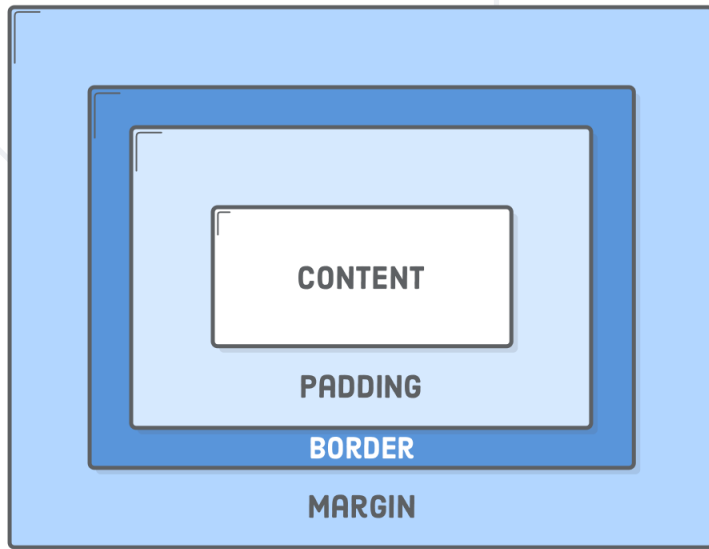


Box Model and Position



SoftUni Team
Technical Trainers



SoftUni



Software University
<https://softuni.bg>

- Block and Inline Elements
- Width and Height
- Padding, Margin, and Border
- Box Sizing
- Semantic Properties
- Position Elements on a Coordinate System

sli.do

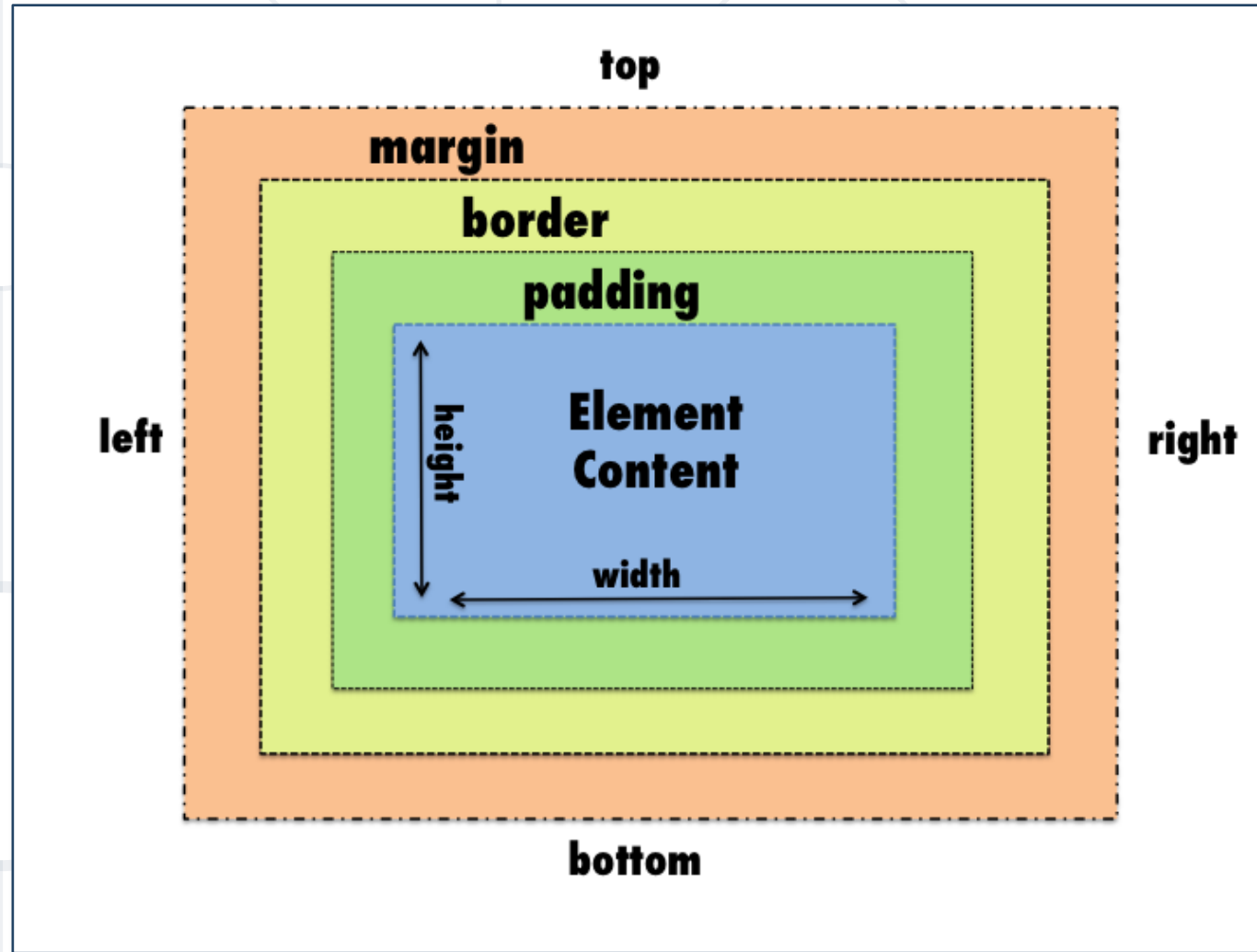
#html-css



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.
- https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model

Box Model – CSS Properties



■ Display

- 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”

Box Model – CSS Properties

- **display: block;**
 - 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".
 - 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.

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


- **display: inline;**
 - Inline elements are those which only occupy the space bounded by the tags defining the element, instead of breaking the flow of the content.

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

- **display: inline-block;**
 - Gives us the ability to use vertical padding and margin on inline elements as well as adding width and height.

