JavaScript

Javascript is one of the Programming language like C, C++,Java,JavaScript, python,php, .dot.

In C,C++, Java is compile based language. But JavaScript is not compile based language.

In Java , if we write SOP("Print ") , If we compile, it will convert to machine level of code.

In JavaScript, we have JavaScriptEngine. This is used to run javascript code in runtime environment.

In webDevelopment we have 3 major Tech stacks.

1. HTML - Markup the elements

2. CSS - for Styling

3. JavaScript - to add the dynamic behaviour.

In browsers we have JS Engines.

edge, chrome, Opera - v8 engine

firefox- spider monkey engine

IE - chakra Engines

Safari- squarrel fish engine.

As per technology grown, JavaScript if runs in only browser, we cannot achieve programming language possibilities. So we got node JS

Node 🡪 runtime environment(JS) 🡪 v8 engine

Principles of execution is same as chrome.

Node JS is an open source, cross-platform/platform independent JavaScript runtime environment.

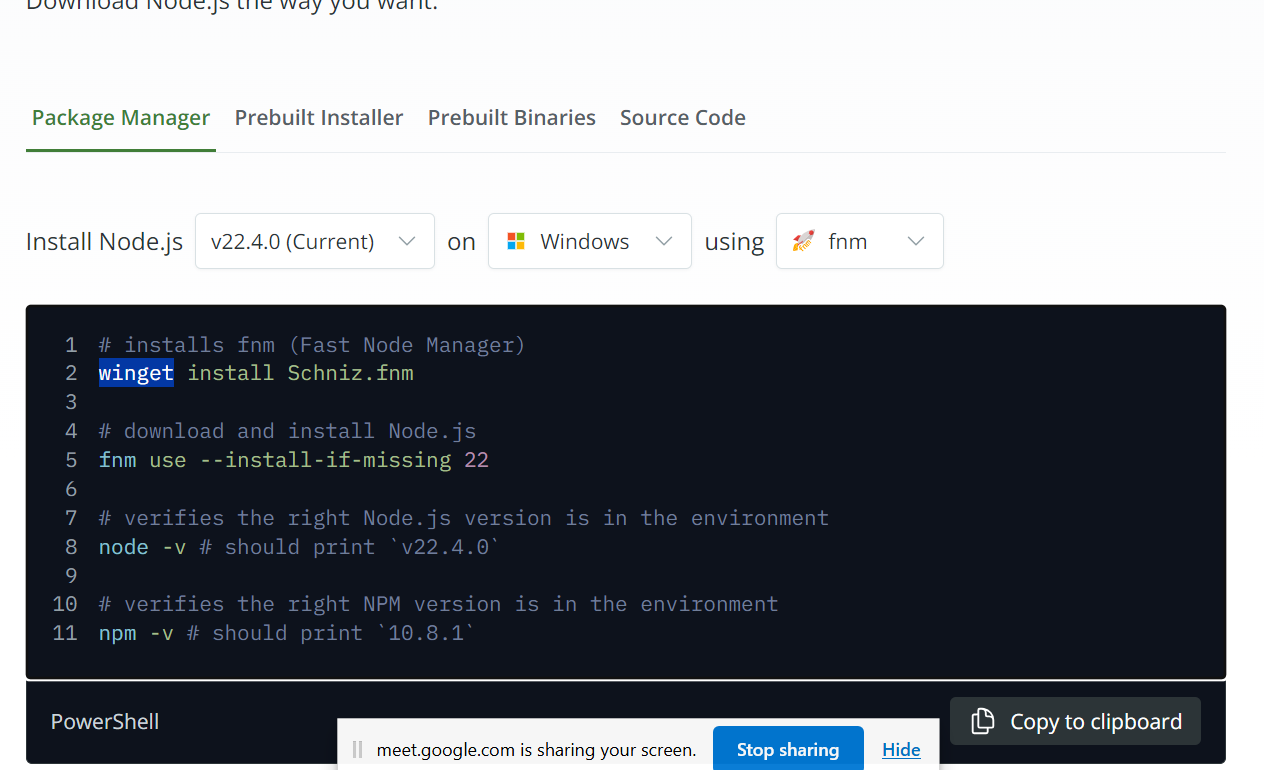
Download the Node JS in System

By using 2 ways we can install the node JS

1. Go to the url [https://nodejs.org/en](https://nodejs.org/en%20) and download the .exec file

It is an excutable file and install it in your system.

1. Go to node js <https://nodejs.org/en/download/package-manager/current>



Add the command in powershell and download.

Check in the command prompt whether our JavaScript has been installed or not

node -v

npm -v

For editing we will take Visual Studio

Variables

-----------------------------------------------------------------------------------------------------

Variables is used to store the piece of data. Variables are containers to store the data.

x->

5

JavaScript Variables can be declared in 4 ways:

* Automatically
* Using var
* Using let
* Using const

Automatically means the variables are defined without any keywords

Ex: x=1, y=2, z=x+y;

Var:  
Var keyword

var x = 5;  
var y = 6;  
var z = x + y;

Var keyword variable is used to declare in global level.

The var keyword was used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

The var keyword should only be used in code written for older browsers.

Let:

Let variable is used to store data and value can change anywhere. Let variable is used in declare in global level and it is used in within the block level.

let varaibleName;

variableName=1;

Const:

const variable is used to store the data and values cannot be changed.

const x=12.

JavaScript Identifiers

All JavaScript variables must be identified with unique names.

These unique names are called identifiers.

The general rules for constructing names for variables (unique identifiers) are:

* Names can contain letters, digits, underscores, and dollar signs.
* Names must begin with a letter.
* Names can also begin with $ and \_ (but we will not use it in this tutorial).
* Names are case sensitive (y and Y are different variables).
* Reserved words (like JavaScript keywords) cannot be used as names.

Datatypes:

1. Number :

Byte, short, int : -1,0,1

Float, double – 12.987, 1233.789;

1. BigInt : (2 power of 53 -1) – 9007199254740991
2. String: sequence of characters (“K7it”,””).
3. Boolean : true or false
4. Null : declaring variable without initializing any values.

Let a;

1. Objects:

Operators

Types of JavaScript Operators

There are different types of JavaScript operators:

* Arithmetic Operators

+, -, %,\*, /

* Assignment Operators

=+, =-,=

* Comparison Operators

==, >, <, <=, >==, !=

* String Operators

Concatenating of string and number, String and string.

let str="java";

str= str+"script";

* Logical Operators

&&, ||, !

* Bitwise Operators

&, |,~, ^, <<,>>,

* Ternary Operators
* Type Operators

Type Conversions

In programming, type conversion is the process of converting data of one [type](https://www.programiz.com/javascript/data-types) to another. For example, converting [string](https://www.programiz.com/javascript/string) data to [number](https://www.programiz.com/javascript/numbers).

There are two types of type conversion in JavaScript:

* Implicit Conversion - Automatic type conversion.
* Explicit Conversion - Manual type conversion.

Here we Explicit Conversions

* Converting Strings to Numbers

The global method Number() converts a variable (or a value) into a number.

Number(“3.2”) -🡪 3.2

Number(“ ”); 🡪 0

Number(“Lucky”); 🡪 NaN

Number(“234 890”); 🡪 NaN

* Converting Numbers to Strings

The global method String() can convert numbers to strings.

String(x);

String(126);

String(100+26);

Number method toString() did the same.

x.toString();

(125).toString();

(100+25).toString();

* Converting Dates to Numbers

The global method Number() can be used to convert dates to numbers.

d = new Date();  
Number(d)

The date method getTime() does the same.

d = new Date();  
d.getTime()

* Converting Dates to String

The global method String() can convert dates to strings.

String(Date())

The Date method toString() does the same.

Date().toString()

* Converting Booleans to Numbers

The global method Number() can also convert booleans to numbers.

Number(false)     // returns 0  
Number(true)

* Converting Booleans to String

The global method String() can convert booleans to strings.

