HTML and CSS

Box Model, Containers and Flexbox

CONTENT

PADDING

BORDER

MARGIN
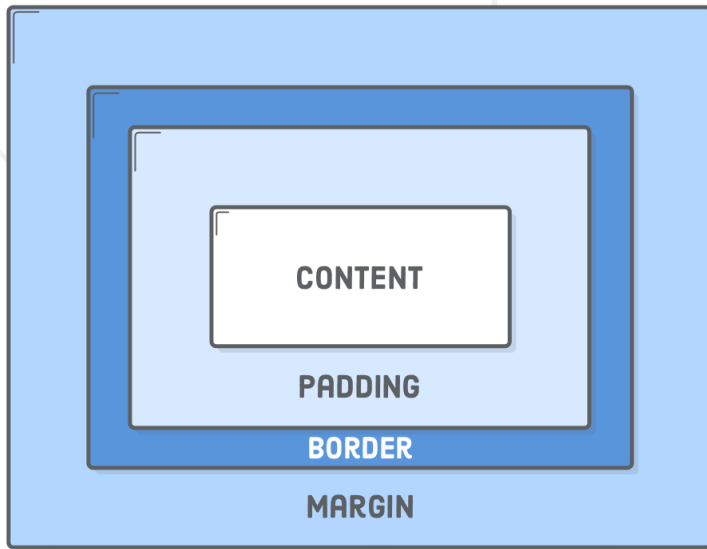
SoftUni Team

Technical Trainers

Software University

SoftUni

sli.do

#QA-FrontEnd

Table of Contents

CSS Box Model

The CSS Box Model

- Fundamental concept in web design and development

- Each **HTML element** is represented as a **rectangular box** by the rendering engine of the browser

- Allows controlling the **spacing** and **sizing** of elements
    - Crucial for **structure** and **consistency**

- Four elements
    - **Content**, **padding**, **border**, **margin**

Software University

CSS Box Model Elements

- **Content**

  - Text and / or media (e.g., images)

- **Padding**

  - Transparent area around the content

- **Border**

  - Wraps content and padding

- **Margin**

  - Transparent area outside the border

The CSS Box Model

- **margin**

  - Sets the margin area on all four sides of an element

  - Shorthand for **margin-top**, **margin-right**, **margin-bottom**, and **margin-left**

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

- **border**

  - Sets the element's border

  - Shorthand for **border-width**, **border-style** and **border-color**

```css
.box {
    border: 1px solid black;
}
```

- **padding**

  - Sets the margin area on all four sides of an element

  - Shorthand for **padding-top**, **padding-right**, **padding-bottom**, and **padding-left**

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

Block & Inline Elements

Block & Inline Elements

- In HTML & CSS, elements are categorized in **two main** types
  - **Block**
  - **Inline**
- They have differences in both **behavior** and **styling**
- Third type → **inline-block**
 - **Combines features** of block and inline elements

Block Elements

- Used for defining the web page's **structure**

- **Start** on a **new line**

- Occupy the **full available width**

- **Accept width** and **height** properties

- Can have **margins** and **paddings**

- Examples

  - `<p>, <div>, <h1>, <section>, …`

Inline Elements

- Used for **styling** segments or small components of a web page

- **Do not start** on a **new line**

- Occupy the **needed width** only

- **Do not accept width** and **height** properties

- Can have **horizontal margins** and **paddings**

- Examples

    - `<span>, <a>, <strong>, <img>, …`

Inline-block Elements

- Used for special cases

  - e.g., horizontally aligned menus

- **Do not start** on a **new line**

- Occupy the **needed width** only

- **Accept width** and **height** properties

- Can have **margins** and **paddings**

- **display**
  - Most important property for controlling an element's layout
  - Specifies how an element is shown on a web page
    - Sets an element's **inner** and **outer** display types
      - **Outer** type defines **element's position** in flow layout
      - **Inner** type defines the layout of the **element's children**

"BLOCK BOX"

"INLINE BOXES"

