1) DIFFERENCE BETWEEN HTTP1.1 AND HTTP.2 ?

HTTP stands for hypertext transfer protocol & it is used in client-server communication

|  |  |
| --- | --- |
| **HTTP/1.1** | **HTTP/2** |
| **It** works on the textual format | It works on the binary protocol |
| There is head of line blocking that blocks all the requests behind it until it doesn’t get it’s all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource Inclining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

**2)write a blog about objects and it’s internal representation in javascript ?**

# Object:

The **Object** type represents one of [JavaScript's data types](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures). It is used to store various keyed collections and more complex entities. Objects can be created using the [Object()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/Object) constructor or the [object initializer / literal syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Object_initializer).

## [Description](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object#description)

Nearly all [objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#objects) in JavaScript are instances of [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object); a typical object inherits properties (including methods) from Object.prototype, although these properties may be shadowed (a.k.a. overridden). The only objects that don't inherit from Object.prototype are those with [null prototype](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object#null-prototype_objects), or descended from other null prototype objects.

Changes to the Object.prototype object are seen by **all** objects through prototype chaining, unless the properties and methods subject to those changes are overridden further along the prototype chain. This provides a very powerful although potentially dangerous mechanism to override or extend object behavior. To make it more secure, Object.prototype is the only object in the core JavaScript language that has [immutable prototype](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/setPrototypeOf#description) — the prototype of Object.prototype is always null and not changeable.

### [Object prototype properties](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object#object_prototype_properties)

You should avoid calling any Object.prototype method, especially those that are not intended to be polymorphic (i.e. only its initial behavior makes sense and no descending object could override it in a meaningful way). All objects descending from Object.prototype may define a custom own property that has the same name, but with entirely different semantics from what you expect. Furthermore, these properties are not inherited by [null-prototype objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object#null-prototype_objects). All modern JavaScript utilities for working with objects are [static](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object#static_methods). More specifically:

* [valueOf()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/valueOf), [toString()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/toString), and [toLocaleString()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/toLocaleString) exist to be polymorphic and you should expect the object to define its own implementation with sensible behaviors, so you can call them as instance methods. However, valueOf() and toString() are usually implicitly called through [type conversion](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#type_coercion) and you don't need to call them yourself in your code.
* [\_\_defineGetter\_\_()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/__defineGetter__), [\_\_defineSetter\_\_()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/__defineSetter__), [\_\_lookupGetter\_\_()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/__lookupGetter__), and [\_\_lookupSetter\_\_()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/__lookupSetter__) are deprecated and should not be used. Use the static alternatives [Object.defineProperty()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty) and [Object.getOwnPropertyDescriptor()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/getOwnPropertyDescriptor) instead.
* The [\_\_proto\_\_](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/proto) property is deprecated and should not be used. The [Object.getPrototypeOf()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/getPrototypeOf) and [Object.setPrototypeOf()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/setPrototypeOf) alternatives are static methods.
* The [propertyIsEnumerable()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/propertyIsEnumerable) and [hasOwnProperty()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty) methods can be replaced with the [Object.getOwnPropertyDescriptor()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/getOwnPropertyDescriptor) and [Object.hasOwn()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwn) static methods, respectively.
* The [isPrototypeOf()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/isPrototypeOf) method can usually be replaced with [instanceof](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/instanceof), if you are checking the prototype property of a constructor.