String(false)      // returns "false"

String(true)       // returns "true"

The Boolean method toString() does the same.

false.toString()   // returns "false"  
true.toString()    // returns "true"

// Sttring to Number

let a= "123a";

let b = Number("123");

let c= Number(a);

console.log(typeof(a));

console.log(b);

console.log(typeof(c));

console.log(c);

// number to String

let  x=123;

let y=String(x);

console.log(typeof(y));

let number=123;

s= (123).toString();

console.log(s);

//dates to Number

let d = new Date();

console.log(typeof(d));

console.log(d);

console.log(Number(d));

date= d.getTime();

console.log(date);

//dates to String

let strDate= String(d);

console.log(typeof(strDate));

console.log(d.toString());

//boolean to Number

let bn=true;

console.log(Number(bn));

// Number to boolean

 let num=-12;

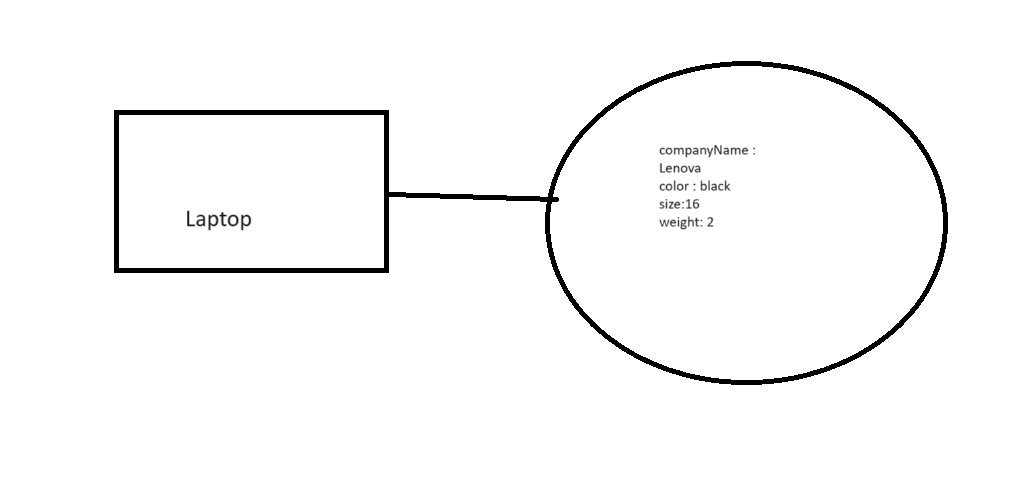
 console.log(Boolean(num));

 // boolean to String

 console.log(typeof(String(bn)));

 console.log(false.toString());

Object:



Here Laptop is an object, for that object we have the properties like companyName , color, size, weight are the properties or key.

let laptop={"company name":"lenova",color:"black",size:16, weight:2};

let laptop={"company name":"lenova",color:"black",size:16, weight:2};

const person={name:"Tharun",color:"white",height:6, weight:60}

const person1={"person name":"Babu",color:"white",height:6.5, weight:65, age:100};

console.log(laptop);

console.log(laptop.name);

console.log(laptop.color);

console.log(person1["person name"]);

console.log(person.name);

console.log(laptop["company name"]);

console.log(person1["age"]);

Here by using 2 ways we can access the properties keys and values.

* 1. Object.propertyName;
  2. Object[“propertyName”]

new Person(“lucky”,9,5,25);

Inside the constructor we can keep the parameters .

Rules:

* 1. Class name should be unique.
  2. Inside our class constructors are mandatory and constructor name constructor.

Class :

Class is a real time entity. Class having both properties and behaviours.

Syntax :

class Person{

    constructor(name, age, height, weight){

        this.name=name;

        this.age=age;

        this.weight=weight;

        this.height=height;

    }

}

 p1= new Person("Tharun", 23, 60,5.6);

const p2= new Person("Gnana",23,60,5.8);

console.log(p2.name);

console.log(p1);

In class, we need to declare constructor with constructor keyword.

To create object for the class we need new operator following with the constructor.

* 1. Ex: P1=Constructor parameters should be in format of (parameter1, parameter2,…)
  2. Class name should not be any JS keyword.
  3. Inside class only constructors, methods, accessors, properties only expected other keywords or identifiers are not allowed.

Like : console.log(p1.age);

class Person{

    constructor(name, age, height, weight){

        this.name=name;

        this.age=age;

        this.weight=weight;

        this.height=height;

    }

}

class car{

    constructor(companyName, color,model,cost ){

        this.companyName = companyName;

        this.model=model;

        this.color=color;

        this.cost=cost;

    }

}

 p1= new Person("Tharun", 23, 60,5.6);

const p2= new Person("Gnana",23,60,5.8);

console.log(p2.name);

console.log(p1);

car1 = new car("Mahindra","black","Thar",2500000);

const car2=car1;

car1.name="Audi";

console.log(car1.name);

Behaviours

Behaviours are also called as Functions.

Function is one of the executable block. Block is nothing but opening braces( { )and ending with closing braces( })

{

} 🡪 block

Syntax:

function functionName(){

}

Rules:

1. Functions should start with function keyword followed with function name and parathesis and block.
2. Inside our parathesis we can keep the parameters, By using comma we can separate the parameters.

(paramater1,parameter2…)

1. If we want Return any value when we call the function, we should use return statement.
2. If we want to call the function, with function name we will call.

function car(){

    console.log("This is first function");

}

car(); // calling statement

function sum(a,b){

    console.log("Addition of the values", (a+b));

}

let a=4, b=20;

sum(2,3);

sum(a,b);

function mul(a,b){

    return a\*b;

}

mulipleValue=mul(a,b);

console.log(mulipleValue)

function evenOrodd(a){

    if(a%2==0){

        return true;

    }else{

        return false;

    }

}

console.log(evenOrodd(7));

Arrow Function

Arrow functions are shorter functions.   
Arrow Function with single statement.

functionName Arrow functions symbol return Value

let myFunction = (a, b) => a \* b;

Parameters inside the function

Arrow functions with multiple statements

let sum = (a, b) => {

Console.log(“Addition”);

return a+b;

}

Calling statement

let result= Sum(12,15);

Example:

console.log(person);

let myfunction =()=> "HelloWorld";

console.log(myfunction());

let sum=(a,b)=> a+b;

console.log(sum(5,4));

let mul=(a,b,c)=>{

    console.log("This function is relates to Multiplications");

    return a\*b\*c;

}

console.log(mul(12,2,3));

div =(a,b)=>{

    return a/b;

}

console.log(div(21,7));

Conditions

If Conditions

It is to validate the value whether it is true or false.

Condition

else false

true, go inside if block

syntax:

if(condition){

} If true it will go for if block , false means it will go for else block.

else{

}

Examples:

Let number=1;

Let flag=true;

If(number==1)

{

Console.log(“valid number”);

}else{

Console.log(“Not Valid Number”);

}

If , else if , else Conditions

If (condition){

}else if(condition){

}else if(condition){

}else{

}

Example:

let string="abc"

if(string=="abcd"){

    console.log("valid number");

}else if(string=="abc"){

    console.log("partially Valid");

}else{

    console.log("not valid");

}

Nested If Condition:

Inside If Condition writing one more if condition , is called nested if condition.

Example:

if(a>0){

    if(a==123){

        console.log("valid number");

    }else{

        console.log("Not Valid");

    }

}else{

    console.log("Please enter value greater than zero")

}

Switch -case Condition:

Switch case is used to validate the condition , with different cases.

Syntax:

Switch(condition){

Case (value) :

Break;

Case (value) :

Break;

Default:

Example:

let day="Wednesday";

switch(day){

    case ("sunday"):

        console.log("Day is Sunday")

        break;

    case ("Monday"):

        console.log("Day is Monday")

        break;

    case ("Tuesday"):

        console.log("Day is Tuesday")

        break;

    case ("Wednesday"):

        console.log("Day is Wednesday")

        break;

    case ("Thursday"):

        console.log("Day is Thursday")

        break;

    case ("Friday"):

        console.log("Day is Friday")

        break;

    default:

        console.log("Its a holiday");

}

Ex:

let i=20;

switch(i%2){

    case(0):

     console.log("It is an even number");

     break;

    case(1):

     console.log("It is odd number");

     break;

    default:

Break : break is a keyword which is used to break the condition directly , not to continue the next line.

Loops

If you want execute the code n no of times we will keep that code in the loop.

1. For
2. While
3. Do while

For loop

Syntax:

For(initialization; condition; increment){

}

Ex: 1 2 4

For( let i=0;i<=10;i++){ // i=0, 0<=10,

Console.log(i); 3

}

o/p:

01………10

* Initialization will execute only one time in the loop process.
* Second it will check the condition.
* If condition true it will go inside the block.
* It will increment the value. ( i++ = i+1)
* After increment, it will check for the condition, if condition true it will go into loop else it will come out from the for loop.

Nested for Loop

Inside one for loop having another for loop, is called Nested for loop.

Syntax:

For(initialization;condition;increment){

For(initialization; condition; increment){

}

}

Example:

let i=1

 for(;i<=10;i++){     //2<=10

    for(let j=0;j<i;j++){ //1<1, 0<2,1<2,2<2

      console.log(j) //0,//0,1

    }

    console.log(i); //1,2

 }

Output:

0

1

0

1

2

0

1

2

3

0

1

2

3

4

While Loop

Syntax:

While(condition){

}

Note: while loop paranthesis we will keep only condition, as like for loop we will not keep initialization, increment.

Ex:

Let n=2;

While(n<=10){

Console.log(n);

n++;

}

Do-while loop

Syntax:

do{

Console.log(n);

n++;

}while(n>5)

DOM

DOM : Document Object Model

By using HTML DOM JavaScript can access, change or remove any elements of HTML document and also can create new elements at any position.

Browser will convert HTML Document to DOM tree structure.

### The HTML DOM Tree of Objects



<HTML>

<head>

<title>MY Title</title>

</head>

<body>

<href>

<a >

<h1>

</body>

Document Object has Properties and Methods.

Using this document object properties & Methods we can

* Select HTML elements
* Modify HTML elements
* Remove/delete HTML elements
* Create HTML elements
* Add/Remove/Change styles to HTML elements.

