**DAY 1: Assignment**

**01 .Objects And Its Internal Representation In JavaScript**

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).

# ****Objects and properties****

A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. You access the properties of an object with a simple dot-notation:

*objectName.propertyName*

Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive. You can define a property by assigning it a value. For example, let’s create an object named myCar and give it properties named make, model, and year as follows:

*var myCar = new Object();  
myCar.make = 'Ford';  
myCar.model = 'Mustang';  
myCar.year = 1969;*

# ****Creating Objects In JavaScript :****

# Create JavaScript Object with Object Literal

One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces as shown below

*let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};*

# Create JavaScript Object with Constructor

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below

*function Vehicle(name, maker) {  
this.name = name;  
this.maker = maker;  
}  
let car1 = new Vehicle(’Fiesta’, 'Ford’);  
let car2 = new Vehicle(’Santa Fe’, 'Hyundai’)  
console.log(car1.name); //Output: Fiesta  
console.log(car2.name); //Output: Santa Fe*

# Using the object.create method

Objects can also be created using the [Object.create()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/create" \t "_blank) method. This method can be very useful, because it allows you to choose the prototype object for the object you want to create, without having to define a constructor function.

*// Animal properties and method encapsulation  
var Animal = {  
 type: 'Invertebrates', // Default value of properties  
 displayType: function() { // Method which will display type of Animal  
 console.log(this.type);  
 }  
};  
// Create new animal type called animal1   
var animal1 = Object.create(Animal);  
animal1.displayType(); // Output:Invertebrates  
// Create new animal type called Fishes  
var fish = Object.create(Animal);  
fish.type = 'Fishes';  
fish.displayType(); // Output:Fishes*

## 02 HTTP1 Vs. HTTP2

* HTTP/1.1 has been around for more than a decade. With Google’s SPDY leading the way in 2015, the IETF (Internet Engineering Task Force) gave us HTTP/2, which introduces several features to reduce page load times. Let’s compare HTTP2 Vs. HTTP1.1 in detail.

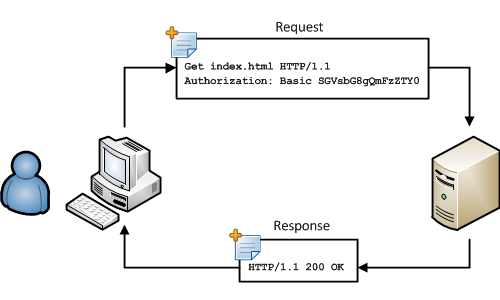
HTTP/2 achieves faster webpage loading without performance optimizations that require extensive human efforts in terms of development. It significantly reduces the complexities that had crept into HTTP/1.1 and gives us a robust protocol which, though not without its flaws, will perhaps stand the test of time. Before making this leap forward, let’s trace our steps back to when the internet was in its infancy to understand how the different versions evolved into the current form.

## **The Beginnings of HTTP & The Internet**

Our story begins in 1969, with a program called Advanced Research Projects Agency Network (ARPANET). ARPANET used packet switching and allowed multiple computers to communicate with each other on a single network. However, this was just a by-product. The original intention behind ARPANET was to design a time-sharing system that allowed research institutes to share their computer resources for effective utilization of processing power.

## **Evolution of HTTP**

HTTP (Hypertext Transfer Protocol) is a set of rules that runs on top of the TCP/IP suite of protocols and defines how files are to be transferred between clients and servers on the world wide web.



## **Key Features of HTTP/1.0:**

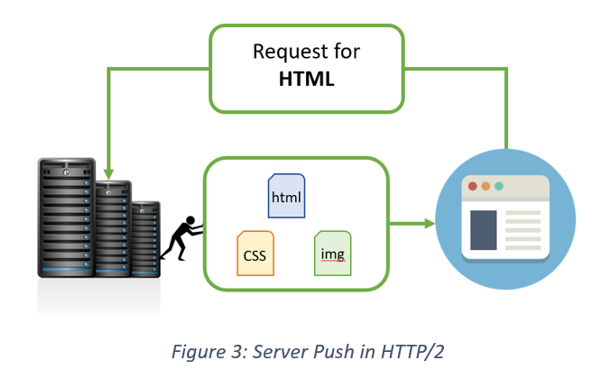
* The concept of headers both for requests (from the client machine) as well as responses (from servers) was introduced. The use of headers such as GET, POST, HEAD added extended flexibility, none of which was possible with the earlier version.
* Version information was now included.
* It allowed a single request/response for every TCP connection.
* Status codes were used to indicate successful requests and to indicate transmission errors.
* The content-type header made it possible to send files other than plain HTML, including scripts and media.

