of type to space the lines apart
 - Leading is achieved in css through the line-height property



- In typography, a font family (**also known as typeface**) is a set of one or more fonts each composed of glyphs that share common design features
- Each font of a typeface has a specific weight, style, condensation, width, slant, italicization, ornamentation, and designer or foundry



- A computer font (or font) is implemented as a digital data file containing a set of graphically related glyphs, characters, or symbols such as dingbats
- Although the term font first referred to a set of movable metal type pieces in one style and size, since the 1990s it is generally used to refer to a set of digital shapes in a single style, scalable to different sizes

- serif
- sans-serif
- monospace
- cursive
- fantasy

Serif

Sans-serif

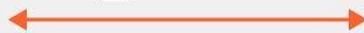
Monospace

Cursive

Fantasy

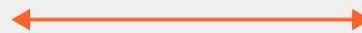
- An explanation of EMs
 - Ems are so-called because they are thought to approximate the size of an uppercase letter M, although 1em is significantly larger than this

px



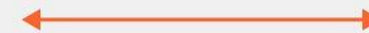
is an
absolute value

rem



multiple of the given
root font-size

em



multiple of the font-size
of the **element on**
which it is used

- Bringhurst describes the EM thus:
 - *The em is a sliding measure. One em is a distance equal to the type size*
 - *In 6-point type, an em is 6 points; in 12 point type an em is 12 points and in 60 point type an em is 60 points*
 - *Thus, a one em space is proportionately same in any size*

- A technique to refer to and automatically download remote fonts was first specified in the CSS2 specification, which introduced the '**@font-face**' construct

```
@font-face {  
  font-family: "Trickster";  
  src:  
    local("Trickster"),  
    url("trickster-COLRv1.otf") format("opentype") tech(color-COLRv1),  
    url("trickster-outline.otf") format("opentype"),  
    url("trickster-outline.woff") format("woff");  
}
```

- At the time, fetching font files from the web was controversial because fonts meant to be used only for certain web pages could also be downloaded and installed in breach of the font license

Web fonts
Web fonts
Web fonts

- In 2010, the WOFF compression method for TrueType and OpenType fonts was submitted to W3C by the Mozilla Foundation, Opera Software and Microsoft, and browsers have since added support

