# FORM VALIDATION

## A Micro Project Report

**Submitted by**


## UDHAYAN V S

## Reg.no: 99220041056

**B.Tech - CSE,**

**CYBER SECURITY**



**Kalasalingam Academy of Research and Education**

**(Deemed to be University)**

**Anand Nagar, Krishnankoil - 626 126**


**FEBRUARY 2024**

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade

## SCHOOL OF COMPUTING

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## BONAFIDE CERTIFICATE

Bonafide record of the work done by **UDHAYAN V S – 99220041056** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Specialization of the Computer Science and Engineering, during the Academic Year Even Semester (2023-24).

Dr. C. BalaSubramanian
Project Guide
Assistant Professor
Computer Science And Engineering
Kalasalingam Academy of
Research and Education
Krishnan kovil – 626126

Mr. N. R. Sathis Kumar
Faculty Incharge
Assistant Professor
Computer Science And Engineering
Kalasalingam Academy of
Research and Education
Krishnan kovil - 626126

Dr. P. Anitha
Evaluator
Assistant Professor
Computer Science And Engineering
Kalasalingam Academy of
Research and Education
Krishnan Kovil – 626126

# Abstract

Form validation is a crucial aspect of web development to ensure that user-submitted data meets specified criteria before it is sent to the server for further processing. In this project, we implement form validation using HTML, CSS, and JavaScript to enhance user experience and data integrity. The abstract of the implementation is as follows:

HTML Structure: The form is structured using HTML, defining input fields, labels, and buttons. Each input field is associated with specific validation rules through attributes like required, pattern, and maxlength.

CSS Styling: CSS is utilized to style the form elements, providing visual feedback to users. This includes highlighting invalid fields, displaying error messages, and styling valid inputs.

JavaScript Validation Logic: JavaScript is employed to perform client-side validation. These functions check each input against predefined rules, such as required fields, data format (e.g., email, number), and character limits.

# Contents

# Chapter 1 – Development Environment Setup

## 1.1 Desktop Environment Setup – Windows

To set up a desktop environment on windows, you'll need a few tools. A basic setup:

**a. Text Editor or Integrated Development Environment (IDE):**

**Visual Studio Code (VS Code):** A popular and lightweight code editor with excellent support for HTML, CSS, and JavaScript. You can download it from https://code.visualstudio.com

**Extensions Installation in VS Code:**

1. Live Server – to host the local server website

2. Live Preview - Hosts a local server in your workspace for you to preview your webpages on.

**b. Web Browser:** Google Chrome or Mozilla Firefox, both browsers have robust developer tools for inspecting and debugging web pages.

## 1.2 Environment Setup – GitHub

Setting up a development environment with GitHub involves not only the tools you use locally but also the integration of version control and collaboration features provided by GitHub.

**a. GitHub Account:** If you don't have one already, sign up for a GitHub account at https://github.com/

**b. GitHub Repository:** Create a new repository on GitHub by clicking the "New" button on the GitHub website.

Choose a name for your repository, add a description if needed, and configure other settings as required.

Optionally, initialize the repository with a README file, a .gitignore file, and a license.

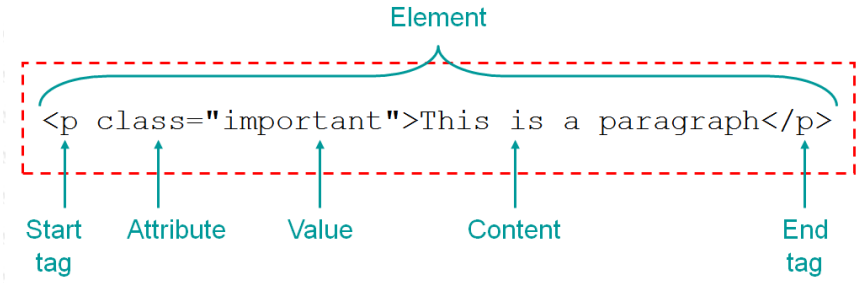
**c. Upload:** Upload a project file or folder in the created Repository.

# Chapter 2 – HTML

## 2.1 Introduction to HTML

1. HTML stands for **Hyper Text Mark Language.**
2. HTML is the standard markup language for creating Web pages.
3. HTML describes the structure of a Web page.
4. HTML consists of a series of elements.
5. HTML elements tell the browser how to display the content.
6. HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

**Syntax:**




## 2.2 History of HTML

Berners-Lee developed the first version of HTML in 1990, primarily as a simple markup language to create and link documents on the web. The first web page, which also served as a tutorial on how to create web pages, was published by Berners-Lee in 1991.

