**1. Write a blog on difference between HTTP 1.1 vs HTTP 2**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **HTTP 1.1** | **HTTP 2** |
| 1 | Development | It was developed in the year 1997. | It was developed in the year 2015. |
| 2 | Binary protocol | It works on the textual format. | It works on the binary protocol. |
| 3 | Prioritization | It doesn’t have prioritization feature; therefore, large files may block rest of the page from loading if they've to load first. | It offers weighted prioritization that will maximize page load speed. |
| 4 | Multiplex | It loads resources one after the other. | It allows multiplexing. It's able to use single TCP connection to send multiple streams of data at once. |
| 5 | Header compression | It compresses data by itself. | It uses HPACK for data compression. |
| 6 | Server push | In this, server only serves requested content to client device. | It allows a server to “push” content to client before the client asks for it. |
| 7 | Connection Management | Multiple connections are often needed to load resources in parallel, which can be less efficient. | Uses a single, multiplexed connection, reducing the need for multiple connections and improving resource loading. |
| 8 | Duplication of data | Problem with HTTP/1.1 is the duplication of data across requests (cookies and other headers). Too many requests mean too much redundant data, which would impact performance. | Techniques like image sprites (combining multiple image requests into a single one) and domain-sharing overcome the problem. |
| 9 | TLS Encryption | HTTPS is optional in HTTP 1.1 | HTTPS is mandatory in HTTP 2 |
| 10 | Browser Support | Most modern web browsers support HTTP 1.1 | HTTP 2 support s Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and Opera. |
| 11 | Security | The client sends a request to a server, and the server sends a response back to the client | A different underlying [protocol called Secure](https://cheapsslweb.com/blog/an-ultimate-guide-on-secure-shell-protocol) Remote Protocol 2 (SRP2) establishes a secure connection between a client and a server |

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

**OBJECTS**

* Objects are more complex, and each object may contain any combination of primitive datatypes as well as reference datatypes.
* An object, is a reference data type. 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.
* For example, if your object is a Car, it will have properties like brand, year, color etc and methods like startEngine, stopEngine, etc.

Two ways to declare objects,

* Using curly braces

e.g.,

const car={

brand:”Ford”,

year:2015

}

* Using new object

eg.,

var car = new object()

car.brand=”Ford”

car.year=2015

**OBJECTS AND PROPERTIES**

* A property of an object can be explained as a variable that is attached to the object.
* The properties of an object define the characteristics of the object.
* We can access the properties of an object in two ways,
* Simple dot-notation

ObjectName.propertyName

Eg.,

var car = new object()

car.brand=”Ford”

car.year=2015

* Square bracket notation

ObjectName[propertyName]

Eg.,

var car = new object()

car[‘brand’]=”Ford”

car[‘year’]=2015

**ADDING NEW PROPERTIES**

car.brand=”Ford”

(or)

car[‘brand’]=”Ford”

**DELETING PROPERTIES**

delete car.brand;

(or)

delete car[“brand”]

**NESTED OBJECTS**

myObj = {  
 name:”Bala”,  
 age:29,  
 car: {  
 brand:”Ford”,  
 year:2015 }  
}

We can access nested objects using the dot notation or the bracket notation:

myObj.car.brand;

myObj.car[“brand”];

myObj[“car”][“brand”];

**NESTED ARRAYS AND OBJECTS**

Values in objects can be arrays, and values in arrays can be objects:

const myObj = {  
 name:”Bala”,  
 age:29,  
 cars: [  
 {name:"Ford", models:["Fiesta", "Focus", "Mustang"]},  
 {name:"BMW", models:["320", "X3", "X5"]},  
 {name:"Fiat", models:["500", "Panda"]}  
 ]  
}

**OBJECT DESTRUCTURING**

const obj={

name:”Kamal”,

age:40

}

const{name,age}=obj;

console.log(name);

**NESTED OBJECT DESTRUCTURING**

const obj={

name:”Kamal”,

age:40,

address:{

city:”Chennai”,

pincode:600117

}

}

const{name,age,address:{city,pincode}}=obj;

console.log(name);

console.log(city);

**UNDEFINED PROPERTY**

Unassigned properties of an object are undefined,

car.color; //undefined

**FOR...IN LOOP**

for (let *variable* in *object*) {  
 *// code to be executed*  
*}*

The block of code inside of the **for...in** loop will be executed once for each property.

const person = {  
 fname:"Ganesh",  
 age: 18  
};  
  
for (let y in person) {  
 txt += person[y];  
}