CSS Display Properties
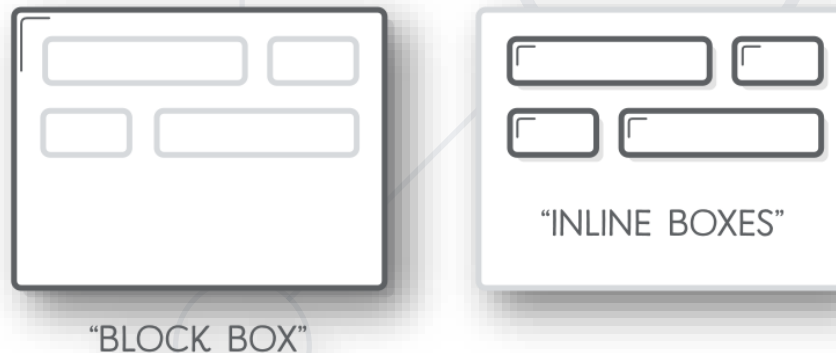
- **display: block;**
  - Displays an element as a **block** element
  - Starts on a **new** line
  - Takes the **whole** width
- **display: inline;**
  - Default value for the **display** property
  - Displays an element as an **inline** element
  - Starts on the **same** line
  - Takes only the **necessary** width

CSS Display Properties

- **`display: inline-block;`**

  - Treat **inline** elements as **block** elements

  - Starts on the **same** line

  - Takes the **specified width** and **height**

  - Respects **margins** and **paddings**

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

CSS Width & Height

CSS Width & Height

- **width**

  - Sets an element's width

- **height**

  - Sets an element's height

```css
.box {
  width: 150px;
  height: 250px;
}
```

CSS Width & Height Values

- **auto**
  - Default value, calculated by the browser
- **length**
  - Defines the value in one of the CSS Units
- **%**
  - Defines the value according to the containing block
- **initial**
  - Sets the width/height to its default value
- **inherit**
  - Inherited from the parent's value

CSS Width & Height Min Values

- Set the **minimum width** and **height** of an element

- **Prevent** the used **values** of the width and height properties from becoming **smaller** than the values specified

  - **min-width**

  - **min-heigth**

```css
.box {
  min-width: 150px;
  min-height: 200px;
}
```

CSS Width & Height Max Values

- Set the **maximum width** and **height** of an element

- **Prevent** the used **values** of the width and height properties from becoming **larger** than the values specified

  - `max-width`

  - `max-height`

```css
.box {
  max-width: 300px;
  max-height: 400px;
}
```
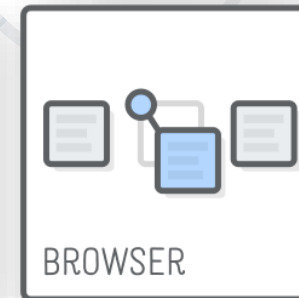
CSS Position

Position

- Specify the type of positioning method used for an element

- Four **major** types
    - Static
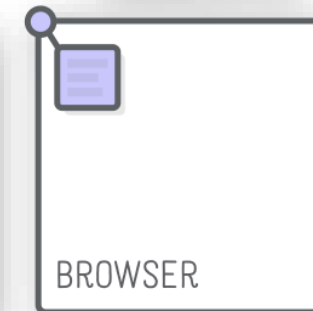   - Relative
    - Absolute
    - Fixed

STATIC

RELATIVE

ABSOLUTE

FIXED

BROWSER

BROWSER

BROWSER

BROWSER
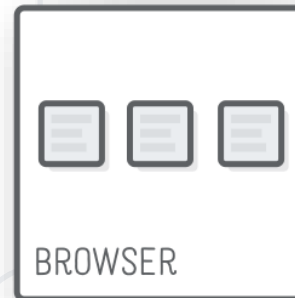
CSS Position

- **position: static;**

  - The default state of every element

  - Places the element into its normal position in the document layout flow

  - **Not affected** by **top**, **bottom**, **left** and **right** properties

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

CSS Position

- **position: relative;**

  - Similar to static positioning

  - Once the positioned element has taken its place in the normal layout flow, its final position can be modified using **top**, **bottom**, **left** and **right** properties