In 1995, HTML 2.0 was published as the first formal specification by the Internet Engineering Task Force (IETF), standardizing many of the elements and attributes used in early web development.

HTML continued to evolve with the introduction of HTML 3.2 and then HTML 4.0, which introduced more sophisticated layout and styling options.

HTML5, the latest major revision of HTML, was introduced in 2008 with the goal of modernizing the language and addressing the needs of contemporary web development.

The HTML5 specification also incorporated features for offline web applications, geolocation, canvas for drawing graphics, and more.

## 2.3 Basic of HTML Tags

HTML tags are the building blocks of HTML documents. They are used to define the structure and content of web pages. Here are some basic HTML tags along with their descriptions:

1. **<!DOCTYPE html>:** This declaration specifies the document type and version of HTML being used.

2. **<html>:** This tag defines the root of an HTML document and contains all other HTML elements.

3. **<head>:** This tag contains meta-information about the document, such as its title, links to stylesheets, and scripts. It's not displayed on the web page itself.

4. **<title>:** This tag sets the title of the document, which appears in the browser's title bar or tab.

5. **<body>:** This tag contains the main content of the document, including text, images, links, etc. Everything that you see on a web page is contained within the body tag.

6. **<h1> to <h6>:** These tags define headings of different levels, with <h1> being the highest level and <h6> being the lowest.

7. **<p>:** This tag defines a paragraph of text.

8. **<a>:** This tag defines a hyperlink, allowing you to link to other web pages or resources.

9. **<img>:** This tag embeds an image into the web page.

10. ** :** This tag defines a division or section in an HTML document, often used for grouping and styling purposes.

11. **<button>:** This tag in HTML is used to create a clickable button on a web page.

12. **<br>:** This tag inserts a line break within text.

13. **<hr>:** This tag inserts a horizontal line, typically used as a thematic break between sections of content.

14. **:** This tag defines text that should be emphasized, often displayed in italics.

15. **<b>:** This tag defines bold text.

16. **<i>:** This tag defines italicized text.

# Chapter 3 CSS

## 3.1 Introduction of CSS

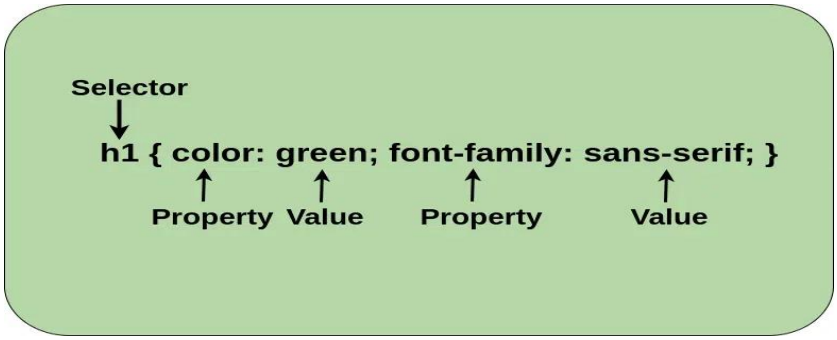
1. CSS stands for **Cascading Style Sheets**.

2. CSS describes how HTML elements are to be displayed on screen, paper, or in other media.

3. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

4. External stylesheets are stored in CSS files.

**Syntax:**




## 3.2 History of CSS

In 1994, Håkon Wium Lie and Bert Bos proposed CSS as part of the proposed HTML standardization process.

**CSS Level 1 (CSS1) -** was officially released in December 1996 as a recommendation by the World Wide Web Consortium (W3C). CSS1 provided basic styling capabilities such as font properties, colors, text alignment, and margins.

**CSS Level 2 (CSS2) -** followed in May 1998, introducing more advanced features such as positioning, floats, and z-index. CSS2 also addressed issues related to accessibility and internationalization.

**CSS Level 3 (CSS3) -** development began in the early 2000s. CSS3 introduced a wide range of new features and enhancements, including advanced selectors, flexible box layout, grid layout, transformations, transitions, animations, and more.

