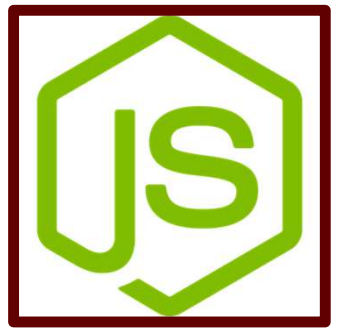
# 22ITC08

# Enterprise Application Development

## UNIT – II

## JavaScript (JS)



# JavaScript

**JavaScript**

**(often shortened to JS)**

**lightweight,**

**interpreted,**

**object-oriented language.**

# JavaScript

- JavaScript, which was originally developed at **Netscape** by **Brendan Eich**, was initially named **Mocha** but soon after was renamed **LiveScript**.
- In late **1995 LiveScript** became a joint venture of Netscape and Sun Microsystems, and its name again was changed, this time to **JavaScript**.
- A **language standard for JavaScript** was developed in the late 1990s by the **European Computer Manufacturers Association (ECMA)** as ECMA-262.
- The **ECMA-262 standard** is now in version 12.

# JavaScript Vs. JScript

- **JavaScript:** It is a scripting language whose trademark is **Oracle** Corporation.
- **JScript:** It is also a scripting language but owned by **Microsoft.**
- **JavaScript:** One need to handle **multiple browser** compatibility by writing code
- **JScript:** It's only support Microsoft **Internet Explorer**.

# JavaScript Comments

**Single-line Comment**

```
<script>
// It is single line comment
document.write("Hello Javascript");
</script>
```

**Multi-line Comment**

```
<script>
/* It is multi line comment. It will not be displayed */
document.write("Javascript multiline comment");
</script>
```

# Why JavaScript?

- JavaScript is one of the **3 languages** all web developers **must** learn:

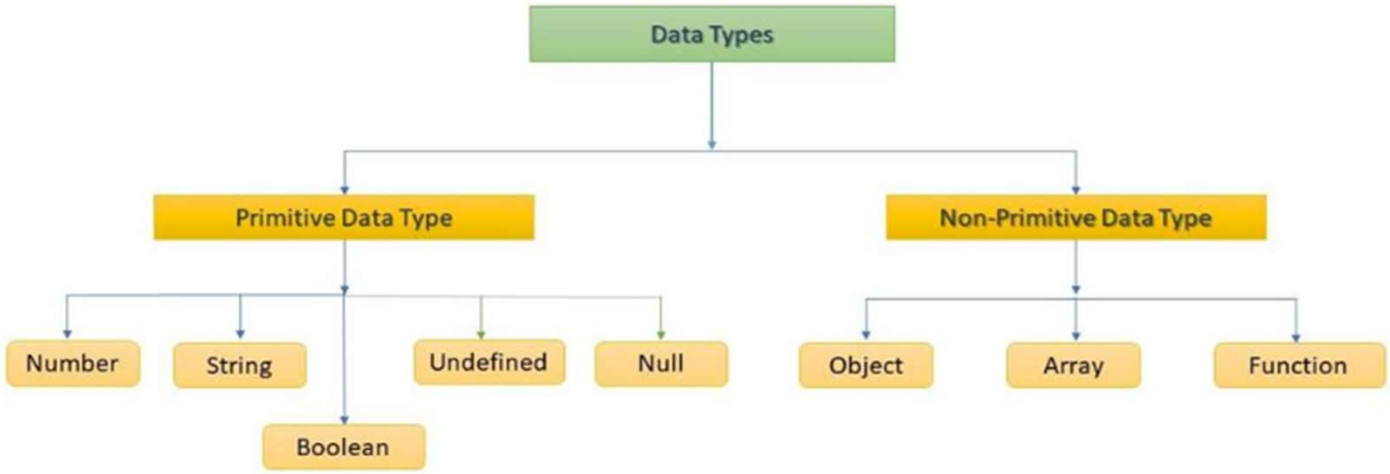
1. **HTML** to define the content of web pages.

2. **CSS** to specify the layout of web pages.

3. **JavaScript** to program the behavior of web pages.

# JavaScript Data Types

- The data type is a basic type of data that can be used in a program.
- In JavaScript, there is no need to specify the **type of variable** because it is **dynamically used by JavaScript Engine**.



# JavaScript Data Types

| Data Types | Description | Example |
|---|---|---|
| String | Represents Textual Data | 'hello', "hello world!" etc |
| Number | An Integer or a Floating-Point Number | 3, 3.234, 3e-2 etc. |
| Boolean | True or False | true and false |
| undefined | A Data type whose variable is not initialized | let a; |
| null | Denotes a Null Value | let a = null; |
| Object | Key-Value Pairs of Data | let student = { }; |

# JavaScript Data Types

```
let length = 16;                                    // Number

let lastName = "Johnson";                           // String

let x = {firstName:"John", lastName:"Doe"};         // Object

const dataChecked = true;                           //Boolean

let name;                                           //undefined

const number = null;                                //null

let person = { firstName: 'John', lastName: 'Doe' }; // Object
```

# JavaScript Data Types

- JavaScript evaluates expressions from **left to right**.
- **Different sequences** can produce **different results.**

let x = 16 + "Volvo"; → **16Volvo**

let x = 16 + 4 + "Volvo"; → **20Volvo**

let x = "Volvo" + 16 + 4; → **Volvo164**

const number1 = 3/0; → **+Infinity**

const number2 = -3/0; → **-Infinity**

# JavaScript Data Types

**To find the type of a variable, you can use the typeof operator.**

```
a=10;
b=3.14;
C="CBIT";
var d=true;
var marks=[90,80,70,98]
var student={rollno:01,name:"Abhilan"}
document.write("<h1>AFTER INITIALIZATION</h1>")
document.write("<br>Datatype of a is ",typeof a)
document.write("<br>Datatype of b is ",typeof b)
document.write("<br>Datatype of c is",typeof c)
document.write("<br>Datatype of d is ",typeof d)
document.write("<br>Datatype of marks is ",typeof marks)
document.write("<br>Datatype of student is ",typeof student)
document.write("<br>Datatype of Null is ",typeof null)
```

# JavaScript Type Conversion

- In programming, type conversion is the process of converting data of one type to another. For example: converting String data to Number.
- There are two types of type conversion in JavaScript.
- **Implicit Conversion - automatic type conversion.**
- **Explicit Conversion - manual type conversion.**

# JavaScript Type Conversion

- JavaScript automatically converts one data type to another (to the right type). This is known as implicit conversion.

## Implicit Conversion to String

```
// numeric string used with + gives string type
let result;

result = '3' + 2;
console.log(result) // "32"

result = '3' + true;
console.log(result); // "3true"

result = '3' + undefined;
console.log(result); // "3undefined"

result = '3' + null;
console.log(result); // "3null"
```

# Implicit Conversion to Number

```
// numeric string used with - , / , * results number type

let result;

result = '4' - '2';
console.log(result); // 2

result = '4' - 2;
console.log(result); // 2

result = '4' * 2;
console.log(result); // 8

result = '4' / 2;
console.log(result); // 2
```

# Non-numeric String Results to NaN

```
// non-numeric string used with - , / , * results to NaN

let result;

result = 'hello' - 'world';
console.log(result); // NaN

result = '4' - 'hello';
console.log(result); // NaN
```

# Implicit Boolean Conversion to Number

```
// if boolean is used, true is 1, false is 0

let result;

result = '4' - true;
console.log(result); // 3

result = 4 + true;
console.log(result); // 5

result = 4 + false;
console.log(result); // 4
```

# Null Conversion to Number

```
// null is 0 when used with number
let result;

result = 4 + null;
console.log(result);  // 4

result = 4 - null;
console.log(result);  // 4
```

# Undefined used with Number, Boolean or Null

```
// Arithmetic operation of undefined with number, boolean or null gives NaN

let result;

result = 4 + undefined;
console.log(result);  // NaN

result = 4 - undefined;
console.log(result);  // NaN

result = true + undefined;
console.log(result);  // NaN

result = null + undefined;
console.log(result);  // NaN
```

# JavaScript Explicit Conversion

```
let result;

// string to number
result = Number('324');
console.log(result); // 324

result = Number('324e-1')
console.log(result); // 32.4

// boolean to number
result = Number(true);
console.log(result); // 1

result = Number(false);
console.log(result); // 0
```

```
let result;
result = Number(null);
console.log(result);  // 0

let result = Number(' ')
console.log(result);  // 0
```

# Convert to String/Boolean Explicitly

```
//number to string
let result;
result = String(324);
console.log(result);  // "324"

result = String(2 + 4);
console.log(result); // "6"

//other data types to string
result = String(null);
console.log(result); // "null"

result = String(undefined);
console.log(result); // "undefined"

result = String(NaN);
console.log(result); // "NaN"

result = String(true);
console.log(result); // "true"

result = String(false);
console.log(result); // "false"
```

```
let result;
result = Boolean('');
console.log(result); // false

result = Boolean(0);
console.log(result); // false

result = Boolean(undefined);
console.log(result); // false

result = Boolean(null);
console.log(result); // false

result = Boolean(NaN);
console.log(result); // false
```

# JavaScript Type Conversion Table

| Value | String Conversion | Number Conversion | Boolean Conversion |
|---|---|---|---|
| 1 | "1" | 1 | `true` |
| 0 | "0" | 0 | `false` |
| "1" | "1" | 1 | `true` |
| "0" | `"0"` | 0 | `true` |
| "ten" | `"ten"` | `NaN` | `true` |
| `true` | "true" | 1 | `true` |
| `false` | "false" | 0 | `false` |
| `null` | "null" | 0 | `false` |
| `undefined` | "undefined" | `NaN` | `false` |

# JavaScript Display Possibilities

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using **document.write()**.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log().**

# JavaScript Display Possibilities

```
<!DOCTYPE html>
        <html>
                        <body>

                    <h2>My First Web Page</h2>
                    <p>My First Paragraph.</p>

                    <p id="demo"></p>

                                        <script>
                                        document.getElementById("demo").innerHTML = 5
+ 6;
                                        </script>

                        </body>
        </html>
```

# JavaScript Display Possibilities

```
<!DOCTYPE html>
        <html>
                        <body>

                        <h2>My First Web Page</h2>
                        <p>My first paragraph.</p>

                        <script>
                        document.write(5 + 6);
                        </script>

                        </body>
        </html>
```

# JavaScript Display Possibilities

```
<!DOCTYPE html>
        <html>
                        <body>

                        <h1>My First Web Page</h1>
                        <p>My first paragraph.</p>

                        <script>
                        window.alert(5 + 6);
                        </script>

            </body>
        </html>
```

# JavaScript Display Possibilities

```
<!DOCTYPE html>
        <html>
                        <body>


                    <script>
                    console.log(5 + 6);
                    </script>


                </body>
        </html>
```

# Methods of window object

| Method | Description |
|---|---|
| alert() | Displays the alert box containing message with ok button. |
| confirm() | Displays the confirm dialog box containing message with ok and cancel button. |
| prompt() | Displays a dialog box to get input from the user. |
| open() | Opens the new window. |
| close() | Closes the current window. |
| setTimeout() | Performs action after specified time like calling function, evaluating expressions etc. |

# JavaScript Object

- This code assigns a **simple value** to a **variable** named car:

**var car = "Fiat";**

- Objects are variables too. But objects can contain many values.
- JavaScript objects are containers for **named values** called properties or methods.

**var car = {type:"Fiat", model:"500", color:"white"};**

**var person = {firstName:"John", lastName:"Doe",**

**age:50}**

# JavaScript Arrays

- An array is a special variable, which can hold more than one value at a time.
- JavaScript arrays are used to store multiple values in a single variable.

