**DAY-1**

**What is HTTP/1.1**

The first usable version of HTTP was created in 1997. Because it went through several stages of development, this first version of HTTP was called HTTP/1.1. This version is still in use on the web.

**What is HTTP/2**

In 2015, a new version of HTTP called HTTP/2 was created. HTTP/2 solves several problems that the creators of HTTP/1.1 did not anticipate. In particular, HTTP/2 is much faster and more efficient than HTTP/1.1. One of the ways in which HTTP/2 is faster is in how it prioritizes content during the loading process.

* HTTP/2 is a **binary protocol**, as opposed to HTTP 1.1 that is plain text. The latter is meant to be human readable (for example sniffing network traffic) meanwhile the former is not. More info in the official FAQ [question](https://http2.github.io/faq/).
* **h2** is HTTP/2 over TLS (protocol negotiation via ALPN).
* A **frame** is the smallest unit of communication within an HTTP/2 connection, consisting of a header and a variable-length sequence of octets structured according to the frame type.
* A **stream** is a bidirectional flow of frames within the HTTP/2 connection. The correspondent concept in HTTP 1.1 is a request/response message exchange.

**Difference between HTTP/1.1 VS HTTP/2**

**HTTP/1.1**

**Multiplexing:**

* HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it.

**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, which often involve several dozen separate resources that the client must request.

**Header compression:**

* Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller.

**HTTP/2**

**Multiplexing:**

* HTTP/2 is able to use a single [TCP](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) connection to send multiple streams of data at once so that no one resource blocks any other resource. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.

**Server push:**

* HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. The server also sends a message letting the client know what pushed content to expect – like if Bob had sent Alice a Table of Contents of his novel before sending the whole thing.

**Header compression:**

* HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading.

**Conclusion:**

In conclusion, the transition from HTTP/1.1 to HTTP/2 represents a significant advancement in web protocol offering several key improvements in performance, efficiency, and functionality.

HTTP/2 introduces multiplexing, header compression, binary protocol framing, server push, stream prioritized these enhancements collectively reduce latency, minimize overhead, and improve resource utilization, result While HTTP/2 adoption requires support from both clients and servers and may entail compatibility challenge making it the preferred choice for modern web applications aiming to deliver optimized performance.

**2) Objects and its internal representation in Javascript ?**

* Objects are fundamental constructs that play a crucial role in data organization, manipulation, and interaction.
* They serve as the building blocks for structuring data and behavior in Javascript applications, offering a powerful.
* An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method. Objects in JavaScript, just as in many other programming languages, can be compared to objects in real life.
* Objects, in JavaScript, is it’s most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).
* In simple terms. “A JavaScript object is a collection of named values having state and behavior (properties and method)”.

For example: Person, car, pen, bike, Personal Computer, Washing Machine etc.

**Creating Objects in Javascript:**

* By object literal
* By creating instance of Objects directly(using new keyword)
* **Internal Representation in Javascript:**

Internally, javascript engines use various techniques to represent objects efficiently.

* **Inline Catching**

An inline function in JavaScript is a function defined within the scope of another function and created at runtime. It allows for the definition of functions that can be used on-the-fly without the need for a separate function declaration.

* **Hidden Class**

Hidden classes are classes that cannot be used directly by the bytecode or other classes. Even though it's mentioned as a class, it should be understood to mean either a hidden class or interface. It can also be defined as a member of the access control nest and can be unloaded independently of other classes.

* **Partical Usecase of object in JS**

An object is a data type that can take in collections of key-value pairs. A major difference between an object and other data types such as strings and numbers in JavaScript is that objects can store different types of data as its values.

* // object const student = { firstName: 'ram', class: 10 };
* const person = { name: 'John', age: 20 };
* let person = { name: 'John', age: 20 };