# 3.3 Basic of CSS Elements

**1. Selectors:** Selectors are patterns used to select the HTML elements you want to style. Here are some common types of selectors:

a. **Element Selector**: Selects elements based on their tag name.
b. **Class Selector:** Selects elements with a specific class attribute.
c. **ID Selector:** Selects a single element with a specific ID attribute.
d. **Universal Selector:** Selects all elements on the page. It's denoted by *.

**2. Properties**: CSS properties are the attributes you can apply to selected elements to style them.

Here are some common properties:

a. **color:** Sets the color of text.
b. **background-color:** Sets the background color of an element.
c. **font-family:** Sets the font family for text.
d. **font-size**: Sets the font size for text.
e. **font-weight:** Sets the boldness of text.
f. **text-align:** Sets the alignment of text.
g. **margin:** Sets the margin around an element.
h. **padding:** Sets the padding within an element.
i. **border:** Sets the border around an element.
j. **width:** Sets the width of an element.
k. **height:** Sets the height of an element.

3. **Values:** Values are assigned to CSS properties to define how they should be applied.

Here are some common values:

a. **Color values:** Keywords (e.g., red, blue), hexadecimal codes (e.g., #ff0000), RGB values (e.g., rgb(255, 0, 0)).
b. **Length values:** Pixels (e.g., 10px), percentages (e.g., 50%), em units (e.g., 1em).
c. **Font values:** Font names (e.g., Arial, Times New Roman), generic font families (e.g., serif, sans-serif).
d. **Text alignment values:** left, center, right.

**4. Selectors and Properties Combined:** CSS rules consist of selectors and the properties and values you want to apply to the selected elements

**5. Box Model:** It consists of content, padding, border, and margin. Understanding the box model is crucial for layout design and positioning of elements.

# Chapter 4 JavaScript

## 4.1 Introduction to JavaScript

JavaScript is a versatile programming language primarily used for creating interactive web pages. It is a key component of web development alongside HTML and CSS. Here's an introduction to JavaScript:

1. **Purpose:** JavaScript (JS) was originally created to add interactivity to web pages. It enables developers to manipulate the content and behavior of web pages dynamically, allowing for interactive features such as form validation, animations, and responsive interfaces.

2. **Dynamic and Versatile:** JavaScript is a dynamic language, which means variables can be assigned different data types and values dynamically during runtime. It is also a versatile language that supports various programming paradigms, including procedural, object-oriented, and functional programming.

3. **Libraries and Frameworks**: JavaScript has a vast ecosystem of libraries and frameworks that extend its capabilities and simplify common tasks.

## 4.2 Basic of JavaScript

JavaScript is a versatile programming language primarily used for creating interactive and dynamic web content. Below are some basic concepts and features of JavaScript:

**1. Variables:** Variables are used to store data values. In JavaScript, you can declare variables using the var, let, or const keywords.

**2. Data Types:** JavaScript supports various data types, including numbers, strings, booleans, arrays, objects, and more.

**3. Operators:** JavaScript includes arithmetic, assignment, comparison, logical, and other types of operators for performing operations on variables and values.

**4. Functions:** Functions are blocks of reusable code that perform a specific task. They can take parameters as input and return a value.

**5. Conditional Statements:** JavaScript supports conditional statements such as if, else if, and else, allowing you to execute different blocks of code based on specified conditions.

# 5. Conclusion and Future Work

In conclusion, building a form validation system using HTML, CSS, and JavaScript is a practical project that demonstrates the integration of front-end technologies to enhance user experience and ensure data integrity. For future work on this form validation project, consider the following enhancements:

**Extended Validation Rules:** Implement additional validation rules such as email validation, password strength checking, date format validation, or custom validation rules specific to your application's requirements.

**Error Messaging**: Enhance the error messaging system to provide more informative and user-friendly error messages, guiding users on how to correct invalid input effectively.

**Form Submission Handling:** Implement server-side validation and error handling to complement client-side validation and ensure data integrity and security on the server-side.