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

■ width

- 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 area
- <https://interactive-examples.mdn.mozilla.net/pages/css/width.html>

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

- **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.

- **min-width**

- The min-width CSS property sets the minimum width of an element. It prevents the used value of the width property from becoming smaller than the value specified for min-width.

```
.box {  
    min-width: 150px;  
}
```

- **max-width**

- The max-width CSS property sets the maximum width of an element. It prevents the used value of the width property from becoming larger than the value specified by max-width.

```
.box {  
    max-width: 150px;  
}
```

■ height

- 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.

- <https://interactive-examples.mdn.mozilla.net/pages/css/height.html>

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

- **min-height**

- The min-height CSS property sets the minimum height of an element. It prevents the used value of the height property from becoming smaller than the value specified for min-height.

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


- **max-height**

- The max-height CSS property sets the maximum height of an element. It prevents the used value of the height property from becoming larger than the value specified for max-height.

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

■ margin

- 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.
- <https://interactive-examples.mdn.mozilla.net/pages/css/margin.html>

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

- **border**

- The border CSS property sets an element's border. It's a shorthand for border-width, border-style, and border-color.
- <https://interactive-examples.mdn.mozilla.net/pages/css/border.html>

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

- **padding**

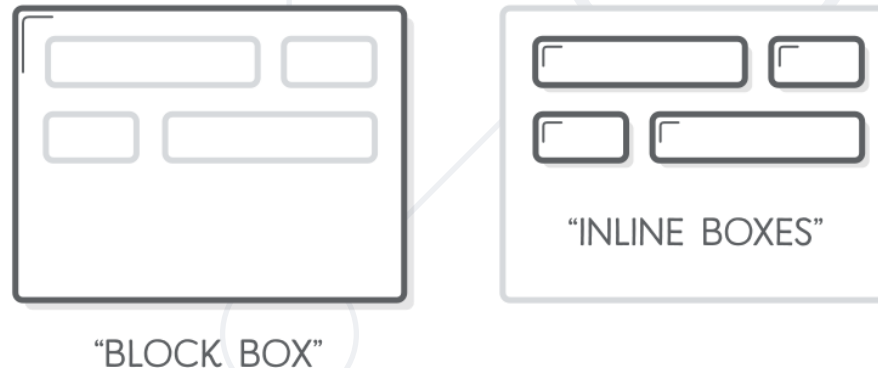
- The padding CSS property sets the padding area on all four sides of an element. It is a shorthand for padding-top, padding-right, padding-bottom, and padding-left.
- <https://interactive-examples.mdn.mozilla.net/pages/css/padding.html>



The CSS Box Model

- **Block** and **Inline** boxes

- 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.



- 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.



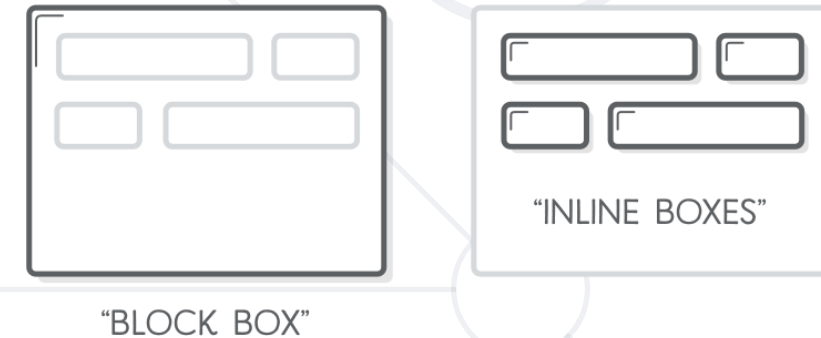
"BLOCK BOX"



"INLINE BOXES"

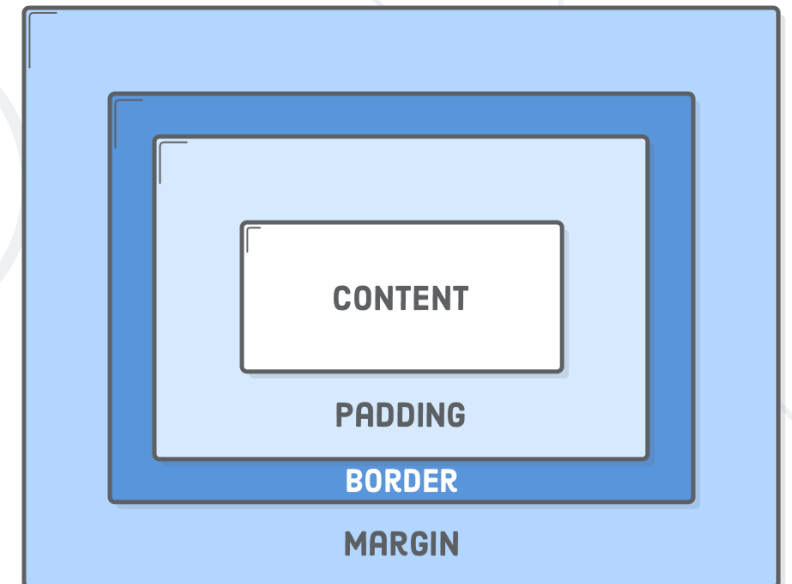
- Outer display type - **Inline**

- 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.



What is the CSS box model?