Methods to select HTML elements

* 1. document.getElementById(“idName”) 🡪 Returns the element with the specified Id

object methodName Argument

* 1. document.getElementByClassName(“className”) 🡪 Returns the list of all elements belongs to the specified class.
  2. Document.getElementsByTagName(“tagName”) 🡪 Returns the list of all elements with specified tag.
  3. Document.querySelector(“.class/#id/tagname”) 🡪 Returns the first object matching CSS style selector.

Example:

<Html>

<head>

</head>

<body>

<div id="first">

<h1>This is heading1</h1>

<p>This is first paragraph</p>

</div>

<div class="special">

<h2>This is heading2</h2>

<p>This is second paragraph</p>

</div>

<div class="special">

<h2>This is heading2</h2>

<p>This is second paragraph</p>

</div>

<script>

let x= document.getElementsByClassName("special")

console.log(x)

let y= document.getElementById("first")

console.log(y)

let z= document.getElementsByTagName("div")

console.log(z)

let a= document.querySelector(".special")

console.log(a)

</script>

</body>

</html>

DOM properties to select HTML Elements.

* Document.body<body>
* Document.head<head>
* Document.title<title>
* Document.Image<image>
* Document.scripts<scripts>
* Document.forms<form>
* Document.anchors <a>
* Document.links <href>

JavaScript Events

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

In html, there are various events which represents that some activity is performed by the user or by the browser. When [javascript](https://www.javatpoint.com/javascript-tutorial) code is included in [HTML](https://www.javatpoint.com/html-tutorial), JS react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, JS handles the HTML events via Event Handlers.

There are types of events.

* 1. Mouse events.
  2. Keyboard events.
  3. Form events
  4. Window/Document events

OnClick Event

<html>

<head> Javascript Events </head>

<body>

<script

function onClick(){

          alert("Registration is successfull");

        }

</script>

<form>

<input type="button" onclick="onClick()" value="Who's this?"/>

</form>

</body>

</html>

Mouse Events

<html>

<head>

<h1> Javascript Events </h1>

</head>

<body>

<script language="Javascript" type="text/Javascript">

function moveOverEvent(){

            alert("please enter the name");

        }

</script>

<p onmouseover="mouseoverevent()"> Keep cursor over me</p>

</body>

</html>

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=" aqua";

}

</script>

</body>

</html>

Keyboard 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>

Load Event

CRUD Operation

C- create -> Post Method -> It will create/save the new data into DB

R- read -> Get Method -> it read the existing data from db

U – update -> Put -> it will modify the existing data

D – delete -> Delete Method -> delete the existing data from db

Form Validation:

LoginForm

<html>

<head>

<title>Login form </title>

</head>

<body>

<script>

function validateform(){

var name=document.getElementById("name").value;

var password=document.getElementById("password").value;

if (name==null || name==""){

alert("Name can't be blank");

return false;

}else if(password==null||password==""){

alert("Password can't be blank.");

return false;

} else if(password.length<6){

alert("Password must be at least 6 characters long.");

return false;

}

}

</script>

<body>

<form name="myform" method="post" onsubmit="return validateform()" >

Name: <input type="text" id="name"><br/>

Password: <input type="password" id="password"><br/>

<input type="submit" value="Login">

</form>

<input type="submit" value="Register">

<a href="D:/JavaScript/RegisterForm.html">

</body>

</html>

Registration Form

<!DOCTYPE html>

<html >

<head>

     <title>Registration Form</title>

    <style>

    .error {

        color: red;

    }

    </style>

</head>

<body>

    <h2>Registration Form</h2>

    <form id="registrationForm" onsubmit="return validateForm()" action="LoginForm.html">

        <label for="username">Username:</label><br>

        <input type="text" id="username" name="username"><br>

        <span id="usernameError" class="error"></span><br>

        <label for="email">Email:</label><br>

        <input type="email" id="email" name="email"><br>

        <span id="emailError" class="error"></span><br>

        <label for="password">Password:</label><br>

        <input type="password" id="password" name="password"><br>

        <span id="passwordError" class="error"></span><br>

        <label for="password">ConfirmPassword:</label><br>

        <input type="password" id="confirmpassword" name="password"><br>

        <span id="confirmpasswordError" class="error"></span><br>

        <input type="submit" value="Register">

    </form>

    <script src="registration.js"></script>

</body>

</html>

Registration.js

function validateForm() {

    // Get form inputs

    var username = document.getElementById("username").value.trim();// trim is method which is used to remove the start and end space of value

    var email = document.getElementById("email").value.trim();

    var password = document.getElementById("password").value.trim();

    var confirmPassword = document.getElementById("confirmpassword").value.trim();

    // Reset previous error messages, initializing the variables

    document.getElementById("usernameError").textContent = "";

    document.getElementById("emailError").textContent = "";

    document.getElementById("passwordError").textContent = "";

    document.getElementById("confirmpasswordError").textContent="";

    // Validate username

    if (username === "") {

        document.getElementById("usernameError").textContent = "Username is required";

        return false;

    }

    // Validate email

    if (email === "") {

        document.getElementById("emailError").textContent = "Email is required";

        return false;

    } else if (!isValidEmail(email)) {

        document.getElementById("emailError").textContent = "Invalid email format";

        return false;

    }

    // Validate password

    if (password === "") {

        document.getElementById("passwordError").textContent = "Password is required";

        return false;

    } else if (!isValidPassword(password)) {

        document.getElementById("passwordError").textContent = "Password should be Minimum eight characters, at least one uppercase letter, one lowercase letter, one number and one special character";

        return false;

    }

    // validate confirm password

    if(confirmPassword === ""){

        document.getElementById("confirmpasswordError").textContent = "confirm password is required";

        return false;

    }

    else if(confirmPassword !== password){

        document.getElementById("confirmpasswordError").textContent = "confirm password is not matched with password";

        return false;

    }

    // If all validations pass, the form can be submitted

   // window.location.href="D:/JavaScript/FormValidation.html";

    return true;

}

function isValidEmail(email) {

    // Basic email validation regex

    var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    return emailRegex.test(email);

}

function isValidPassword(password) {

    // Basic email validation regex

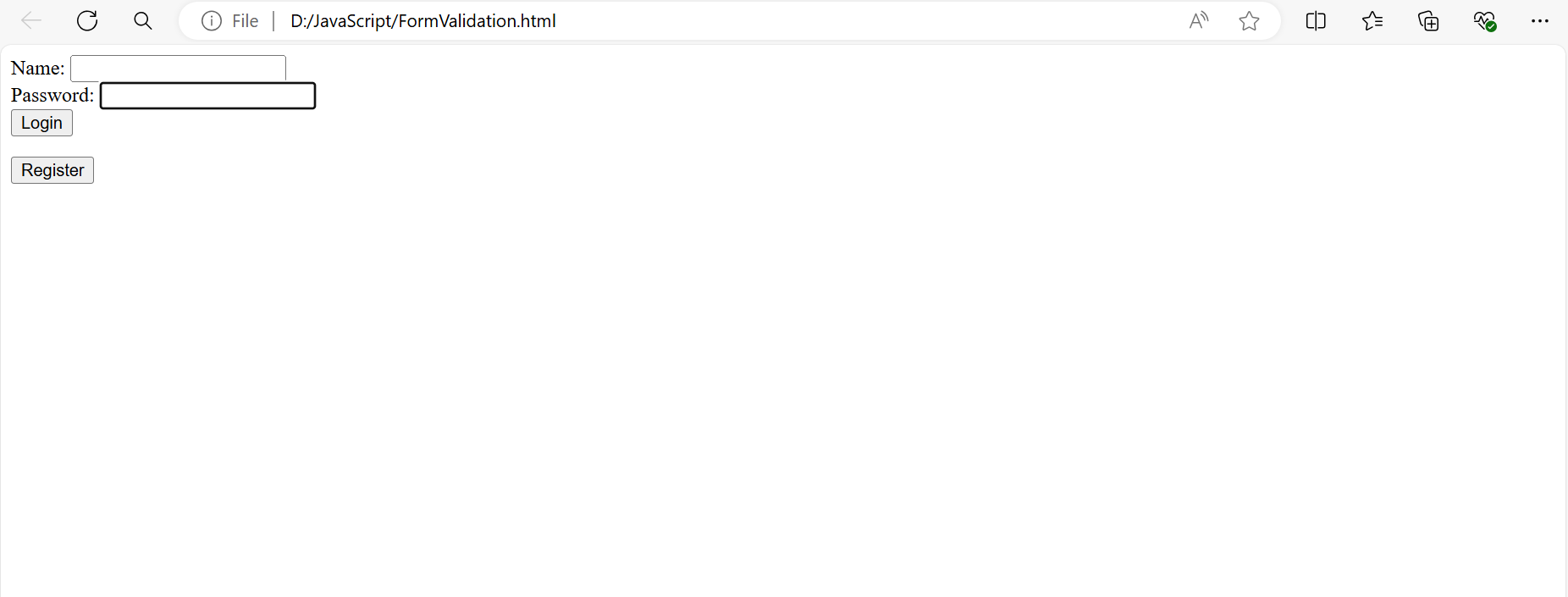
    var passwordRegex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/;

    return passwordRegex.test(password);

