

## Agenda

- BootStrap
- Breakpoints
- Layout
- Grid
- Components
- Javascript Introduction
- Builtin Objects
- Language Fundamentals
- Functions

## BootStrap

- It is the CSS Framework for developing responsive and mobile-first websites.
- Bootstrap 5 is the newest version of Bootstrap

## BootStrap Breakpoints

- Breakpoints are customizable widths that determine how your responsive layout behaves across device.
- They are the building blocks of responsive design.
- Use media queries to architect your CSS by breakpoint.
- Media queries are a feature of CSS that allow you to conditionally apply styles based on a set of browser and operating system parameters.
- We most commonly use min-width in our media queries.

## Available breakpoints

- Bootstrap includes six default breakpoints, sometimes referred to as grid tiers, for building responsively.
1. Extra Small (None) - <576px (For mobile no prfefix)
  2. Small(sm) - ≥576px
  3. Medium(md) - ≥768px
  4. Large(lg) - ≥992px
  5. Extra large(xl) - ≥1200px
  6. Extra extra large(xxl) - ≥1400px

## Container

- Containers are the most basic layout element in Bootstrap and are required when using our default grid system.
- Containers are used to contain, pad, and (sometimes) center the content within them.
- Bootstrap comes with three different containers:
  1. .container, which sets a max-width at each responsive breakpoint
  2. .container-{breakpoint}, which is width: 100% until the specified breakpoint
  3. .container-fluid, which is width: 100% at all breakpoints
    - a full width container, spanning the entire width with some padding on right and left

## Grid System

- Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content.
1. grid supports six responsive breakpoints.
    - Breakpoints are based on min-width media queries, meaning they affect that breakpoint and all those above it (e.g., `.col-sm-4` applies to `sm`, `md`, `lg`, `xl`, and `xxl`).
    - This means you can control container and column sizing and behavior by each breakpoint.
  2. Containers center and horizontally pad your content.
  3. Rows are wrappers for columns.
    - Each column has horizontal padding (called a gutter) for controlling the space between them.
    - This padding is then counteracted on the rows with negative margins to ensure the content in your columns is visually aligned down the left side.
    - Rows also support modifier classes to uniformly apply column sizing and gutter classes to change the spacing of your content.
  4. Columns are incredibly flexible.
    - There are 12 template columns available per row, allowing you to create different combinations of elements that span any number of columns.
    - Column classes indicate the number of template columns to span (e.g., `col-4` spans four). widths are set in percentages so you always have the same relative sizing.
  5. Row columns
    - Use the responsive `.row-cols-*` classes to quickly set the number of columns that best render your content and layout.
    - Whereas normal `.col-*` classes apply to the individual columns (e.g., `.col-md-4`), the row columns classes are set on the parent `.row` as a shortcut. With `.row-cols-auto` you can give the columns their natural width.

## Gutters

- These are the padding between your columns, used to responsively space and align content in the Bootstrap grid system.
- Gutters are also responsive and customizable. Gutter classes are available across all breakpoints, with all the same sizes as our margin and padding spacing.
- Change horizontal gutters with `.gx-*` classes, vertical gutters with `.gy-`, or all gutters with `.g-` classes.
- `.g-0` is also available to remove gutters.
- `gx-1`(horizontal padding of 2px) and `gy-1` (vertical margin of 4px)
- `gx-2`(horizontal padding of 4px) and `gy-2` (vertical margin of 8px)
- `gx-3`(horizontal padding of 8px) and `gy-3` (vertical margin of 16px)
- `gx-4`(horizontal padding of 12px) and `gy-4` (vertical margin of 24px)
- `gx-5`(horizontal padding of 24px) and `gy-5` (vertical margin of 48px)

## Margin and Padding

- Assign responsive-friendly margin or padding values to an element or a subset of its sides with shorthand classes.
- Includes support for individual properties, all properties, and vertical and horizontal properties.
- Spacing utilities that apply to all breakpoints, from xs to xxl, have no breakpoint abbreviation in them.
- This is because those classes are applied from min-width: 0 and up, and thus are not bound by a media query. The remaining breakpoints, however, do include a breakpoint abbreviation.
- The classes are named using the format {property}{sides}-{size} for xs and {property}{sides}-{breakpoint}-{size} for sm, md, lg, xl, and xxl.
- Where property is one of:
  - m - for classes that set margin
  - p - for classes that set padding
- Where sides is one of:
  - t - for classes that set margin-top or padding-top
  - b - for classes that set margin-bottom or padding-bottom
  - s - (start) for classes that set margin-left or padding-left in LTR, margin-right or padding-right in RTL
  - e - (end) for classes that set margin-right or padding-right in LTR, margin-left or padding-left in RTL
  - x - for classes that set both \*-left and \*-right
  - y - for classes that set both \*-top and \*-bottom
  - blank - for classes that set a margin or padding on all 4 sides of the element
- Where size is one of:
  - 0 - for classes that eliminate the margin or padding by setting it to 0
  - 1 - (by default) for classes that set the margin or padding to \$spacer \* .25
  - 2 - (by default) for classes that set the margin or padding to \$spacer \* .5
  - 3 - (by default) for classes that set the margin or padding to \$spacer
  - 4 - (by default) for classes that set the margin or padding to \$spacer \* 1.5
  - 5 - (by default) for classes that set the margin or padding to \$spacer \* 3
  - auto - for classes that set the margin to auto
- spacer is calculated in terms of rem (root em) an unit of measurement in css.
- 1 rem is 8px for padding and 16px for margin

## Typography

- arranging text to make it readable, clear, and visually appealing
- we can change the text font, style, color of the text
- To use heading font in the paragraph or div use class h1..h6
- Traditional heading elements are designed to work best in the meat of your page content.
- When you need a heading to stand out, we can use a display heading display1..display6