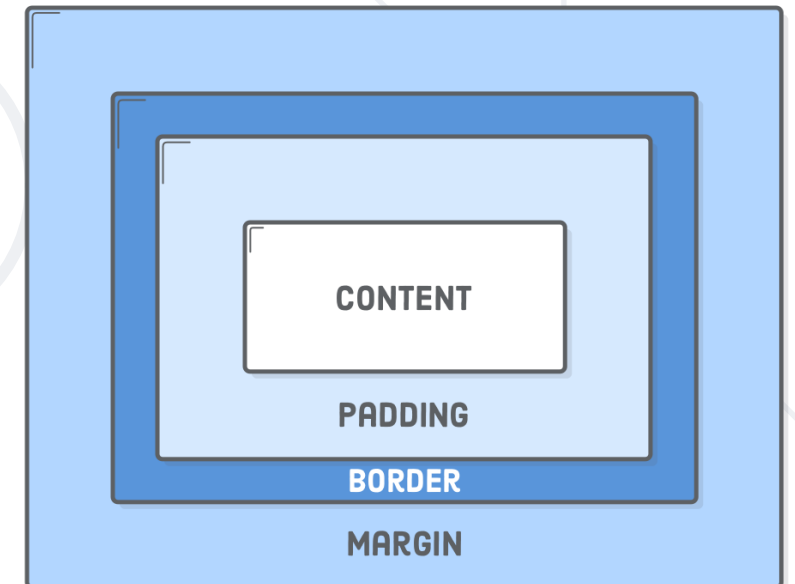
- The CSS box model defines how the different parts of a box — margin, border, padding, and content — work together to create a box that you can see on a page. Inline boxes use just some of the behavior defined in the box model.
- To add complexity, there is a standard and an alternate box model. By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



What is the CSS box model?

■ Parts of a box

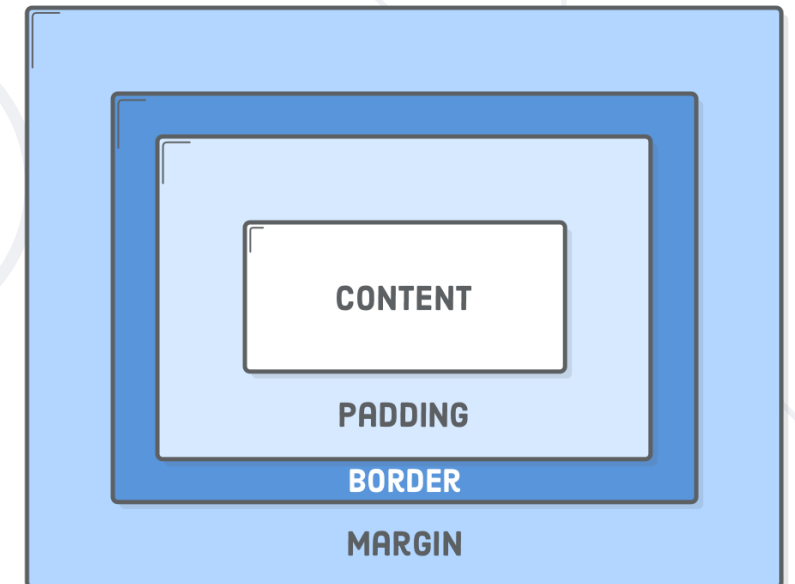
- Making up a block box in CSS we have the:
 - **Content box:** The area where your content is displayed; size it using properties like inline-size and block-size or width and height.
 - **Padding box:** The padding sits around the content as white space; size it using padding and related properties.



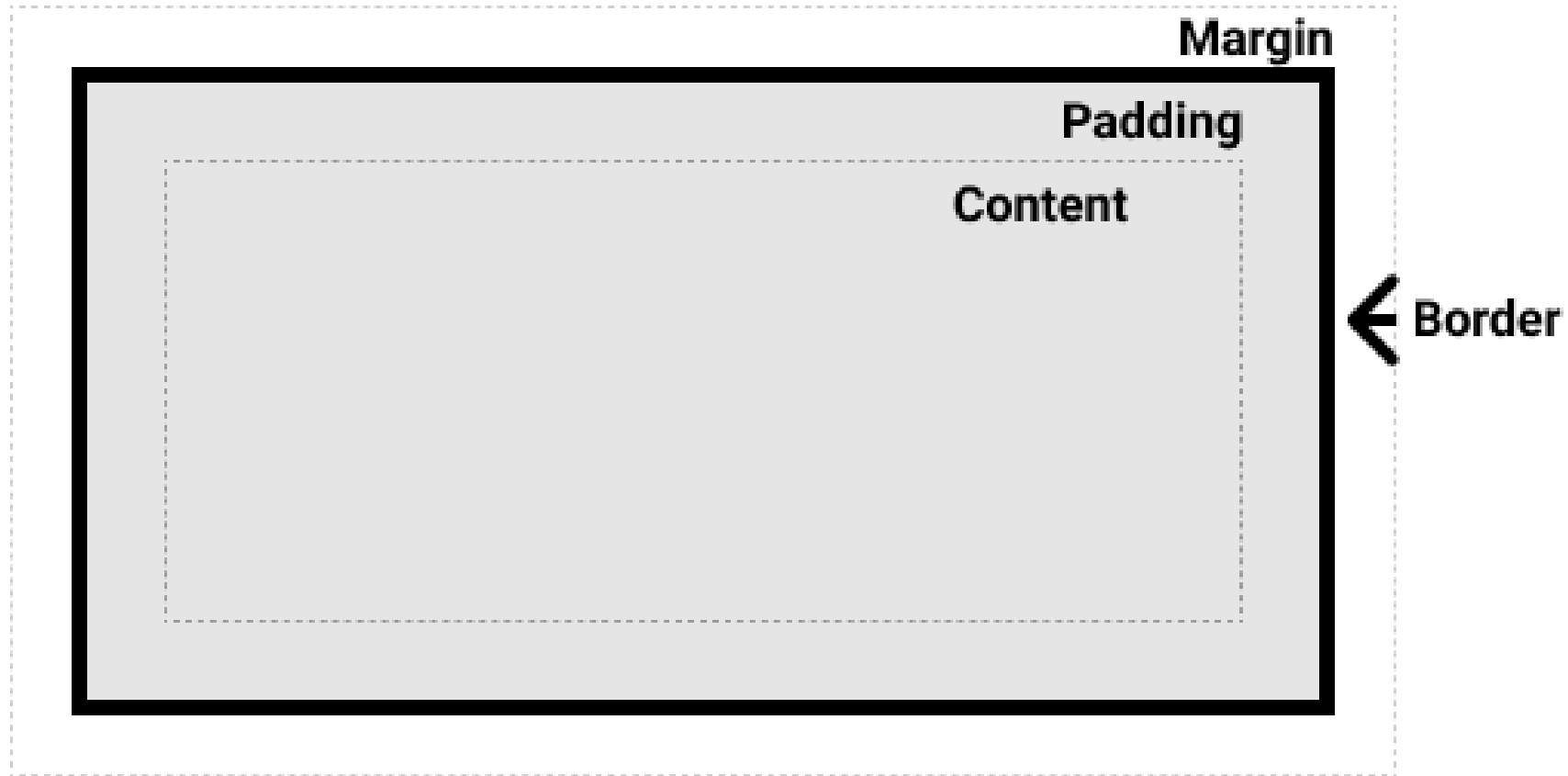
What is the CSS box model?