**Creating an Array**

**var *array_name* = [*item1*, *item2*, ...];**

**var *array_name* = new Array(“item1”, ”item2”, ”item3");**

**var cars = ["Saab", "Volvo", "BMW"];**

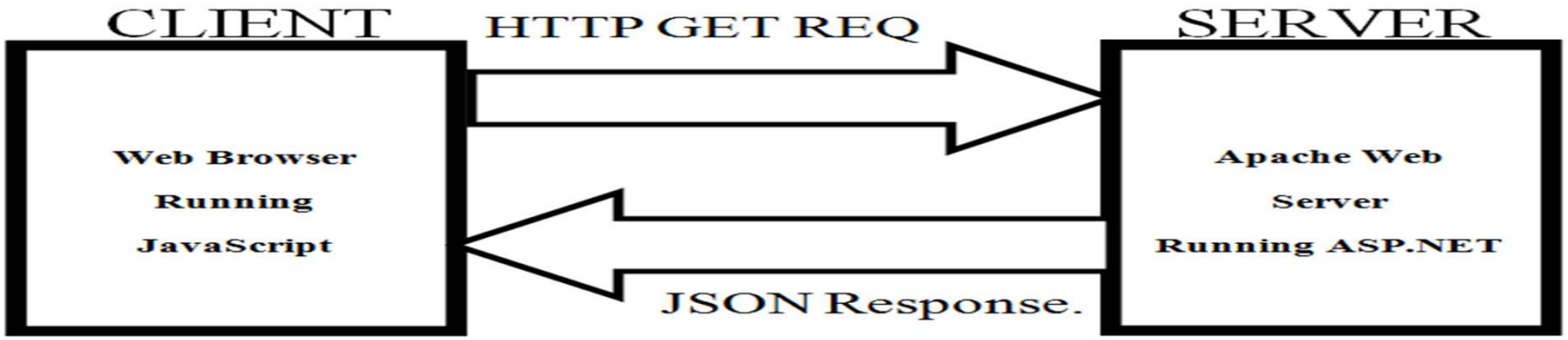
**var cars = new Array("Saab", "Volvo", "BMW");**

**Array indexes start with 0.**

- [0] is the first element. [1] is the second element.

# JSON

- JSON is a format for storing and transporting data.
- JSON is often used when data is sent from a server to a web page.
- JSON stands for **JavaScript Object Notation.**
- JSON is lightweight data interchange format.
- JSON is language independent.
- JSON is "self-describing" and easy to understand.



# JSON

- It is a **textual way to represent objects** by using two structures:

  **Collections of name-value pairs**

  **Arrays of values**.

- The primary reason to use JSON instead of XML is to eliminate the complexity of parsing.
- JSON is easy for people to read and write, and it is easy for machines to parse and generate.
- **JSON is a way to represent JavaScript objects as strings.**

# JavaScript Functions and Events

- **A JavaScript function is a block of JavaScript code, that can be executed when "called" for.**
- For example, a function can be called when an **event** occurs, like when the user clicks a button.
- Developers can use these events to execute JavaScript coded responses, which cause **buttons to close windows, messages to be displayed to users, data to be validated**, and virtually any other type of response imaginable.

# JSON SYNTAX RULES

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

```
{
"employees":
        [
                {"firstName":"John", "lastName":"Doe"},
                {"firstName":"Anna", "lastName":"Smith"},
                {"firstName":"Peter", "lastName":"Jones"}
        ]
}
```

# JSON.stringify()

```
<!DOCTYPE html>
	<html>
		<body>

			<h2>Create JSON string from a JavaScript object.</h2>

			<p id="demo"></p>

		<script>
			var obj = { name: "John", age: 30, city: "New York" };
			var myJSON = JSON.stringify(obj);
			document.getElementById("demo").innerHTML = myJSON;
		</script>

		</body>
	</html>
```

# Parse the data From SERVER

```
<!DOCTYPE html>
        <html>
                <body>

                        <h2>Create Object from JSON String</h2>

                        <p id="demo"></p>

                <script>
                var txt = '{"name":"John", "age":30, "city":"New York"}'
                var obj = JSON.parse(txt);
                document.getElementById("demo").innerHTML = obj.name + ", " +
obj.age;
                </script>

                </body>
</html>
```

# Storing Data

```
<!DOCTYPE html>
        <html>
                <body>
                <h2>Store and retreive data from local storage.</h2>
                <p id="demo"></p>
                <script>
                                var myObj, myJSON, text, obj;
                //Storing data:
                                myObj = { "name":"John", "age":31, "city":"New York" };
                                myJSON = JSON.stringify(myObj);
                                localStorage.setItem("testJSON", myJSON);
                //Retrieving data:
                                text = localStorage.getItem("testJSON");
                                obj = JSON.parse(text);
                                document.getElementById("demo").innerHTML = obj.name;
                </script>
                </body>
        </html>
```

# Differences between JSON and XML

| JSON | XML |
| --- | --- |
| JSON stands for JavaScript Object Notation. | XML stands for eXtensible Markup Language. |
| It is data-oriented. | XML is document-oriented. |
| JSON doesn't use end tag | XML have end tag. |
| JSON is shorter | XML is larger than JSON. |
| JSON is quicker to read and write | XML is not. |
| JSON can use arrays | XML do not have an array. |

# JavaScript Functions and Events

```
<!DOCTYPE html>
        <html>
                <head>
                        <script>
                                function myFunction()
                                {
         document.getElementById("demo").innerHTML = "Paragraph
changed.";
                                }                                </script>
                </head>
                <body>
        <h1>A Web Page</h1>
        <p id="demo">A Paragraph</p>
        <button type="button" onclick="myFunction()">Try it</button>
        </body>
        </html>
```

# JavaScript Functions and Events

**The change in the state of an object is known as an Event.**

- Mouse events
- Keyboard events
- Form events
- Window/Document events

# JS Mouse events

| Event Performed | Event Handler | Description |
|---|---|---|
| click | onclick | When mouse click on an element |
| mouseover | onmouseover | When the cursor of the mouse comes over the element |
| mouseout | onmouseout | When the cursor of the mouse leaves an element |
| mousedown | onmousedown | When the mouse button is pressed over the element |
| mouseup | onmouseup | When the mouse button is released over the element |
| mousemove | onmousemove | When the mouse movement takes place. |

# JS Keyboard/Form events

| Event Performed | Event Handler | Description |
|---|---|---|
| Keydown & Keyup | onkeydown & onkeyup | When the user press and then release the key |

| Event Performed | Event Handler | Description |
|---|---|---|
| focus | onfocus | When the user focuses on an element |
| submit | onsubmit | When the user submits the form |
| blur | onblur | When the focus is away from a form element |
| change | onchange | When the user modifies or changes the value of a form element |

# JS Window/Document Events

| Event Performed | Event Handler | Description |
|---|---|---|
| load | onload | When the browser finishes the loading of the page |
| unload | onunload | When the visitor leaves the current webpage, the browser unloads it |
| resize | onresize | When the visitor resizes the window of the browser |
| load | onload | When the browser finishes the loading of the page |

# JavaScript Events and Objects

| Event | Object |
|---|---|
| **onLoad** | **Body** |
| **onUnload** | **Body** |
| **onMouseOver** | **Link, Button** |
| **onMouseOut** | **Link, Button** |
| **onSubmit** | **Form** |
| **onClick** | **Button, Checkbox, Submit, Reset, Link** |

# JavaScript Click Event

```
<html>
        <head> Javascript Events </head>
                <body>
                <script language="Javascript" type="text/Javascript">
                 <!--
                 function clickevent()
                {
         document.write(“CBIT");
                }
                 //-->
                </script>
                <form>
                <input type="button" onclick="clickevent()" value="Who's this?"/>
                </form>
                </body>
</html>
```

# JavaScript Mouse Event

```
<html>
        <head>
                <h1> Javascript Events </h1>
        </head>
        <body>
                <script language="Javascript" type="text/Javascript">
                 <!--
                        function mouseoverevent()
                        {
                  alert("CBIT");
                        }
                 //-->
                </script>
                        <p onmouseover="mouseoverevent()"> Keep cursor over me</p>
        </body>
</html>
```

# JavaScript Focus Event

```
<html>
        <head> Javascript Events</head>
        <body>
        <h2> Enter something here</h2>
        <input type="text" id="input1" onfocus="focusevent()"/>
            <script>
            <!--
             function focusevent()
            {
          document.getElementById("input1").style.background=" CBIT";
            }
            //-->
            </script>
        </body>
</html>
```

# JavaScript Keydown Event

```
<html>
        <head> Javascript Events</head>
        <body>
            <h2> Enter something here</h2>
            <input type="text" id="input1" onkeydown="keydownevent()"/>
            <script>
            <!--
            function keydownevent()
            {
          document.getElementById("input1");
          alert("Pressed a key");
            }
            //-->
            </script>
        </body>
</html>
```

# JavaScript Load Event

```
<html>
        <head>Javascript Events</head>
        </br>
        <body onload="window.alert('Page successfully loaded');">
            <script>
    <!--
    document.write("The page is loaded successfully");
    //-->
    </script>
        </body>
</html>
```

# JavaScript Strings

- JavaScript strings are for storing and manipulating text.

```
<!DOCTYPE html>
  <html>
        <body>
              <h2>JavaScript Strings</h2>
              <p id="demo"></p>
        <script>
        let text = "John Doe";  // String written inside quotes
        document.getElementById("demo").innerHTML = text;
        </script>
        </body>
  </html>
```

# JavaScript Strings

| Method | Description |
|---|---|
| charAt(index) | returns the character at the specified index |
| concat() | joins two or more strings |
| replace() | replaces a string with another string |
| split() | converts the string to an array of strings |
| substr(start, length) | returns a part of a string |
| substring(start,end) | returns a part of a string |
| slice(start, end) | returns a part of a string |
| toLowerCase() | returns the passed string in lower case |
| toUpperCase() | returns the passed string in upper case |
| trim() | removes whitespace from the strings |
| includes() | searches for a string and returns a boolean value |
| search() | searches for a string and returns a position of a match |

# JavaScript Strings

```
const text1 = 'hello';
const text2 = 'world';
const text3 = '    JavaScript    ';

console.log(text1.concat(' ', text2));     // "hello world"
console.log(text1.toUpperCase());                  // HELLO
console.log(text3.trim());                                 // JavaScript
console.log(text1.split());                                // ["hello"]
console.log(text1.slice(1, 3));                    // "el"
console.log(text3.search("Script"));     //9
console.log(text1.charAt("2"));                     //l
console.log(text1.substr(0,3));                    //hel
console.log(text1.substring(0,4));                  //hell
```

# JS Date and Time

- In JavaScript, **date and time are represented by the Date object**.
- The Date object provides the **date and time information** and also provides **various methods**.

## Creating Date Objects

new Date()

new Date(milliseconds)

new Date(Date string)

new Date(year, month, day, hours, minutes, seconds, milliseconds)

# JS Date and Time

```
const timeNow = new Date();
console.log(timeNow);                          // shows current date and time

const time1 = new Date(0);
console.log(time1);                                    // Thu Jan 01 1970 05:30:00

const time2 = new Date(100000000000)
console.log(time2);                                    // Sat Mar 03 1973 15:16:40


const date1 = new Date("2020-07-01");
console.log(date1);                                    // Wed Jul 01 2020 05:45:00
GMT+0545
const date2 = new Date("2020-07-01T12:00:00Z");
console.log(date2);                                    // Wed Jul 01 2020 17:45:00
GMT+0545

const time3 = new Date(2020, 1, 20, 4, 12, 11, 0);
console.log(time3);                                    // Thu Feb 20 2020 04:12:11
```

# Object Oriented Programming (OOP) in JS

- As JavaScript is widely used in Web Development and some of the **Object Oriented** mechanisms supported by **JavaScript.**
- There are certain features or mechanisms which makes a Language Object-Oriented like:

1. **Object**
2. **Classes**
3. **Encapsulation**
4. **Inheritance**

# Object Oriented Programming (OOP) in JS

## Object

By using arrays we can store a group of individual objects and it is not possible to store key-value pairs.

If we want to represent a group of key-value pairs then we should go for Objects.

**Array: A group of individual objects**

**Object: A group of key-value pairs**

**JavaScript objects store information in the form of key-value pairs.**

An Object is an **instance** of a class.

