STUDENT NAME: FAVOUR OLUWASEGUN EYITAYO

COURSE: FRONT-END 201

TUTOR: MR JOSHUA E

ASSIGNMENT QUESTIONS;

1. Read and write about the HTTP Status code
2. Difference between HTTP and HTTPS
3. Read and write about HTTP request methods

**A**.

**HTTP Status Code**

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack.

The HTTP status codes tell you what 's happening when browsers try to contact your website. They show when things go right and when things go wrong.

Therefore, they mean a lot for SEO (Search Engine Optimization) and Google.

An HTTP status code is a message a website 's server sends to the browser to indicate whether or not that request can be fulfilled. Status codes are embedded in the HTTP header of a page to tell the browser the result of its request.

**The different status codes and what they say**

There are five ranges of HTTP status codes. Each range defines where the error was encountered and the number defines what the actual error was.

Here are the 5 ranges and what they mean:

1xx: Informational

2xx: Success!

3xx: Redirect. The requested page has moved somewhere else.

4xx: Client error. There 's something wrong with the way the browser asked for the page.

5xx: Server error. Something went wrong with the way the server tried to send the page.

**HTTP Status Codes Important for SEO**

Obviously, all status codes are important "you should know how healthy your site is" but there are certain ones that particularly important for SEO and anyone working on a website.

**200: Success**

Seeing pages with a 200 status is the outcome you 're hoping for. Servers return 200 status codes, or any code within the 2xx range, when things are working as intended. This means the server, browser and visitor are all happy.

**301: Permanent redirect**

A server returns a 301 HTTP response when the requested URL has moved permanently to a new URL. If a user tries to visit the old URL, it will return a 301 HTTP status, pointing the browser to the new URL. If you move a page without adding a 301 redirect, users trying to visit the old URL will see a 404 error. Plus, using a 301 HTTP status will pass full link juice to the final URL.

**302: Found**

The 302 HTTP status tells the browser that the requested page has been found, but exists at a different URL. The browser then requests that updated URL. Since this status is a bit ambiguous, it 's better to use 301 redirects when you permanently move a page. 302 redirects do still pass full link juice, though.

**404: Not Found**

The most well-known of the 4xx status codes, servers return a 404 error when the browser requests a URL that the server can 't find. These responses can be pretty bad from a user experience because they frustrate people trying to get to a desired page. From an SEO perspective, a site with lots of 404 errors tells Google that it possibly isn 't maintained very well and won't offer users a good experience.

Monitor your site 's 404 errors via Google Search Console and aim to minimize the number of errors.

What impact does your 404 page have on SEO?

Avoid 404 errors by redirecting pages when you move them, keeping your links up to date and making sure deleted pages return a 410 HTTP status.

Since it 's impossible to prevent humans from ever reaching a page that returns a 404 HTTP status, make the most of it by creating a custom 404 page.

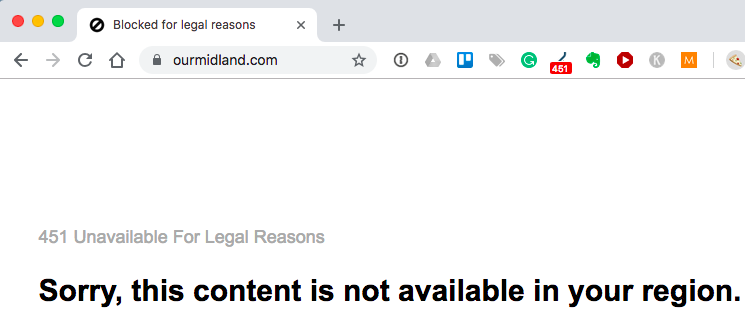
**410: Gone**

Servers return a 410 HTTP status when the URL requested by a browser has been removed. For a human user, the result is the same as a 404. They 'll see the designated error page.

For search engines, however, a 410 says that the page has been deleted and they shouldn 't index that URL.

Before you remove a page, ask yourself if you can add a 301 redirect to a different, relevant page

elsewhere on your site.



**500: Internal server error**

Servers return a 500 HTTP status when the browser makes a valid request, but an internal error in the server prevents it from returning the page. Search engines dislike this error code because it 's very generic" they don 't know what 's supposed to be happening.

500 errors are usually caused by errors in code or a database. If your website returns a lot of 500 errors, you must fix these issues ASAP.

**503: Service unavailable**

Servers return the 503 HTTP status if they are unavailable when the browser makes the request.

Use 503 responses during planned server maintenance periods. A 503 tells a search engine that, even though they can 't access anything now, if they come back in a little bit chances are it will work.

**Managing HTTP Status Codes**

The HTTP status codes of your pages is a big part of SEO, and for search engines in general. Knowing when and how HTTP statuses occur is key in maximizing user experience and minimizing the number of errors people and Google see.

For instance, knowing how to properly delete a page using a 410 vs. when to use a 301 redirect is vital in maintaining the health of your website in Google 's eyes.

To check the HTTP status codes that your website generates, log in to your Google Search Console account. Then check the Coverage report to see how many URLs Google encounters that return error codes.