- **Parts of a box**

- 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.

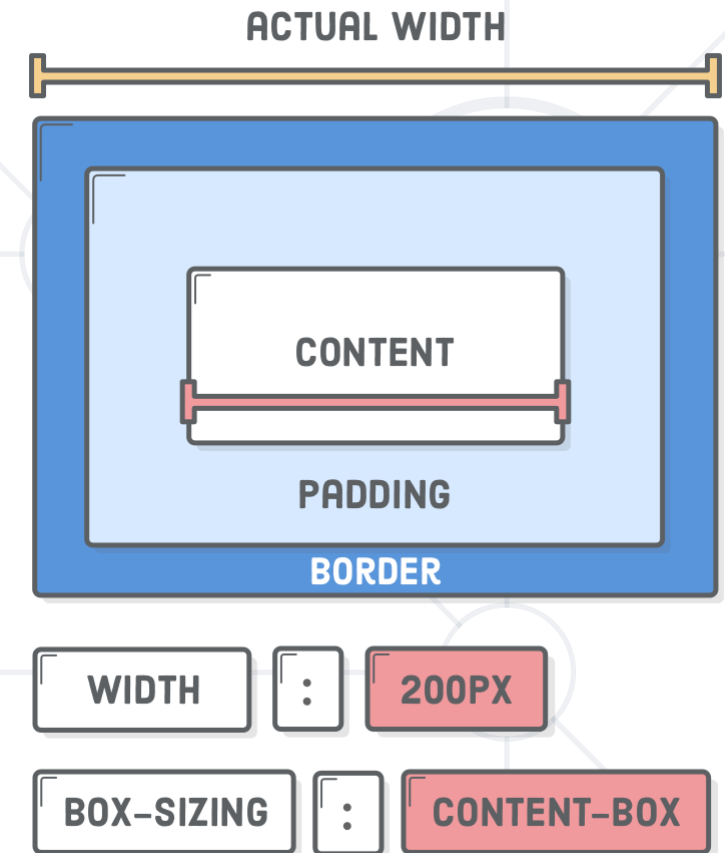


What is the CSS box model?



The standard CSS box model

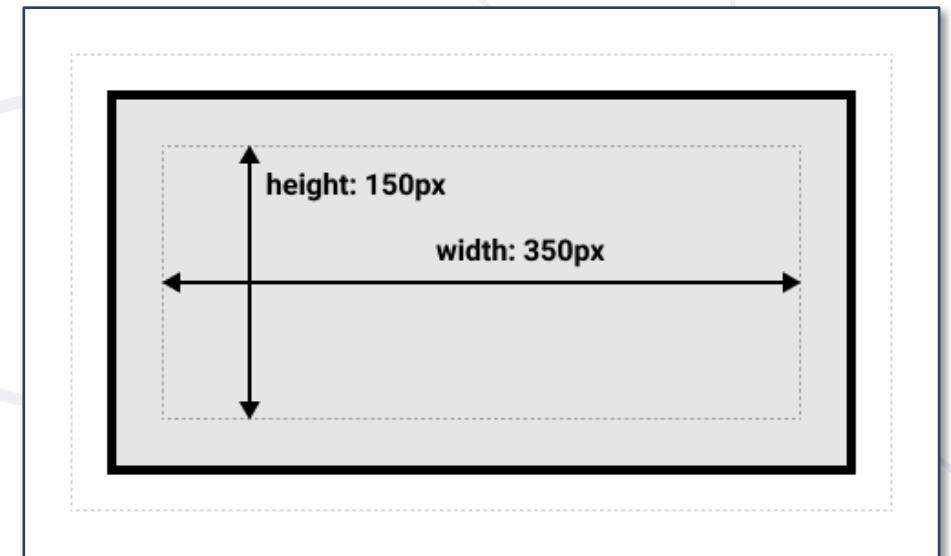
- In the standard box model, if you give a box an inline-size and a block-size (or width and a height) attributes, this defines the inline-size and block-size (width and height in horizontal languages) of the content box.
- Any padding and border is then added to those dimensions to get the total size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



