DAY:1

18/08/2021

**Difference between HTTP1.1 AND HTTP/2**

**HTTP**

* HyperText Transfer Protocol
* Application protocol i.e., foundation for data communication for www.
* Based on client-server model
* Client-Servers are communicating via requests and responses.

**http1.1**

* 3rd version of HTTP
* Introduced persistent connections for improved performances and laid the foundation for standard requests.
* When websites became more resource intensive, http1.1 began to show its limitations.
* Specifically, one outstanding request per TCP connection
* Slowing down page load times.

**http/2:**

* Released by IETP (Internet Engineering Task Force) in 2015.
* 2nd major version of Internet Protocol and derived from SPDY protocol.
* Focus-reducing Latency via TCP pipelining and providing mandatory compression.

Goals of developing http/2:

* Request multiplexing over a single protocol
* Protocol negotiation mechanism (i.e., electing protocol either http1.1/http2 or other).
* High Level compatibility with http1.1
* Page load speed improvements
* Binary protocol
* Compression of request headers
* Server Push
* Request pipelining
* HOL (Head of Line)-Package blocking
* **Request multiplexing:**
* Can send multiple requests for data in parallel over a **single TCP**connection
* Reduces additional round trip time ( rtt),making your website load faster without any optimization, and makes domain sharding simultaneously.
* **Header Compression:**
* Compress large number of header frames
* HPACK specification used
* Both client and server maintain a list of headers used in previous clients and server requests.
* **Binary Protocol:**
* Transforming from text to binary protocol
* Uses binary commands (o’s and 1’s) to complete request response cycles
* Eases complications with framing and simplifies implementation of commands, which are intermixed due to commands with texts and optional spaces
* **Server PUSH:**
* Allows the server to send the additional cacheable information to the client, that is not requested, but it is anticipated in future requests.
* The server can prioritize pushed resources

* **Browser compatibility:**
* Web browsers only support http/2 via encrypted connections
* Increasing user and application security.

**Objects and Its Internal Representation in JavaScript**

Objects, in JavaScript, it is most important data-type and forms the building blocks for modern JavaScript.

Objects are more complex and each object may contain any combination of the primitive data-types types (Number, String, Boolean, null, undefined and symbol) 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.

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.

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. 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

# ****Creating Objects In JavaScript :****

# 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:

Example

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()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/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);  
 }  
};

Blocks:

<https://docs.google.com/document/d/1-LGT0Qg8qHR0yt2nSiwvmGoX-WYepEVl7iWrnIVuG_8/edit?usp=sharing>