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

CSS Position

- **position: fixed;**

    - Position is relative to the viewport

    - Elements stays in the same place even if the page is scrolled

    - Affected by properties **top**, **bottom**, **left** and **right**

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

CSS Position

- **position: absolute;**

  - Element is positioned based on the two-dimensional coordinate system
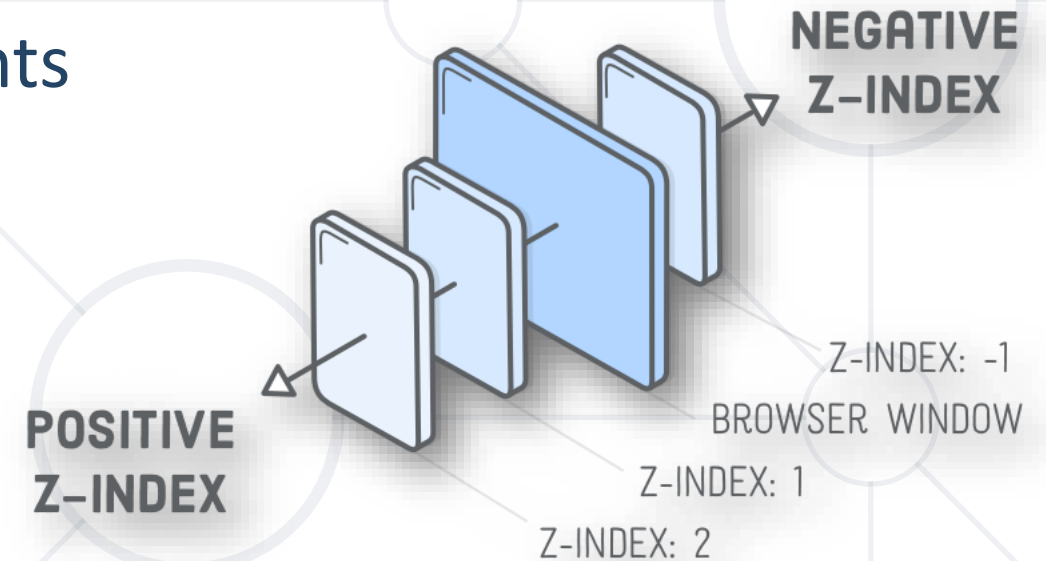
  - Relative to the nearest positioned ancestor

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

CSS Position

- **position: sticky;**
  - Based on the user's scrolling
  - Resembles **position: relative**
 - Once the given offset position is met, resembles **position: fixed**