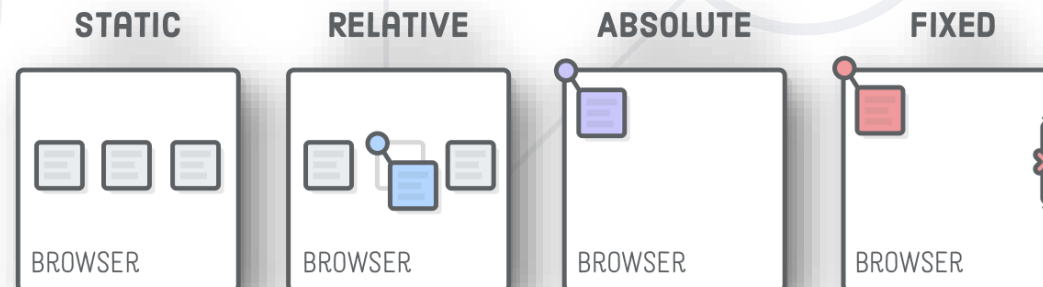


Google Fonts



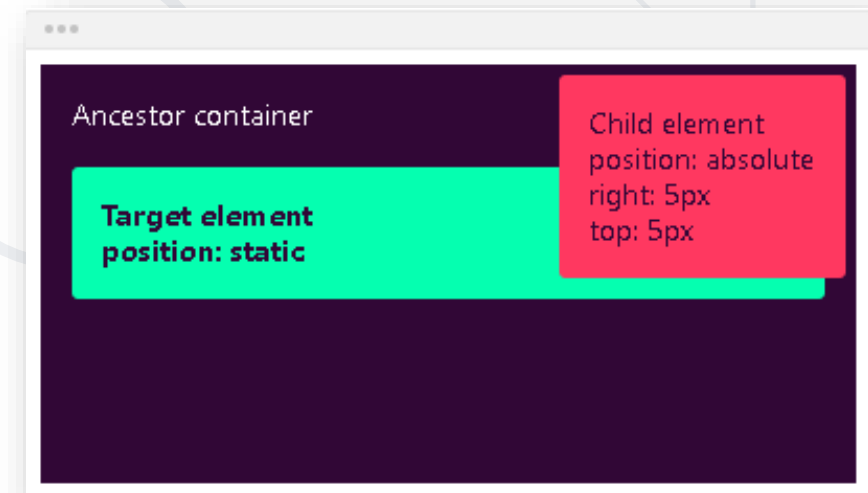
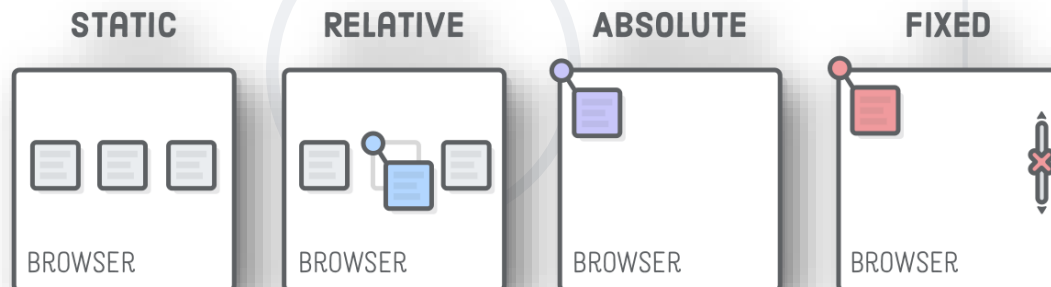
Position

- The position property specifies the type of positioning method used for an element (**static**, **relative**, **fixed**, **absolute** or **sticky**)
- Elements are then positioned using the **top**, **bottom**, **left**, and **right** properties
- However, these properties will not work unless the position property is set first
- They also work differently depending on the position value
- *Reference Documentation*



- Static - the **default** state of every element
 - Puts the element into its **normal position** in the document layout flow
- It will **NOT** react to the following properties: **top, bottom, left, right, z-index**

```
div {  
  position: static;  
}
```



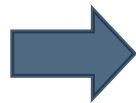
Position Relative

- It looks like static positioning, but once the positioned element has taken its place, you can then modify its final position with the **positional properties**

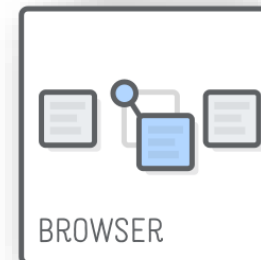
```
  

```

```
img.new {  
  position: relative;  
  top: -200px;  
  right: 150px;  
}
```

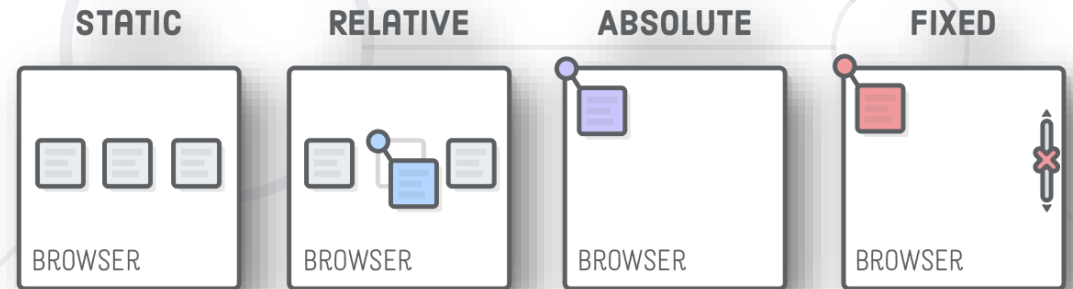


RELATIVE



- This way we have to position the element based on a two dimensional coordinate system. We can use left, top, bottom, right to place the element exactly where we want

```
.box {  
  position: absolute;  
}
```