# Object Oriented Programming (OOP) in JS

To create empty object:

var nums={}

OR

var nums=new Object()

1st Way:

2nd Way:

nums["fno"]=100

nums["sno"]=200

nums.fno=100

nums.sno=200

var variableName={ key1:value1,key2:value2,...};

var college={ name:‘CBIT', year: 1979, Dept:‘IT' };

| Arrays | Object |
| --- | --- |
| 1) Arrays can be used to represent individual values | 1) Objects can be used to represent key-value pairs |
| 2) Order will be maintained in Arrays | 2) Order concept not applicable for Objects |
| 3) By using index we can access and update data in arrays | 3) By using key we can access and update data in Objects |

**var College=[“CBIT",“MGIT",“VCE",“VNRVJIET"];**

**var college={ name:‘CBIT', year: 1979, Dept:‘IT' };**

# Object Oriented Programming (OOP) in JS

```
//Defining object
let person =
{
   first_name:'CBIT',
   last_name: 'MGIT',
            //method
            getFunction ()
            {
      return (`The name of the person is${person.first_name} ${person.last_name}`)
            },
   //object within object
                        phone_number :
                        {
                         mobile:'12345',
                         landline:'6789'
                         }
}
console.log(person.getFunction());
console.log(person.phone_number.landline);
```

# Object Oriented Programming (OOP) in JS

**Classes–:**

Classes are **blueprint** of an Object.

*JavaScript classes, introduced in ECMAScript 2015*

**JavaScript Class Syntax**

1. Use the **keyword class** to create a class.

**2. Always add a method named constructor():**

```
class ClassName
 {
  constructor() { ... }
}
```

# Object Oriented Programming (OOP) in JS

```
// Defining class using es6
class Vehicle
{
    constructor(name, maker, engine)
    {
        this.name = name;
        this.maker = maker;
        this.engine = engine;
    }
    getDetails()
    {
        return (`The name of the bike is ${this.name}.`)
    }
}
    // Making object with the help of the constructor
    let bike1 = new Vehicle('CBIT', 'Suzuki', '1340cc');
    let bike2 = new Vehicle('MGIT', 'Kawasaki', '998cc');

    console.log(bike1.name);
    console.log(bike2.maker);
    console.log(bike1.getDetails());
```

# Object Oriented Programming (OOP) in JS

- **Encapsulation:** The process of **wrapping properties and functions** within a **single unit** is known as encapsulation.

```
class person
{
   constructor(name,id)
   {
      this.name = name;
      this.id = id;
   }
   add_Address(add)
   {
      this.add = add;
   }
   getDetails()
   {
      console.log(`Name is ${this.name},Address is: ${this.add}`);
   }
}

let person1 = new person('CBIT',45);
person1.add_Address('Hyderabad');
person1.getDetails();
```

# Object Oriented Programming (OOP) in JS

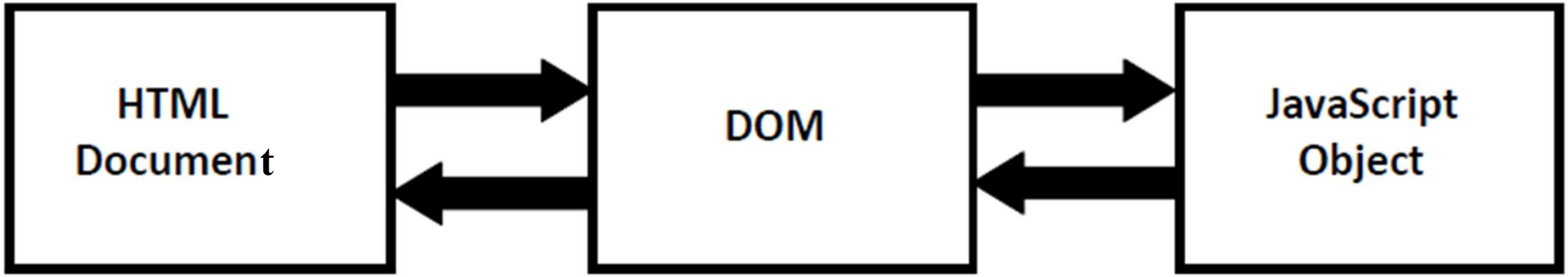
- **Inheritance** – It is a concept in which some properties and methods of an Object are being used by another Object.
- Unlike most of the OOP languages where classes inherit classes, JavaScript Objects inherit Objects i.e. certain features (property and methods) of one object can be reused by other Objects.

# Object Oriented Programming (OOP) in JS

```
//Inheritance example
class person
{
   constructor(name)
   {
      this.name = name;
   }
   //method to return the string
   toString(){      return (`Name of person: ${this.name}`); }
}
class student extends person
{
   constructor(name,id)
   {
      //super keyword for calling the above class constructor
      super(name);
      this.id = id;
   }
   toString()
   {
      return (`${super.toString()},Student ID: ${this.id}`);
   }
}
let student1 = new student('Abhilan',22);
console.log(student1.toString());
```

# Document Object Model (DOM)

- DOM acts as interface between JavaScript and HTML, CSS.
- Browser will construct DOM.
- All HTML tags will be stored as JavaScript objects.



# Document Object Model (DOM)

**To display Document to the console:**

Just type on JavaScript console: document

Then we will get total HTML Page/Document.

**To Display DOM Objects on the Console:**

console.dir(document)

Observe that Root element of DOM is document.

**DOM Attributes:**

1) document.URL :This is original URL of the website.

2) document.body :It returns everything inside body.

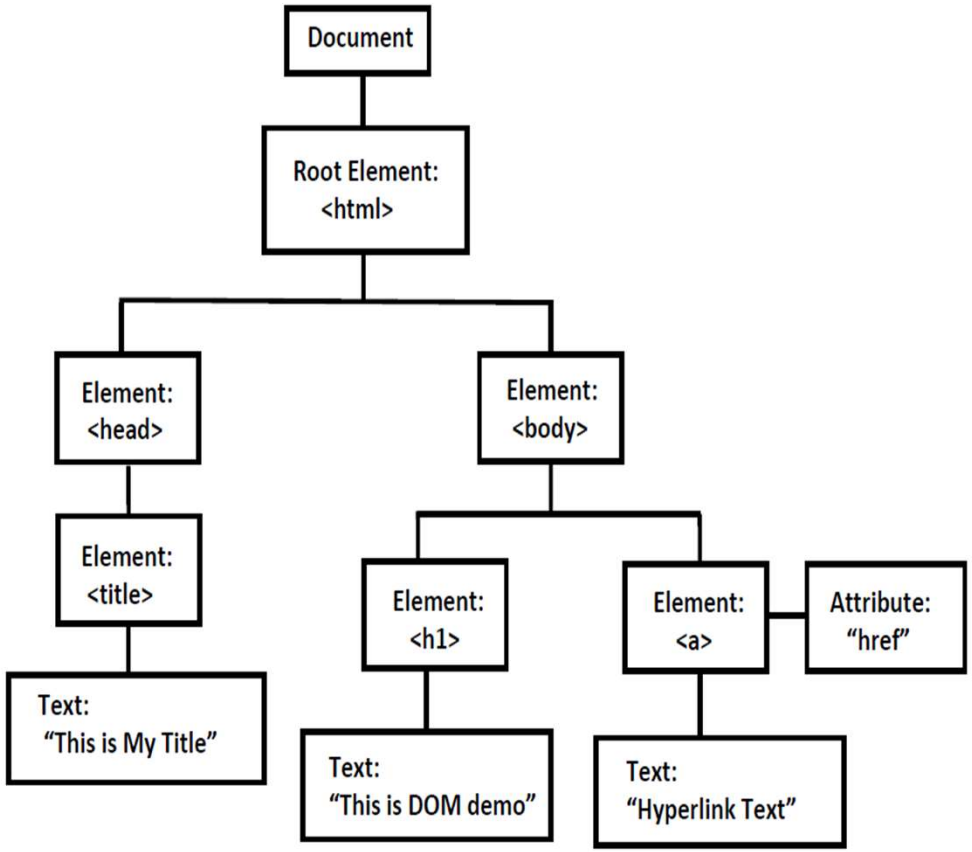
3) document.head :It returns head of the page.

4) document.links :It returns list of all links of the page.

# Document Object Model (DOM)

**https://software.hixie.ch/utilities/js/live-dom-viewer/**

```
<!DOCTYPE html>
    <html>
        <head>
            <title>This is My Titile</title>
        </head>
        <body>
            <h1>This is DOM Demo</h1>
            <a href="#">HeperlinkText</a>
        </body>
    </html>
```



# Document Object Model (DOM)

**How to grab HTML Elements from the DOM?**

1) **document.getElementById()**
   Returns element with the specified id.
2) **document.getElementsByClassName()**
   Returns list of all elements belongs to the specified class.
3) **document.getElementsByTagName()**
   Returns list of all elements with the specified tag.
4) **document.querySelector()**
   Returns the first object matching CSS style selector.
5) **document.querySelectorAll()**
   Returns all objects Matches the CSS Style Selector.

## JavaScript Regular Expressions

- In JavaScript, a **Reg**ular **Ex**pression (RegEx) is an object that describes a sequence of characters used for defining a search pattern.

**/^a...s$/**

- The pattern is: any five letter string starting with a and ending with s.

**https://regex101.com/**

**Create a RegEx**

- **Using a regular expression literal:**

```
cost regularExp = /abc/;
```

- **Using the RegExp() constructor function:**

```
const reguarExp = new RegExp('abc');
```

# JavaScript Regular Expressions

## Using String Methods

- In JavaScript, regular expressions are often used with the two string methods: **search() and replace().**

- The search() method uses an expression to search for a match, and returns the position of the match.

1. The search method returns the position in the String object at which the pattern matched.
2. **If there is no match, search returns –1.**
3. The position of the first character in the string is 0.

- The replace() method returns a modified string where the pattern is replaced.

# JavaScript Regular Expressions

```
var str = "Rabbits are furry";

var position = str.search(/bits/);

if (position >= 0)

document.write("'bits' appears in position", position,"<br />");

else

document.write("'bits' does not appear in str <br />");
```

**output:**

**'bits' appears in position 3**

# JavaScript Regular Expressions

```
<!DOCTYPE html>
<html>
    <body>
        <h2>JavaScript String Methods</h2>
        <p>Replace "Microsoft" with “Google" in the paragraph below:</p>
        <button onclick="myFunction()">Try it</button>
        <p id="demo">Please visit Microsoft!</p>
            <script>
            function myFunction()
            {
              let text = document.getElementById("demo").innerHTML;
              document.getElementById("demo").innerHTML =  text.replace("Microsoft",“Google");
            }
            </script>
    </body>
</html>
```

# JavaScript Regular Expressions

| Method | Description |
|---|---|
| exec() | Executes a search for a match in a string and returns an array of information. It returns null on a mismatch. |
| test() | Tests for a match in a string and returns true or false. |
| match() | Returns an array containing all the matches. It returns null on a mismatch. |
| matchAll() | Returns an iterator containing all of the matches. |
| search() | Tests for a match in a string and returns the index of the match. It returns -1 if the search fails. |
| replace() | Searches for a match in a string and replaces the matched substring with a replacement substring. |
| split() | Break a string into an array of substrings. |

# JavaScript Regular Expressions

```
const string = 'Find me';
const pattern = /me/;
// search if the pattern is in string variable
const result1 = string.search(pattern);
console.log(result1); // 5
// replace the character with another character
const string1 = 'Find me';
string1.replace(pattern, 'found you'); // Find found you
// splitting strings into array elements
const regex1 = /[\s,]+/;
const result2 = 'Hello world! '.split(regex1);
console.log(result2); // ['Hello', 'world!', '']
// searching the phone number pattern
const regex2 = /(\d{3})\D(\d{3})-(\d{4})/g;
const result3 = regex2.exec('My phone number is: 555 123-4567.');
console.log(result3); // ["555 123-4567", "555", "123", "4567"]
```

# JavaScript Regular Expressions

**MetaCharacters are characters that are interpreted in a special way by a RegEx engine.**

**[]** - **Square brackets** -Square brackets specify **a set of characters** you wish to match.

**.** - **Period**-A period matches any **single character** (except newline '\n').

^ - **Caret**-The caret symbol ^ is used to check if a string **starts with a certain character**.

**$** - **Dollar**-The dollar symbol $ is used to check if a string **ends with a certain character**.

* - **Star**-The star symbol * matches **zero or more occurrences** of the pattern left to it.

+ - **Plus**-The plus symbol + matches **one or more occurrences** of the pattern left to it.