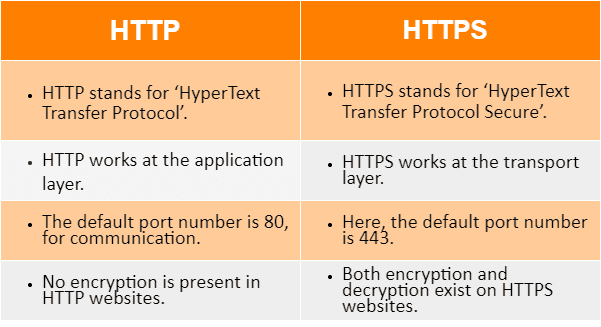
Monitor your site 's crawl errors encountered by Google.

These errors must be fixed to ensure that your website is being indexed correctly by search engines and navigated by users.

B.

**Different between HTTP and HTTPS**

HTTPS is HTTP with encryption and verification. The only difference between the two protocols is that HTTPS uses TLS (SSL) to encrypt normal HTTP requests and responses, and to digitally sign those requests and responses. As a result, HTTPS is far more secure than HTTP. A website that uses HTTP has http:// in its URL, while a website that uses HTTPS has <https://.>



**C**.

**HTTP Request Methods – What are HTTP Requests?**

**HTTP Request Methods**

The internet boasts a vast array of resources hosted on different servers. For you to access these resources, your browser needs to be able to send a request to the servers and display the resources for you. HTTP (Hypertext Transfer Protocol), is the underlying format that is used to structure request and responses for effective communication between a client and a server. The message that is sent by a client to a server is what is known as an HTTP request. When these requests are being sent, clients can use various methods.

Therefore, HTTP request methods are the assets that indicate the specific desired action to be performed on a given resource. Each method implements a distinct semantic, but there are some standard features shared by the various HTTP request methods.

**What Are HTTP Request Methods?**

An HTTP request is an action to be performed on a resource identified by a given Request-URL. Request methods are case-sensitive, and should always be noted in upper case. There are various HTTP request methods, but each one is assigned a specific purpose.

**How Do HTTP Requests Work?**

HTTP requests work as the intermediary transportation method between a client/application and a server. The client submits an HTTP request to the server, and after internalizing the message, the server sends back a response. The response contains status information about the request.

**What Are the Various Types of HTTP Request Methods?**

**GET**

GET is used to retrieve and request data from a specified resource in a server. GET is one of the most popular HTTP request techniques. In simple words, the GET method is used to retrieve whatever information is identified by the Request-URL.

Read more about GET.

**HEAD**

The HEAD technique requests a reaction that is similar to that of GET request, but doesn’t have a message-body in the response. The HEAD request method is useful in recovering meta-data that is written according to the headers, without transferring the entire content. The technique is commonly used when testing hypertext links for accessibility, validity, and recent modification.

**POST**

Another popular HTTP request method is POST. In web communication, POST requests are utilized to send data to a server to create or update a resource. The information submitted to the server with POST request method is archived in the request body of the HTTP request. The HTTP POST method is often used to send user-generated data to a server. One example is when a user uploads a profile photo.

Read more about POST.

**PUT**

PUT is similar to POST as it is used to send data to the server to create or update a resource. The difference between the two is that PUT requests are idempotent. This means that if you call the same PUT requests multiple times, the results will always be the same.

**DELETE**

Just as it sounds, the DELETE request method is used to delete resources indicated by a specific URL. Making a DELETE request will remove the targeted resource.

**PATCH**

A PATCH request is similar to POST and PUT. However, its primary purpose is to apply partial modifications to the resource. And just like a POST request, the PATCH request is also non-idempotent. Additionally, unlike POST and PUT which require a full user entity, with PATCH requests, you may only send the updated username.

Read more about PATCH.

**TRACE**

TRACE requests are used to invoke a remote, application loop-back test along the path to the target resource. The TRACE method allows clients to view whatever message is being received at the other end of the request chain so that they can use the information for testing or diagnostic functions.

**CONNECT**

The CONNECT request method is used by the client to create a network connection to a web server over a particular HTTP. A good example is SSL tunneling. In a nutshell, CONNECT request establishes a tunnel to the server identified by a specific URL.

**HTTP PUT vs POST Request Methods**

Both PUT and POST request methods are used to facilitate data transmission between a client and a server, and despite having a similar role of sending data to create and update resources, they have their subtle differences as shown in the following table.

## HTTP PUT vs POST Request Methods

Both PUT and POST request methods are used to facilitate data transmission between a client and a server, and despite having a similar role of sending data to create and update resources, they have their subtle differences as shown in the following table.

|  |  |
| --- | --- |
| PUT | POST |
| It is idempotent, meaning that putting a resource twice will have no effect | It is not idempotent, and thus calling a POST request repeatedly is discouraged |
| Identity is selected by the client | Identity is returned by the server |
| Operates as specific | Operates as abstract |

## HTTP GET vs POST

Now that we have an idea of what GET and POST requests are, let’s compare the two;

|  |  |
| --- | --- |
| GET | POST |
| Parameters in this method are saved in the browser’s history | Parameters are not archived in the browser history or other web server logs |
| Can be bookmarked | Cannot be bookmarked |
| Features a restriction on data length. This is because the GET method adds data to the URL for it to be sent, and  we know the maximum URL length is 2048 characters | There are no restriction on data length |
| There is no impact when you hit the reload/back button. | Should you hit the reload/back button, sent data will be resubmitted |
| Has restriction on data type as the only allowed data type is ASCII characters | There is no restriction on data type, and binary data is also allowed |
| Information is visible to everyone in the URL | Information is not displayed in the URL thus not visible to everyone |
| Can be [cached](https://rapidapi.com/blog/api-glossary/cache/) | Can’t be cached |