HTML Forms and Javascript Validation

HTML Forms

In HTML, forms are used to collect user input.

An HTML form contains form elements.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

```
<form>
  .
  .
  form elements
  .
  .
</form>
```

The <input> Element

The `<input>` element is the most important form element.

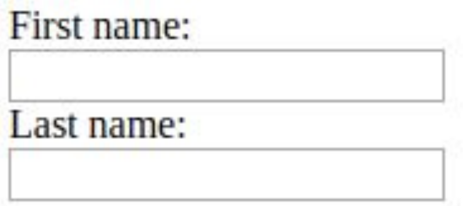
The `<input>` element can be displayed in several ways, depending on the type attribute. Here are some examples:

Type	**Description**
<input type="text">	Defines a one-line text input field
<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)
<input type="checkbox">	Defines a checkbox

Input Type Text

`<input type="text">` defines a one-line text input field:

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

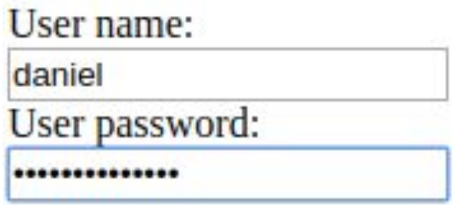


Input Type Password

`<input type="password">` defines a password field:

The characters in a password field are masked (shown as asterisks or circles).

```
<form>
  User name:<br>
  <input type="text" name="username"><br>
  User password:<br>
  <input type="password" name="psw">
</form>
```


User name:

daniel

User password:

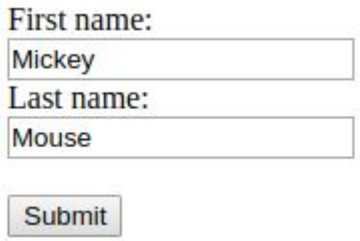
••••••••••••••

Input Type Submit

`<input type="submit">` defines a button for submitting form data to a **form-handler**.

The form-handler is typically a **server page** with a script for processing input data. We will learn this later when we talk about PHP.

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```


First name:

Mickey

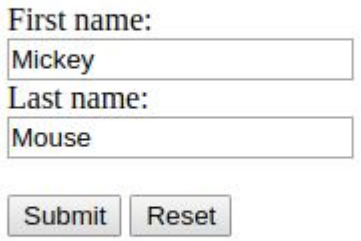
Last name:

Mouse

Input Type Reset

`<input type="reset">` defines a reset button that will reset all form values to their default values:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

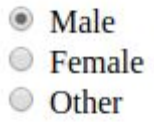

First name:
Mickey
Last name:
Mouse

Input Type Radio

`<input type="radio">` defines a radio button.

Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

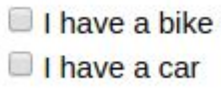


Input Type Checkbox

`<input type="checkbox">` defines a checkbox.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
</form>
```

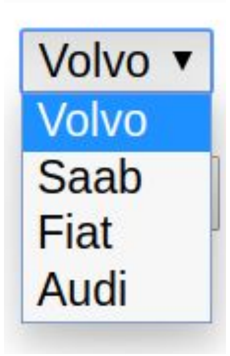


☐ I have a bike
☐ I have a car

The <select> Element

The `<select>` element defines a drop-down list
The `<option>` elements defines an option that can be selected.
By default, the first item in the drop-down list is selected.

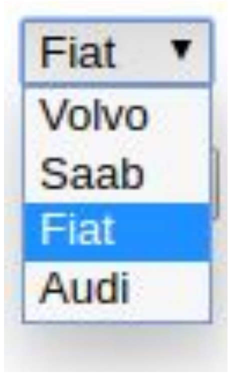
```
<form>
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
</form>
```



The <select> Element

To define a pre-selected option, add the selected attribute to the option

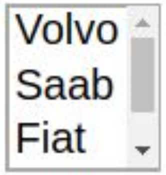
```
<form>
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat" selected>Fiat</option>
    <option value="audi">Audi</option>
  </select>
</form>
```



The <select> Element

Use the multiple attribute to allow the user to select more than one value:
Use the size attribute to specify the number of visible values:

