

IF120203-Web Programming

# 05: Introduction to Bootstrap, Grid, and Flex

PRU/SPMI/FR-BM-18/0222

PRADITA UNIVERSITY



# Goals for Today

---

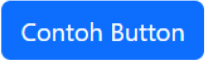

- To learn about the basic of Bootstrap
- To be able to understand how bootstrap as a CSS Framework works
- To understand how grid and flex works for web development
- To know when to apply flex or grid in any cases

# Introduction to Bootstrap

---

- Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.
- Bootstrap makes responsive web design a reality. It makes it possible for a web page or app to detect the visitor's screen size and orientation and automatically adapt the display accordingly.

# Example: Button - CSS Native



Contoh Button

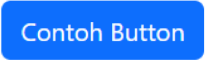

## HTML

```
<button class="contoh-button">Contoh Button</button>
```

## CSS

```
.contoh-button {  
  padding: 7px 12.5px;  
  background-color: #0d6efd;  
  border-radius: 6px;  
  border: none;  
  color: white;  
  transition: 0.2s all ease;  
}
```

# Example: Button - Bootstrap



Contoh Button

## HTML

```
<button class="btn btn-primary">Contoh Button</button>
```

## CSS

No CSS required to make this button with Bootstrap

P.S. unless you want to customize it

# Benefit of using Bootstrap

---

- A consistent framework that supports major of all browsers and CSS compatibility fixes
- Lightweight and customizable
- Responsive structures and styles
- Good documentation and community support

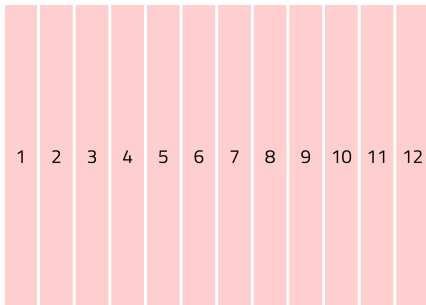
# Introduction to Grid

---

- Grid in CSS is a layout technique that divides a webpage into rows and columns, creating a structured grid structure.
- Grid in CSS is a powerful layout system that simplifies the way we arrange elements on a web page. When paired with frameworks like Bootstrap, it becomes even more accessible and efficient.
- Instead of relying solely on float-based layouts or complex positioning, Grid offers a straightforward and intuitive way to manage page structure.

# Introduction to Grid

- Grid in Bootstrap is a size setting that is displayed on the monitor. Grid Bootstrap functions to make settings for the width of each web component so that we can freely adjust the responsiveness of the website pages that we create with Bootstrap. Bootstrap has 12 grid classes.





# Grid Customization - 1

---

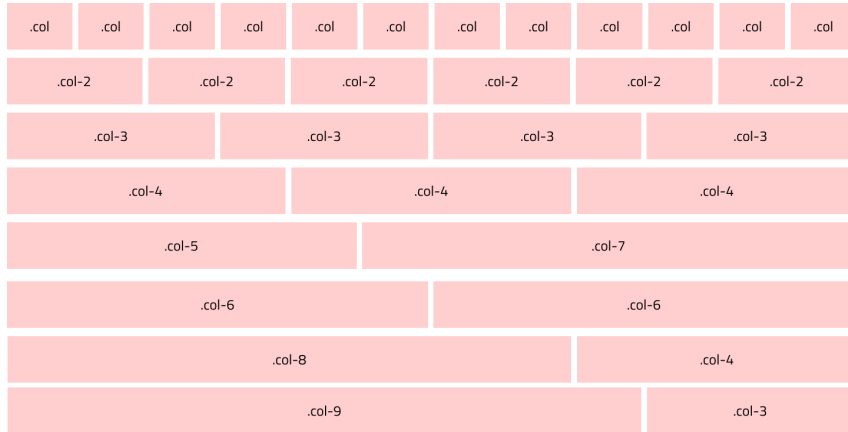
- Bootstrap's grid system is based on a 12-column layout, where the screen width is divided into 12 equal parts. This provides a convenient way to create responsive designs that adapt to different screen sizes. Each column is represented by a class, typically named col-\*, where the \* is a number from 1 to 12.
- When designing your layout, you can use these column classes to define the width of each column within a row. For example, col-6 means the column occupies half the width of the row, while col-3 means the column takes up one-fourth of the row's width.

# Grid Customization - 2

---

- If the total column count in a row exceeds 12, Bootstrap's grid system will automatically wrap the overflowing columns onto the next line. This ensures that your content remains visually coherent and doesn't break the layout. Here's how this works:
- 1. Overflowing Columns: Let's say you have a row with columns col-6, col-4, col-3, and col-2. The total column count is  $6 + 4 + 3 + 2 = 15$ , which exceeds 12.
- 2. Automatic Wrapping: In this scenario, Bootstrap will wrap the overflowing columns to the next line while maintaining the column order. So, the first row will contain columns col-6 and col-4, and the second row will have columns col-3 and col-2.

