**JavaScript-intro**

Write a blog on Difference between HTTP1.1 vs HTTP2? Javascript Multiplexing: HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, 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. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.

Server push: Typically, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpages, which often involve several dozen separate resources that the client must request. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. The server also sends a message letting the client know what pushed content to expect – like if Bob had sent Alice a Table of Contents of his novel before sending the whole thing.

Header compression: Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading.

Write a blog about objects and its internal representation in Javascript? Objects are important data types in javascript. Objects are different than primitive datatypes (i.e. number, string, boolean, etc.). Primitive data types contain one value but Objects can hold many values in form of Key: value pair. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

Every object has some property associated with some value. These values can be accessed using these properties associated with them.

var myCar = new Object();

myCar.make = 'Suzuki';

myCar.model = 'Altros';

myCar.year = 1978;

myCar.wheels = 2;

After creating myCar object, the value inside the object can be accessed using keys.

i.e.

myCar.year

Output: 1978

These values can be accessed using brackets notation also.

myCar[year]

Output: 1978

The syntax for adding a property to an object is : ObjectName.ObjectProperty = propertyValue; The syntax for deleting a property from an object is: delete ObjectName.ObjectProperty; The syntax to access a property from an object is: objectName.property  
//or objectName["property”]  
//or objectName[expression]  
So, conclusion and simple definition for Java Script properties is “Properties are the values associated with a JavaScript object”.

Object methods An object method is an object property containing a function definition.

i.e.,

Let’s assume to start the car there will be a mechanical functionality. function(){return ignition.on}

and so similar is to stop/brake/headlights on & off, etc. So, conclusion and simple definition for Java Script Object methods is “Methods are actions that can be performed on objects.”

Create JavaScript Object with Object Literal One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces as shown below let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'}; Create JavaScript Object with Constructor Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below function Vehicle(name, maker) { this.name = name; this.maker = maker; } let car1 = new Vehicle(’Fiesta’, 'Ford’); let car2 = new Vehicle(’Santa Fe’, 'Hyundai’) console.log(car1.name); //Output: Fiesta console.log(car2.name); //Output: Santa Fe Using the JavaScript Keyword new The following example also creates a new JavaScript object with four properties: var person = new Object(); person.firstName = “John”; person.lastName = “Doe”; person.age = 50; person.eyeColor = “blue”; Using the Object.create method Objects can also be created using the Object.create() method. This method can be very useful, because it allows you to choose the prototype object for the object you want to create, without having to define a constructor function. // Animal properties and method encapsulation var Animal = { type: 'Invertebrates', // Default value of properties displayType: function() { // Method which will display type of Animal console.log(this.type); } }; // Create new animal type called animal1 var animal1 = Object.create(Animal); animal1.displayType(); // Output:Invertebrates // Create new animal type called Fishes var fish = Object.create(Animal); fish.type = 'Fishes'; fish.displayType(); // Output:Fishes

**About**

Javascript

**Topics**

**Resources**

[Readme](https://github.com/udhaymsd/Javascript-intro#readme)

**Stars**

[**0** stars](https://github.com/udhaymsd/Javascript-intro/stargazers)

**Watchers**

[**1** watching](https://github.com/udhaymsd/Javascript-intro/watchers)

**Forks**

[**0** forks](https://github.com/udhaymsd/Javascript-intro/network/members)

[**Releases**](https://github.com/udhaymsd/Javascript-intro/releases)

No releases published

[Create a new release](https://github.com/udhaymsd/Javascript-intro/releases/new)

[**Packages**](https://github.com/users/udhaymsd/packages?repo_name=Javascript-intro)

No packages published  
[Publish your first package](https://github.com/udhaymsd/Javascript-intro/packages)

**Footer**

© 2022 GitHub, Inc.

**Footer navigation**

* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Stat](https://www.githubstatus.com/)