```
<form>
  <select name="cars" size="3" multiple>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
</form>
```



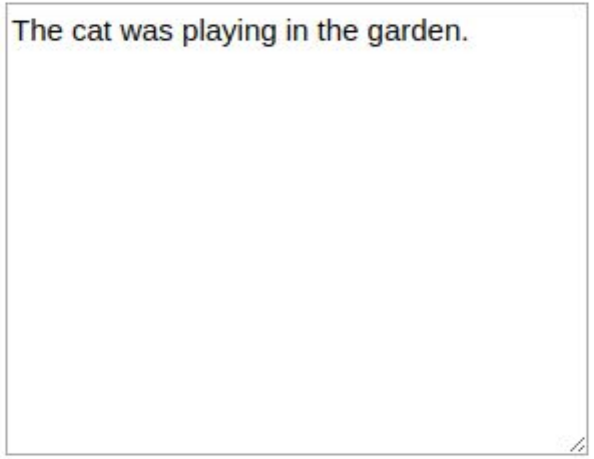
The <textarea> Element

The `<textarea>` element defines a multi-line input field (a text area)

The rows attribute specifies the visible number of lines in a text area.

The cols attribute specifies the visible width of a text area.

```
<form>
  <textarea name="message" rows="10" cols="30">
     The cat was playing in the garden.
  </textarea>
</form>
```


The cat was playing in the garden.

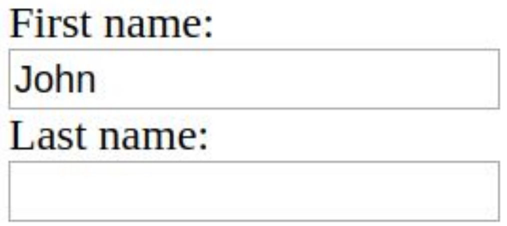
You can also define the size of the text area by using CSS properties width and height

HTML Input Attributes

The value Attribute

The value attribute specifies the initial value for an input field:

```
<form>
  First name:<br>
  <input type="text" name="firstname" value ="John">
  <br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

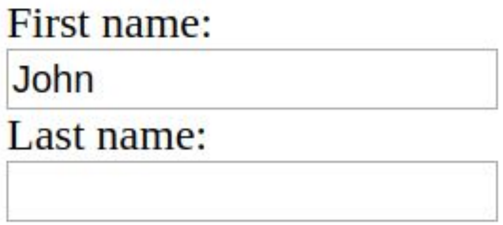


First name:
John
Last name:

The readonly Attribute

The readonly attribute specifies that the input field is read only (cannot be changed):

```
<form>
  First name:<br>
  <input type="text" name="firstname" value ="John" readonly>
  <br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```


First name:

John

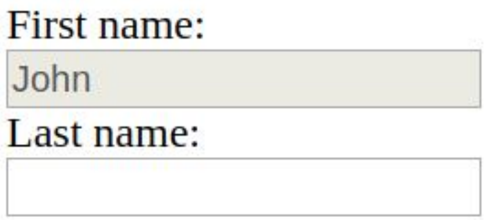
Last name:

The disabled Attribute

The disabled attribute specifies that the input field is disabled.

A disabled input field is unusable and un-clickable, and its value will not be sent when submitting the form

```
<form>
  First name:<br>
  <input type="text" name="firstname" value ="John" disabled>
  <br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```


First name:

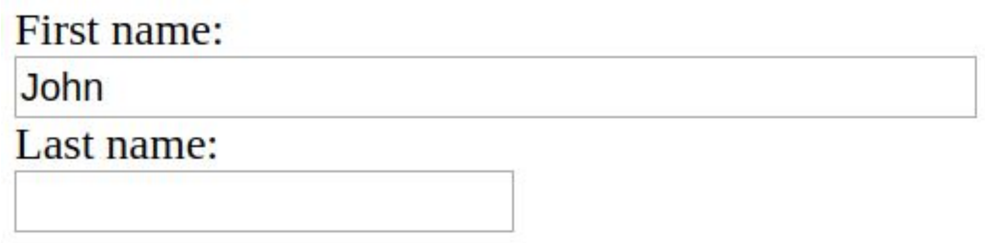
John

Last name:

The size Attribute

The size attribute specifies the size (in characters) for the input field:

```
<form>
  First name:<br>
  <input type="text" name="firstname" value="John" size="40">
  <br>Last name:<br>
  <input type="text" name="lastname">
</form>
```


First name:

John

Last name:

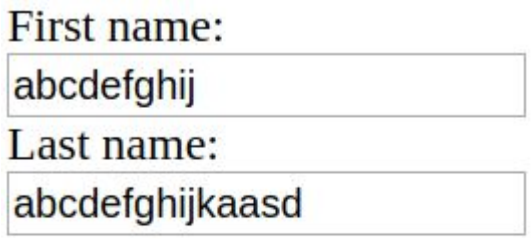
The maxlength Attribute

The maxlength attribute specifies the maximum allowed length for the input field:

With a maxlength attribute, the input field will not accept more than the allowed number of characters.