**?** - **Question Mark** -The question mark symbol ? matches **zero or one occurrence**.

| - **Alternation-** Vertical bar | is used for alternation (**or operator**).

# JavaScript Regular Expressions

| Name | Equivalent Pattern | Matches |
|---|---|---|
| \d | [0-9] | A digit |
| \D | [^0-9] | Not a digit |
| \w | [A-Za-z_0-9] | A word character (alphanumeric) |
| \W | [^A-Za-z_0-9] | Not a word character |
| \s | [ \r\t\n\f] | A white-space character |
| \S | [^ \r\t\n\f] | Not a white-space character |

**/\d\.\d\d/ // Matches a digit, followed by a period, // followed by two digits.**

**/\D\d\D/ // Matches a single digit.**

**/\w\w\w/ // Matches three adjacent word characters**

# 22ITC08

# EAD

# UNIT-II

# BOOTSTRAP

# Introduction of Bootstrap

- **Powerful, extensible, and feature-packed frontend toolkit.**
- Bootstrap is the most commonly used **framework for Front-End** Development.
- Bootstrap providing several pre defined libraries for css and java script.
- **Current version of Bootstrap is: 5.2.x**

**https://getbootstrap.com/**

**https://www.w3schools.com/howto/**

# Bootstrap 4 VS Bootstrap 5

| BASIS OF | BOOTSTRAP 4 | BOOTSTRAP 5 |
|---|---|---|
| **Grid System** | It has 5 tier (xs, sm, md, lg, xl). | It has 6 tier (xs, sm, md, lg, xl, xxl). |
| **Color** | It has limited colors. | Extra colors added with the looks, A card improved color palette. |
| **Jquery** | It has jquery and all related plugins. | Jquery is removed and switched to vanilla JS with some working plugins |
| **Internet Explorer** | Bootstrap 4 supports both IE 10 and 11. | Bootstrap 5 doesn't support IE 10 and 11. |
| **Form elements** | Radio buttons, checkboxes have different look in different OS and browsers. | The look of form elements will not change, on different OS or browser. |
| **Bootstrap Icons** | Bootstrap 4 doesn't have its own SVG icons, we have to use font-awesome for icons. | Bootstrap 5 have its own SVG icons |
| **Jumbotron** | It supports. | It doesn't support jumbotron. |

# Connectivity of Bootstrap in page

We can **connect Bootstrap** with HTML by using the following **2 ways.**

- **By using CDN**
- **Locally**

**CDN [ Content Delivery Network ]**

Add the following in the <head> part of our html

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" >
```

# Content Delivery Network

```
<!DOCTYPE html>
    <html lang="en" dir="ltr">
        <head>
            <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

            <meta charset="utf-8">
            <title></title>
        </head>
        <body>
            <h1>This is First Bootstrap Demo</h1>
            </body>
    </html>
```

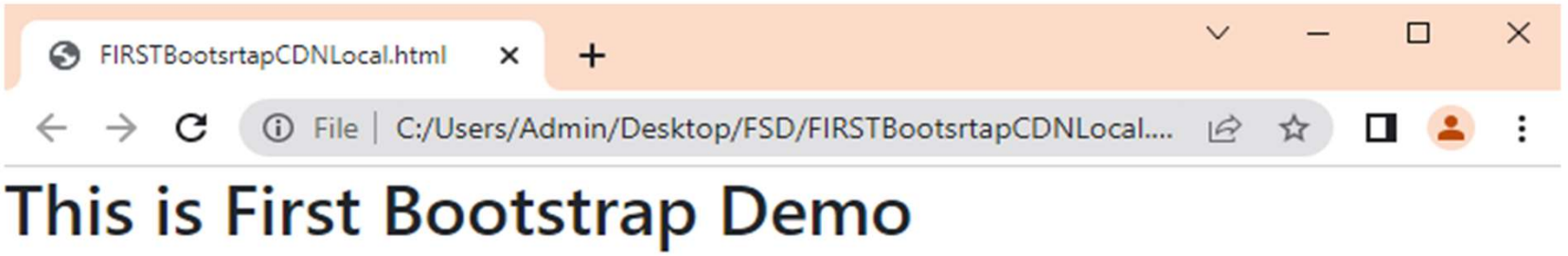
# Use Bootstrap Locally

- Download bootstrap.css file from the link getbootstrap.com .
- zip file contains bootstrap.css file.
- copy this file in our application folder and add the following link in html.

<link rel="stylesheet" href="bootstrap.css">

# Use Bootstrap Locally

```
<!DOCTYPE html>
    <html lang="en" dir="ltr">
        <head>
            <link href="bootstrap.min.css" rel="stylesheet">

            <meta charset="utf-8">
            <title></title>
        </head>
        <body>
            <h1>This is First Bootstrap Demo</h1>
            </body>
    </html>
```



# Container and Container-fluid

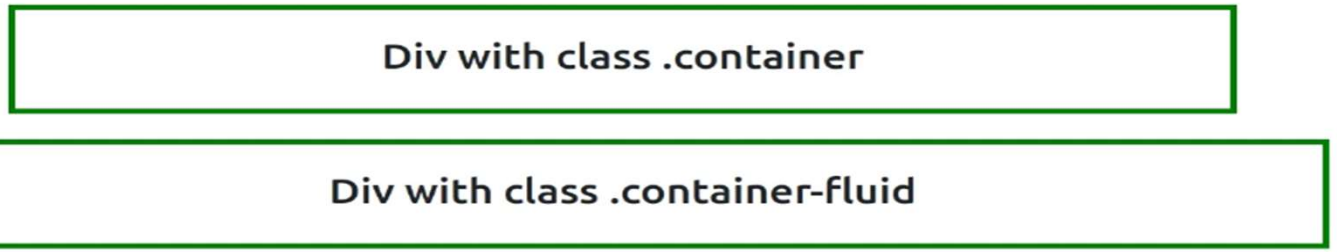
- In bootstrap, the **container** is used to set the content's margin.

**.container**

- The .container class provides a responsive fixed width container.

**.container-fluid**

- The .container-fluid class provides a full-width container which spans the entire width of the viewport.



# Container and Container-fluid

```
<!DOCTYPE html>
    <head>
        <title>Bootstrap columns</title>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
    <style>
        .container, .container-fluid
        {
        padding-top: 30px;
        padding-bottom: 30px;
        border: 6px solid green;
        margin-bottom: 30px;
        }
    </style>
    </head>
        <body>
         <div class="container">
            <h1 class="text-center">Div with class .container</h1>
        </div>
        <div class="container-fluid">
            <h1 class="text-center">Div with class .container-fluid</h1>
        </div>
        </body>
</html>
```

# Container and Container-fluid

```
.container-fluid
{
  width: 100%;
  padding-right: var(--bs-gutter-x, 0.75rem);
  padding-left: var(--bs-gutter-x, 0.75rem);
  margin-right: auto;
  margin-left: auto;
}
```

```
.container{
  width: 100%;
  padding-right: var(--bs-gutter-x, 0.75rem);
  padding-left: var(--bs-gutter-x, 0.75rem);
  margin-right: auto;
  margin-left: auto;
}
@media (min-width: 576px) {  .container {    max-width: 540px;  }}
@media (min-width: 768px) {  .container {    max-width: 720px;  }}
@media (min-width: 992px) {  .container {    max-width: 960px;  }}
@media (min-width: 1200px) {  .container {    max-width: 1140px;  }}
@media (min-width: 1400px) {  .container {    max-width: 1320px;  }}
```

# Container and Container-fluid

| Viewport width | .container width |
|---|---|
| Less than 576px | The container is set to 100% width |
| 576px to 767px | The container is set to 540px width |
| 768px to 991px | The container is set to 720px width |
| 992px to 1199px | The container is set to 960px width |
| 1200px to 1399px | The container is set to 1140px width |
| 1400px + | The container is set to 1320px width |

# Bootstrap Component - Button

- Most of the styles in Bootstrap are implemented as class styles.
- We can use these classes directly in our web application to improve look and feel.

**Commonly Used Classes :**

1) Container Class

2) Button Related Classes

3) Jumbotron Classes

# Bootstrap Component - Button

## Container Class:

**To center our elements we should use container class.**

```
<div class="container">
    <h1>This is First Bootstrap Demo</h1>
</div>
```

```
<div class="container">
  <h2>Button Styles</h2>
  <button type="button" class="btn">Basic</button>
  <button type="button" class="btn btn-default">Default</button>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
```

Button Styles

Basic Default Primary Success Info Warning Danger Link

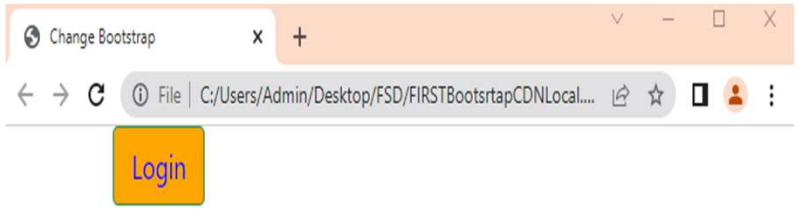
# Bootstrap Component - Button

**The button classes can be used on an <a>, <button>, or <input> element:**

```
<div class="container">
  <h2>Button Tags</h2>
  <a href="#" class="btn btn-success btn-lg" role="button">Link Button</a>
  <button type="button" class="btn btn-success btn-lg">Button</button>
  <button type="button" name="button" class="btn btn-success btn-lg active"> Login</button>
  <button type="button" name="button" class="btn btn-success btn-lg" disabled="disabled">Login</button>
  <a href="https://getbootstrap.com" class="btn btn-success btn-lg">Click Here for Bootstrap Documentation</a>
</div>
```

# How to Change Default Styles of Bootstrap

```
<html lang="en" dir="ltr">
    <head>
        <link rel="stylesheet" href="Bootstrap.css">
        <style type="text/css">
        .btn-success
        {
        background: orange;
        color: blue;
        }
        </style>
    </head>
     <body>
        <div class="container">
        <button type="button" name="button" class="btn btn-success btn-lg">Login</button>
        </div>
    </body>
 </html>
```



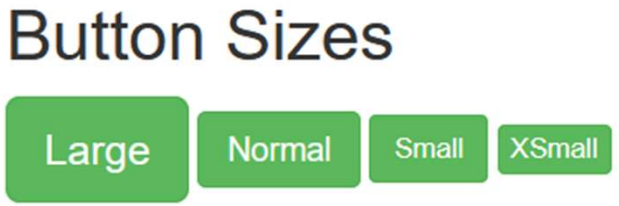
# Bootstrap Component - Button

- Bootstrap provides four **button sizes**.
- classes that define the different sizes are:

  **.btn-lg**

  **.btn-sm**

  **.btn-xs**




```
<div class="container">
  <h2>Button Sizes</h2>
  <button type="button" class="btn btn-success btn-lg">Large</button>
  <button type="button" class="btn btn-success">Normal</button>
  <button type="button" class="btn btn-success btn-sm">Small</button>
  <button type="button" class="btn btn-success btn-xs">XSmall</button>
</div>
```

# Bootstrap Component - Button

## Button Groups

- Bootstrap allows you to **group a series of buttons** together (on a single line) in a button group.
- Use a  element with class .btn-group to create a button group

** **

# Bootstrap Component- Jumbotron

- A **jumbotron** indicates a big box for calling extra attention to some special content or information.
- A **jumbotron** is displayed as a grey box with rounded corners. It also enlarges the font sizes of the text inside it.
- Inside a **jumbotron** you can put nearly any valid HTML, including other Bootstrap elements/classes.

# Bootstrap Component- Jumbotron