The standard CSS box model

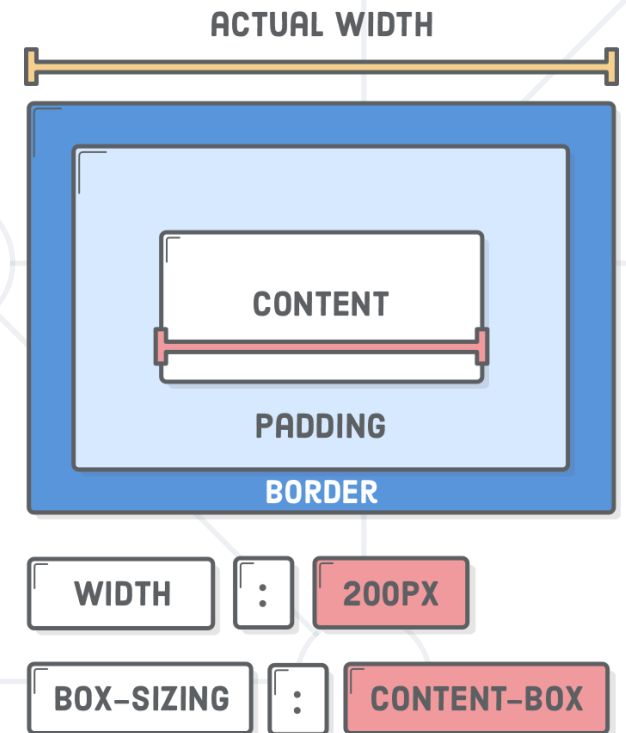
- If we assume that a box has the following CSS:
- The actual space taken up by the box will be 410px wide ($350 + 25 + 25 + 5 + 5$) and 210px high ($150 + 25 + 25 + 5 + 5$).

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



The alternative CSS box model

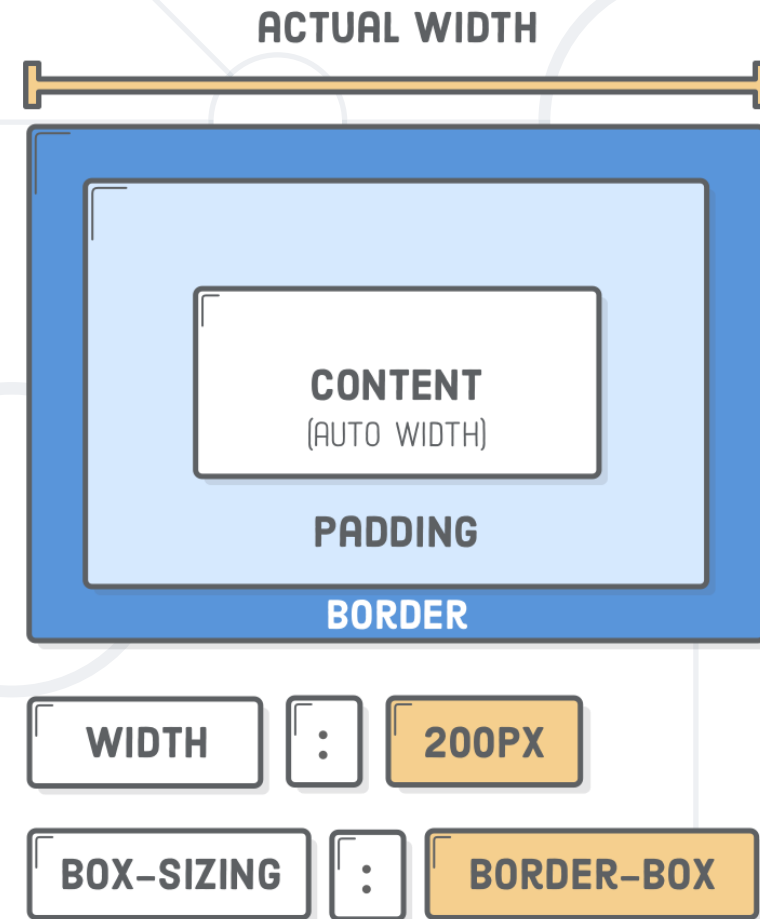
- In the alternative box model, any width is the width of the visible box on the page.
- The content area width is that width minus the width for the padding and border (see image below).
- No need to add up the border and padding to get the real size of the box.



The alternative CSS box model

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

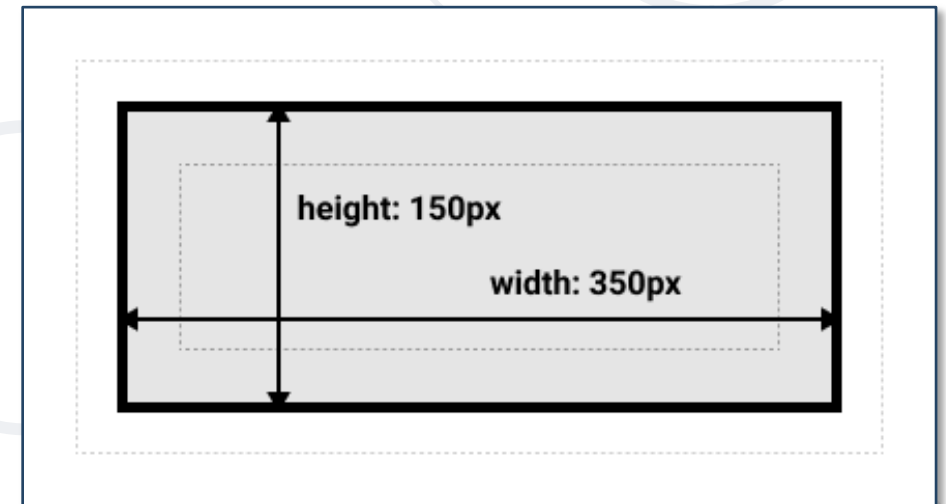
```
.box {  
  box-sizing: border-box;  
}
```