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

CSS Position

- **z-index**

  - Specifies stack order of an element

  - Used for overlapping other elements

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


NEGATIVE
Z-INDEX

POSITIVE
Z-INDEX

Z-INDEX: -1
BROWSER WINDOW
Z-INDEX: 1
Z-INDEX: 2

CSS Positioning Properties

- Control placement of an element within its containing element

- Four positioning properties for direction

  - **top**

  - **bottom**

  - **left**

  - **right**

- Set the margin edge for the positioned box

- Combined with the different types of **position** property
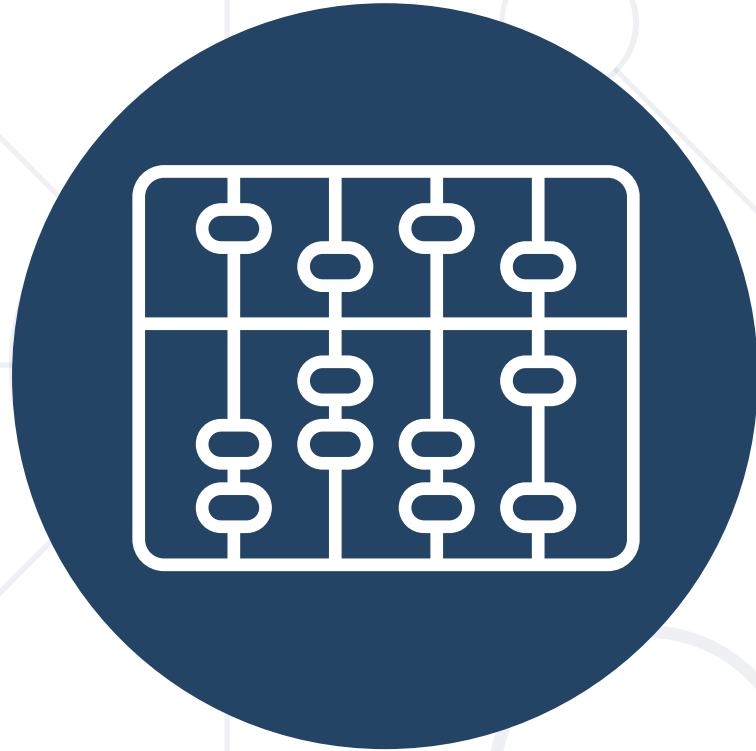
Flexbox

What is Flexbox?

- **Flexbox** == Flexible Box Layout Module

- Powerful layout method for arranging elements

  - Organize the elements in **rows** and **columns** within a container

  - Design flexible layout **without positioning**
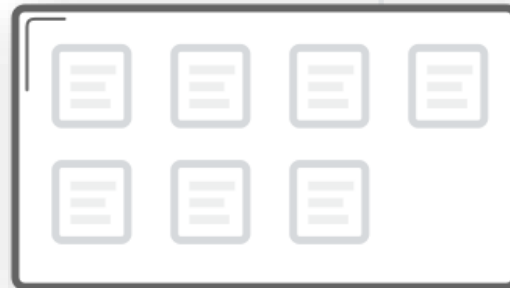
"FLEX CONTAINER"

"FLEX ITEMS"