## **Key Features of HTTP/1.1:**

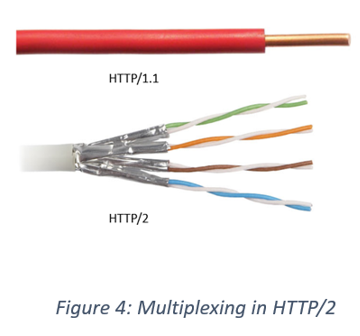
* It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it was possible to have persistent connections. It allowed multiple requests/responses per TCP connection.
* The Upgrade header was used to indicate a preference from the client that made it possible to switch to a more preferred protocol if found appropriate by the server.
* HTTP/1.1 provided support for chunk transfers that allowed streaming of content dynamically as chunks and for additional headers to be sent after the message body. This enhancement was particularly useful in cases where values of a field remained unknown until the content had been produced. For example, when the content had to be digitally signed, it was not possible to do so before the entire content gets generated.
* Other features that reinforced its stability were introduced such as:
  + pipelining (the second request is sent before the response to the first is adequately served)
  + content negotiation (an exchange between client and server to determine the media type, it also provides the provision to serve different versions of a resource at the same URI)
  + cache control (used to specify caching policies in both requests and responses)

## **Key Features of HTTP/2:**

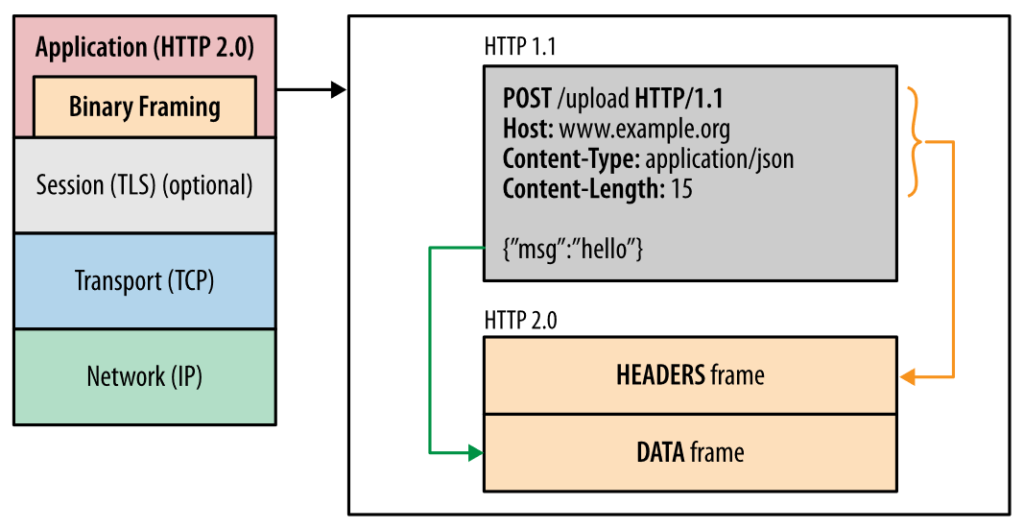
* It introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny the server push; however, in most cases, this feature adds a lot of efficiency to the process.

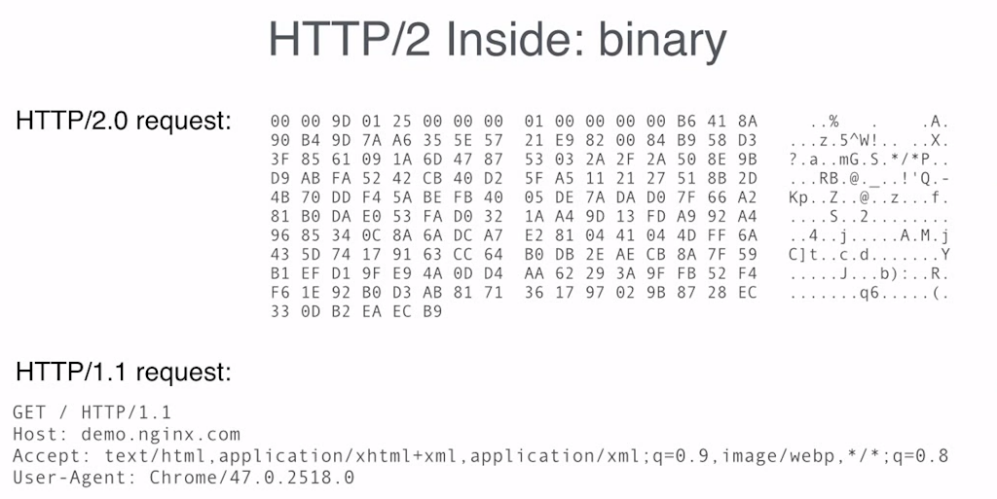


* Introduces the concept of multiplexing that interleaves the requests and responses without head-of-line blocking and does so over a single TCP connection.



