

## 12.4 HTML Forms

---

How do forms work?

The pages you've created so far have been great. However, the visitor could only passively receive the information you give — they couldn't share anything themselves.

Adding forms to your code allows the user to submit information to your server. For example, take this form:

### **Tell me about yourself:**

What's your name?

Which High School Musical was the best?

- ✓ High School Musical
- High School Musical 2
- High School Musical 3: Senior Year
- High School Musical: The Musical: The Series

Once Roger presses submit, the form immediately sends the server each of the input names alongside Roger's chosen values. To make sure the server can differentiate the various types of information, it's sent out from the browser in **name-value** pairs. Here's what it could look like:

NAME  
username=Roger  
VALUE

*\*Note: Don't change the name of a form control unless you're absolutely confident the server will be able to understand the change.*

So what happens to this data? Well, the server can then either process it using another programming language, like Java or PHP, or store it in a database somewhere.

On Roger's side, he receives a server-generated message based on the value he chose.

**Thanks, Roger!**

**You voted for High School Musical**

### Adding forms

The main benefit of using forms is that they allow you to group individual input elements together. This way, instead of having to submit each input to the server separately, you can do it all at once.

*\*Note: You can use JavaScript to submit separate inputs programmatically, as you'll see later on in this course. However, until then, HTML forms will be the main tool for forwarding user data to the server.*

As you saw in the video, forms can be created with a simple `<form>` element, within which all future inputs will be added. There are two main attributes that come along with it: `action` and `method`.

## action attribute

The action attribute is mandatory for the `<form>` element to work.

*\*Note: Technically, instead of the `action` attribute, you can also use a `formaction` attribute on the button itself. But for now, let's stick to the first one.*

The `action` attribute specifies the URL for the server which receives the submitted information from the form.

```
<form action="/action_page.php">  
<h1> This is a form! </h1>  
</form>
```

TRY IT OUT

### method attribute

Another required addition to your form element is the `method` attribute. It specifies the HTTP method for transmitting your submitted data. There are two ways of doing so: GET and POST.

#### GET

**Description:** The form values are added to the end of `action` attribute's URL. You don't need to specify it as it's the default method. For example, the code snippet above uses the GET method.

**When to use:** When you're getting something from the server (rather than sending it forward to a database).

#### POST

**Description:** The form values are sent through HTTP methods.

**When to use:** If your form has a file upload function, contains confidential data, or you wish to send the information to a database.

**Task:** Go back to our code editor (you can use the *Try it Out* button below). Change the form's method to POST.

TRY IT OUT

## 12.5 Basic Inputs

---

The text here says it's a form. But try running the code. There's nothing there! How can it be a form if we don't have any inputs?

```
<form action="/action_page.php">
<h1> This is a form! </h1>
</form>
```

TRY IT OUT

Let's change that, shall we?

`<input>`

To add an interactive element to your form, an `<input>` tag is used, alongside a mandatory `type` attribute. The `type` attribute specifies what kind of data your HTML form can receive from the user. These are the values it can take in HTML5:

button	checkbox	color	date	datetime-local	email
file	hidden	image	month	number	password
radio	range	reset	search	submit	tel
text	time	url	week		

However, you don't need to remember all of these. In fact, here's a list of the main types you'll probably use most often:

Type	Definition
text	Allows typing a line of text (usually under 32 characters)
date	Provides access to a calendar to choose a date
email	Allows typing an email address and validates it
checkbox	Allows selecting multiple options from a predefined set
radio	Allows selecting a single option from a predefined set
submit	Creates a submit button needed to send the input to the server

The `<input>` element does not have any content and therefore is an empty element, i.e. does not need a closing tag.

*\*Note: If you forget to specify the `type` attribute, the form element will automatically set it to text.*

### Name-value pair

It's important to remember that, for the server to know which input the user added what information to, forms use name-value pairs. For example, in login forms, you wouldn't want your server mixing up the password with the email, would you?

That's why each input should also have a `name` attribute. Whatever the user adds to the input — be it their name, favorite sushi place, or file upload — automatically becomes the value.

*\*Note: The `name` attribute you set will not appear in the browser screen.*



### id attribute

The `id` attribute helps you give an identifying name to your element. It must be unique to the element. We discussed it when learning about anchoring, however, it will keep coming back — so make sure you remember it!

*\*Note: ID syntax is case-sensitive and must not contain any whitespaces.*

### Basic type attribute values

If a `type` attribute has a text value, a single-line text input is added into your code.

```
<input type="text" name="Name" id="name">
```

TRY IT OUT

With the `password` value, the browser generates a textbox similar to the previous one but with automatically hidden characters. If you want, you can also specify `maxlength`, i.e. how many characters the user may enter. This will allow you to make sure your forms are not flooded with almost infinite characters.

```
<input type="password" name="password" maxlength="30">
```

TRY IT OUT

*\*Note: The `maxLength` attribute is, however, easily removable with dev tools, so precaution should also be taken at the back end.*

If you want to collect emails, you would use the `email` value. This way, the browser will check if the user provided the correct email format (i.e. @emailengine).

```
<input type="email" name="email" >
```

TRY IT OUT

Checkbox values allow your page visitors to check one or multiple answers to a question. It works great for multiple choice questions or agreeing to Terms and Conditions.

```
<p>Please select your favorite flower out of the following:  
  <br>  
  <input type="checkbox" name="flower"  
value="rose" > Rose  
<input type="checkbox" name="flower"  
value="tulip"> Tulip  
<input type="checkbox" name="flower"  
value="orchid"> Orchid </p>
```

TRY IT OUT

Finally, you need to make sure your user's efforts are not in vain and that their information is sent forward to the server. To submit all the form elements, we add the **submit** button. The text you set in the **value** attribute will appear on the button.

*\*Note: Unlike the other type values, the submit button does not require a name. However, you can still add one.*

```
<input type="submit" name="submit"  
value="Submit answers">
```

TRY IT OUT

How the submit button is displayed strongly varies on the browser used. If you wish to control its appearance, you can use CSS to style it or, alternatively, upload an image of a button.