The maxlength attribute does not provide any feedback. If you want to alert the user, you must write JavaScript code.

```
<form>
  First name:<br>
  <input type="text" name="firstname" maxlength="10">
  <br>Last name:<br>
  <input type="text" name="lastname">
</form>
```



First name:
abcdefghij
Last name:
abcdefghijkaasd

Javascript Form Validation

Form Validation

Form validation is the process of verifying that user input in a form matches the specified rules for that input field.

We already saw how to make sure the maximum length of a text entered in an `<input type="text">` can be controlled using the maxlength attribute

We can do the same thing and a lot more in Javascript.

For example, we can check if the text entered in the email field is a valid email address, or if the user has entered a secure password or not.

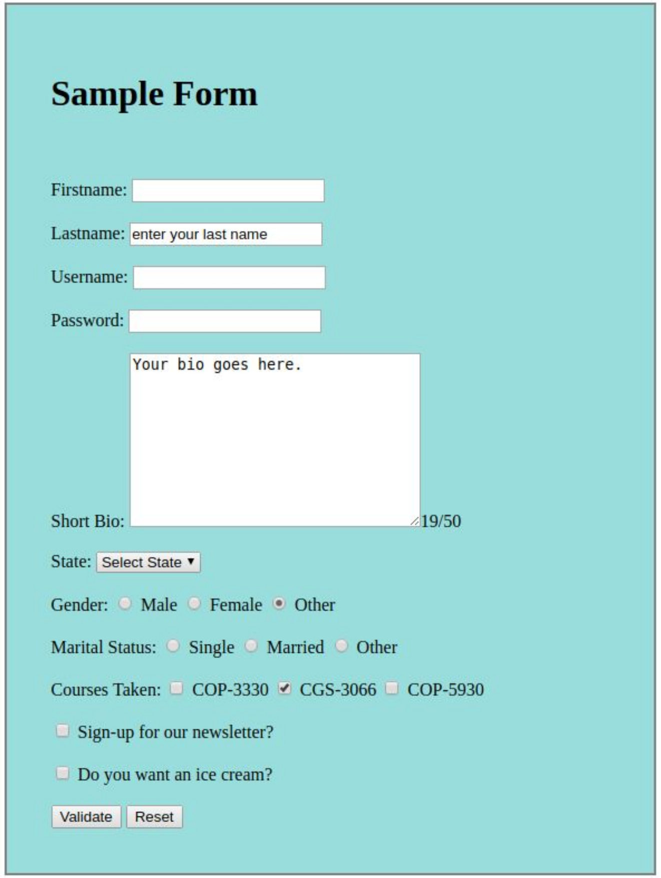
Example

We have this form. Let's implement some validation rules for user input

We want to do this by using javascript

First, lets create a javascript file and link it to our html file.

```
<head>
 <title>Form Validation</title>
 <link rel="stylesheet" href="css/style.css">
 <script src="js/validate.js" defer></script>
</head>
```



Sample Form

Firstname:

Lastname: enter your last name

Username:

Password:

Short Bio: Your bio goes here. 19/50

State: Select State

Gender: Male Female [✓ Other]

Marital Status: Single Married Other

Courses Taken: COP-3330 [✓ CGS-3066] COP-5930

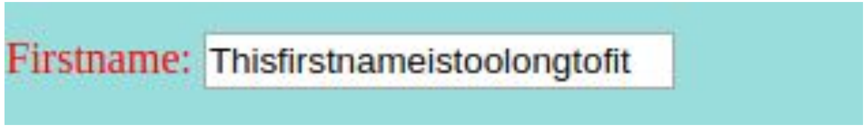
Sign-up for our newsletter?

Do you want an ice cream?