}

Output:





Regex Pattern for Password Validation:

Minimum eight characters, at least one letter and one number:

"^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$"

Minimum eight characters, at least one letter, one number and one special character:

"^(?=.\*[A-Za-z])(?=.\*\d)(?=.\*[@$!%\*#?&])[A-Za-z\d@$!%\*#?&]{8,}$"

Minimum eight characters, at least one uppercase letter, one lowercase letter and one number:

"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)[a-zA-Z\d]{8,}$"

Minimum eight characters, at least one uppercase letter, one lowercase letter, one number and one special character:

"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$"

Minimum eight and maximum 10 characters, at least one uppercase letter, one lowercase letter, one number and one special character:

"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,10}$"

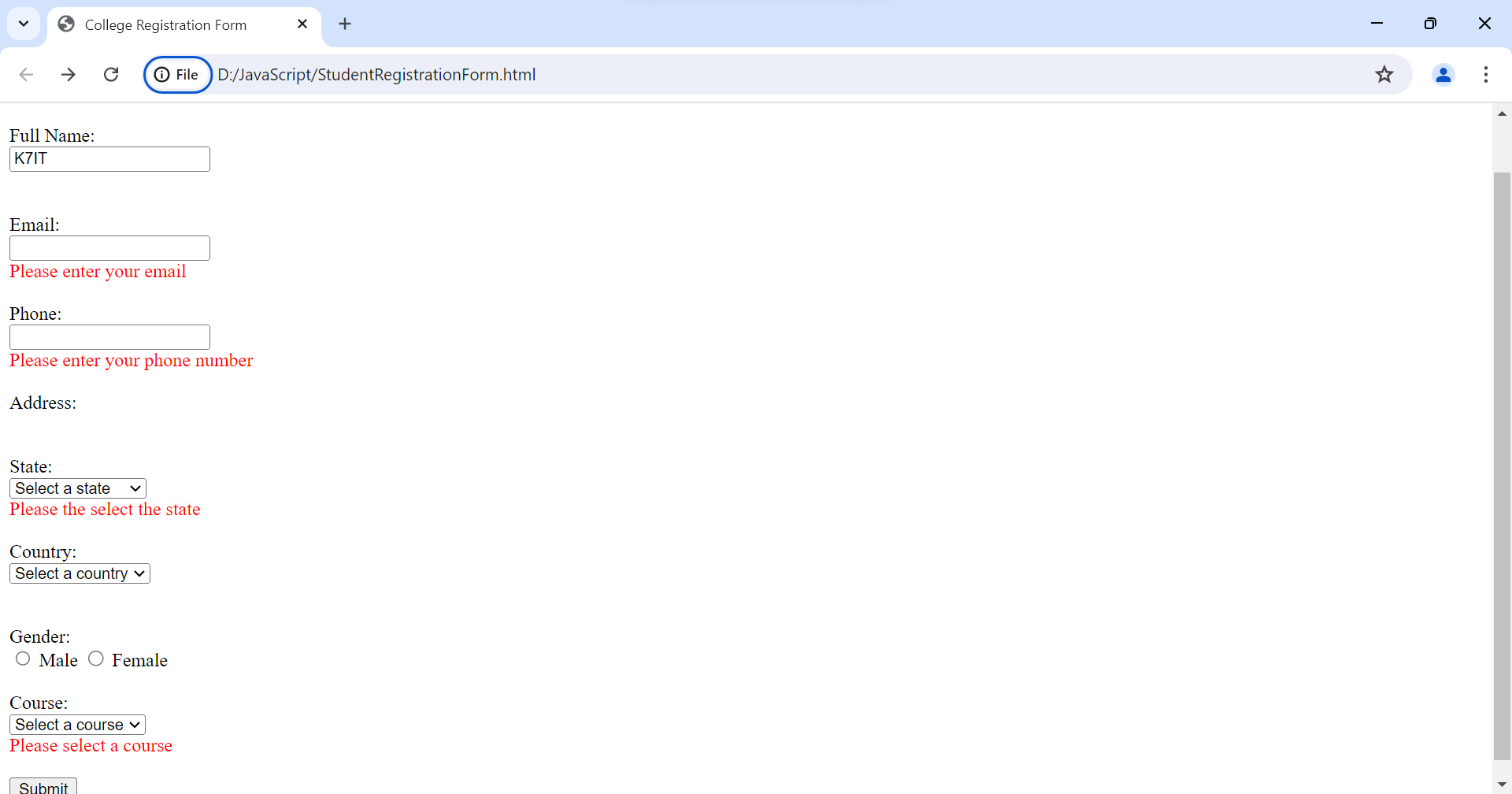
Student Registration Form

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <title>College Registration Form</title>  <style>      .error {          color: red;      }  </style>  </head>  <body>      <h2>College Registration Form</h2>      <form id="registrationForm" onsubmit="return validateForm()" action="LoginForm.html">          <label for="fullname" required>Full Name:</label><br>          <input type="text" id="fullname" name="fullname" required><br>          <span id="fullnameError" class="error"></span><br><br>          <label for="email">Email:</label><br>          <input type="email" id="email" name="email"><br>          <span id="emailError" class="error"></span><br><br>          <label for="phone">Phone:</label><br>          <input type="tel" id="phone" name="phone"><br>          <span id="phoneError" class="error"></span><br><br>          <label for="address">Address:</label><br>          <address id="address" name="address"></address><br><br>          <label for="state">State:</label><br>          <select id="state" name="state">              <option value="">Select a state</option>              <option value="state">AndhraPradesh</option>              <option value="state">Telanga</option>              <option value="state">Karnataka</option>              <option value="state">Kerala</option>              <!-- Add more states as needed -->          </select><br>          <span id="stateError" class="error"></span><br><br>          <label for="country">Country:</label><br>          <select id="country" name="country">              <option value="">Select a country</option>              <option value="country">INDIA</option>              <option value="country">USA</option>              <option value="country">RUSSIA</option>              <!-- Add more countries as needed -->          </select><br>          <span id="countryError" class="error"></span><br><br>          <label for="gender">Gender:</label><br>          <input type="radio" id="male" name="gender" value="male">          <label for="male">Male</label>          <input type="radio" id="female" name="gender" value="female">          <label for="female">Female</label><br><br>          <label for="course">Course:</label><br>          <select id="course" name="course">              <option value="">Select a course</option>              <option value="engineering">Engineering</option>              <option value="medical">Medical</option>              <option value="arts">Arts</option>              <option value="aeronatical">Aeronatical</option>          </select><br>          <span id="courseError" class="error"></span><br><br>          <input type="submit" value="Submit">      </form>      <script src="StudentRegistration.js"></script>  </body>  </html> |

StudentRegistration.js

|  |
| --- |
| function validateForm() {      var fullname = document.getElementById("fullname").value;      var email = document.getElementById("email").value;      var phone = document.getElementById("phone").value;      var address = document.getElementById("address").value;      var state = document.getElementById("state").value;      var country = document.getElementById("country").value;      var course = document.getElementById("course").value;      var gender = document.querySelector('input[name="gender"]:checked'); // to get which  radio button is selected      var fullnameError = document.getElementById("fullnameError");      var emailError = document.getElementById("emailError");      var phoneError = document.getElementById("phoneError");      var courseError = document.getElementById("courseError");      var stateError =  document.getElementById("stateError");      // Resetting errors      fullnameError.innerHTML = "";      emailError.innerHTML = "";      phoneError.innerHTML = "";      courseError.innerHTML = "";      stateError.innerHTML = "";      var isValid = true;      // Validate fullname      if (fullname === "") {          fullnameError.innerHTML = "Please enter your full name";          isValid = false;      }      // Validate email      if (email === "") {          emailError.innerHTML = "Please enter your email";          isValid = false;      } else if (!isValidEmail(email)) {          emailError.innerHTML = "Invalid email format";          isValid = false;      }      // Validate phone      if (phone === "") {          phoneError.innerHTML = "Please enter your phone number";          isValid = false;      } else if (!isValidPhone(phone)) {          phoneError.innerHTML = "Invalid phone number format";          isValid = false;      }      // Validate course      if (course === "") {          courseError.innerHTML = "Please select a course";          isValid = false;      }      // Validate gender      if (!gender) {          alert("Please select your gender");          isValid = false;      }      //validate State      if(state===""){          stateError.innerHTML="Please the select the state";          isValid= false;      }      return isValid;  }  // Function to validate email format  function isValidEmail(email) {      var emailRegex = /\S+@\S+\.\S+/;      return emailRegex.test(email);  }  // Function to validate phone number format  function isValidPhone(phone) {      var phoneRegex = /^\d{10}$/;      return phoneRegex.test(phone);  } |

Output:



CSS

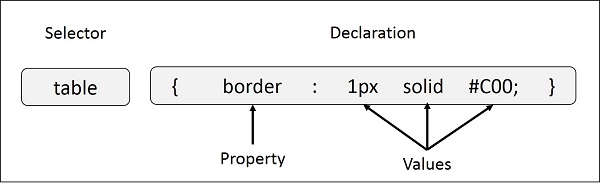
* Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.
* CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
* CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantages

* CSS saves time − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
* Pages load faster − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
* Easy maintenance − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
* Superior styles to HTML − CSS have a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* Multiple Device Compatibility − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
* Global web standards − Now HTML attributes are being deprecated and it is being recommended to use CSS. So, it’s a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Syntax:

selector {property: value}



Here table is a selector and border is a property and given value *1px solid #C00* is the value of that property.