The Flex Model

- **Flex container**

  - **Main** element that holds other elements

  - Defines the **layout** and **positioning** of all its **children**

  - Used for creating **distinct sections** in a webpage

- **Items**

  - **Direct child** element of a container element

  - Managed by the parent container element's properties
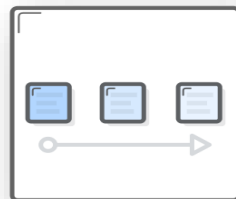
The Flex Model
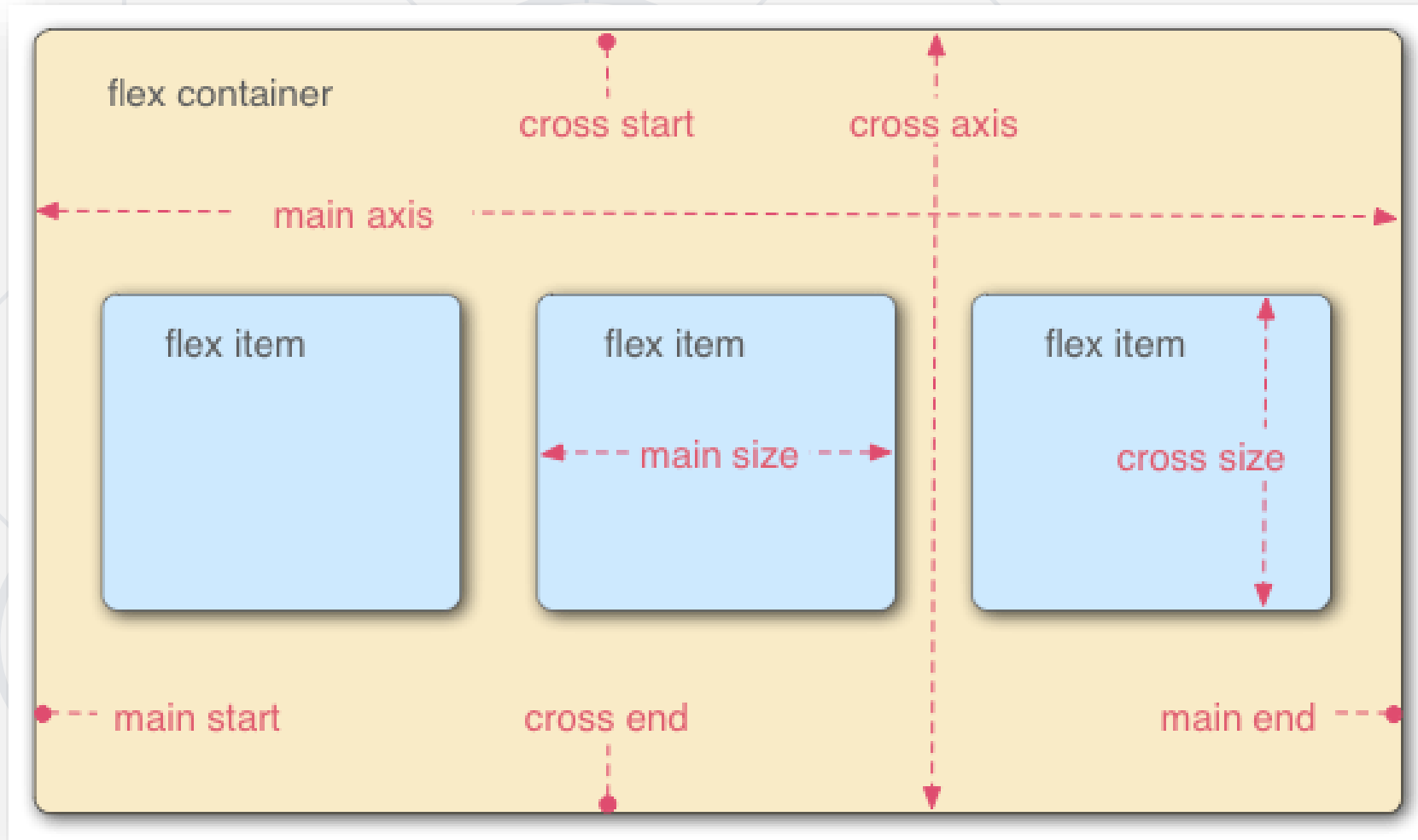
Container Properties

display: flex;

- Define the flex container

- Enable the flex context for the direct children of the container

```css
.box {
  display: flex;
}
```
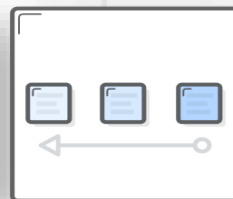
Container Properties

- **flex-direction**

  - Specify which direction the main axis runs (which direction the flexbox children are laid out in)
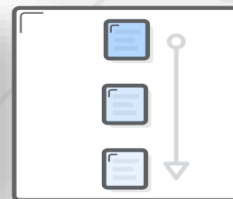
  - Default value → **row**

```css
.box {
    flex-direction: row | row-verse | column | column-verse;
}
```
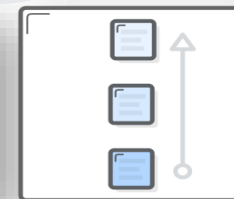

ROW   ROW–REVERSE   COLUMN   COLUMN–REVERSE

- **flex-wrap**

  - Control the children overflow of the container

```
.box {
    flex-wrap: wrap | nowrap | wrap-verse;
}
```


NO WRAPPING
FLEX-WRAP: NOWRAP;

WITH WRAPPING
FLEX-WRAP: WRAP;

Container Properties

- **justify-content**

  - Define the items alignment along the **main** axis

```css
.box {
  justify-content: flex-start | center | flex-end | space-around | space-between;
}
```


FLEX-START   CENTER   FLEX-END   SPACE-AROUND   SPACE-BETWEEN

Container Properties

- **align-items**
  - Define the items alignment along the **cross** axis

```
.box {
  align-items: stretch | flex-start | flex-end | center |
justify-content | align-items;
}
```
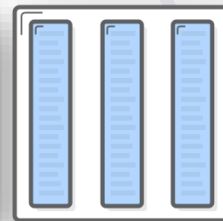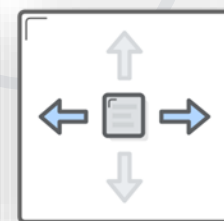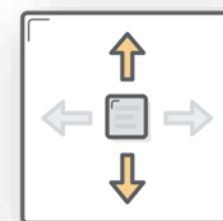