What if a field is not valid?

We have to somehow notify the user, if the input value for a field is not valid

In this example, we will just turn the label for the invalid field to red



In our form, labels are inside <span> elements, so we can just change the style of the corresponding <span> element.

```
<span id="firstname">Firstname: </span><input type="text" name="firstname">
```

What if a field is not valid?

We have talked about several ways to change the style of an HTML element in javascript.

In our CSS file, we have the following rule:

```
.error {
  color:red;
}
```

Therefore, we can just add .error class to the <span> element of the invalid field to make its label, red.

When Should We Validate the Input?

We can validate input fields on different events:

- as the user types in the values
- when the form is submitted
- ...

Here, we will validate all input at once, when the submit button is clicked (form is submitted)

Note that **'submit'** is an **event** that is defined for **form** elements. So we can add an **eventListener** for **'submit' event** to our **form**.

Listen to Submit Event on the Form

In order to listen to 'submit' event on our form, we should add the following lines to our javascript file:

```
const myform = document.querySelector("#myForm");
myform.addEventListener('submit', formSubmitted);
```

Adding these lines, will tell the browser to execute formSubmitted function each time the form is submitted.

Now we can define formSubmitted function and add our validation rules inside it.

Validating Firstname and Lastname

Validation Rules:

- firstname should be non empty and less than 20 characters long
- lastname should be non empty and less than 30 characters long

```
function formSubmitted(event) {
    if(!myform.firstname.value || myform.firstname.value.length > 20)
    {
        document.querySelector("#firstname").classList.add("error");
    }
    if(!myform.lastname.value || myform.lastname.value.length > 30)
    {
        document.querySelector("#lastname").classList.add("error");
    }
}
```

Validating Firstname and Lastname

What did we do?

- created the function to validate fields
- checked the value for firstname to be non-empty and it's length to be < 20 characters long
- checked the value for lastname to be non-empty and it's length to be < 30 characters long
- If any of those fields are invalid, we add class **"error"** to the label associated with the field:

```
document.querySelector("#lastname").classList.add("error");
```

Validating Username

Validation Rules:

- username should be non empty and between 6 - 12 characters long
- username cannot have @ character in it

```
if(!myform.username.value || myform.username.value.length > 12 ||
    myform.username.value.length < 6 || myform.username.value.includes("@"))
    {
        document.querySelector("#username").classList.add("error");
    }
```

Validating Password

Validation Rules:

- password should be at least 6 characters long
- password should have at least 1 uppercase letter and 1 lowercase letter

```
if(!myform.password.value || myform.password.value.length < 6)
    {
        document.querySelector("#password").classList.add("error");
    }
if(myform.password.value == myform.password.value.toUpperCase()
    || myform.password.value == myform.password.value.toLowerCase())
    {
        document.querySelector("#password").classList.add("error");
    }
```

Validating State

Validation Rules:

- User must select a state

Note: the default value for state is 0

```
if(myform.state.value == "0")
    {
        document.querySelector("#state").classList.add("error");
    }
```

Validating Marital Status

Validation Rules:

- If the user selected "married", add Married to the bio field text

```
if(myform.marital.value == "Married")
     {
          myform.bio.value += " Married!";
     }
```

Validating Courses Taken

Validation Rules:

- user should not have taken CGS-3066 before

Note: We have multiple checkbox elements with the name 'course' in our HTML form. Loop through all of them and see which one is selected.

```
for (let i=0; i<myform.course.length; i++) {
    if (myform.course[i].checked && myform.course[i].value == "3066") {
        document.querySelector("#course").classList.add("error");
    }
}
```

Validating Newsletter and Ice Cream!

Validation Rules:

- if the user selects to receive newsletter, alert them about possible spam!
- User cannot receive newsletter and ice cream at the same time!

```
if(myform.news.checked) {
    alert("We will spam you!");

    if(myform.icecream.checked) {
        myform.news.checked = false;
        alert("You can't have ice cream and get our newsletter!");
    }
}
```

Great Job!

We have implemented a set of validation rules for our form in javascript.

We learned to do several things:

- dynamically change the style of html elements based on validation results
- alert the user about validation results
- dynamically change form element values depending on validation results

But, what if we wanted to save this information to a file or a database on our server?

For that, we need a server-side programming language.

Next time, we will start talking about PHP.