The alternative CSS box model

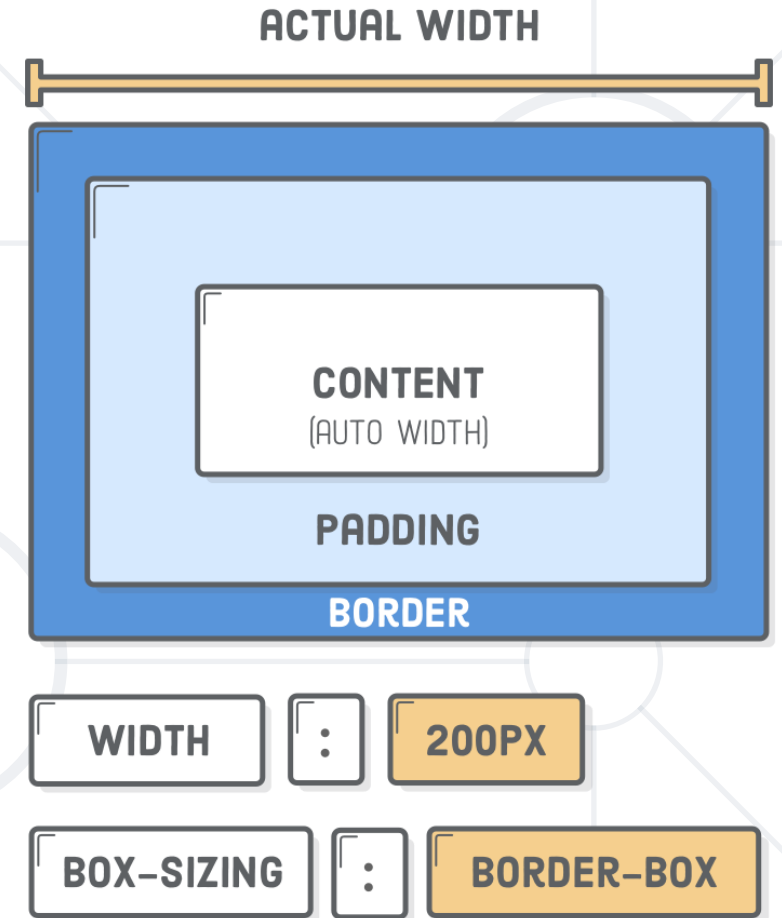
- If we assume the box has the same CSS as above:
- Now, the actual space taken up by the box will be 350px in the inline direction and 150px in the block direction.

```
.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;  
}
```





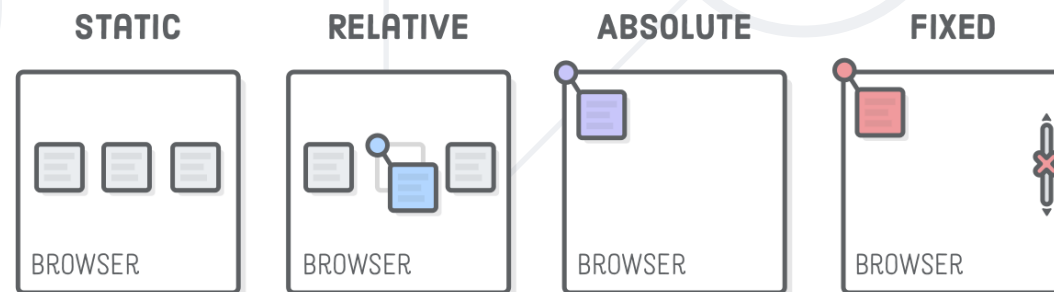
Box Model - Let's try it ...

Demo



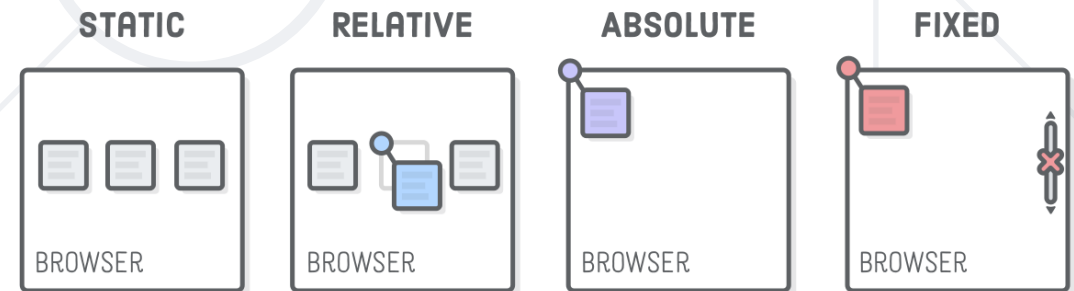
Position

- 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.
- <https://developer.mozilla.org/en-US/docs/Web/CSS/position>



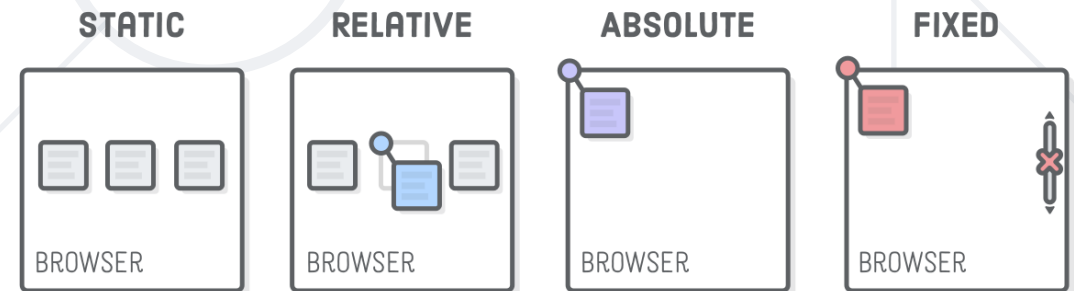
- **position: static;**
 - The default state of every element — it just means "put the element into its normal position in the document layout flow — nothing special to see here."

