1. **Difference between HTTP1.1 vs HTTP2**

**What is HTTP?**

**HTTP** stands for **H**yper **T**ext **T**ransfer **P**rotocol

Communication between client computers and web servers is done by sending **HTTP Requests** and receiving **HTTP Responses**

HTTP Request / Response

Communication between clients and servers is done by **requests** and **responses**:

1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

Difference between HTTP1.1 vs HTTP2

|  |  |
| --- | --- |
| **HTTP1.1** | **HTTP2** |
| HTTP/1.1, the first standardized version of HTTP, was introduced in 1997. It presented significant performance optimizations (over HTTP/0.9 and HTTP/1.0) and transformed the way requests and responses were exchanged between clients and servers. | At the beginning of 2010, Google introduced an experimental protocol, SPDY, which supported multiplexing (multiple requests/responses sent and received asynchronously over a single TCP connection) but as it gained traction IETF’s HTTP Working Group came up with HTTP/2 in 2015, which is based on the SPDY protocol. |
| HTTP/1.1 is slower than HTTP/2. | HTTP/2 is much faster and more efficient than HTTP/1.1. |
| HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. | [HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource.](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) |
| It compresses data by itself. | It uses HPACK for data compression. |
| It uses requests resource In lining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| works on the textual format. | It works on the binary protocol. |

**2. Objects and its internal representation in Javascript**

* 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).
* Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types. An object, is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.

[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.