When ordering products online,

you fill out your address and your credit card number,

then you click on an Order button

that confirms your order.

Because you are learning

more about front-end development,

you might be wondering what exactly

happens when you click on that order button.

In this video, you will learn what happens

when forms are submitted in the web browser.

By now, you know that the web browser communicates with

a web server using a HTTP request response cycle.

This means that the web browser

sends requests to the web server,

and the web server sends back a response.

Up to now, the main type

of requests you've been introduced

to were for resources such as HTML documents,

images, CSS files, and JavaScript files.

But it is also possible to

send data as part of a request.

This is how form send data to the web server.

In fact, there are two ways a form

can send data to the web server using

the HTTP GET method or the HTTP POST method.

You can specify which method the form should

use with the method attribute of the form element.

Let's examine an example form and see how

the data can be sent using the two different methods.

This login form, excepts a username and password.

It also has a login button that

submits the form to the web server.

How would it work if the method attribute is set to GET?

When the Login button is clicked,

the form data is sent as part of the request URL.

This means that the user data is appended to

the end of the URL in the web browser navigation bar.

The web server receives the HTTP GET request,

and extracts the form data from the URL.

While this is an easy way to submit data,

it has three key problems.

First, the length of a URL is limited to around

2,000 characters depending on

the web browser you're using.

Some web browsers allow more,

but there are inconsistencies between browsers.

So if you have a large form,

some data may be lost when it

sent via the GET Method to the web server.

Second, the length of a requested URL

is also limited on some web servers.

Popular web server software such

as Apache web server or Engine X,

have a default limit of around 4,096 characters.

Again, if you have a large form,

some data may be lost.

The final problem is security.

Since the data is included as part of the URL,

it means that your data is stored in

your web browser history and

possibly in the request logs on the web server.

If you're transmitting personal information

such as addresses or credit card numbers,

this is a major privacy and security risk.

Now, let's focus on how the form would

work if we set the method attribute to post.

When the form is submitted using the post method,

the form data is inserted into

the content of the HTTP request.

When the submit button is pressed,

it will send an HTTP post request

with the data contained in the body of the request.

The HTTP post method is more

secure than the HTTP GET method.

However, the data could still be read by

a third party listening to the HTTP request.

To secure this completely,

HTTPS is used to encrypt the data

so that only the sender and

receiver can understand the data.

Once the web server receives the request,

it processes it and sends back an HTTP response.

If the request was successful,

the response will direct

the web browser to a new webpage.

If errors occurred, there are many ways this can be

handled by the webpage as explained in a previous video.

You should now be familiar with how

forms are sent to web servers,

and the difference between GET and POST.

You've been asked to help develop a new online store. Security is very important to the customer. Which form method should be used to improve security?



Post



Get

Correct

That's right. The HTTP *POST* method is more secure than the HTTP *GET* method. When a form is submitted using the Post method, the form data is inserted into the content of the HTTP request instead of appended at the end of the URL as is done with the *GET* method.