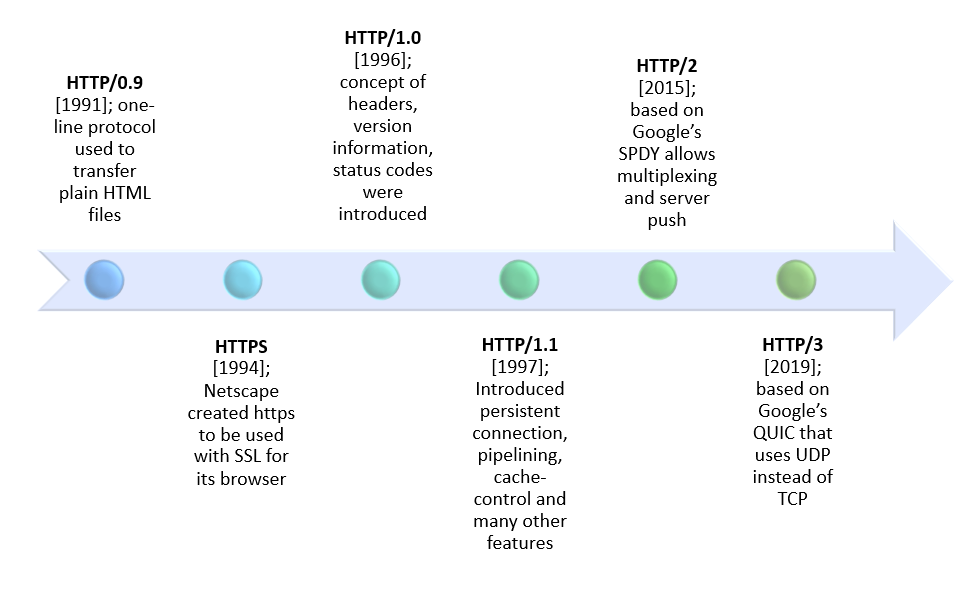
* It is a binary protocol i.e. only binary commands in the form of 0s and 1s are transmitted over the wire. The binary framing layer divides the message into frames that are segregated based on their type – Data or Header. This feature greatly increases efficiency in terms of security, compression and multiplexing.





* HTTP/2 uses HPACK header compression algorithm that is resilient to attacks like CRIME and utilizes static Huffman encoding.

HTTP/3, the next version in the series, is based on Google’s QUIC which, unlike its precursors is a drastic shift to UDP. Given the gradual adoption rate of HTTP/2, HTTP/3 with its security challenges (that comes into play the moment we switch from TCP to UDP) is expected to face some difficulties.



## **HTTP/1.x vs HTTP/2: A Comparative Study**

HTTP2 Vs. HTTP1 is not a debate at all. HTTP2 is much faster and more reliable than HTTP1. HTTP1 loads a single request for every TCP connection, while HTTP2 avoids network delay by using multiplexing.

HTTP is a network delay sensitive protocol in the sense that if there is less network delay, then the page loads faster. However, an impressive increase in network bandwidth only slightly improves page load time. This is key to understanding the differences in performance efficiencies between the different versions of HTTP. Back in the day when people used dial up modems web pages were simple and it was the actual data transfer between the server and the client that contributed towards the largest chunk of the page load time. Today the actual downloading of resources from server takes a negligible portion of the total page load time due to the tremendous increase in bandwidth availability. It is the time taken to establish the TCP connection and making requests that impacts performance. It was initially recommended to use only two connections per hostname but today most browsers use six connections per hostname. When we talk about http vs http2 in terms of performance it is important to note that a lot of performance optimizations adopted by HTTP/1.1 introduced complexities in terms of developmental efforts as well as network congestion that HTTP/2 attempts to address.

|  |  |  |  |
| --- | --- | --- | --- |
| Differentiator | HTTP/1.0 | HTTP/1.1 | HTTP/2 |
| Year | 1991 | 1997 | 2015 |
| Key Features | For every TCP connection there is only one request and one response. HTTP1 Protocol | It supports connection reuse i.e. for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution. HTTP2 Protocol | Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested. HTTP3 Protocol |
| Status Code | Can define 16 status codes; the error prompt is not specific enough. | Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient. | Underlying semantics of HTTP such as headers, status codes remains the same. |
| Authentication Mechanism | Uses basic authentication scheme which is unsafe since username and passwords are transmitted in clear text or base64 encoded. | It is relatively secure since it uses digest authentication, NTLM authentication. | Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate\_Security. |
| Caching | Provides support for caching via the If-Modified-Since header. | Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags. | HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream. |
| Web Traffic | HTTP/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. However, TCP starts slowly and with domain sharding (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion. | | HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays. |