1. HTTP Request Methods ?

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. Each of them implements a different semantic, but some common features are shared by a group of them: e,g. a request method can be safe, idempotent, or cacheabe.

**GET**

The GET method requests a representation of the specified resource. Requests using Get should only retrieve data

**POST**.

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

**PUT**

The PUT method replaces all current representations of the target resource with the request payload.

**DELETE**

The DELETE method deletes the specified resource.

1. How does browser read URL ?

* You type a URL in your browser and press Enter.
* Browser looks up IP address for the domain.
* Brower initiates TCP connection with the server.
* Browser sends the HTTP request to the server.
* Server processes request and sends back a response.
* Browser renders the content.

1. HTTP Response status code:

* Informational responses(100-199)

1. 100 Continue
2. 101 Switching protocols

* Successful response(200-299)

1. 201 Created
2. 202 Accepted

* Redirection message(300-399)

1. 302 Found

* Client error responses(400-499)

1. 401 Unauthorized
2. 403 Forbidden
3. 404 Not Found

* Server error responses(500-599)

1. 500 Internal Server Error
2. 501 Not Implemented

1. Difference between HTTP/1.1 VS HTTP/2.0 ?

|  |  |  |
| --- | --- | --- |
|  | **HTTP/1.1** | **HTTP/2.0** |
| Release Year | 1997 | 2015 |
| Key Features | 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 implement due to issues such as head-of-line blocking and was not a feasible solution. | 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. |
| Status Code | 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 | 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 | 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. 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. |

1. Write a blog about objects and its internal representation in Javascript ?

Object in Javascript are used to store keyed collection of various data more complex entities

An object can be created with{..} with an optional list of properties. A property is a “key:value” pair, where key is a string(also called as property name)and value can be anything.

An empty object can be created using one of the two methods

**Let user=new Object( );** // object constructor syntax

**Let user={ };** // object literal syntax

We can immediately put some properties into {…}as “key:value” pairs:

**Let user={**

**Name:”Rajesh”,**

**Age:25**

**};**

A property has a key (also known as “name” or “identifier”)before the colon ”:” and avalue to the right of it.

**In the user object, there are two properties:**

1. The first property has the name **“ name ”** and the value **“Rajesh**”.

2. The second one has the name “**age”** and the value **30**.

**Property values are accessible using the dot notation:**

// get property values of the object:

**Alert( user.name ):** //Rajesh

**Alert( user.age ):** // 30

The value can be any type.Let’s add a Boolean one:

**User.isAdmin = true;**

To remove a property, we can use the delete operator:

**Delete user.age;**

We can also use multiword property names, but then they must be quoted:

**Let user = {**

**Name:”Rajesh”,**

**Age:25,**

**“Likes birds”:true // multiword property name must be quoted**

**};**

For multiword properties, the dot access doesn’t work:

**// this would give a syntax error**

**User.likes birds = true**

Javascript doesn’t understand that. It thinks that we address user.likes, and then gives a syntax error when comes across unexpected birds.

The dot requires the key to be a valid variable identifier. That implies: contains no spaces, doesn’t include special characters($ and\_ are allowed).

There’s an alternative “square bracket notation ” that works with any string:

**Let user ={ };**

**// set**

**user[“likes birds”]=true;**

**// get**

**Alert(user[“likes birds”]); //true**

**// delete**

**Delete user[“likes birds”];**