Ex:

style{

h1 {

color: #36CFFF;

}

}

Types of Selectors

* [Universal Selectors](https://www.tutorialspoint.com/css/css_selectors.htm#universal_selectors)
* [Element Selectors](https://www.tutorialspoint.com/css/css_selectors.htm#element_selectors)
* [Class Selectors](https://www.tutorialspoint.com/css/css_selectors.htm#class_selectors)
* [Id Selectors](https://www.tutorialspoint.com/css/css_selectors.htm#id_selectors)
* [Attribute Selectors](https://www.tutorialspoint.com/css/css_selectors.htm#attribute_selectors)
* [Group Selectors](https://www.tutorialspoint.com/css/css_selectors.htm#group_selectors)

## CSS Universal Selector

Universal selector, denoted by an asterisk mark (\*), is a special selector that matches all elements in an HTML document. These are generally used to add a same length margin and padding to all the elements in document.

### Syntax

\* {

margin: 0;

padding: 0;

}

As per the above syntax, the universal selector is used to apply a margin and padding of 0 to all HTML elements.

### Example

|  |
| --- |
| <html>  <head>  <style>  \* {  margin: 0;  padding: 0;  background-color: peachpuff;  color: darkgreen;  font-size: 25px;  }  </style>  </head>  <body>  <h1>Universal selector (\*)</h1>  <div>  Parent element  <p>Child paragraph 1</p>  <p>Child paragraph 2</p>  </div>  <p>Paragraph 3</p>  </body>  </html> |

CSS Element Selector

A element selector targets an HTML element, such as [<h1>](https://www.tutorialspoint.com/html/html_hn_tag.htm), [<p>](https://www.tutorialspoint.com/html/html_p_tag.htm), etc. This is used when we want to apply similar style to all the <p> tags or <h1> tags in the document.

Syntax:

p {

color: green;

}

h1 {

text-decoration-line: underline;

}

Example:

|  |
| --- |
| <html>  <head>  <style>  div {  border: 5px inset gold;  width: 300px;  text-align: center;  }  p {  color: green;  }  h1 {  text-decoration-line: underline;  }  </style>  </head>  <body>  <div>  <h1>Element selector</h1>  <p>Div with border </p>  <p>Text aligned to center</p>  <p>Paragraph with green color</p>  <p>h1 with an underline</p>  </div>  </body>  </html> |

CSS Class Selector

A class selector targets an element with a specific value for its [class attribute](https://www.tutorialspoint.com/html/html_class_attribute.htm) to style it. A class in CSS is denoted by "." (period) symbol.

Syntax

.sideDiv {

text-decoration-line: underline;

}

.topDiv {

color: green;

font-size: 25px;

}

Example

Following example demonstrates the use of a class selector, where .style-div, .topDivs and .bottomDivs are class selectors:

|  |
| --- |
| <html>  <head>  <style>  .style-div {  border: 5px inset gold;  width: 300px;  text-align: center;  }  .topDivs{  font-weight: bold;  font-size: 30px;  }  .bottomDivs{  color: green;  font-size: 20px;  }  </style>  </head>  <body>  <div class="style-div">  <div class="topDivs">  Hello World  </div>  <div class="topDivs">  Learn CSS  </div>  <div class="bottomDivs">  From  </div>  <div class="bottomDivs">  K7IT  </div>  </div>  </body>  </html> |

CSS ID Selector

An ID selector targets single element with a particular value for [id attribute](https://www.tutorialspoint.com/html/html_id_attribute.htm) to style it. An id in CSS is denoted by "#" (hash) symbol. Same class can be applied to multiple elements, but an id is unique for an element.

Syntax

#style-p {

color: green;

font-size: 25px;

}

#style-h1 {

text-decoration-line: underline;

color: red;

}

Example

Following example demonstrates the use of an id selector, where #style-div, #tutorial and #stylePoint are the id selectors applied on the elements:

|  |
| --- |
| <html>  <head>  <style>  #style-div {  border: 5px inset gold;  width: 300px;  text-align: center;  }  #tutorial{  color: green;  font-size: 20px;  }  #stylePoint{  color: black;  font-size: 15px;  font-weight: bold;  }  </style>  </head>  <body>  <div id="style-div">  <div id="tutorial">  Tutorials  <span id="stylePoint">  Point  </span>  </div>  <p>  Here we used ids to  style different elements.  </p>  </div>  </body>  </html> |

**CSS Attribute Selector**

An attribute selector targets an element based on a specific attribute or attribute values on an element.

For a detailed explanation of attribute selectors, refer this [attribute selector](https://www.tutorialspoint.com/css/css_attribute_selectors.htm) article.

**Syntax**

/\* Style all anchor tag with target attribute \*/

a[target] {

background-color: peachpuff;

}

/\* Style all anchor tag that links to tutorialspoint \*/

a[href="https://www.tutorialspoint.com"] {

background-color: peachpuff;

}

Example

|  |
| --- |
| <html>  <head>  <style>  a[href]{  font-size: 2em;  }  a[target] {  background-color: peachpuff;  color: blueviolet;  }  /\* Attribute with value have more priority\*/  /\* Hence black background applied to CSS link\*/  a[target="\_self"] {  background-color: black;  }  </style>  </head>  <body>  <h2>Attribute selector</h2>  <p>  Styling applied to anchor element:  </p>  <a href="https://www.k7itech.com/">  Tutorialspoint  </a>  <br><br>  <a href="/html/index.html" target="\_blank">  HTML Tutorial  </a>  <br><br>  <a href="/css/index.htm" target="\_self">  CSS Tutorial  </a>  </body>  </html> |

CSS Group Selector

CSS group selector allow us to apply same style to multiple elements at a time. Name of elements can be separated by commas. This method is recommended as it keep CSS concise and avoid redundancy.

Syntax

/\* Apply same background color for h1 and h2 \*/

h1, h2 {

background-color: grey;

}

Example

|  |
| --- |
| <html>  <head>  <style>  /\* This applies to both <h1> and <h2> elements \*/  h1, h2 {  background-color: grey;  padding: 4px;  }  /\*Applies to all paragraphs, elements with class\*/  /\*'highlight', and element with ID 'hightlightSpan'\*/  p, .highlight, #hightlightSpan {  background-color: yellow;  padding: 10px;  }  </style>  </head>  <body>  <h1>CSS Selectors</h1>  <h2>Group Selectors</h2>  <p>This is a paragraph.</p>  <div class="highlight">  This is div  </div>  <br>  <span id="hightlightSpan">  This is span  </span>  </body>  </html> |

Example 2:

|  |
| --- |
| <html>  <head>     <style>        .style-div {           border: 5px inset rgb(0, 255, 106);           width: 300px;           text-align: center;        }        .topDivs{           font-weight: bold;           font-size: 30px;        }        .bottomDivs{           color: green;           font-size: 20px;        }        #tutorial{           color: rgb(238, 90, 90);           font-size: 20px;        }        #stylePoint{           color: black;           font-size: 15px;           font-weight: bold;        }        p[name]{          color:rgb(255, 0, 17);        }        a[target]{           color:yellow;        }        a[href="https://www.k7itech.com/"]{          color:darkgoldenrod;        }        p,h2{          color:chartreuse;          font-weight: bolder;          background-color: darksalmon;        }     </style>  </head>  <body>     <div class="style-div">           <div class="topDivs">              Hello World           </div>           <div class="topDivs">              Learn CSS           </div>           <div class="bottomDivs">              From           </div>           <div class="bottomDivs">              K7IT           </div>           <div id="tutorial">              Tutorials              <span id="stylePoint">                 Point              </span>           </div>           <p>              Here we used ids to              style different elements.           </p>       </div>     <h2>Attribute selector</h2>     <p name="paragraph">        Styling applied to anchor element:     </p>     <a href="https://www.k7itech.com/" target="new">        Tutorialspoint     </a>     <br><br>     <a href="LoginForm.html" target="\_blank">        HTML Tutorial     </a>     <br><br>     <a href="Register.html" target="\_self">        CSS Tutorial     </a>  </body>  </html> |

**Embedded CSS - The <style> Element**

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside the <head>...</head> tags. Rules defined using this syntax will be applied to all the elements available in the document.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style type = "text/css" media = "all">  body {  background-color: linen;  }  h1 {  color: maroon;  margin-left: 40px;  }  </style>  </head>  <body>  <h1 >This is a heading</h1>  <p>This is a paragraph.</p>  </body>  </html> |

Inline CSS - The *style* Attribute

You can use *style* attribute of any HTML element to define style rules. These rules will be applied to that element only

Syntax:

<element style = "...style rules....">

Example:

|  |
| --- |
| <html>  <head>  </head>  <body>  <h1 style = "color:#36C;">  This is inline CSS  </h1>  </body>  </html> |

**External CSS - The <link> Element**

The <link> element can be used to include an external stylesheet file in your HTML document.

