|  |  |  |
| --- | --- | --- |
| **S.No** | ***HTTP1.1*** | ***HTTP2*** |
| 1 | http1.1 is the first version of http. | http2 is the new version of http, which is faster and more efficient than http1.1 |
| 2 | Every request and response contains headers, and these headers are not compressed, leading t increased overhead in terms of data transfer | It uses header compression, which reduces overhead by efficiently compressing these headers, resulting in reduced bandwidth usage and faster load times. |
| 3 | Doesn't support server push. The client needs to make individual requests for each resource, even if they are predictable. | Introduces server push, allowing the server to proactively send resources to the client that it predicts the client will request, reducing the need for additional round trips. |
| 4 | This uses textual format(ASCII) for messages, which can be human readable but less efficient for machines | Employs a binary framing layer for its protocol, increasing efficiency in parsing and reducing errors, but sacrificing human readability. |
| 5 | Doesn't have built-in flow control mechanisms, which can lead to congestion and performance issues in case of varying bandwidth. | implements flow control at the protocol level, enabling better management of data streams and preventing overwhelming the receiver. |
| 6 | This loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. | It is able to use a single TCP(Transmission control Protocol) connection to send multiple streams of data at once so that no one resource blocks any other resource. |
| 7 | Requires multiple connections for parallel downloads, leading to increased latency. | Facilitates connection reuse allowing multiple requests and responses to be sent over a single connection, reducing latency and improving performance |
| 8 | Errors and retries van be inefficient due to limitations in multiplexing and pipelining. | Optimizes error handling by separating frames within a stream, enabling more efficient retries and error recovery. |
| 9 | Simpler to implement due to its straight forward nature and widespread use. | More complex due to its binary framing, multiplexing, prioritization, and other advanced features. |
| 10 | Fully backward compatible with older HTTP version | Designed with backward compatibility in mind but requires negotiation between client and server to upgrade from HTTP1.1 |

*Question:1*

*Blog on Diff Between HTTP1.1 and HTTP2:-*

*Question:2*

*Blog on Objects and its internal representation in javascript:-*

1. 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.
2. Internal properties  
     
   They are called “internal”, because they are not directly accessible via the language, but they do influence its behavior. Internal properties have special names that are written in double square brackets. Two examples: The internal property [[Prototype]] points to the prototype of an object.
3. Property values can be values of any type, including other objects, which enables building complex data structures. There are two types of object properties: The data property and the accessor property. Each property has corresponding attributes.
4. Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
5. objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

For Eg. If your object is a student, it will have properties like name, age, address, id, etc and methods like updateName, UpdateAddress, etc.

1. 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

1. Unassigned properties of an object are [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) (and not [null](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null)).

For Example:

Var Car = {

Brand:”KIA”,

Colour:”Red”,

Milage:14

}

Console.log(Car.seater); // Undefined.

1. Properties of JavaScript objects can also be accessed or set using a bracket notation (for more details see [property accessors](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_Accessors)). Objects are sometimes called associative arrays, since each property is associated with a string value that can be used to access it. So, for example, you could access the properties of the myCar object as follows:

myCar['make'] = 'Ford';  
myCar['model'] = 'Mustang';  
myCar['year'] = 1969;

1. The following example also creates a new JavaScript object with four properties:Example: var person = new Object();  
   person.firstName = “John”;  
   person.lastName = “Doe”;  
   person.age = 50;  
   person.eyeColor = “blue”;
2. You can also access properties by using a string value that is stored in a variable:

var propertyName = 'make';  
 myCar[propertyName] = 'Ford';propertyName = 'model';  
 myCar[propertyName] = 'Mustang';

---------------------------------------------THANK YOU------------------------------------------------------