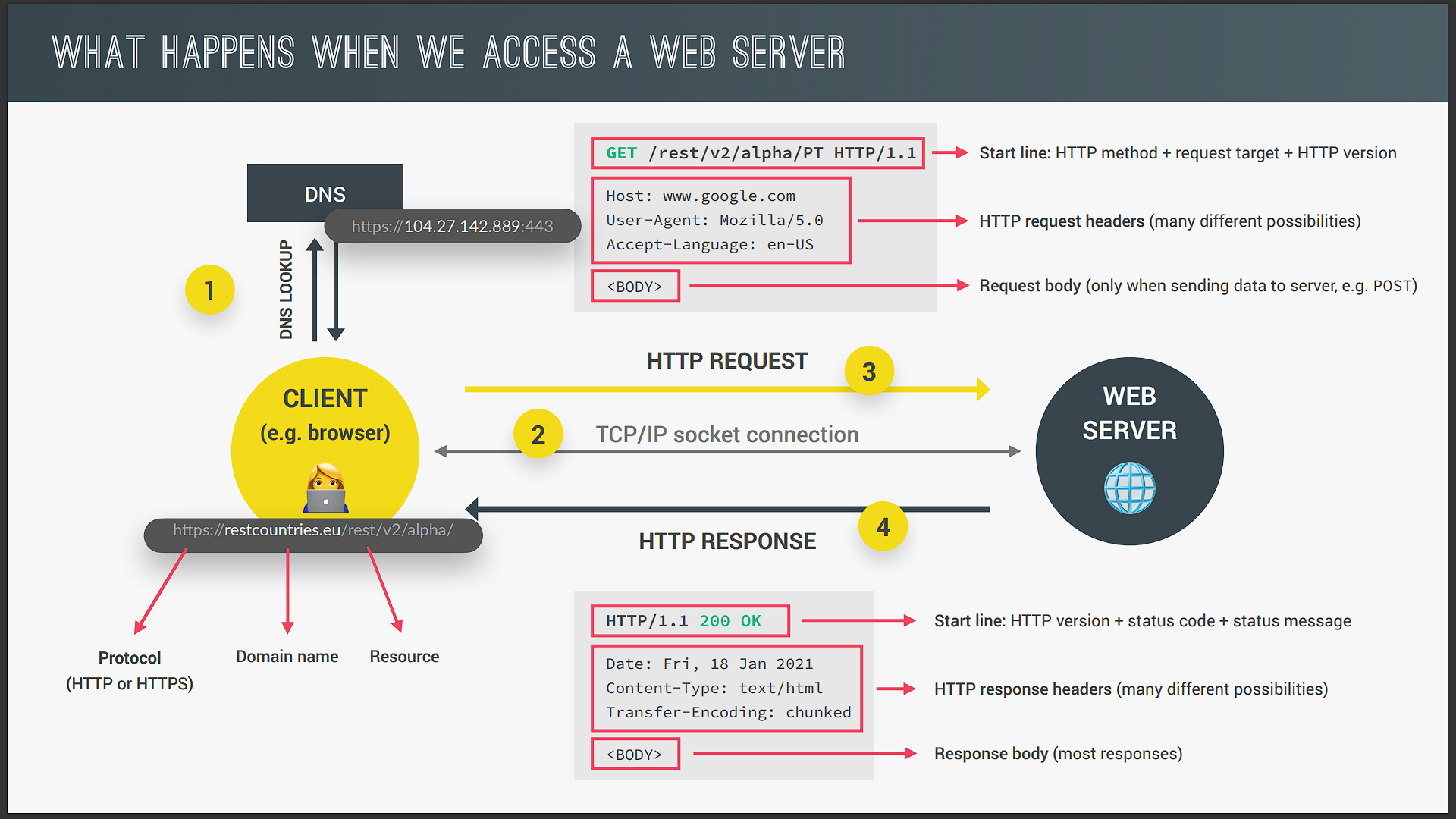


So this is the diagram that we already saw when we first talked about AJAX calls. So just to recap whenever we try to access a Web server the browser which is the client sends a request to the server and the server will then send back a response and that response contains the data or the Web page that we requested. And that's right this process works the exact same way no matter if we're accessing an entire Web page or just some data from a Web API. And this whole process actually has a name and it's called the Request-response model or also the Client-server architecture. But anyway let's now dive a bit deeper into this.

So let's use the example of the URL that we actually accessed in the last video to get our country data. Now every URL gets an HTTP or HTTPS which is for the protocol that will be used on this connection. And we're gonna talk about this a bit later in the video. Then we have the domain name which is restcountries.eu in this case. And also after a slash we have to socalled resource that we want to access. And in this case that's /rest/V and so on. Now this domain name restcountries.eu is actually not the real address of the server that we're trying to access. It's really just a nice name that is easy for us to memorize. But what this means is that we need a way of kind of converting the domain name to the real address of the server. And that happens through a socalled DNS. So DNS stands for domain name server and domain name servers are a special kind of server. So they are basically like the phone books of the Internet.