Position Absolute

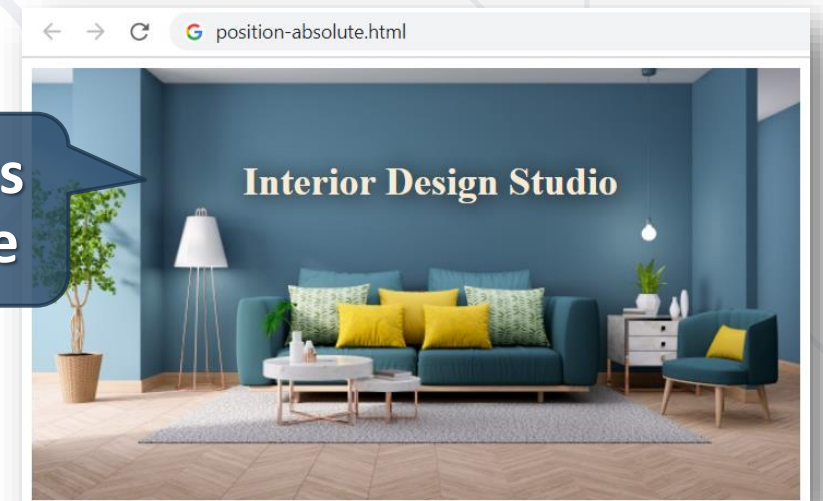
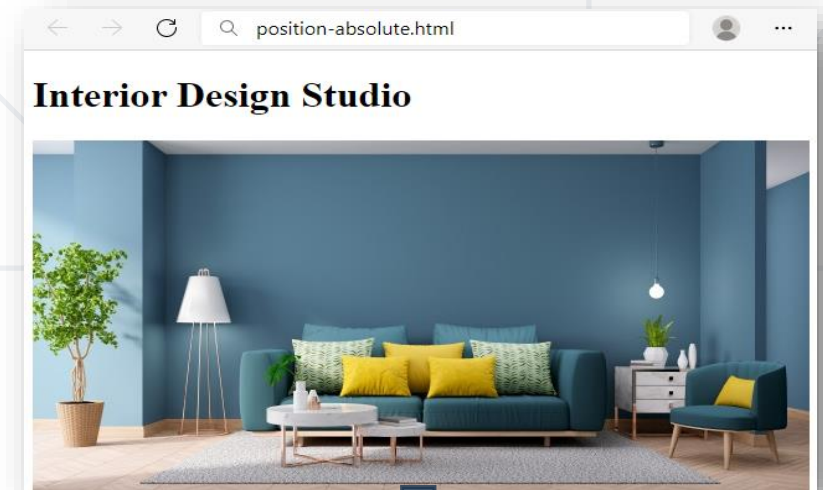
- Absolute positioning → from the **upper left corner** of the parent

```
<h1>Interior Design Studio</h1>  

```

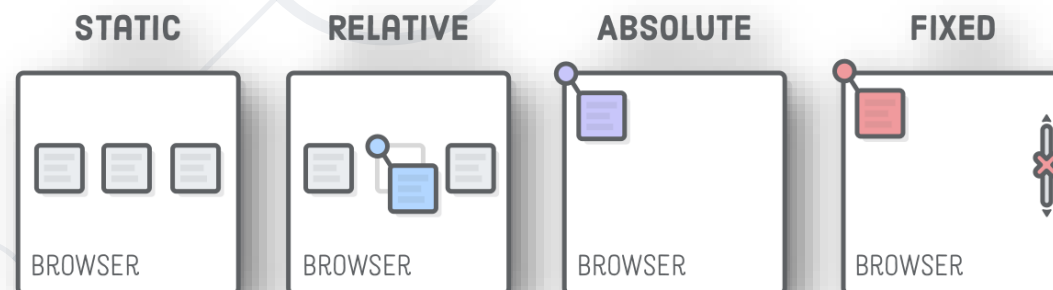
```
h1 {  
  position: absolute;  
  top: 60px;  
  left: 180px;  
  color: antiquewhite;  
  text-shadow: 1px 1px 20px black;  
}
```