- All the text related font, style and color information is available under utilities section in the documentation

## Flex

- It manages the layout, alignment, and sizing of grid columns, navigation, components, and more with a full suite of responsive flexbox utilities.
- Apply display utilities to create a flexbox container and transform direct children elements into flex items.
- Flex containers and items are able to be modified further with additional flex properties.
- Responsive variations for flex are .d-flex and .d-inline-flex.
- Set the direction of flex items in a flex container with direction utilities.
- Use .flex-row to set a horizontal direction (the browser default), or .flex-row-reverse to start the horizontal direction from the opposite side.
- Use .flex-column to set a vertical direction, or .flex-column-reverse to start the vertical direction from the opposite side.
- Use justify-content utilities on flexbox containers to change the alignment of flex items on the main axis (the x-axis to start, y-axis if flex-direction: column). Choose from start (browser default), end, center, between, around, or evenly.
- Use align-items utilities on flexbox containers to change the alignment of flex items on the cross axis (the y-axis to start, x-axis if flex-direction: column). Choose from start, end, center, baseline, or stretch (browser default).

## JavaScript

- Invented by Brendan Eich in 1995 at Netscape Corporation for Netscape2
- It is a Scripting language for web development
- It is an interpreted language
- Used to develop server side programs (Node) as well
- It is an Object Oriented Programming Language
- Use script tag to write JS code in html page
- Loosely typed language
- It is used to dynamically modify the html pages. It has full integration with HTML/CSS
- All major browsers support and by default enabled for javascript
- ECMAScript is the official name of the language.
- ECMAScript versions have been abbreviated to ES1, ES2, ES3, ES5, and ES6.
- Since 2016, versions are named by year (ECMAScript 2016, 2017, 2018, 2019, 2020).

## Types of javascript

### 1. Internal

- It is inserted into the documents by using the script tag
- script tag provides a block to write the java script programs

```
<script>
  JS code goes here
</script>
```

## 2. External

- To use the predefined programs of any javascript library.

```
<script src = "myscript.js"></script>
```

## Built-in Objects

### 1. console

- represents the web console (terminal)
- use log method to write output on console

### 2. window

- used to display alert, prompt or confirmation

### 3. document:

- represents DOM (Document Object Model)
- Collection of objects of all the elements present inside the page

## Variables

- It is a container to store the data
- To declare a variable data type MUST NOT be used in its declaration
- To create variables in JS we can use

### 1. var

- It is a normal variable whose value can be changed multiple times
- var is not used as it is deprecated, it is recommended to use let instead

### 2. let

- It is a normal variable whose value can be changed multiple times

### 3. const

- It is a variable whose value cannot be changed once initialized.

- Syntax: let name = initial value; const name = initial value; E.g.

```
let num = 100; // number
const salary = 4.5; // number
let = "test"; // string
const firstName = 'steve'; // string
let canVote = true; // Boolean
```

## Variable Scope

### 1. global

- a variable declared outside any function
- can be declared with or without var keyword
- can be accessed outside or inside any function

- E.g.

```
num = 100;  
var salary = 4.5;
```

- can be declared inside a function without using a var keyword
- E.g.

```
function function1() {  
  // global  
  firstName = "test";  
}
```

## 2. Local

- Must be declared inside a function with keyword var
- Can NOT be accessed outside the function in which it is declared
- E.g.

```
function function1() {  
  // local  
  var firstName = "test";  
}
```

## Data Types

- In JS, all Data Types are inferred (automatically decided by JS .....
- Types

### 1. number:

- It supports both whole and decimal numbers
- E.g.
  - num = 100;
  - salary = 4.5;

### 2. string: collection of characters E.g. - firstName = "steve"; - lastName = 'Jobs';

### 3. boolean:

- may have only true or false value
- E.g.
  - canVote = true;
  - canVote = false;

### 4. undefined:

### 5. object:

## Built-in Values

### 1. NaN

- Not a Number
- Is of type number
- E.g. `console.log(parseInt("test"));`

### 2. Infinity:

- When a number is divided by 0
- E.g. `answer = 10 / 0; // Infinity`

### 3. undefined:

## Function

- Function must be declared along with its definition
- To declare a function, function keyword is used
- Syntax: `function () { // function body }`

## Function Alias

- Another way/name to call a function
- Syntax: `var = ;`
- E.g.

```
function function1() {  
  console.log("inside function1");  
}  
  
// function alias  
var myFunction1 = function1;  
myFunction1();
```

## Properties of function

- A function CAN NOT decide the data type of parameters
- Only caller decides the data type of parameters
- E.g.

```
function function1 (p1) {  
  console.log("p1 = " + p1 + " type = " + typeof(p1));  
}  
  
function1(10); // number  
function1("test"); // string  
function1(true); // boolean
```

- If a program contains multiple functions with same name then only bottom-most function's definition will be used
- E.g.

```
function function1 () {  
  console.log("function1 - 1");  
}  
  
function function1 () {  
  console.log("function1 - 2");  
}  
  
function function1 () {  
  console.log("function1 - 3");  
}  
  
// output: function1 - 3  
function1();
```

- A function can be called with excess number of parameters
- Excess parameters can be used by using a hidden parameter arguments
- E.g.

```
function function1 (p1, p2) {  
  // body  
}  
function1(10, 20, 30, 40, 50); // p1 = 10, p2 = 20
```

- A function can be called with less number of parameters
- E.g.

```
function function1 (p1, p2) {  
  // body  
}  
function1(); // p1 = undefined, p2 = undefined  
function1(10); // p1 = 10, p2 = undefined  
function1(10, 20); // p1 = 10, p2 = 20
```

- A function can be called before its declaration
- E.g.

```
function1();  
function function1() {  
  // body  
}
```