# 6. References


1. https://www.coursera.org/learn/html-css-javascript-for-web developers/home/week/1
2. https://www.coursera.org/learn/html-css-javascript-for-web-developers/home/week/2
3. https://www.coursera.org/learn/html-css-javascript-for-web-developers/home/week/3
4. https://www.coursera.org/learn/html-css-javascript-for-web-developers/home/week/4
5. https://www.coursera.org/learn/html-css-javascript-for-web-developers/home/week/5
6. https://www.w3schools.com/
7. https://www.geeksforgeeks.org/
8. https://github.com/Udhayan-V-S/Udhayan-V-S.github.io/tree/main/Project/FORM%20VALIDATION

# 7. Appendix

## HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="image/x-icon" href="img/registration-form.png">
    <link rel="stylesheet" href="style.css">
    <script src="script.js" defer></script>
    <title>Register</title>
</head>
<body>
    <div class="container">
        <form action="" id="form">
            <h1>Register</h1>
            <div class="input-group">
                <label for="username">Username</label>
                <input type="text" id="username" name="username">
                <div class="error"></div>
            </div>
            <div class="input-group">
                <label for="email">Email</label>
                <input type="text" id="email" name="email">
                <div class="error"></div>
            </div>
            <div class="input-group">
```

```
            <label for="password">Password</label>
            <input type="password" id="password" name="password">
            <div class="error"></div>
        </div>
        <div class="input-group">
            <label for="cpassword">Confirm Password</label>
            <input type="password" id="cpassword" name="cpassword">
            <div class="error"></div>
        </div>
        <button type="submit">Register</button>
    </form>
  </div>
</body>
</html>
```

## CSS

```
body{
  background: rgb(227, 227, 227);
  background: linear-gradient(0deg, rgb(2, 25, 26) 0%, rgb(39, 43, 102) 86%);
  background-attachment: fixed;
  margin:0;
  font-family: 'Poppins', sans-serif;
}
#form{
  width:400px;
  margin:20vh auto 0 auto;
  background-color: whitesmoke;
  border-radius: 5px;
  padding:30px;
}
```

```
h1{
    text-align: center;
    color:#402e71;
}

#form button{
    background-color: #402e71;
    color:white;
    border: 1px solid #000000;
    border-radius: 5px;
    padding:10px;
    margin:20px 0px;
    cursor:pointer;
    font-size:20px;
    width:100%;
}
.input-group{
    display:flex;
    flex-direction: column;
    margin-bottom: 15px;
}
.input-group input{
    border-radius: 5px;
    font-size:20px;
    margin-top:5px;
    padding:10px;
    border:1px solid rgb(10, 26, 67) ;
}
.input-group input:focus{
    outline:0;
```

```
}
.input-group .error{
    color:rgb(242, 18, 18);
    font-size:16px;
    margin-top: 5px;
}
.input-group.success input{
    border-color: #0cc477;
}
.input-group.error input{
    border-color:rgb(206, 67, 67);
}
```

## JavaScript

```
const form = document.querySelector('#form')
const username = document.querySelector('#username');
const email = document.querySelector('#email');
const password = document.querySelector('#password');
const cpassword = document.querySelector('#cpassword');

form.addEventListener('submit',(e)=>{
    if(!validateInputs()){
        e.preventDefault();
    }
})
function validateInputs(){
    const usernameVal = username.value.trim()
    const emailVal = email.value.trim();
    const passwordVal = password.value.trim();
    const cpasswordVal = cpassword.value.trim();
```

```
let success = true
if(usernameVal===''){
    success=false;
    setError(username,'Username is required')
}
else{
    setSuccess(username)
}
if(emailVal===''){
    success = false;
    setError(email,'Email is required')
}
else if(!validateEmail(emailVal)){
    success = false;
    setError(email,'Please enter a valid email')
}
else{
    setSuccess(email)
}
if(passwordVal === ''){
    success= false;
    setError(password,'Password is required')
}
else if(passwordVal.length<8){
    success = false;
    setError(password,'Password must be atleast 8 characters long')
}
else{
    setSuccess(password)
}
```

```
    if(cpasswordVal === ''){
        success = false;
        setError(cpassword,'Confirm password is required')
    }
    else if(cpasswordVal!==passwordVal){
        success = false;
        setError(cpassword,'Password does not match')
    }
    else{
        setSuccess(cpassword)
    }
    return success;
}
//element - password, msg- pwd is reqd
function setError(element,message){
    const inputGroup = element.parentElement;
    const errorElement = inputGroup.querySelector('.error')

    errorElement.innerText = message;
    inputGroup.classList.add('error')
    inputGroup.classList.remove('success')
}
function setSuccess(element){
    const inputGroup = element.parentElement;
    const errorElement = inputGroup.querySelector('.error')
    errorElement.innerText = '';
    inputGroup.classList.add('success')
    inputGroup.classList.remove('error')
}
const validateEmail = (email) => {
```

```
    return String(email)
      .toLowerCase()
      .match(
        /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
      );
  };
```

# 8. Certification