1. Write a blog on Difference between HTTP1.1 vs HTTP2

* Ans:
* [**Introduction**](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference#introduction)
* The Hypertext Transfer Protocol, or HTTP, is an application protocol that has been the de facto standard for communication on the World Wide Web since its invention in 1989. From the release of HTTP/1.1 in 1997 until recently, there have been few revisions to the protocol. But in 2015, a reimagined version called HTTP/2 came into use, which offered several methods to decrease latency, especially when dealing with mobile platforms and server-intensive graphics and videos. HTTP/2 has since become increasingly popular, with some estimates suggesting that around a third of all websites in the world support it. In this changing landscape, web developers can benefit
* from understanding the technical differences between HTTP/1.1 and HTTP/2, allowing them to make informed and efficient decisions about evolving best practices.
* After reading this article, you will understand the main differences between HTTP/1.1 and HTTP/2, concentrating on the technical changes HTTP/2 has adopted to achieve a more efficient Web protocol.

## [Background](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference#background)

* To contextualize the specific changes that HTTP/2 made to HTTP/1.1, let’s first take a high-level
* look at the historical development and basic workings of each.
* [**HTTP/1.1**](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference#http-1-1)
* Developed by Timothy Berners-Lee in 1989 as a communication standard for the World Wide Web, HTTP is a top-level application protocol that exchanges information between a client computer and a local or remote web server. In this process, a client sends a text-based request to a server by calling a *method* like GET or POST. In response, the server sends a resource like an HTML page back to the client.
* For example, let’s say you are visiting a website at the domain www.example.com. When you navigate to this URL, the web browser on your computer sends an HTTP request in the form of a text-based message, similar to the one shown here:
* GET /index.html HTTP/1.1

### [HTTP/2](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference#http-2)

HTTP/2 began as the SPDY protocol, developed primarily at Google with the intention of reducing web page load latency by using techniques such as compression, multiplexing, and prioritization. This protocol served as a template for HTTP/2 when the Hypertext Transfer Protocol working group httpbis of the [IETF (Internet Engineering Task Force)](https://www.ietf.org/) put the standard together, culminating in the publication of HTTP/2 in May 2015. From the beginning, many browsers supported this standardization effort, including Chrome, Opera, Internet Explorer, and Safari. Due in part to this browser support, there has been a significant adoption rate of the protocol since 2015, with especially high rates among new sites.

From a technical point of view, one of the most significant features that distinguishes HTTP/1.1 and HTTP/2 is the binary framing layer, which can be thought of as a part of the application layer in the internet protocol stack. As opposed to HTTP/1.1, which keeps all requests and responses in plain

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

https://github.com/reach2arunprakash/javascript-101/tree/master/ppt

* Write a blog about objects and its internal representation in Javascript
* Ans

# **bjects And Its Internal Representation In JavaScript**

[[Bablu Verma](https://medium.com/@babluv623?source=post_page-----c5b576b20a99--------------------------------)](https://medium.com/@babluv623?source=post_page-----c5b576b20a99--------------------------------)

[[Analytics Vidhya](https://medium.com/analytics-vidhya?source=post_page-----c5b576b20a99--------------------------------)](https://medium.com/analytics-vidhya?source=post_page-----c5b576b20a99--------------------------------)

[Bablu Verma](https://medium.com/@babluv623?source=post_page-----c5b576b20a99--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Febe812476810&operation=register&redirect=https%3A%2F%2Fmedium.com%2Fanalytics-vidhya%2Fobjects-and-its-internal-representation-in-javascript-c5b576b20a99&user=Bablu+Verma&userId=ebe812476810&source=post_page-ebe812476810----c5b576b20a99---------------------post_header-----------)

Published in

[Analytics Vidhya](https://medium.com/analytics-vidhya?source=post_page-----c5b576b20a99--------------------------------)

·

4 min read

·

Nov 5, 2020

5

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.

Loosely speaking, 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, in the context of an object.

For Eg. If your object is a student, it will have properties like name, age, address, id, etc and methods like updateAddress, updateNam, etc.

# **Objects and properties**

A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. 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:

var myCar = new Object();  
myCar.make = 'Ford';  
myCar.model = 'Mustang';  
myCar.year = 1969;

Unassigned properties of an object are [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) (and not [null](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null)).

myCar.color; // undefined

Properties of JavaScript objects can also be accessed or set using a bracket notation (for more details see [property accessors](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_Accessors)). Objects are sometimes called associative arrays, since each property is associated with a string value that can be used to access it. So, for example, you could access the properties of the myCar object as follows:

myCar['make'] = 'Ford';  
myCar['model'] = 'Mustang';  
myCar['year'] = 1969;

An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string. However, any property name that is not a valid JavaScript identifier (for example, a property name that has a space or a hyphen, or that starts with a number) can only be accessed using the square bracket notation. This notation is also very useful when property names are to be dynamically determined (when the property name is not determined until runtime). Examples are as follows:

// four variables are created and assigned in a single go,   
// separated by commas  
var myObj = new Object(),  
 str = 'myString',  
 rand = Math.random(),  
 obj = new Object();  
myObj.type = 'Dot syntax';  
myObj['date created'] = 'String with space';  
myObj[str] = 'String value';  
myObj[rand] = 'Random Number';  
myObj[obj] = 'Object';  
myObj[''] = 'Even an empty string';console.log(myObj);

You can also access properties by using a string value that is stored in a variable:

var propertyName = 'make';  
myCar[propertyName] = 'Ford';propertyName = 'model';  
myCar[propertyName] = 'Mustang';

You can use the bracket notation with [for...in](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in) to iterate over all the enumerable properties of an object. To illustrate how this works, the following function displays the properties of the object when you pass the object and the object's name as arguments to the function:

function showProps(obj, objName) {  
 var result = ``;  
 for (var i in obj) {  
 // obj.hasOwnProperty() is used to filter out properties from the object's prototype chain  
 if (obj.hasOwnProperty(i)) {  
 result += `${objName}.${i} = ${obj[i]}\n`;  
 }  
 }  
 return result;  
}

So, the function call showProps(myCar, "myCar") would return the following:

myCar.make = Ford