```
<div class="container">
  <div class="jumbotron">
    <h1>Bootstrap Tutorial</h1>
    <p>Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile-first projects on the web.</p>
  </div>
  <p>This is some text.</p>
  <p>This is another text.</p>
</div>
```

# Bootstrap Component- Grid

- Bootstrap’s grid system uses a series of containers, rows, and columns to layout and align content.
- Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.
- If you do not want to use all 12 columns individually, you can group the columns together to create wider columns.

https://chrome.google.com/webstore/detail/bootstrap-grid-overlay/mnlklmelflkheijccafopdohgclfefcg

# Bootstrap Component- Grid

**Why we need Grid?**

- To display our application layout properly on multiple devices of multiple screen sizes like desktop, laptop, tab, mobile etc, we should go for grid system.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 |
| span 4 | | | span 4 | | | span 4 | | | | | |
| span 4 | | | span 8 | | | | | | | | |
| span 6 | | | | | | span 6 | | | | | |
| span 12 | | | | | | | | | | | |

# Bootstrap Component- Grid

**Bootstrap 5 grid system has six classes**

**.col-** (extra small devices - screen width less than 576px)

**.col-sm-** (small devices - screen width equal to or greater than 576px)

**.col-md-** (medium devices - screen width equal to or greater than 768px)

**.col-lg-** (large devices - screen width equal to or greater than 992px)

**.col-xl-** (xlarge devices - screen width equal to or greater than 1200px)

**.col-xxl-** (xxlarge devices - screen width equal to or greater than 1400px)

# Bootstrap Component- Grid

## How to implement Grid?

We can implement grid by using 2 classes

| | XSmall | Small | Medium | Large | Extra Large |
|---|---|---|---|---|---|
| Class prefix | .col- | .col-sm- | .col-md- | .col-lg- | .col-xl- |
| Screen width | <576px | >=576px | >=768px | >=992px | >=1200px |

**1. row class: to define row**

** **

2. Within the row we can define columns by using the following class

**col-screenSize-noOfColumns**

# Bootstrap Component- Grid

**CASE 1:**

```
<div class="row">
        <div class="col-lg-3 box">Element-1</div>
        <div class="col-lg-6 box">Element-2</div>
        <div class="col-lg-3 box">Element-3</div>
</div>
```

?

**CASE 2:**

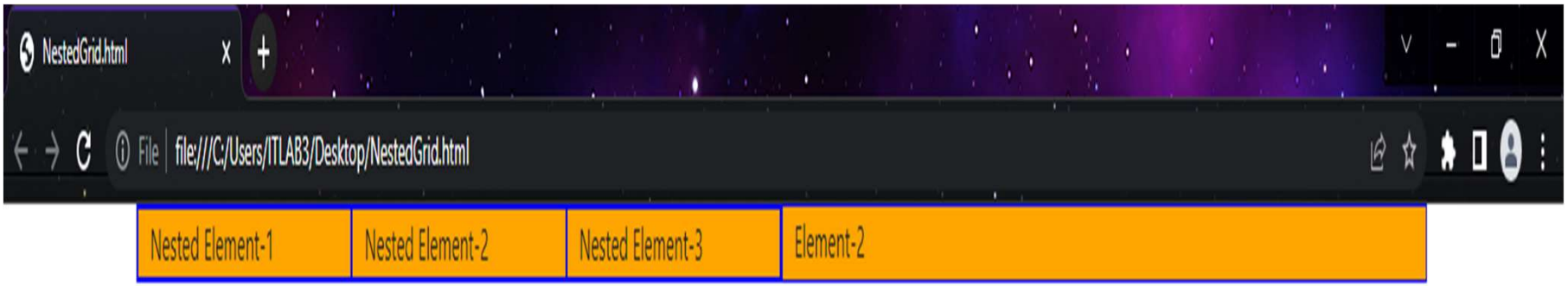
```
<div class="row">
        <div class="col-md-3 box">Element-1</div>
        <div class="col-md-3 box">Element-2</div>
        <div class="col-md-3 box">Element-3</div>
        <div class="col-md-3 box">Element-4</div>
</div>
```

?

# Bootstrap Component- Grid

## Nested Grid: Grid inside Grid

```
<div class="container">
    <div class="row">
        <div class="col-lg-6 box">
            <div class="row">
            <div class="col-lg-4 box">Nested Element-1</div>
            <div class="col-lg-4 box">Nested Element-2</div>
            <div class="col-lg-4 box">Nested Element-3</div>
        </div>
    </div>
        <div class="col-lg-6 box">Element-2 </div>
    </div>
</div>
```



# Bootstrap Component- Table

**A basic Bootstrap 5 table has a light padding and horizontal dividers.**

The **.table** class adds **basic styling** to a table.

The **.table-striped** class **adds zebra-stripes** to a table.

The **.table-bordered** class **adds borders on all sides** of the table and cells.

The **.table-hover** class adds a **hover effect** on table rows.

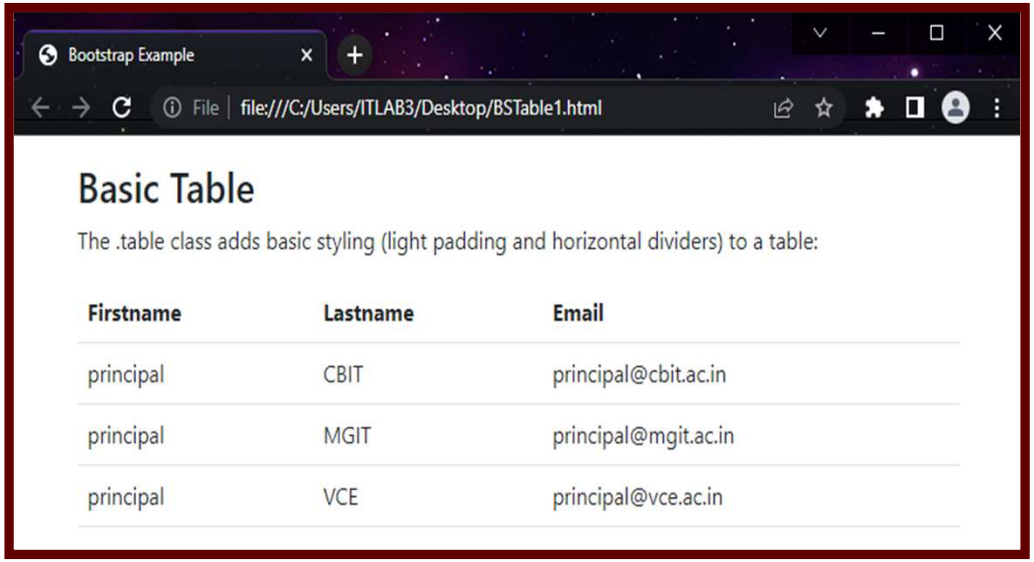
The .**table-dark** class adds a **black background** to the table.

The .**table-dark** and .**table-striped** to create a **dark, striped table**.

The **.table-borderless** class **removes borders** from the table.

# Bootstrap Component- Table

```
<table class="table">
  <thead>
    <tr>
      <th>Firstname</th>
      <th>Lastname</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>principal</td>
      <td>CBIT</td>
      <td>principal@cbit.ac.in</td>
    </tr>
    <tr>
      <td>principal</td>
      <td>MGIT</td>
      <td>principal@mgit.ac.in</td>
    </tr>
    <tr>
      <td>principal</td>
      <td>VCE</td>
      <td>principal@vce.ac.in</td>
    </tr>
  </tbody>
</table>
```



# Bootstrap Component- Table

**Contextual classes** can be used to color the whole table (<table>), the table rows (<tr>) or table cells (<td>).

| Default | Defaultson | def@somemail.com |
|---|---|---|
| Primary | Joe | joe@example.com |
| Success | Doe | john@example.com |
| Danger | Moe | mary@example.com |
| Info | Dooley | july@example.com |
| Warning | Refs | bo@example.com |
| Active | Activeson | act@example.com |
| Secondary | Secondson | sec@example.com |
| Light | Angie | angie@example.com |
| Dark | Bo | bo@example.com |

# Bootstrap Component- Table

| Class | Description |
|---|---|
| `.table-primary` | Blue: Indicates an important action |
| `.table-success` | Green: Indicates a successful or positive action |
| `.table-danger` | Red: Indicates a dangerous or potentially negative action |
| `.table-info` | Light blue: Indicates a neutral informative change or action |
| `.table-warning` | Orange: Indicates a warning that might need attention |
| `.table-active` | Grey: Applies the hover color to the table row or table cell |
| `.table-secondary` | Grey: Indicates a slightly less important action |
| `.table-light` | Light grey table or table row background |
| `.table-dark` | Dark grey table or table row background |

# Bootstrap Component- Table

```
<table class="table">
      <thead>
         <tr>
            <th>Firstname</th>
            <th>Lastname</th>
            <th>Email</th>
         </tr>
      </thead>
      <tbody>
         <tr>
            <td>Default</td>
            <td>Defaultson</td>
            <td>def@somemail.com</td>
         </tr>
         <tr class="table-primary">
            <td>Primary</td>
            <td>Joe</td>
            <td>joe@example.com</td>
         </tr>
       </tbody>
   </table>
```

# Bootstrap Component- Table

## Responsive Tables

The **.table-responsive** class **adds a scrollbar** to the table.

| Class | Screen width |
|---|---|
| `.table-responsive-sm` | < 576px |
| `.table-responsive-md` | < 768px |
| `.table-responsive-lg` | < 992px |
| `.table-responsive-xl` | < 1200px |
| `.table-responsive-xxl` | < 1400px |

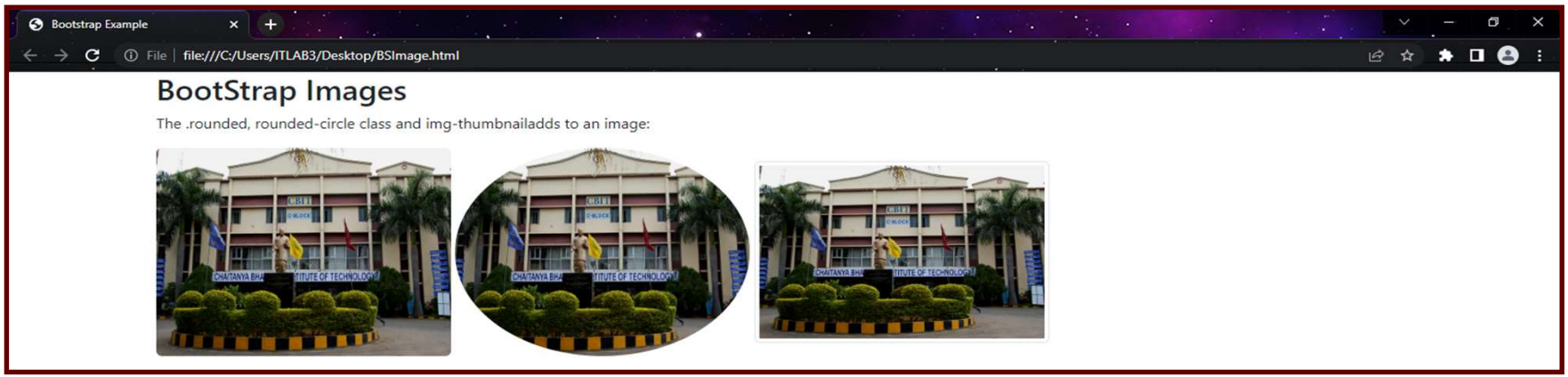
### Responsive Table

The .table-responsive class adds a scrollbar to the table when needed:

| # | Firstname | Lastname | Age | City | Country | Sex | Example | Example | Example |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Anna | Pitt | 35 | New York | USA | Female | Yes | Yes | Yes |

# Bootstrap Component- Image