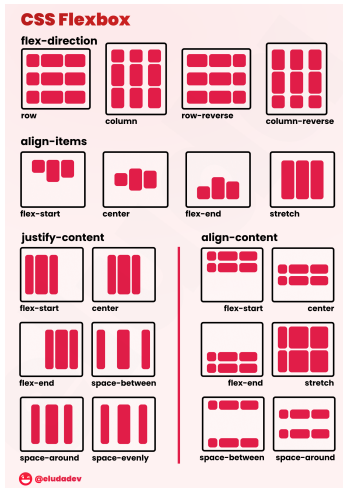
# Grid Visualization



# Introduction to Flex

## - 1

Flexbox, short for Flexible Box Layout, is a CSS layout model designed to provide an efficient way to arrange and distribute space among items in a container, even when their sizes are unknown or dynamic. Flexbox is particularly well-suited for creating complex layouts and aligning items within a container in a more predictable and consistent manner.



# Introduction of Flex - 2

---

- In the Flexbox model, you work with two main components: the flex container and the flex items.
- 1. Flex Container: The parent element that holds the flex items. To create a flex container, you apply the `display: flex;` or `display: inline-flex;` property to the container element. This establishes a new flex formatting context for the container and its direct children become flex items.
- 2. Flex Items: The child elements of the flex container. These items can be arranged horizontally (in a row) or vertically (in a column). Flex items can grow, shrink, and be aligned based on various properties.

# Introduction of Flex - 3

---

- flex-direction: This property defines the main axis along which the flex items will be placed. The values can be row (default), row-reverse, column, or column-reverse.
- justify-content: This property aligns flex items along the main axis. When the flex-direction is set to row, it controls horizontal alignment, and when set to column, it controls vertical alignment. Common values include flex-start, flex-end, center, space-between, and space-around.
- align-items: This property aligns flex items along the cross axis (the axis perpendicular to the main axis). When the flex-direction is row, it controls vertical alignment, and when set to column, it controls horizontal alignment. Common values include flex-start, flex-end, center, baseline, and stretch.

# Introduction of Flex - 4

---

- When you change the flex-direction to column, the behavior of justify-content and align-items switches because the main axis and cross axis orientations change:
- With flex-direction: column, the main axis becomes vertical and the cross axis becomes horizontal.
- justify-content now controls vertical alignment, similar to how align-items worked in the row direction. It aligns flex items vertically within the column.
- align-items now controls horizontal alignment, similar to how justify-content worked in the row direction. It aligns flex items horizontally within the column.

# JQuery Reverse Button

---

```
$(document).ready(function () {  
    $("#reverse").click(function () {  
        const fromSelect = $("#from");  
        const toSelect = $("#to");  
        const temp = fromSelect.val();  
        fromSelect.val(toSelect.val());  
        toSelect.val(temp);  
    });  
    // Next is Convert Button
```



# JQuery Convert Button - Part 1

---

```
$("#convert").click(async function () {  
  const fromSelect = $("#from");  
  const toSelect = $("#to");  
  const apiKey = "  
    fca_live_iq1WvbaON67X9adHTaJiqEszDhP6jtDz7louUbWv";  
  const fromCurrency = fromSelect.val();  
  const toCurrency = toSelect.val();
```

# JQuery Convert Button - Part 2

```
const encodedFromCurrency = encodeURIComponent(fromCurrency)
;
const encodedToCurrency = encodeURIComponent(toCurrency)
;

const url = `https://api.freecurrencyapi.com/v1/latest?
  apikey=${apiKey}&currencies=${encodedToCurrency}&
  base_currency=${encodedFromCurrency}`;
const options = {
  method: "GET",
};
```

# JQuery Convert Button - Part 3

```
try {  
    const response = await fetch(url, options);  
    const data = await response.json(); // Parse the  
        response body as JSON  
    // console.log(data); // For debugging purposes  
    const rate = data["data"][toSelect.val()]; // Get the  
        rate value for the selected currency  
    const amount = $("#amount").val();  
    const converted = (amount * rate).toLocaleString(  
        undefined, {  
            minimumFractionDigits: 2,  
            maximumFractionDigits: 2,  
        });  
}
```

# JQuery Convert Button - Part 4

```
let fromCurrencySymbol;  
    if (fromSelect.val() === "IDR") {  
        fromCurrencySymbol = "Rp. ";  
    } else if (fromSelect.val() === "USD") {  
        fromCurrencySymbol = "$ ";  
    } else if (fromSelect.val() === "EUR") {  
        fromCurrencySymbol = "€ ";  
    } else if (fromSelect.val() === "GBP") {  
        fromCurrencySymbol = "£ ";  
    }  
}
```

# JQuery Convert Button - Part 5

```
let toCurrencySymbol;  
    if (toSelect.val() === "IDR") {  
        toCurrencySymbol = "Rp. ";  
    } else if (toSelect.val() === "USD") {  
        toCurrencySymbol = "$ ";  
    } else if (toSelect.val() === "EUR") {  
        toCurrencySymbol = "€ ";  
    } else if (toSelect.val() === "GBP") {  
        toCurrencySymbol = "£ ";  
    }  
}
```

# JQuery Convert Button - Part 6

---

```
$("#from-result").text(fromCurrencySymbol + amount);  
$("#from-currency").text(fromSelect.val());  
$("#to-result").text(toCurrencySymbol + converted);  
$("#to-currency").text(toSelect.val());  
const exchangeRateLabel = $("#exchange-rate");  
const formattedRate = parseFloat(rate).toFixed(2);  
const exchangeRateText = '1 ${fromSelect.val()} = ${  
    formattedRate} ${toSelect.val()}';
```

# JQuery Convert Button - Part 7

---

```
        exchangeRateLabel.text(exchangeRateText);  
    } catch (error) {  
        console.error(error);  
    }  
});  
});
```

**Thank You**