<h1> frees its
original place



- The element is removed from the normal document flow, and no space is created for the element in the page layout
- The element is positioned relative to its initial containing block, which is the viewport in the case of visual media
- Its final position is determined by the values of top, right, bottom, and left

```
div {  
  position: fixed;  
}
```

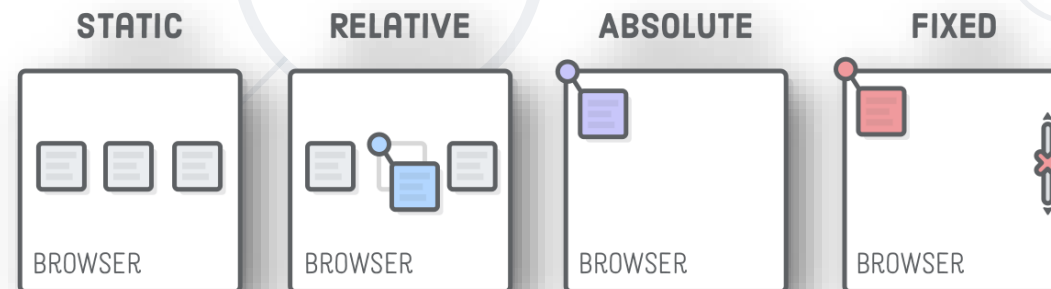


- This value always creates a new stacking context
- In printed documents, the element is placed in the same position on every page



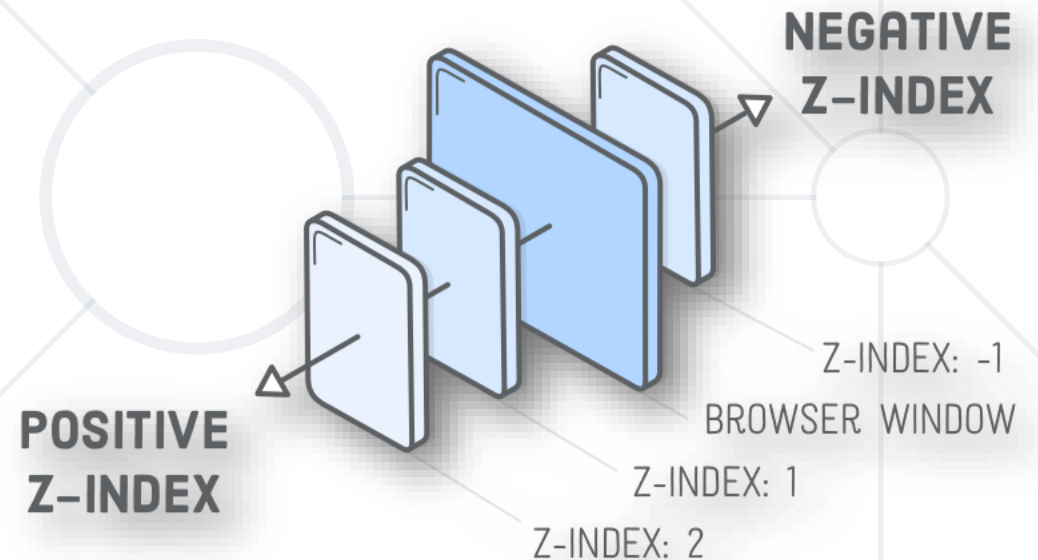
- The element is positioned according to the normal flow of the document, and then offset relative to its nearest scrolling ancestor and containing block (nearest block-level ancestor), including table-related elements, based on the values of top, right, bottom, and left.
- The offset does not affect the position of any other elements.

```
.box {  
  position: sticky;  
}
```

