2.Difference between HTTP1.1 and HTTP2

|  |  |
| --- | --- |
| HTTP1.1 | HTTP2 |
| HTTP1.1 is slower and not reliable | HTTP2 is much faster and more reliable than HTTP.1 |
| HTTP1.1 loads resource one after the other, so if one resource cannot be loaded, it blocks all the resource behind it. | HTTP2 is able to use single TCP connection to send multiple streams of data at once so that no one resource blocks ant other resource. |
| It is less secure | It is more secure than HTTP1.1 |
| It uses binary protocols | It uses multiplexing |
| The textual protocols used in HTTP1.1 | Binary protocols consume less bandwidth and are more efficient and are less error prone than the textual protocols used in HTTP1.1 |
| It can initiate single request in TCP connection. | IT can initiate multiple requests in TCP connection. Ie, MUltiplexing |

3.1.Objects and its internal representation in java script

🡪In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

🡪 An object type is simply a collection of properties in the form of name and value pairs. Notice from the list that null and undefined are primitive JavaScript data types, each being a data type containing just one value. ... A function is a special kind of object that has an executable script block associated with it. The built-in objects are Date, Math, String, Array, and Object. Each is used in a unique and not-quite-consistent way

🡪 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:

**const myCar = new Object();**

**myCar.make = 'Ford';**

**myCar.model = 'Mustang';**

**myCar.year = 1969;**

**🡪** The above example could also be written using an object initializer, which is a comma-delimited list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}):

**const myCar = {**

**make: 'Ford',**

**model: 'Mustang',**

**year: 1969**

**};**

**🡪** 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 JavaScript, almost "everything" is an object.

* Booleans can be objects (if defined with the new keyword)
* Numbers can be objects (if defined with the new keyword)
* Strings can be objects (if defined with the new keyword)
* Dates are always objects
* Maths are always objects
* Regular expressions are always objects
* Arrays are always objects
* Functions are always objects
* Objects are always objects

All JavaScript values, except primitives, are objects.

🡪 Objects are variables too. But objects can contain many values.Object values are written as **name : value** pairs (name and value separated by a colon).