```
<div class="container">
  <h2>BootStrap Images</h2>
  <p>The .rounded, rounded-circle class and img-thumbnailadds to an image:</p>
  <img src="cbit.jpg" class="rounded" alt="CBIT" width="304" height="236">
  <img src="cbit.jpg" class="rounded-circle" alt="CBIT" width="304" height="236">
  <img src="cbit.jpg" class="img-thumbnail" alt="CBIT" width="304" height="236">
</div>
```

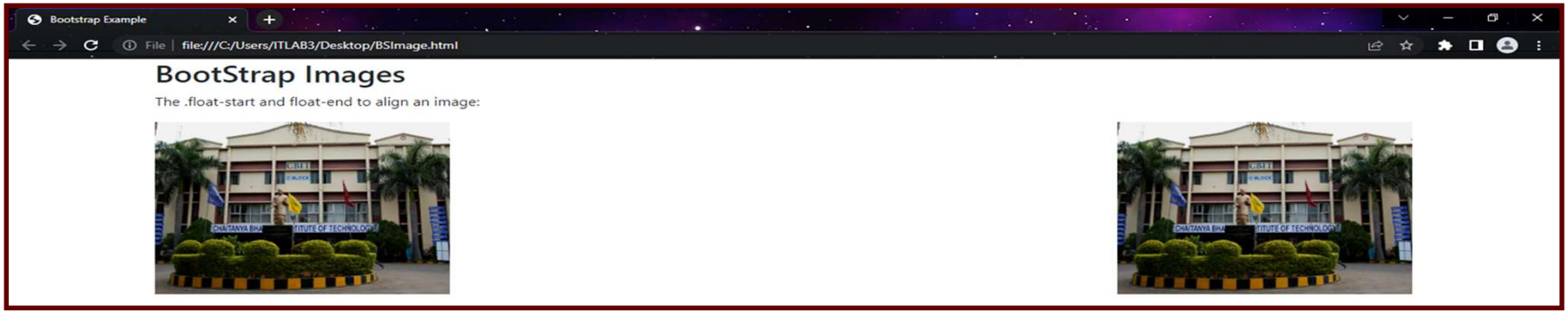


# Bootstrap Component- Image

## Aligning Images

Float an image to the **left with the .float-start** class or to the **right with .float-end**.

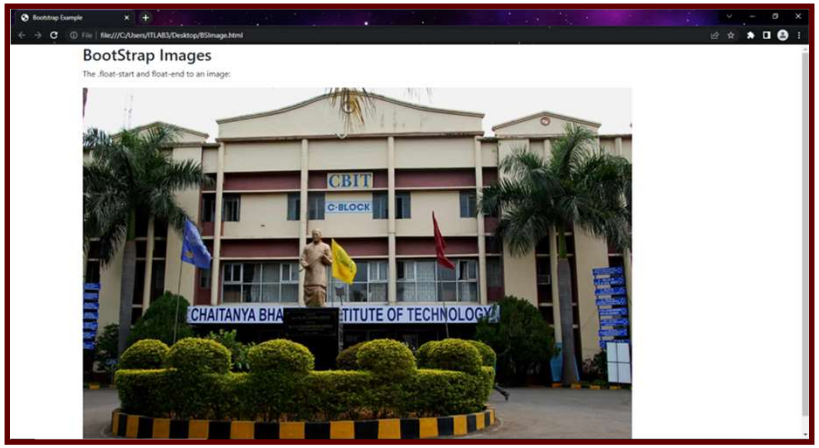
```
<div class="container">
  <h2>BootStrap Images</h2>
  <p>The .float-start and float-end to align an image:</p>
  <img src="cbit.jpg" class="float-start" alt="cbit" width="304" height="236">
  <img src="cbit.jpg" class="float-end" alt="cbit" width="304" height="236">
</div>
```



# Bootstrap Component- Image

➢Images come in all sizes. **Responsive Images** automatically adjust to fit the size of the screen.

```
<div class="container">
  <h2>BootStrap Images</h2>
  <p>The img-fluid to an image:</p>
   <img class="img-fluid" src="cbit.jpg" alt="CBIT-Hyd" width="1100" height="500">
</div>

</body>
</html>
```



# Bootstrap Component- Alert

- Bootstrap 5 provides an easy way to create predefined alert messages.

- **Example .alert-success**


```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  A simple danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  A simple info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
  A simple light alert—check it out!
</div>
<div class="alert alert-dark" role="alert">
  A simple dark alert—check it out!
</div>
```

# Bootstrap Component- Alert

## Link color

- Use the .alert-link utility class to quickly provide matching colored links within any alert.

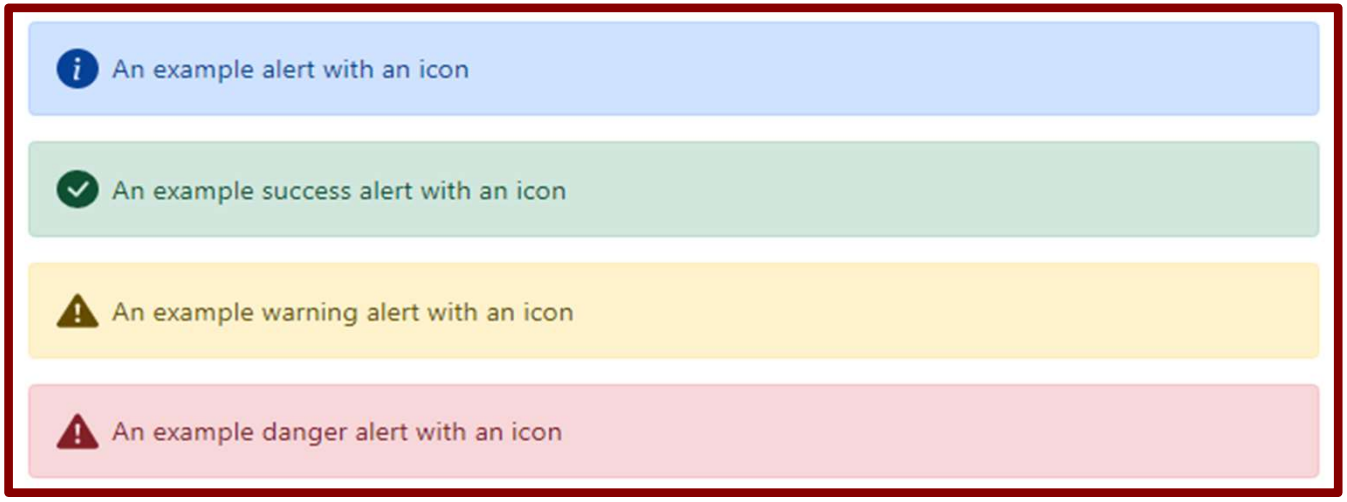
```
<div class="alert alert-primary" role="alert">
    A simple primary alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-secondary" role="alert">
    A simple secondary alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-success" role="alert">
    A simple success alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-danger" role="alert">
    A simple danger alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-warning" role="alert">
    A simple warning alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-info" role="alert">
    A simple info alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-light" role="alert">
    A simple light alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
  <div class="alert alert-dark" role="alert">
    A simple dark alert with <a href="#" class="alert-link">an example link</a>. Give it a click if you like.
  </div>
```

# Bootstrap Component- Alert

# Bootstrap Component- Alert

## Icons

- Similarly, you can use **Bootstrap Icons** to create alerts with icons. Depending on your icons and content, you may want to add more utilities or custom styles.



# Bootstrap Component- Alert

```
<div class="m-4">
    <!-- Success Alert -->
    <div class="alert alert-success alert-dismissible d-flex align-items-center fade show">
        <i class="bi-check-circle-fill"></i>
        <strong class="mx-2">Success!</strong> Your message has been sent successfully.
        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
    </div>

    <!-- Error Alert -->
    <div class="alert alert-danger alert-dismissible d-flex align-items-center fade show">
        <i class="bi-exclamation-octagon-fill"></i>
        <strong class="mx-2">Error!</strong> A problem has been occurred while submitting your data.
        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
    </div>

    <!-- Warning Alert -->
    <div class="alert alert-warning alert-dismissible d-flex align-items-center fade show">
        <i class="bi-exclamation-triangle-fill"></i>
        <strong class="mx-2">Warning!</strong> There was a problem with your network connection.
        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
    </div>
```

# Bootstrap Component- Progress Bar

A progress bar can be used to show how far a user is in a process.

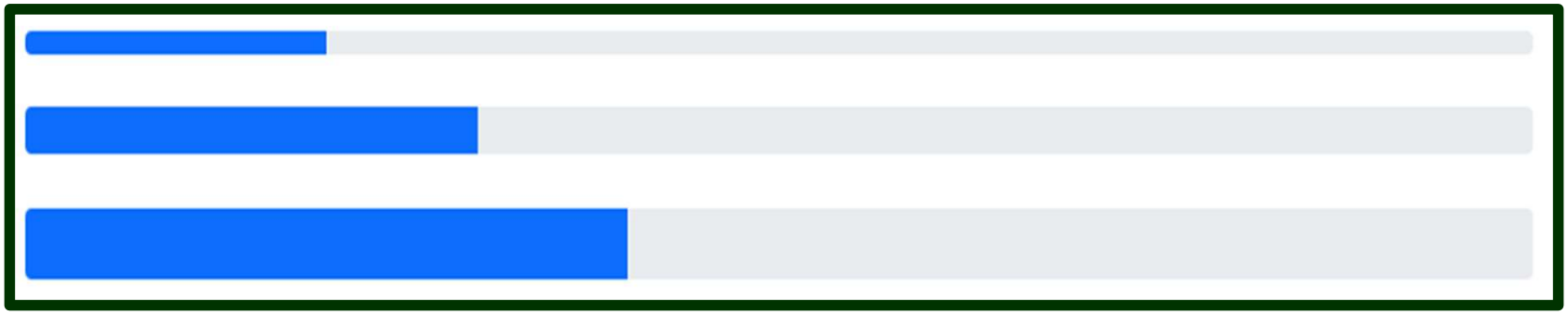
.progress class to create a default progress bar.

.progress-bar to indicate the progress so far.

# Bootstrap Component- Progress Bar

## Progress Bar Height

```
<div class="progress" style="height:20px">
  <div class="progress-bar" style="width:40%;height:20px"></div>
</div>
```



# Bootstrap Component- Progress Bar

## Progress Bar Labels

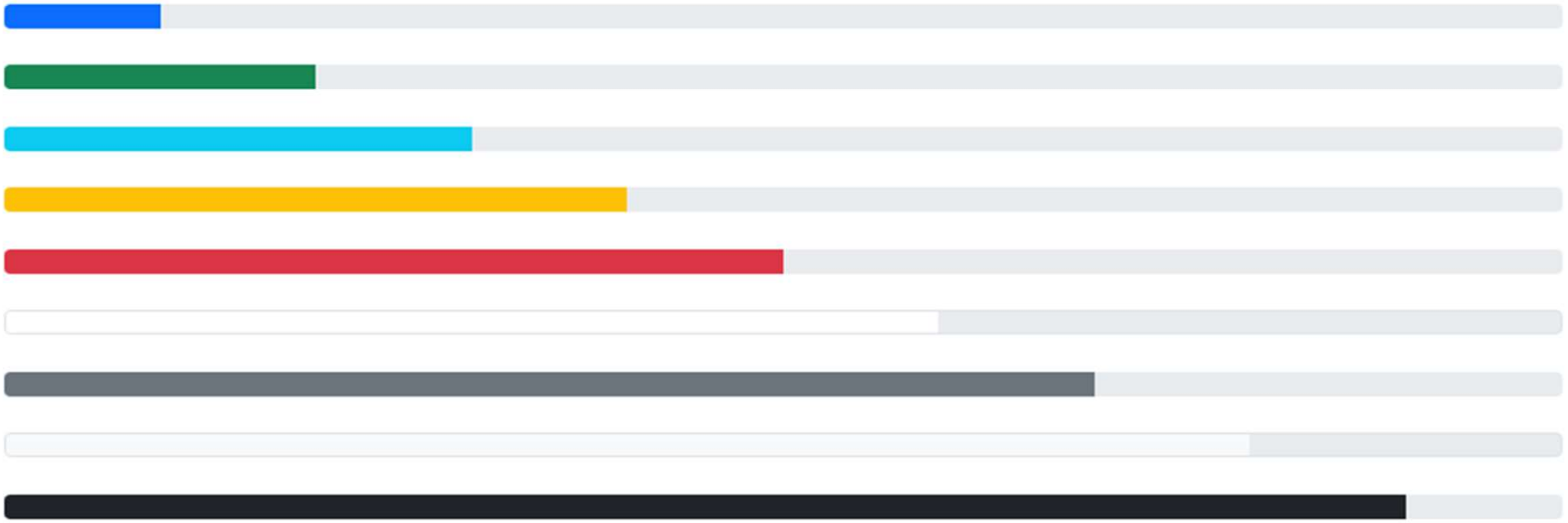
```
<div class="progress">
  <div class="progress-bar" style="width:70%">70%</div>
</div>
```

Add text inside the progress bar to show the visible percentage:

# Bootstrap Component- Progress Bar

## Colored Progress Bars

- By default, the progress bar is blue (primary).



# Bootstrap Component- Progress Bar

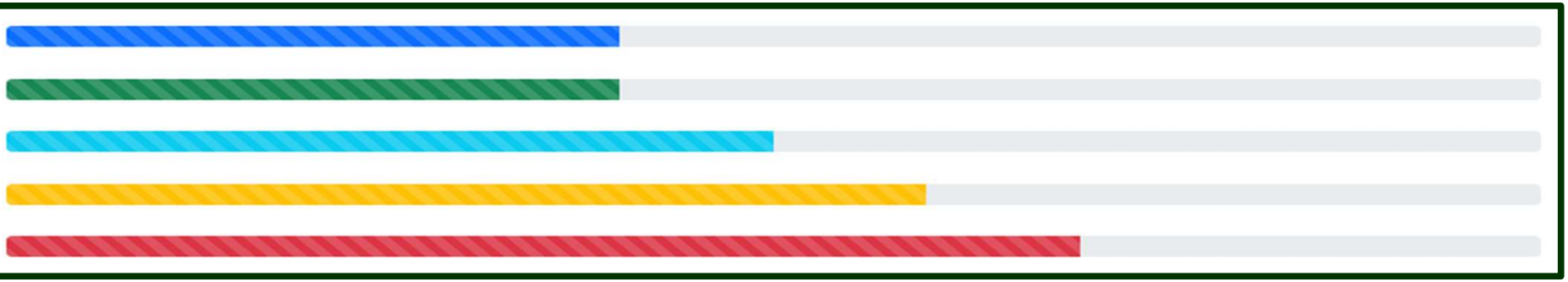
```
<!-- Blue -->
<div class="progress">
    <div class="progress-bar" style="width:10%"></div>
  </div>