An external style sheet is a separate text file with .css extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

Syntax:

<head>

<link type = "text/css" href = "..." media = "..." />

</head>

Example:

<head>

<link type = "text/css" href = "mystyle.css" media = " all" />

</head>

Imported CSS - @import Rule

@import is used to import an external stylesheet in a manner similar to the <link> element.

Syntax:

<head>

@import "URL";

</head>

Syntax 2:

<head>

@import url("URL");

</head>

Example

<head>

@import "mystyle.css";

</head>

CSS Rules Overriding

We have discussed four ways to include style sheet rules in a HTML document. Here is the rule to override any Style Sheet Rule.

* Any inline style sheet takes highest priority. So, it will override any rule defined in <style>...</style> tags or rules defined in any external style sheet file.
* Any rule defined in <style>...</style> tags will override rules defined in any external style sheet file.
* Any rule defined in external style sheet file takes lowest priority, and rules defined in this file will be applied only when above two rules are not applicable.

**Menu bar Navigation code**

|  |
| --- |
| <html>  <head>  <title>  menu navigation  </title>  <style>  .menu{  background-color: #333;  color: white;  padding: 10px;  text-align: center;  }  .menu a{  color:white;  padding: 10px 20px;  text-decoration: none;  display:inline-block;  }  .menu a:hover{  background-color: #575757;  }  .content{  padding: 20px;  display: none;  }  .active {  display: block; /\* Show the active content section \*/  }  </style>  </head>  <body>  <div class="menu">  <a href="#" onclick="showContent('home')">Home</a>  <a href="#" onclick="showContent('about')">about</a>  <a href="#" onclick="showContent('services')">services</a>  <a href="#" onclick="showContent('contact')">contact</a>  </div>  <div id="home" class="content">  <h2>Welcome to K7IT </h2>  <p>This is K7IT home Page</p>  </div>  <div id="about" class="content">  <h2>About K7IT</h2>  <p>K7 Infotech where innovation meets excellence in software solutions. Established with a passion for harnessing technology to drive business success, we specialize in developing bespoke software that transforms ideas into reality. Our dedicated team of tech enthusiasts and industry experts collaborates seamlessly to deliver cutting-edge applications tailored to meet the unique needs of our clients across various sectors.</p>  </div>  <div id="services" class="content">  <h2>Services of K7IT</h2>  <p id="ui">UI/UX design is more than aesthetics—it's about crafting intuitive interfaces that enhance user satisfaction, streamline interactions, and drive meaningful engagement. By blending creativity with user research and technology, we create seamless digital experiences that delight and inspire.</p>  <h2>Web Apps - JAVA/J2EE</h2>  <p> Web Apps powered by Java/J2EE leverage robust frameworks and libraries to deliver scalable, secure, and enterprise-grade solutions. With Java's versatility and J2EE's robust architecture, these applications ensure high performance, reliability, and seamless integration across diverse platforms, meeting the complex demands.</p>  </div>  <div id="contact" class="content">  <h2>Contact Us</h2>  <p>Please call us or email to us for any queries and our team at K7 Infotech will evaluate your queries and revert back appropriately  #17,LV Nilayam,  3rd cross, Jayanthi Nagar Ext, Horamavu,  Bangalore Pin code -560043  </p>  </div>  </div>  <script>  function showContent(id) {  // Hide all content sections  const contents = document.querySelectorAll('.content');  contents.forEach(content => content.classList.remove('active'));  // Show the selected content section  const selectedContent = document.getElementById(id);  if (selectedContent) {  selectedContent.classList.add('active');  }  }  // Optionally show the home section by default  showContent('home');  </script>  </body>  </html> |

**Menu with dropdown options**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Dropdown Menu Example</title>  <style>  body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  }  .navbar {  overflow: hidden;  background-color: #333;  }  .navbar a {  float: left;  display: block;  color: #f2f2f2;  text-align: center;  padding: 14px 20px;  text-decoration: none;  }  .navbar a:hover {  background-color: #ddd;  color: black;  }  .dropdown {  float: left;  overflow: hidden;  }  .dropdown .dropbtn {  font-size: 16px;  border: none;  outline: none;  color: #f2f2f2;  padding: 14px 20px;  background-color: #333;  cursor: pointer;  }  .dropdown-content {  display: none;  position: absolute;  background-color: #f9f9f9;  min-width: 160px;  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);  z-index: 1;  }  .dropdown-content a {  float: none;  color: black;  padding: 12px 16px;  text-decoration: none;  display: block;  text-align: left;  }  .dropdown-content a:hover {  background-color: #ddd;  }  .show {  display: block;  }  .content {  display: none; /\* Hide all content sections by default \*/  padding: 20px;  margin-top: 50px; /\* Adjust based on header height \*/  border: 1px solid #ddd;  border-radius: 5px;  }  </style>  </head>  <body>  <div class="navbar">  <a href="#home" onclick="showSection('home')">Home</a>  <a href="#about" onclick="showSection('about')">About</a>  <div class="dropdown">  <button class="dropbtn" onclick="toggleDropdown()">Services</button>  <div class="dropdown-content" id="dropdownContent">  <a href="#UI/UX" onclick="showSection('UI/UX')">UI/UX</a>  <a href="#Backend" onclick="showSection('Backend')">Backend</a>  <a href="#Testing" onclick="showSection('Testing')">Testing</a>  </div>  </div>  <a href="#contact" onclick="showSection('contact')">Contact</a>  </div>  <div id="home" class="content">  <h2>Welcome to K7IT</h2>  <p>This is K7IT home Page</p>  </div>  <div id="about" class="content">  <h2>About K7IT</h2>  <p>K7 Infotech where innovation meets excellence in software solutions. Established with a passion for harnessing technology to drive business success, we specialize in developing bespoke software that transforms ideas into reality. Our dedicated team of tech enthusiasts and industry experts collaborates seamlessly to deliver cutting-edge applications tailored to meet the unique needs of our clients across various sectors.</p>  </div>  <div id="UI/UX" class="content">  <h2>Services of K7IT</h2>  <p>UI/UX design is more than aesthetics—it's about crafting intuitive interfaces that enhance user satisfaction, streamline interactions, and drive meaningful engagement. By blending creativity with user research and technology, we create seamless digital experiences that delight and inspire.</p>  </div>  <div id="Backend" class="content">  <h2>Web Apps - JAVA/J2EE</h2>  <p>Web Apps powered by Java/J2EE leverage robust frameworks and libraries to deliver scalable, secure, and enterprise-grade solutions. With Java's versatility and J2EE's robust architecture, these applications ensure high performance, reliability, and seamless integration across diverse platforms, meeting the complex demands.</p>  </div>  <div id="Testing" class="content">  <h2>Manual and Automation</h2>  <p>Testing ensures that software applications function as expected. Our testing services include both manual and automated approaches to ensure high quality and reliability of your software solutions.</p>  </div>  <div id="contact" class="content">  <h2>Contact Us</h2>  <p>Please call us or email us for any queries and our team at K7 Infotech will evaluate your queries and revert back appropriately:</p>  <p>#17, LV Nilayam, 3rd Cross, Jayanthi Nagar Ext, Horamavu, Bangalore Pin code - 560043</p>  </div>  <script>  function toggleDropdown() {  const dropdownContent = document.getElementById('dropdownContent');  if (dropdownContent.classList.contains('show')) {  dropdownContent.classList.remove('show');  } else {  dropdownContent.classList.add('show');  }  }  function showSection(sectionId) {  // Hide all sections  const sections = document.querySelectorAll('.content');  sections.forEach(section => {  section.style.display = 'none';  });  // Show the selected section  const selectedSection = document.getElementById(sectionId);  if (selectedSection) {  selectedSection.style.display = 'block';  }  // Hide the dropdown menu after selection (if necessary)  const dropdownContent = document.getElementById('dropdownContent');  if (dropdownContent.classList.contains('show')) {  dropdownContent.classList.remove('show');  }  }  // Close dropdown if clicked outside  window.onclick = function(event) {  if (!event.target.matches('.dropbtn')) {  const dropdownContent = document.getElementById('dropdownContent');  if (dropdownContent.classList.contains('show')) {  dropdownContent.classList.remove('show');  }  }  }  // Optionally show the home section by default  showSection('home');  </script>  </body>  </html> |