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



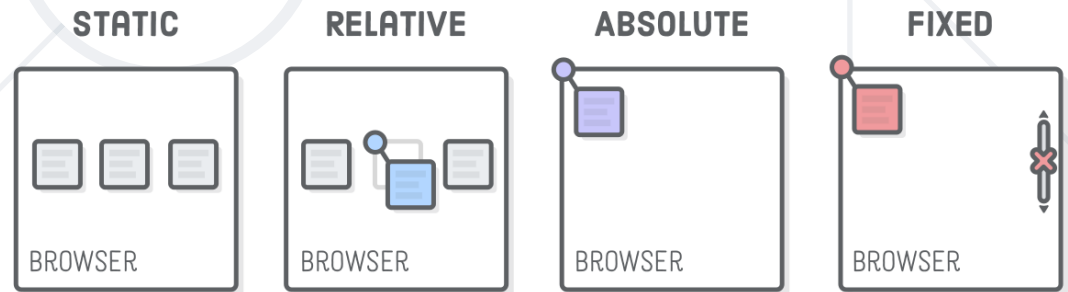
- **position: relative;**
 - Very similar to static positioning, except that once the positioned element has taken its place in the normal layout flow, you can then modify its final position, including making it overlap other elements on the page.

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



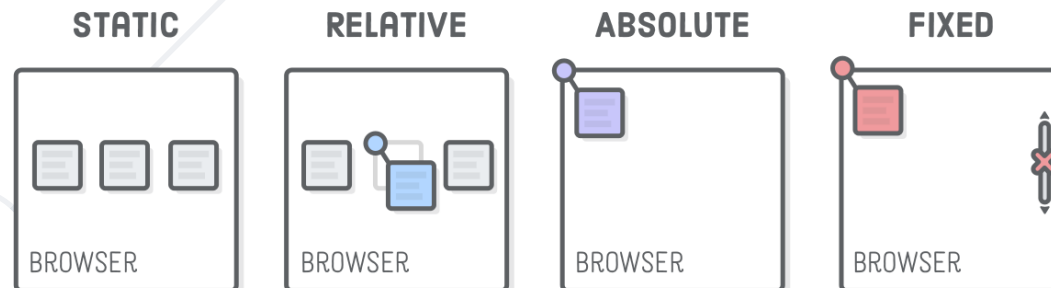
- **position: absolute;**
 - This way we have to position the element based on a two dimensional coordinate system. We can use left, top, bottom, right to place the element exactly where we want.

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



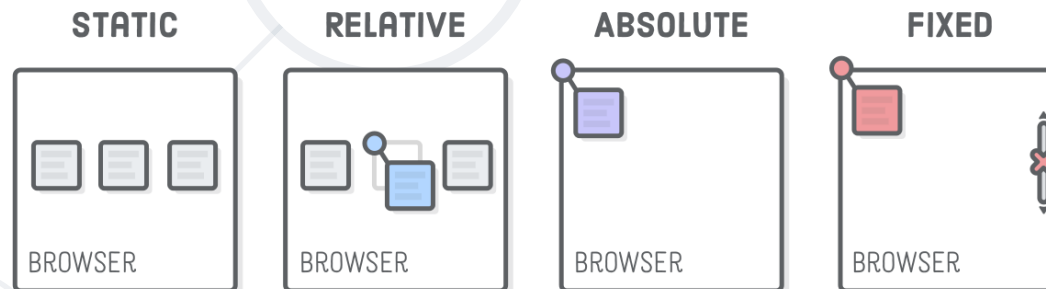
- **position: fixed;**
 - The element is removed from the normal document flow, and no space is created for the element in the page layout. The element is positioned relative to its initial containing block, which is the viewport in the case of visual media. Its final position is determined by the values of top, right, bottom, and left.
 - This value always creates a new stacking context. In printed documents, the element is placed in the same position on every page.

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



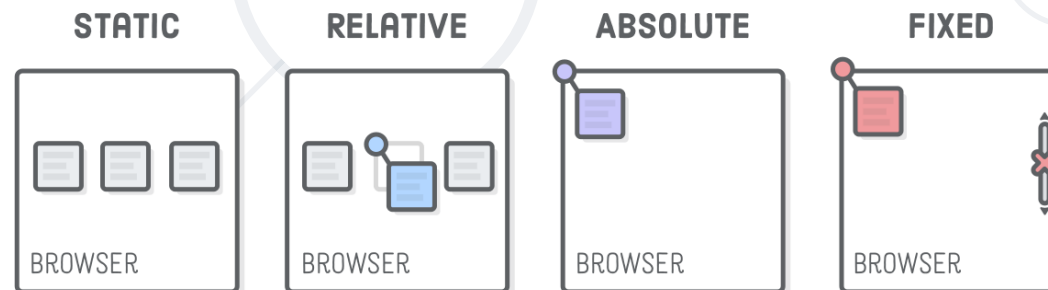
- **position: sticky;**
 - The element is positioned according to the normal flow of the document, and then offset relative to its nearest scrolling ancestor and containing block (nearest block-level ancestor), including table-related elements, based on the values of top, right, bottom, and left. The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

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



- **position: sticky;**
 - This value always creates a new stacking context. Note that a sticky element "sticks" to its nearest ancestor that has a "scrolling mechanism" (created when overflow is hidden, scroll, auto, or overlay), even if that ancestor isn't the nearest actually scrolling ancestor.