  <!-- Green -->
  <div class="progress">
    <div class="progress-bar bg-success" style="width:20%"></div>
  </div>

  <!-- Turquoise -->
  <div class="progress">
    <div class="progress-bar bg-info" style="width:30%"></div>
  </div>

  <!-- Orange -->
  <div class="progress">
    <div class="progress-bar bg-warning" style="width:40%"></div>
  </div>

  <!-- Red -->
  <div class="progress">
    <div class="progress-bar bg-danger" style="width:50%"></div>
  </div>

</div>
```

# Bootstrap Component- Progress Bar

## Striped Progress Bars

**Use the .progress-bar-striped class to add stripes to the progress bars.**

```
<div class="progress">
  <div class="progress-bar progress-bar-striped" style="width:40%"></div>
</div>
```



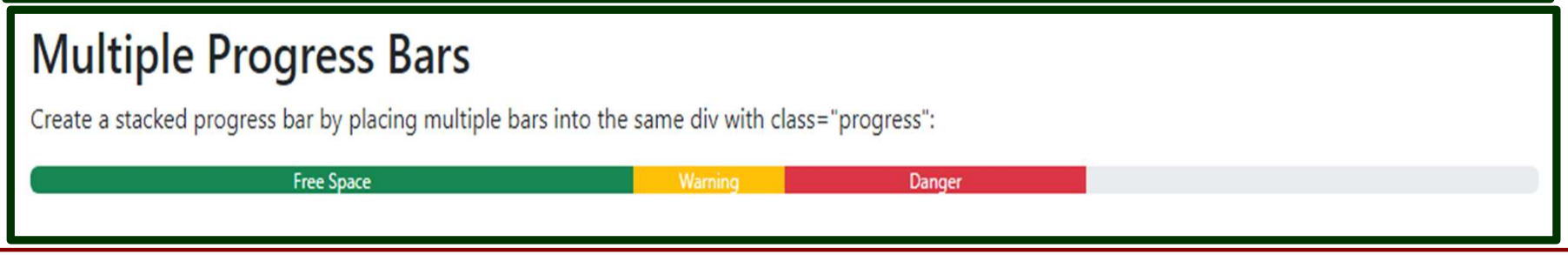
# Bootstrap Component- Progress Bar

## Animated Progress Bar

Add the `.progress-bar-animated` class to animate the progress bar:

## Multiple Progress Bars

```
<div class="container mt-3">
  <h2>Multiple Progress Bars</h2>
  <p>Create a stacked progress bar by placing multiple bars into the same div with class="progress":</p>
  <div class="progress">
    <div class="progress-bar bg-success" style="width:40%">   Free Space   </div>
    <div class="progress-bar bg-warning" style="width:10%">   Warning      </div>
    <div class="progress-bar bg-danger" style="width:20%">    Danger       </div>
  </div>
</div>
```



# Bootstrap Component- Spacing

**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

x - for classes that set both *-left and *-right

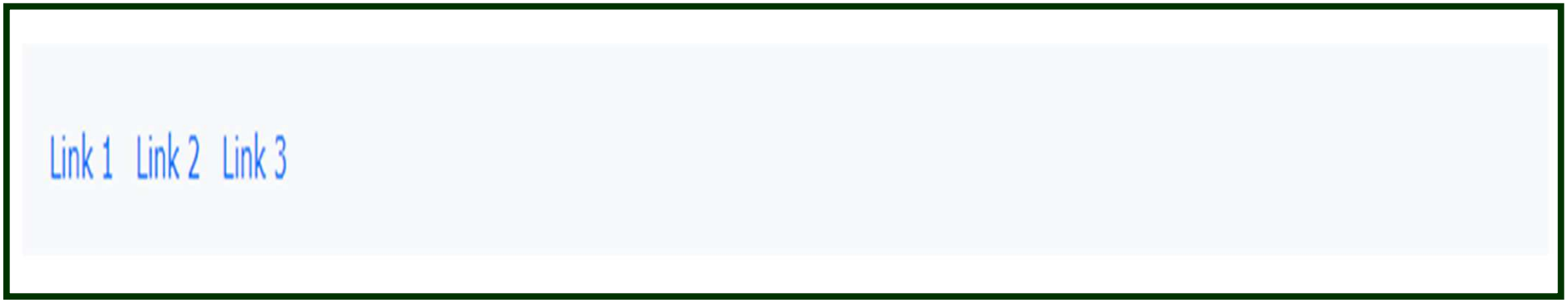
y - for classes that set both *-top and *-bottom

# Bootstrap Component- Navbar

- A navigation bar is a **navigation header** that is placed at the top of the page.
- Navbars require a wrapping **.navbar** with **.navbar-expand{-sm|-md|-lg|-xl|-xxl}** for responsive collapsing and color scheme classes.
- Navbars and their contents are fluid by default. Change the container to limit their horizontal width in different ways.
- **Navbars are responsive by default**, but you can easily modify them to change that.

# Bootstrap Component- Navbar

- To add links inside the navbar, use either an **<ul>** element (or a ) **with class="navbar-nav".** Then add **<li>** elements **with a .nav-item class followed by an <a> element with a .nav-link class**.



# Bootstrap Component- Navbar

- Remove the **.navbar-expand-* class** to create a navigation bar that will always be **vertical**.

```
<!-- A grey horizontal navbar that becomes vertical on small screens -->
<nav class="navbar navbar-expand-sm bg-light">

  <div class="container-fluid">
    <!-- Links -->
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="#">Link 1</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link 2</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link 3</a>
      </li>
    </ul>
  </div>

</nav>
```

# Bootstrap Component- Navbar

## Centered Navbar

- Add the **.justify-content-center** class to center the navigation bar.

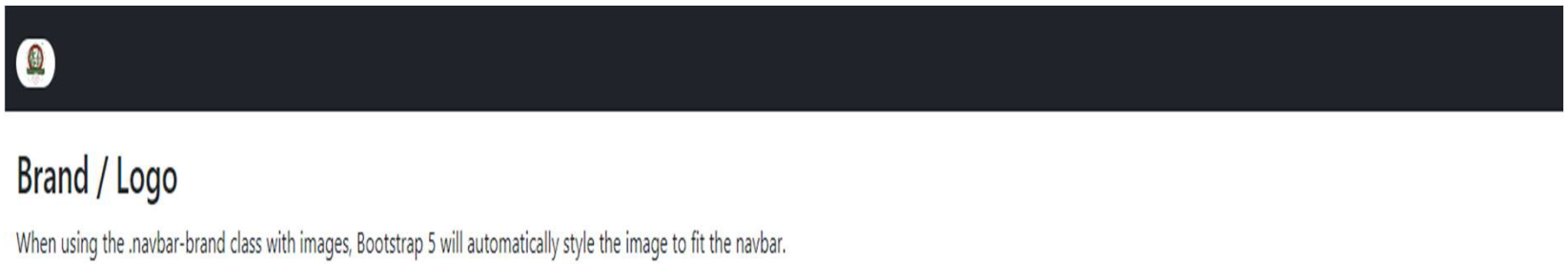

## Colored Navbar

- Use any of the .bg-color classes to change the background color of the navbar. **(.bg-primary, .bg-success, .bg-info, .bg-warning etc)**

# Bootstrap Component- Navbar

## Brand / Logo

- The .navbar-brand class is used to highlight the **brand/logo/project name** of your page.
- When using the **.navbar-brand** class with images, Bootstrap 5 will automatically style the image to fit the navbar vertically.



# Bootstrap Component- Navbar

## Navbar With Dropdown

```
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Dropdown</a>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Link</a></li>
    <li><a class="dropdown-item" href="#">Another link</a></li>
    <li><a class="dropdown-item" href="#">A third link</a></li>
  </ul>
</li>
```

# Bootstrap Component- Navbar

## Fixed Navigation Bar

- The navigation bar can also be fixed at the top or at the bottom of the page.
- The .fixed-top class m

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark fixed-top">
  ...
</nav>
```

- Use the .fixed-bottom class to make the navbar stay at the bottom of the page.

# Bootstrap Component- Navbar/Tabs/Pills

- **Nav**

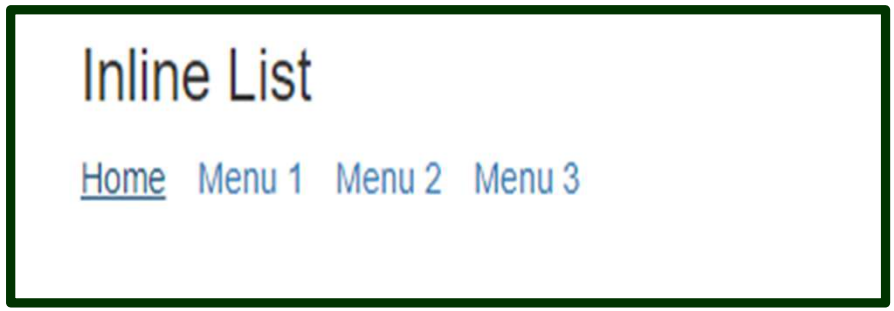

- **Tab**


- **Pill**

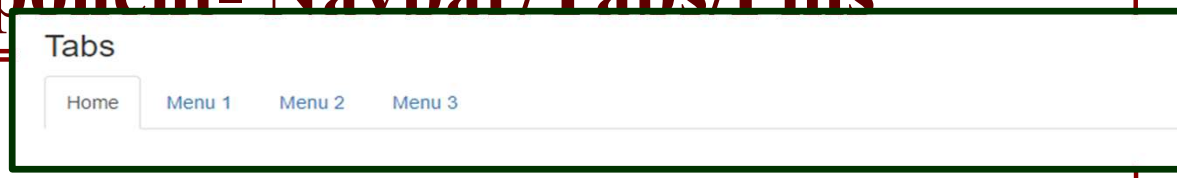
# Bootstrap Component- Navbar/Tabs/Pills

- If you want to create a horizontal menu of the list , add the **.list-inline class** to **<ul>**.

```
<div class="container">
  <h3>Inline List</h3>
  <ul class="list-inline">
    <li><a href="#">Home</a></li>
    <li><a href="#">Menu 1</a></li>
    <li><a href="#">Menu 2</a></li>
    <li><a href="#">Menu 3</a></li>
  </ul>
</div>
```