sideNavigation bar

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <title>Side Navigation Menu Example</title>  <style>  body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  scroll-behavior: smooth; /\* Smooth scrolling \*/  }  .sidebar {  height: 100%;  width: 250px; /\* Width of the sidebar \*/  position: fixed; /\* Fixed sidebar \*/  top: 0;  left: 0;  background-color: #333;  padding-top: 20px;  z-index: 1000; /\* Ensures sidebar is above other content \*/  }  .sidebar a {  display: block;  color: #f2f2f2;  padding: 14px 20px;  text-decoration: none;  text-align: left;  }  .sidebar a:hover {  background-color: #575757;  }  .sidebar a.active {  background-color: #4CAF50;  color: white;  }  .content {  margin-left: 270px; /\* Same as the width of the sidebar \*/  padding: 20px;  }  .section {  padding: 20px;  border-bottom: 1px solid #ddd; /\* Border for visibility \*/  min-height: 600px; /\* Added height for scrolling effect \*/  }  </style>  </head>  <body>  <div class="sidebar">  <a href="#home" id="nav-home" class="active">Home</a>  <a href="#about" id="nav-about">About</a>  <a href="#services" id="nav-services">Services</a>  <a href="#contact" id="nav-contact">Contact</a>  </div>  <div class="content">  <div id="home" class="section">  <h1>Home Section</h1>  <p>Welcome to the home section.</p>  </div>  <div id="about" class="section">  <h1>About Section</h1>  <p>Information about us.</p>  </div>  <div id="services" class="section">  <h1>Services Section</h1>  <p>Details of our services.</p>  </div>  <div id="contact" class="section">  <h1>Contact Section</h1>  <p>Contact information.</p>  </div>  </div>  <script>  // Handle click events to update active class  document.querySelectorAll('.sidebar a').forEach(anchor => {  anchor.addEventListener('click', function(e) {  e.preventDefault();  document.querySelector(this.getAttribute('href')).scrollIntoView({  behavior: 'smooth'  });  document.querySelectorAll('.sidebar a').forEach(link => {  link.classList.remove('active');  });  this.classList.add('active');  });  });  </script>  </body>  </html> |

**How to send Email Notification when submit the Contact Us form.**

**1. Set Up Your EmailJS Account**

1. **Create an Account:**
   * Sign up at [EmailJS](https://www.emailjs.com/).
2. **Add an Email Service:**
   * In the EmailJS dashboard, navigate to the **Email Services** section and add a new email service (e.g., Gmail, Outlook).
3. **Create an Email Template:**
   * Go to the **Email Templates** section.
   * Create a template with placeholders for dynamic content. For example:

Subject: {{subject}}

Message: {{message}}

1. **Get Your User ID, Template ID, and Service ID:**
   * Go to the **Integration** section in the dashboard.
   * Copy your **User ID**, **Service ID**, and **Template ID**.

HTML

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Dropdown Menu Example</title>      <link rel="stylesheet" href="menudropdown.css">  </head>  <body>      <div class="video-background">          <video autoplay muted loop id="background-video">              <source src="6774640-uhd\_3840\_2160\_30fps.mp4" type="video/mp4">              Your browser does not support the video tag.          </video>      </div>      <div class="navbar">          <a href="#"><img src="logo.png" alt="Logo" class="logo"></a>          <a href="#home" onclick="showSection('home')">Home</a>          <a href="#about" onclick="showSection('about')">About</a>          <div class="dropdown">              <button class="dropbtn" onclick="toggleDropdown()">Services</button>              <div class="dropdown-content" id="dropdownContent">                  <a href="#UI/UX" onclick="showSection('UI/UX')">UI/UX</a>                  <a href="#Backend" onclick="showSection('Backend')">Backend</a>                  <a href="#Testing" onclick="showSection('Testing')">Testing</a>              </div>          </div>          <a href="#" id="openRegisterModal" class="amodel">Register</a>          <a href="#" id="openLoginModal" class="amodel">Login</a>          <a href="#" id="openContactModal" class="amodel">Contact Us</a>      </div>      <a href="#" id="openRegisterModal" class="amodel">Register</a>      <a href="#" id="openLoginModal"class=amodel>Login</a>  </div>  <div id="registrationModal" class="modal">      <div class="modal-content">          <span id="closeRegisterModal" class="close">&times;</span>          <h2>Registration Form</h2>          <form id="registrationForm">              <label for="name">Name:</label>              <input type="text" id="name" name="name" required>              <label for="email">Email:</label>              <input type="email" id="email" name="email" required>              <label for="password">Password:</label>              <input type="password" id="password" name="password" required>              <button type="submit">Submit</button>          </form>      </div>  </div>   <!-- Modal Structure for Login -->   <div id="loginModal" class="modal">      <div class="modal-content">          <span id="closeLoginModal" class="close">&times;</span>          <h2>Login Form</h2>          <form id="loginForm">              <label for="loginEmail">Email:</label>              <input type="email" id="loginEmail" name="loginEmail" required>              <label for="loginPassword">Password:</label>              <input type="password" id="loginPassword" name="loginPassword" required>              <button type="submit">Login</button>          </form>      </div>  </div>    <!-- Contact Us Modal -->  <div id="contactModal" class="modal">      <div class="modal-content">          <span id="closeContactModal" class="close">&times;</span>          <h2>Contact Us</h2>          <form id="contactForm">              <label for="contactName">Name:</label>              <input type="text" id="contactName" name="contactName" required>              <label for="contactEmail">Email:</label>              <input type="email" id="contactEmail" name="contactEmail" required>              <label for="contactMessage">Message:</label>              <textarea id="contactMessage" name="contactMessage" rows="4" required></textarea>              <button type="submit">Send Message</button>          </form>          <p id="formResponse"></p>      </div>  </div>      <!-- Main Content Sections -->      <div id="home" class="content">          <h2>Welcome to K7IT</h2>          <p>This is K7IT home Page</p>      </div>      <div id="about" class="content">          <h2>About K7IT</h2>          <p>K7 Infotech where innovation meets excellence in software solutions...</p>      </div>      <div id="UI/UX" class="content">          <h2>UI/UX Design</h2>          <p>UI/UX design is more than aesthetics—it's about crafting intuitive interfaces...</p>      </div>      <div id="Backend" class="content">          <h2>Backend Development</h2>          <p>Web Apps powered by Java/J2EE leverage robust frameworks...</p>      </div>      <div id="Testing" class="content">          <h2>Testing Services</h2>          <p>Testing ensures that software applications function as expected...</p>      </div>      <div id="contact" class="content">          <h2>Contact Us</h2>          <p>Please call us or email us for any queries and our team at K7 Infotech will evaluate your queries and revert back appropriately:</p>          <p>#17, LV Nilayam, 3rd Cross, Jayanthi Nagar Ext, Horamavu, Bangalore Pin code - 560043</p>      </div>      <script src="https://cdn.emailjs.com/dist/email.min.js"></script>      <script>          emailjs.init("u9shrl9BEIDZl8ZwT"); // User-id/ publicKey      </script>      <script src="MenuDropDown.js"></script>  </body>  </html> |

# JavaScript

|  |
| --- |
| // Toggle Dropdown  function toggleDropdown() {      const dropdownContent = document.getElementById('dropdownContent');      dropdownContent.classList.toggle('show');  }  function showSection(sectionId) {      // Hide all sections      document.querySelectorAll('.content').forEach(section => {          section.style.display = 'none';      });      // Show the selected section      const selectedSection = document.getElementById(sectionId);      if (selectedSection) {          selectedSection.style.display = 'block';      }      // Hide the dropdown menu      document.getElementById('dropdownContent').classList.remove('show');  }  // Handle Modals  function openModal(modal) {      if (modal) modal.style.display = 'block';  }  function closeModal(modal) {      if (modal) modal.style.display = 'none';  }  // Event listeners for opening modals  document.getElementById('openRegisterModal')?.addEventListener('click', (event) => {      event.preventDefault();      openModal(document.getElementById('registrationModal'));  });  document.getElementById('openLoginModal')?.addEventListener('click', (event) => {      event.preventDefault();      openModal(document.getElementById('loginModal'));  });  document.getElementById('openContactModal')?.addEventListener('click', (event) => {      event.preventDefault();      openModal(document.getElementById('contactModal'));  });  // Event listeners for closing modals  document.getElementById('closeRegisterModal')?.addEventListener('click', () => closeModal(document.getElementById('registrationModal')));  document.getElementById('closeLoginModal')?.addEventListener('click', () => closeModal(document.getElementById('loginModal')));  document.getElementById('closeContactModal')?.addEventListener('click', () => closeModal(document.getElementById('contactModal')));  // Close modals if clicked outside  window.addEventListener('click', (event) => {      if (event.target.classList.contains('modal')) {          closeModal(event.target);      }      if (!event.target.matches('.dropbtn')) {          const dropdownContent = document.getElementById('dropdownContent');          if (dropdownContent && dropdownContent.classList.contains('show')) {              dropdownContent.classList.remove('show');          }      }  });    // Handle Contact Form Submission  document.addEventListener('DOMContentLoaded', () => {      const contactForm = document.getElementById('contactForm');      const formResponse = document.getElementById('formResponse');      contactForm.addEventListener('submit', function(event) {          event.preventDefault();  // Prevent the default form submission          // Send the email using EmailJS          emailjs.sendForm('service\_g1e9nvv', 'template\_oabpgn9', contactForm)              .then(function(response){                  // Success message                  formResponse.textContent = 'Your message has been sent successfully!';                  formResponse.style.color = 'green';                  contactForm.reset();  // Clear the form              }, function(error) {                  formResponse.textContent = 'Sorry, there was an error sending your message.';                  formResponse.style.color = 'red';              });      });  });  // Optionally show the home section by default  showSection('home'); |