So the first step that happens when we access any Web server is that the browser makes a request to a DNS and this special server will then simply match the web address of the URL to the server's real IP address all right. And actually this all happens through your Internet service provider but the complete details don't really matter here. What you need to retain from this first part is that the domain is not the real address and that a DNS will convert the domain to the real IP address. And then after the real IP address has been sent back to the browser we can finally call it. So this is how the real address looks like. So it still has the protocol but then comes the IP address. And also the port that we access on the server. And this port number is really just to identify a specific service that's running on a server. So you can think of it like a sub address okay. This port number has nothing to do with the /rest/V resource that we want to access. So that resource will actually be sent over in the HTTP request as we will see in a moment.

Okay and that actually wraps up the first step. So once we have the real IP address a TCP socket connection is established between the browser and the server. And so they are now finally connected. And this connection is typically kept alive for the entire time that it takes to transfer all files of the Website or all data. Now what are TCP and IP? Well TCP is the Transmission Control Protocol. And IP is the Internet Protocol. And together they are communication protocols that define exactly how data travels across the Web. They are basically the Internet's fundamental control system because again they are the ones who set the rules about how data moves on the Internet. And don't worry if that sounds confusing we are actually gonna learn a bit more about TCP/IP a bit later in this video.

But anyway now it's time to finally make our request. And the request that we make is an HTTP request where HTTP stands for Hypertext Transfer Protocol. So after TCP/IP HTTP is another communication protocol. And by the way a communication protocol is simply a system of rules that allows two or more parties to communicate. Now in the case of HTTP it's just a protocol that allows clients and Web servers to communicate. And that works by sending requests and response messages from client to server and back. Now a request message will look something like this. The beginning of the message is the most important part called the start line. And this one contains the HTTP method that is used in the request then the request target and the HTTP version. So about the HTTP methods there are many available but the most important ones are; GET for simply requesting data POST for sending data and PUT and PATCH to basically modify data. So you'll see that an HTTP request to a server is not only for getting data but we can also send data now right. Now about the request target. This is where the server is told that we want to access the rest/V/alpha resource in this case remember that? So we had this in the URL before and now it is simply sent as the target in the HTTP request. And so then the server can figure out what to do with it. Now if this target was empty so if it was just a slash basically then we would be accessing the website's route which is just restcountries.eu in this example. Then the next part of the request are the request headers which is just some information that we sent about the request itself. There are tons of standard different headers like what browser is used to make the request at what time the user's language and many many more.

Now finally in the case we're sending data to the server. There will also be a request body and that body will contain the data that we're sending for example coming from an HTML form. So that is the HTTP request. And I hope that it makes sense to you. Now of course it's not us developers who manually write these HTTP requests but it's still helpful and valuable that you understand what an HTTP request and also a response look like.

Also I want to mention that there's also HTTPS as you probably know. And the main difference between HTTP and HTTPS is that HTTPS is encrypted using TLS or SSL which are yet some are protocols but I'm not gonna bore you with these. But besides that the logic behind HTTP requests and responses still applies to HTTPS alright? So our request is formed and now it hits the server which will then be working on it until it has our data or Web page ready to send back. And once it's ready it will send it back using as you can guess an HTTP response. And the HTTP response message actually looks quite similar to the request. So also with a start line headers and a body. Now in this case the start line has besides the version also a status code and a message. And these are used to let the client know whether the request has been successful or failed. For example means okay. And the status code that everyone knows is which means page not found. So that is where this code that everyone already knew comes from okay. Then the response headers are information about the response itself. So just like before and there are a ton available and we can also make up our own actually.

And finally the last part of the response is again the body which is present in most responses and this body usually contains the JSON data coming back from an API or the HTML of the Web page that we requested or something like that. So we talked in great detail about the most important parts here which are the HTTP request and the response. But in our imaginary example we only just did one request to restcountries.eu and got one response back right? And that's how it's gonna work when all we do is to access an API. However if it's a Web page that we're accessing then there will be many more requests and responses. And that's because when we do the first request all we get back is just the initial HTML file. That HTML file will then get scanned by the browser for all the assets that it needs in order to build the entire Web page like JavaScript CSS files image files or other assets. And then for each different file there will be a new HTTP request made to the server. So basically this entire back and forth between client and server happens for every single file that is included in the Web page.

However there can be multiple requests and responses happening at the same time but the amount is still limited because otherwise the connection would start to slow down. But anyway when all the files have finally arrived then the Web page can be rendered in the browser according to the HTML CSS and JavaScript specifications that you already know. Now as a final piece of the puzzle let's talk again about TCP/IP and figure out how this request and response data is actually sent across the Web okay.

So we said before that TCP and IP are the communication protocols that define how data travels across the Web. Now I'm not gonna go into a lot of detail here but here is what you need to know. So first the job of TCP is to break the requests and responses down into thousands of small chunks called packets before they are sent. Once the small packets arrive at their final destination TCP will reassemble all the packets into the original request or response. And this is necessary so that each packet can take a different route through the Internet because this way the message arrives at the destination as quick as possible which would not be possible if we sent the entire data simply as a big chunk. So that would be like trying to go through dense traffic with like the biggest bus that you can imagine. So probably not a good idea. Now as a second part the job of the IP protocol is to actually send and route these packets through the Internet. So it ensures that they arrive at the destination they should go using IP addresses on each packet.

Okay and that's it. That's a broad overview of what really happens behind the scenes of the Web. And I hope that you found this information useful and also interesting and not all too confusing. But anyway now let's go back to our JavaScript and AJAX calls.