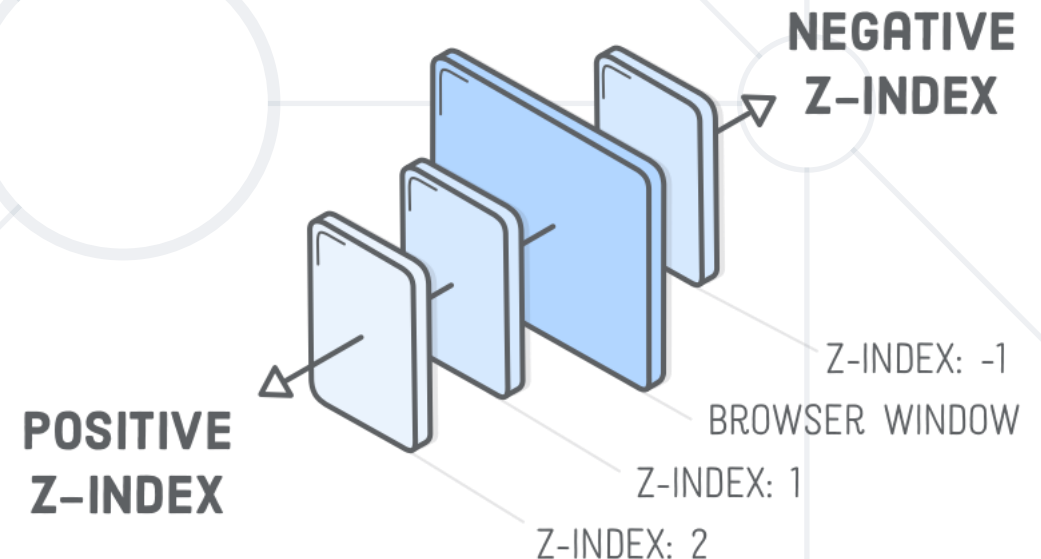
```
.box {  
  position: sticky;  
}
```



■ z-index

- The z-index CSS property sets the z-order of a positioned element and its descendants or flex items. Overlapping elements with a larger z-index cover those with a smaller one.
- <https://interactive-examples.mdn.mozilla.net/pages/css/z-index.html>

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





CSS Position - Let's try it ...

Demo

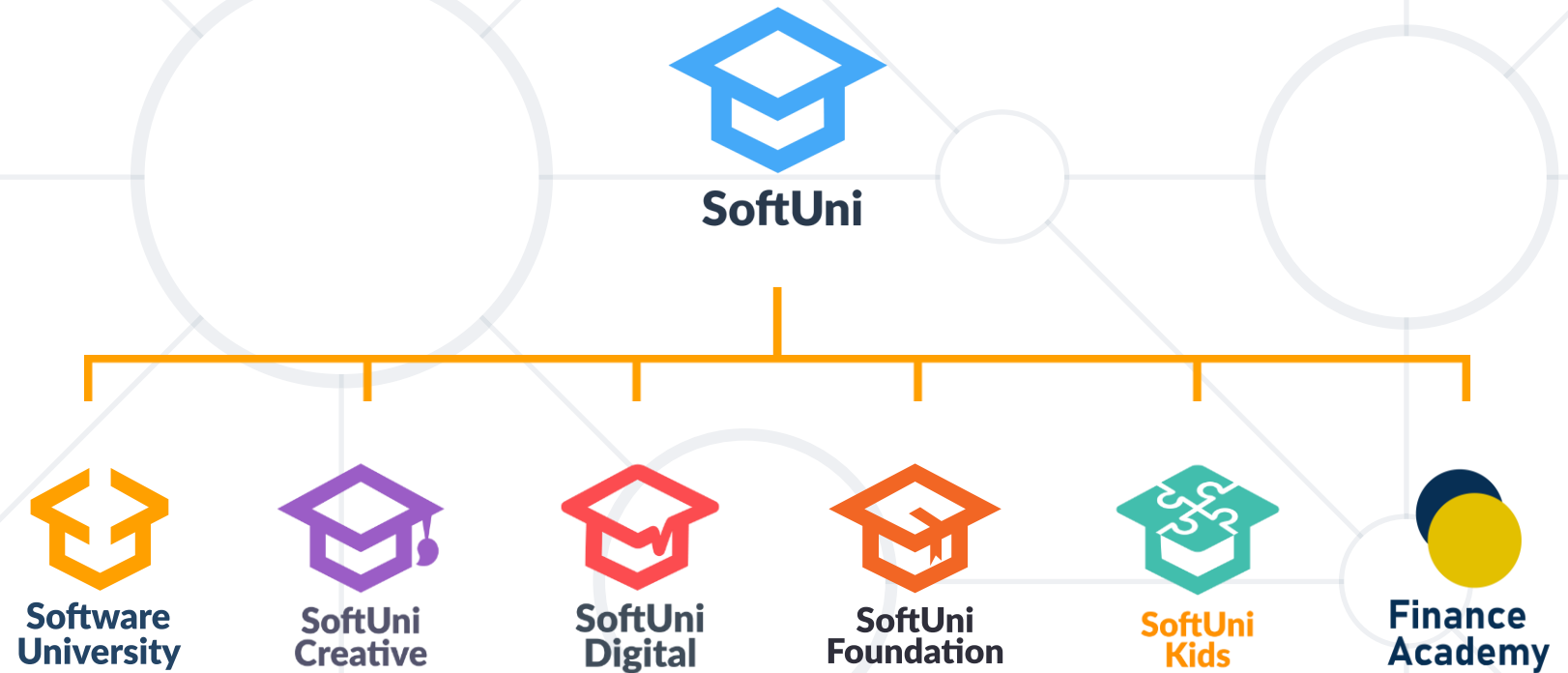
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model
- <https://developer.mozilla.org/en-US/docs/Web/CSS/display>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing>
- <https://css-tricks.com/the-css-box-model>
- <https://css-tricks.com/box-sizing>
- <https://www.paulirish.com/2012/box-sizing-border-box-ftw/>

- <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/>

- Block and Inline Elements
- Width and Height
- Padding, Margin, and Border
- Box Sizing
- Semantic Properties
- Position Elements on a Coordinate System



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**

 **Flutter**TM
International

INDEAVR
Serving the high achievers



AMBITIONED

 **DRAFT
KINGS**



**SOFTWARE
GROUP**



BOSCH



Postbank

Решения за твоето утре

 **PHAR
VISION**



SmartIT

DXC
TECHNOLOGY

createX

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- 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

- Software University Forums

- forum.softuni.bg