CSS

|  |
| --- |
| html,body {              font-family: Arial, sans-serif;              margin: 0;              padding: 0;              height : 100%;              overflow :hidden;          }          .logo {              height: 30px; /\* Adjust height as needed \*/              margin: 0 20px; /\* Adjust margin for spacing \*/              float: left; /\* Ensure the logo is aligned correctly in the navbar \*/          }            .navbar {              overflow: hidden;              background-color: #98d8ea;          }          .navbar a {              float: left;              display: block;              color: #f2f2f2;              text-align: center;              padding: 14px 20px;              text-decoration: none;          }          .navbar a[amodel] {              float: right;              display: block;              color: #f2f2f2;              text-align: center;              padding: 14px 20px;              text-decoration: none;          }          .navbar a:hover {              background-color: #d9e2ea;              color: rgb(18, 75, 110);          }          .dropdown {              float: left;              overflow: hidden;          }          .dropdown .dropbtn {              font-size: 16px;              border: none;              outline: none;              color: #f2f2f2;              padding: 14px 20px;              background-color: #7ab1db;              cursor: pointer;          }          .dropdown-content {              display: none;              position: absolute;              background-color: #f9f9f9;              min-width: 160px;              box-shadow: 0px 8px 16px 0px rgba(14, 3, 3, 0.2);              z-index: 1;          }          .dropdown-content a {              float: none;              color: black;                  : 12px 16px;              text-decoration: none;              display: block;              text-align: left;          }          .dropdown-content a:hover {              background-color: #ddd;          }          .show {              display: block;          }          .content {              display: none; /\* Hide all content sections by default \*/              padding: 20px;              margin-top: 50px; /\* Adjust based on header height \*/              border: 1px solid #ddd;              border-radius: 5px;          }  .video-background {      position: fixed;      top: 0;      left: 0;      width: 100%;      height: 100%;      overflow: hidden; /\* Hide any overflow from the video \*/      z-index: -1; /\* Place the video behind other content \*/  }  #background-video {      width: 100%;      height: 100%;      object-fit: cover; /\* Ensure the video covers the entire area \*/  }  /\* Content styling \*/  .content {      position: absolute;      top: 50%;      left: 50%;      transform: translate(-50%, -50%); /\* Center content horizontally and vertically \*/      color: white;      text-align: center;      padding: 20px;      z-index: 1; /\* Ensure content is above the video \*/      background: rgba(0, 0, 0, 0.5); /\* Optional: Add a semi-transparent background for better text visibility \*/      border-radius: 8px; /\* Optional: Add rounded corners to the background \*/  }  /\* menudropdown.css \*/  /\* Existing styles for navbar and other elements \*/  /\* Modal background \*/  .modal {      display: none; /\* Hidden by default \*/      position: fixed;      z-index: 2; /\* Sit on top of the page \*/      left: 0;      top: 0;      width: 100%;      height: 100%;      overflow: auto; /\* Enable scroll if needed \*/      background-color: rgba(0, 0, 0, 0.4); /\* Black background with opacity \*/  }  /\* Modal content \*/  .modal-content {      background-color: #fefefe;      margin: 15% auto; /\* Centered \*/      padding: 20px;      border: 1px solid #888;      width: 80%; /\* Could be more or less, depending on screen size \*/      max-width: 600px; /\* Maximum width \*/      position: relative;  }  /\* Close button \*/  .close {      color: #aaa;      float: right;      font-size: 28px;      font-weight: bold;  }  .close:hover,  .close:focus {      color: black;      text-decoration: none;      cursor: pointer;  }  /\* Form styles \*/  form {      display: flex;      flex-direction: column;  }  form label {      margin-top: 10px;  }  form input {      padding: 8px;      margin-bottom: 10px;      border: 1px solid #ddd;      border-radius: 4px;  }  form button {      padding: 10px;      background-color: #333;      color: white;      border: none;      border-radius: 4px;      cursor: pointer;  }  form button:hover {      background-color: #555;  }  #contactForm {      display: flex;      flex-direction: column;      max-width: 600px;      margin: auto;  }  #contactForm label {      margin-top: 10px;  }  #contactForm input, #contactForm textarea {      padding: 10px;      margin-top: 5px;      border: 1px solid #ccc;      border-radius: 4px;  }  #contactForm button {      margin-top: 10px;      padding: 10px;      border: none;      background-color: #007bff;      color: white;      border-radius: 4px;      cursor: pointer;  }  #contactForm button:hover {      background-color: #0056b3;  }  #formResponse {      margin-top: 10px;  } |

Output:

|  |
| --- |
| html,body {              font-family: Arial, sans-serif;              margin: 0;              padding: 0;              height : 100%;              overflow :hidden;          }          .logo {              height: 30px; /\* Adjust height as needed \*/              margin: 0 20px; /\* Adjust margin for spacing \*/              float: left; /\* Ensure the logo is aligned correctly in the navbar \*/          }            .navbar {              overflow: hidden;              background-color: #98d8ea;          }          .navbar a {              float: left;              display: block;              color: #f2f2f2;              text-align: center;              padding: 14px 20px;              text-decoration: none;          }          .navbar a[amodel] {              float: right;              display: block;              color: #f2f2f2;              text-align: center;              padding: 14px 20px;              text-decoration: none;          }          .navbar a:hover {              background-color: #d9e2ea;              color: rgb(18, 75, 110);          }          .dropdown {              float: left;              overflow: hidden;          }          .dropdown .dropbtn {              font-size: 16px;              border: none;              outline: none;              color: #f2f2f2;              padding: 14px 20px;              background-color: #7ab1db;              cursor: pointer;          }          .dropdown-content {              display: none;              position: absolute;              background-color: #f9f9f9;              min-width: 160px;              box-shadow: 0px 8px 16px 0px rgba(14, 3, 3, 0.2);              z-index: 1;          }          .dropdown-content a {              float: none;              color: black;                  : 12px 16px;              text-decoration: none;              display: block;              text-align: left;          }          .dropdown-content a:hover {              background-color: #ddd;          }          .show {              display: block;          }          .content {              display: none; /\* Hide all content sections by default \*/              padding: 20px;              margin-top: 50px; /\* Adjust based on header height \*/              border: 1px solid #ddd;              border-radius: 5px;          }  .video-background {      position: fixed;      top: 0;      left: 0;      width: 100%;      height: 100%;      overflow: hidden; /\* Hide any overflow from the video \*/      z-index: -1; /\* Place the video behind other content \*/  }  #background-video {      width: 100%;      height: 100%;      object-fit: cover; /\* Ensure the video covers the entire area \*/  }  /\* Content styling \*/  .content {      position: absolute;      top: 50%;      left: 50%;      transform: translate(-50%, -50%); /\* Center content horizontally and vertically \*/      color: white;      text-align: center;      padding: 20px;      z-index: 1; /\* Ensure content is above the video \*/      background: rgba(0, 0, 0, 0.5); /\* Optional: Add a semi-transparent background for better text visibility \*/      border-radius: 8px; /\* Optional: Add rounded corners to the background \*/  }  /\* menudropdown.css \*/  /\* Existing styles for navbar and other elements \*/  /\* Modal background \*/  .modal {      display: none; /\* Hidden by default \*/      position: fixed;      z-index: 2; /\* Sit on top of the page \*/      left: 0;      top: 0;      width: 100%;      height: 100%;      overflow: auto; /\* Enable scroll if needed \*/      background-color: rgba(0, 0, 0, 0.4); /\* Black background with opacity \*/  }  /\* Modal content \*/  .modal-content {      background-color: #fefefe;      margin: 15% auto; /\* Centered \*/      padding: 20px;      border: 1px solid #888;      width: 80%; /\* Could be more or less, depending on screen size \*/      max-width: 600px; /\* Maximum width \*/      position: relative;  }  /\* Close button \*/  .close {      color: #aaa;      float: right;      font-size: 28px;      font-weight: bold;  }  .close:hover,  .close:focus {      color: black;      text-decoration: none;      cursor: pointer;  }  /\* Form styles \*/  form {      display: flex;      flex-direction: column;  }  form label {      margin-top: 10px;  }  form input {      padding: 8px;      margin-bottom: 10px;      border: 1px solid #ddd;      border-radius: 4px;  }  form button {      padding: 10px;      background-color: #333;      color: white;      border: none;      border-radius: 4px;      cursor: pointer;  }  form button:hover {      background-color: #555;  }  #contactForm {      display: flex;      flex-direction: column;      max-width: 600px;      margin: auto;  }  #contactForm label {      margin-top: 10px;  }  #contactForm input, #contactForm textarea {      padding: 10px;      margin-top: 5px;      border: 1px solid #ccc;      border-radius: 4px;  }  #contactForm button {      margin-top: 10px;      padding: 10px;      border: none;      background-color: #007bff;      color: white;      border-radius: 4px;      cursor: pointer;  }  #contactForm button:hover {      background-color: #0056b3;  }  #formResponse {      margin-top: 10px;  } |

Results:

