**1. Write a blog about objects and its internal representation in JavaScript**

Objects are one of the data types in JS. Objects stores key-value pairs.

The object values may contain data types such as numbers, strings, Booleans, arrays.

The Objects variables don’t actually store the value instead objects are reference data types which means they contains the address of the memory that is allocated to the value.

* **Creating objects in JS:-**

let santosh\_personal\_info =

{Name: “Santosh Kumar Singh”, Age:25, Gender: “Male”, E-mail: “[xxxxxxxxx@gmail.com](mailto:xxxxxxxxx@gmail.com)”, Contact\_details: [1111111, 22222222222]};

* **Adding the items in JS Objects:-**

To add a new item to object you will need to use:

santosh\_personal\_info.Address = “Jamshedpur, Jharkhand”;

Output:

{Name: “Santosh Kumar Singh”, Age:25, Gender: “Male”, E-mail: “[xxxxxxxxx@gmail.com](mailto:xxxxxxxxx@gmail.com)”, Contact\_details: [1111111, 22222222222], Address = “Jamshedpur, Jharkhand”};

* **Deleting the items in JS Objects:-**

To delete an item inside object you will need to use:

delete santosh\_personal\_info.Age;

Output:

{Name: “Santosh Kumar Singh”, Gender: “Male”, E-mail: “[xxxxxxxxx@gmail.com](mailto:xxxxxxxxx@gmail.com)”, Contact\_details: [1111111, 22222222222], Address = “Jamshedpur, Jharkhand”};

* **Accessing the items in JS:-**

1. If you want to access name:

santosh\_personal\_info[“Name”] or santosh\_personal\_info.Name

Output:

Santosh Kumar Singh

2. If you want to access Contact\_details:

santosh\_personal\_info[“Contact\_details”] or santosh\_personal\_info.Contact\_details

Output:

[1111111, 22222222222]

* **If you want to view all the keys at once:-**

To view all the keys as an array of strings at once:

Object.keys(variable\_name\_of\_object)

Output:

[“Name”, “Gender”, “E-mail”, “Contact\_details”, “Address”]

* **If you want to view all the values at once:-**

To view all the values as an array at once:

Object.values(variable\_name\_of\_object)

Output:

[ “Santosh Kumar Singh”, “Male”, ”[xxxxxxxxx@gmail.com](mailto:xxxxxxxxx@gmail.com)”, [1111111, 22222222222], “Jamshedpur, Jharkhand”]

**2. Write a blog on Difference between HTTP1.1 vs HTTP2**

* HTTP stands for hypertext transfer protocol & it is used in client-server communication for request-response.
* Before exchanging request-response there is a TCP connection established between the client-server.
* HTTP is an application layer protocol.

**HTTP 1.1**

* It was released back in 1997.
* Status codes were first introduced with http1.1.
* Encryption is a choice.
* It also allowed the connection to be persistent with pipelining, which means for every subsequent request the connection need not to be re-established instead the same connection could be used to serve multiple request-response.
* Uses compression technique to decrease the size of the file.
* Came up with the concept of caching.

**Problems/Drawbacks: -**

* HTTP/1.1 transfer all the requests & responses in the plain text message form.
* The request-response cycle can only be carried out one at a time which means that until all the response is given to an ongoing request, new requests are not being acknowledged. (This is a.k.a head of line blocking)

However, later for the same version queuing was introduced which allowed all the request to be queued & then server sends the response at once for all the queued request. To overcome the HOL blocking parallel connection could be used, but the number of parallel TCP connection which could be established with http1.1 was 6.

* Only compresses the data not the header. So header was as additional burden everytime.

**HTTP 2**

* All data gets converted to binary (0’s & 1’s) to process data as opposed to plain text.
* Security was primary focus with http2 as everything is encrypted. (TLS/SSL).
* Significant performance improvement also came with http2 in terms of load time reduction.
* It is fully compatible with http1.1.

**Features:**

1. **Stream priority**

single/same TCP connection is used for recieving multiple rsponse. when a [client](https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/) makes a request for a webpage, the server sends data all at once, instead of sending one thing after another. This method of data delivery is known as multiplexing. Multiplexing resolves the head-of-line blocking issue in HTTP/1.1 by ensuring that no message has to wait for another to finish.

1. **Multiplexing**

The binary framing layer organizes messages into parallel streams of data. When a client sends requests to a server, it can prioritize the responses it is requesting by assigning a weight between 1 and 256 to each stream. The higher number indicates higher priority. This allows developers to decide which page resources will load first.

1. **Server push:**

Typically, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpages. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. This capability allows the server to send additional cacheable information to the client that isn’t requested but is anticipated in future requests.

1. **Header compression:**

A common method of optimizing web applications is to use compression algorithms to reduce the size of HTTP messages that travel between the client and the server. Small files load more quickly than large ones.

HTTP/2 uses HPACK compression to shrink the size of headers. HTTP/2 can split headers from their data, resulting in a header frame and a data frame which then gets compressed.

**Summary with a daily life example: -**

Modern website’s function much more like applications, with a constant two-way stream of data being an essential part of their functionality.

For example, when you’re typing something in a Google Doc, every keystroke sends data to Google’s servers. Google’s servers process that data, and then send updates back to your browser with the text you’ve typed, along with other helpful information like suggestions, the last edit status of the document, and much more.

Over HTTP/1.1, each of your keypresses would initiate a new connection to the server, to send each character you typed over the wire. That’s a boatload of connections which is time consuming.

With HTTP/2 though, it’s essentially a constant two-way stream over a single connection. Google’s server is always listening for data coming from your browser, and your browser is always listening for data to come back from Google. There’s no more send data, wait for response, update the screen, send more data, wait for a response, etc. Instead, everything happens in real-time. In this way, a web “page” like a Google Doc can update itself so frequently as to feel like a native application on your computer.