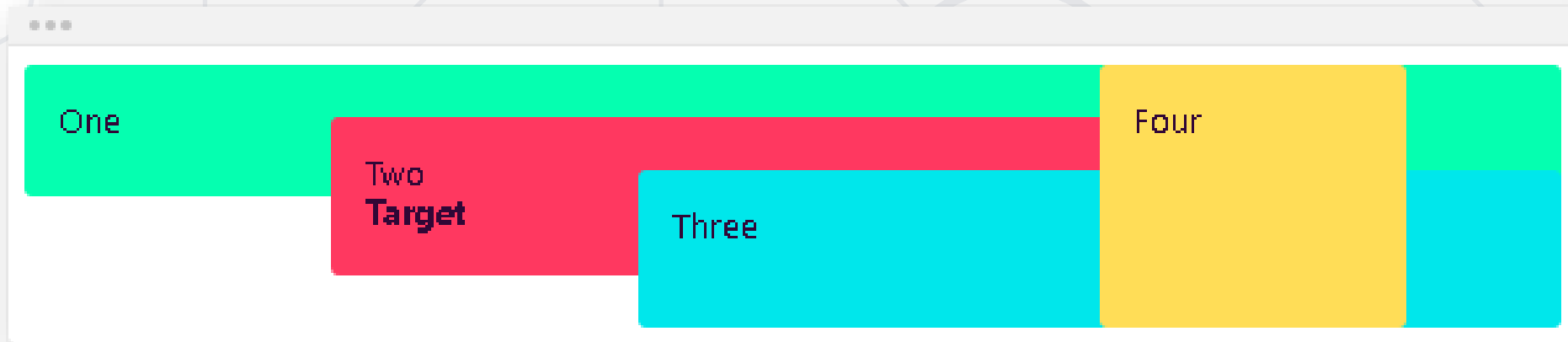


- The **z-index** CSS property sets the z-order of a positioned element and its descendants or flex items
- Overlapping elements with a larger **z-index** cover those with a smaller one
- Example

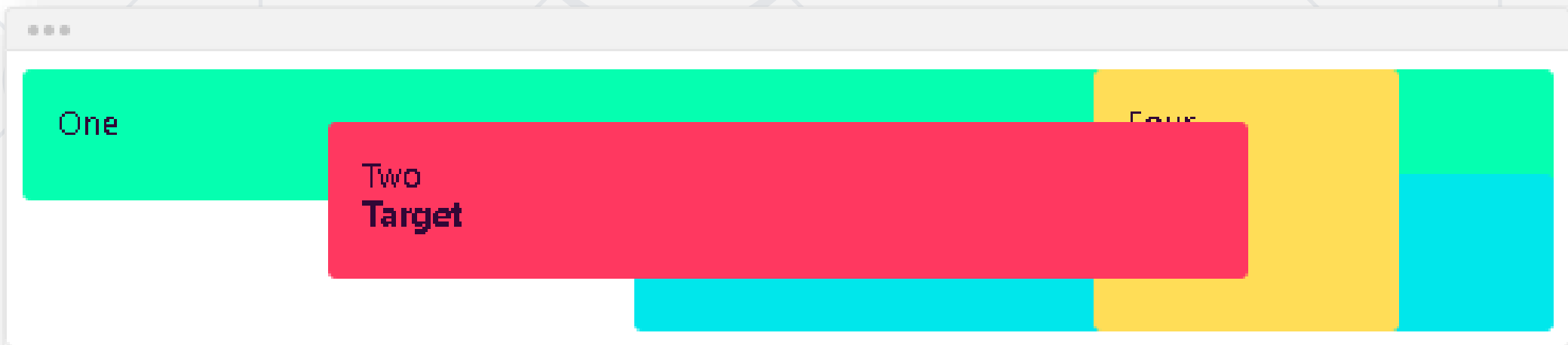
```
.box {  
  z-index: [number];  
}
```