FLEX-START   CENTER   FLEX-END   STRETCH   JUSTIFY-CONTENT   ALIGN-ITEMS

Container Properties

- **align-content**

  - Define the container's lines alignment along the **cross** axis

```
.box {
  align-items: flex-start | flex-end | center | space-between
| space-around | stretch | start | end;
}
```

Container Properties

- **gap**

  - Sets the gaps (gutters) between rows and columns

```
.box {
  gap: 2em;
}
```

Items Properties

- **order**

  - Set the order to lay out an item in a flex container

    - Items in a container are sorted by ascending order value and then by their source code order

```css
.box {
    order: 1;
}
```


FLEX-DIRECTION
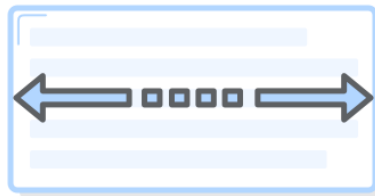(WHOLE CONTAINER)

ORDER
(INDIVIDUAL ITEMS)

Item Properties

- **flex-grow**

  - Define the ability to grow an item

```css
.box {
    flex-grow: 1;
}
```


FIXED
FLEX: INITIAL;

FLEXIBLE
FLEX: 1;

FIXED
FLEX: INITIAL;

Item Properties

- **flex-shrink**

  - Define the ability to shrink an item

    ```css
    .box {
        flex-shrink: 1;
    }
    ```

- **flex-basis**

  - Set the initial main size of a flex item

    ```css
    .box {
        flex-basis: 20em;
    }
    ```

Summary

- The **CSS Box** model is a fundamental concept in web development

- Using various properties, allows **controlling** the **spacing** and **sizing** of elements

- **Inline** & **Block** elements define the behavior and styling of HTML elements

- **Flexbox** == powerful layout method for arranging elements in a web page

    - Container

    - Items

Questions?
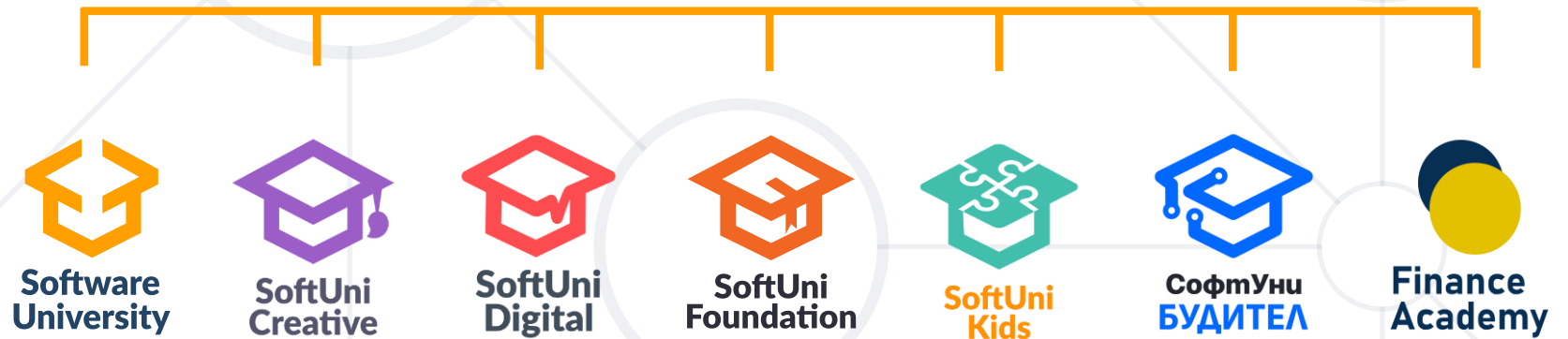
Trainings @ Software University (SoftUni)

- 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

License

- 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