# Bootstrap Component- Navbar/Tabs/Pills




## Tabs

Tabs are created with **<ul class="nav nav-tabs">.**

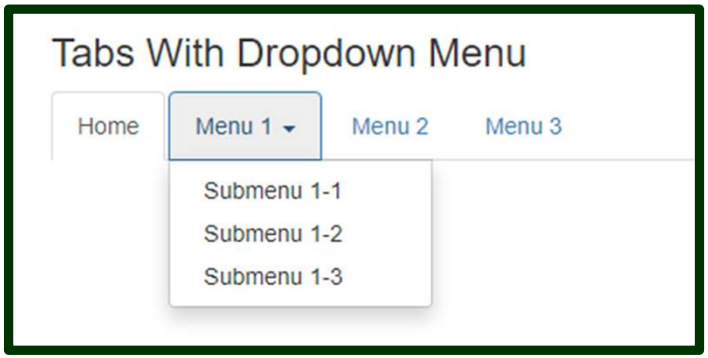
Note: Also mark the **current page** with **<li class="active">.**

```
<div class="container">
  <h3>Tabs</h3>
  <ul class="nav nav-tabs">
    <li class="active"><a href="#">Home</a></li>
    <li><a href="#">Menu 1</a></li>
    <li><a href="#">Menu 2</a></li>
    <li><a href="#">Menu 3</a></li>
  </ul>
</div>
```

# Bootstrap Component- Navbar/Tabs/Pills

## Tabs With Dropdown Menu

```
<div class="container">
  <h3>Tabs With Dropdown Menu</h3>
  <ul class="nav nav-tabs">
    <li class="active"><a href="#">Home</a></li>
    <li class="dropdown">
      <a class="dropdown-toggle" data-toggle="dropdown" href="#">Menu 1 <span class="caret"></span></a>
      <ul class="dropdown-menu">
        <li><a href="#">Submenu 1-1</a></li>
        <li><a href="#">Submenu 1-2</a></li>
        <li><a href="#">Submenu 1-3</a></li>
      </ul>
    </li>
    <li><a href="#">Menu 2</a></li>
    <li><a href="#">Menu 3</a></li>
  </ul>
</div>
```



# Bootstrap Component- Navbar/Tabs/Pills

## Pills

- Pills are created with **<ul class="nav nav-pills">.** Also mark the **current page** with **<li class="active">.**

```
<ul class="nav nav-pills">
  <li class="active"><a href="#">Home</a></li>
  <li><a href="#">Menu 1</a></li>
  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>
</ul>
```

# Bootstrap Component- Navbar/Tabs/Pills

## Vertical Pills

- Pills can also be displayed vertically. Just add the **.nav-stacked** class.

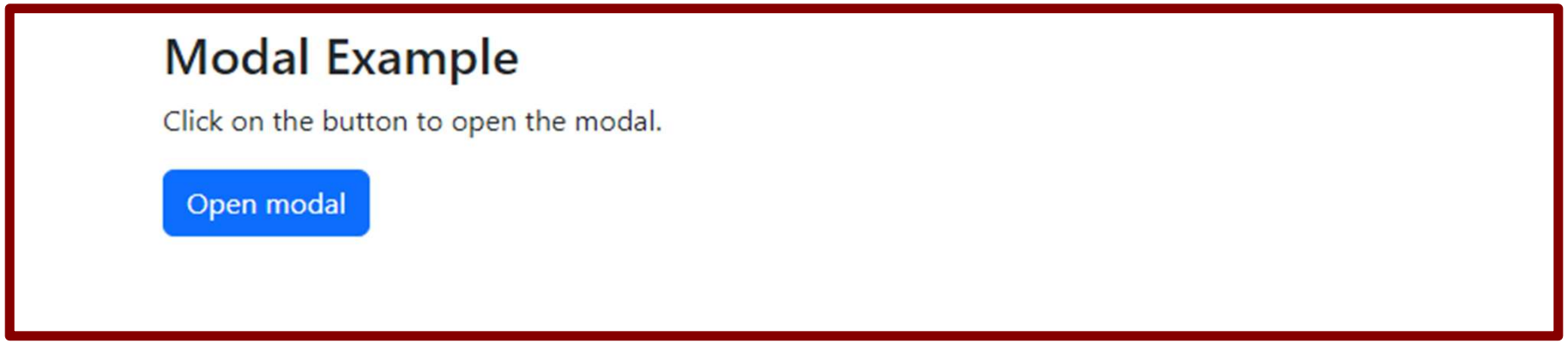
```
<ul class="nav nav-pills nav-stacked">
  <li class="active"><a href="#">Home</a></li>
  <li><a href="#">Menu 1</a></li>
  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>
</ul>
```

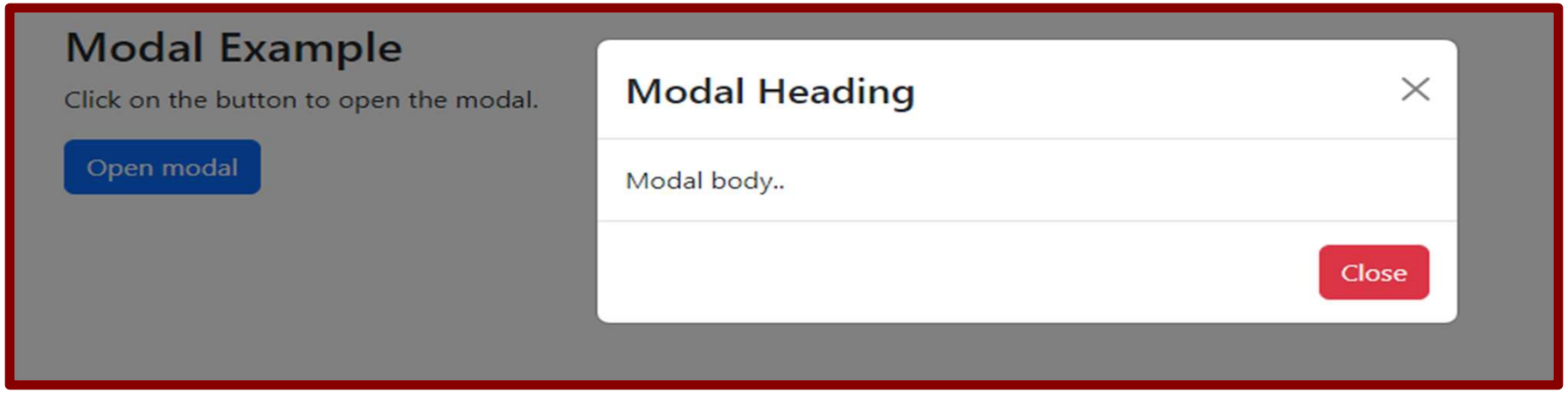
# Bootstrap Component- Modal

- Use Bootstrap's JavaScript modal plugin to add dialogs to your site for lightboxes, user notifications, or completely custom content.

```
<div class="modal" id="myModal">
  <div class="modal-dialog">
    <div class="modal-content">

      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Modal Heading</h4>
        <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
      </div>

      <!-- Modal body -->
      <div class="modal-body">
        Modal body..
      </div>

      <!-- Modal footer -->
      <div class="modal-footer">
        <button type="button" class="btn btn-danger" data-bs-dismiss="modal">Close</button>
      </div>

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

# Bootstrap Component- Modal



# Bootstrap Component- Modal

## Add animation

- Use the .fade class to add a fading effect when opening and closing the modal.

```
<!-- Fading modal -->
<div class="modal fade"></div>

<!-- Modal without animation -->
<div class="modal"></div>
```

## Modal Size

**Add the size class to the  element with class .modal-dialog.**

**.modal-sm** class for small modals.

```
<div class="modal-dialog modal-sm">
```

**.modal-lg** class for large modals

**.modal-xl** for extra large modals

# Bootstrap Component- Modal

## Full Screen Modals

- If you want the modal to span the whole width and height of the page, use the **.modal-fullscreen** class

## Responsive Full Screen Modals- .modal-fullscreen-*-*

| Class | Description |
|---|---|
| **.modal-fullscreen-sm-down** | **Fullscreen below 576px** |
| .modal-fullscreen-md-down | Fullscreen below 768px |
| **.modal-fullscreen-lg-down** | **Fullscreen below 992px** |
| .modal-fullscreen-xl-down | Fullscreen below 1200px |
| **.modal-fullscreen-xxl-down** | **Fullscreen below 1400px** |

# Bootstrap Component- Form

- Bootstrap provides several classes for forms.

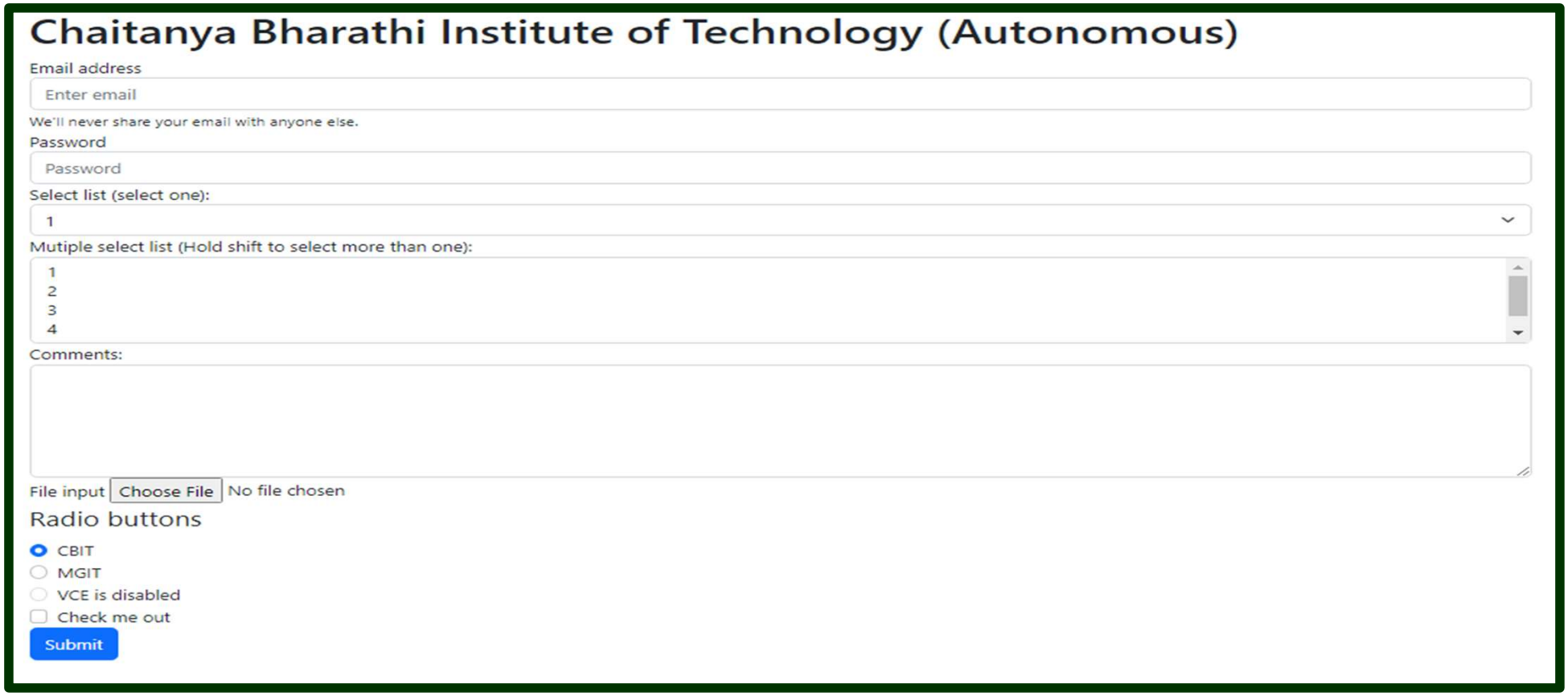
**1) form-group: Removed and Replaced by the Grid System in BS5**

It is responsible to maintain proper space between elements so that elements will be arranged properly.

**2) form-control:**

This class is responsible to make width as 100% for elements <input>, <textarea>, and <select>. It is also responsible for border styling.

# Bootstrap Component- Form



# Bootstrap Component- Form Validation

- The **.was-validated** or **.needs-validation** to the **<form> element**, depending on whether you want to provide validation feedback before or after submitting the form.
- The input fields will have a **green (valid)** or **red (invalid)** border to indicate what's missing in the form.
- You can also add a **.valid-feedback** or **.invalid-feedback** message to tell the user explicitly what's missing, or needs to be done before submitting the form.

# Bootstrap Component- Form Validation