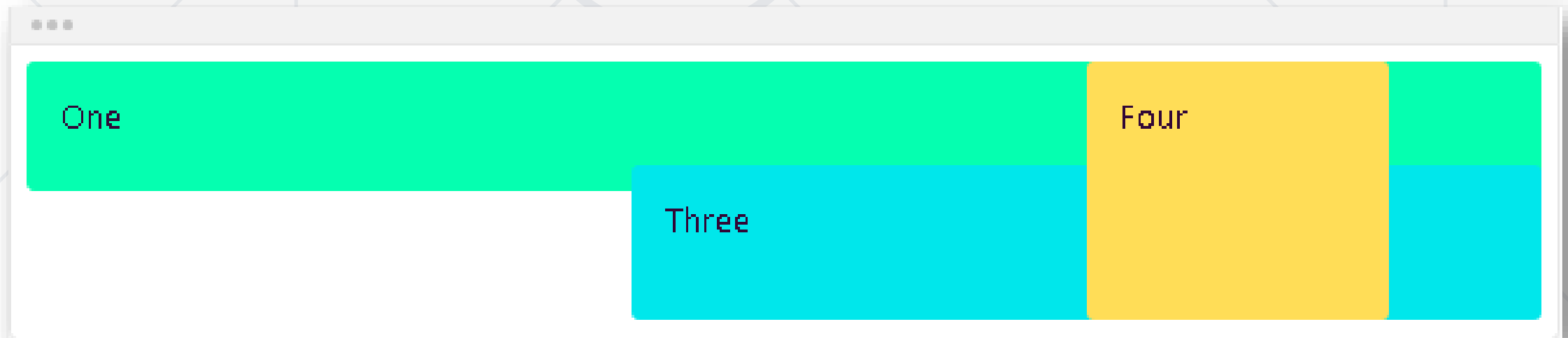
- Defines the order of the elements on the **z-axis**. It only works on positioned elements (**anything apart from static**)
 - Default value: **z-index: auto;**
 - The order is defined by the order in the HTML code:



- The z-index value is relative to the other ones
- The target element is move in **front** of its siblings
 - **z-index: 1;**



- You can use **negative** values
- The target element is moved **behind** its **siblings**
 - **z-index: -1;**



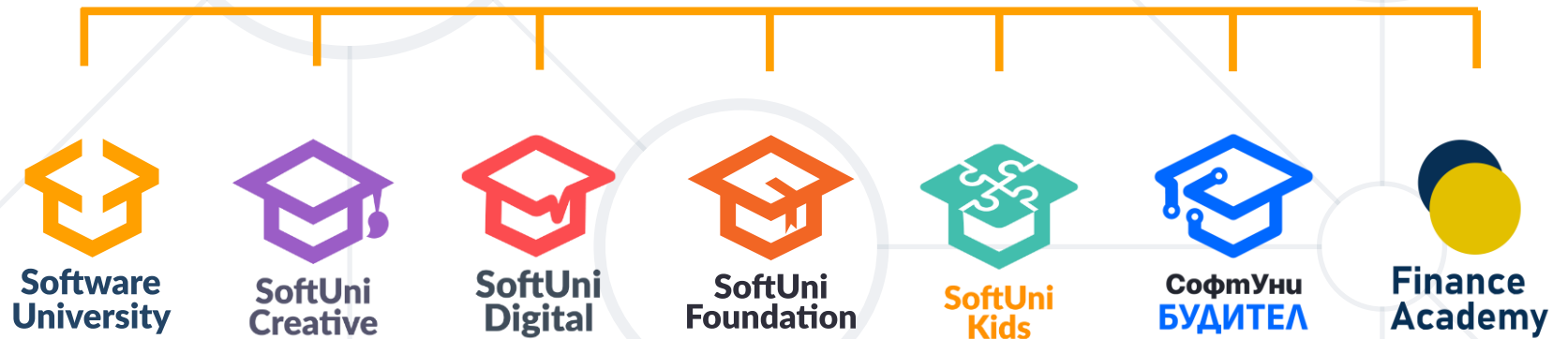
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model
- <https://developer.mozilla.org/en-US/docs/Web/CSS/display>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing>
- <https://css-tricks.com/the-css-box-model>
- <https://css-tricks.com/box-sizing>
- <https://www.paulirish.com/2012/box-sizing-border-box-ftw/>

- <https://developer.mozilla.org/en-US/docs/Web/CSS/position>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/z-index>
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index
- <https://interactive-examples.mdn.mozilla.net/pages/css/position.html>
- <https://css-tricks.com/video-screencasts/198-about-the-position-property/>
- <https://css-tricks.com/almanac/properties/p/position/>
- <https://css-tricks.com/position-sticky-2/>

- What is Box Model?
- Width and Height to the elements
- What are the Padding, Border and Margin?
- What is Box-sizing?
- How to reset Box-sizing?
- Positioning properties
- Z-Index



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, about.softuni.bg
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity

