**OBJECTS AND INTERNAL REPRESENTATION IN JS:**

**Objects:**

in JavaScript, is it’s most important datatype 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 datatypes all store a single value each.

Example:

let school = {

name : “Shahrukh”,

location : “chennai”,

B’day : “1996”

}

In this example “name”, “location”, “b’day” are all “keys” and “shahrukh”, “channai” and 1996 are values of these keys respectively. Each of these keys is referred to as properties of the object. An object in JavaScript may also have a function as a member, in which case it will be known as a method of that object.

.Property names can also be strings with more than one space separated words. In which case, these property names must be enclosed in quotes :  
let school = {  
“school name” : “Vivekananda School”,  
}  
Like property names which are numbers, they must also be accessed using the bracket notation.

**Ex:**

let car1 = new Vehicle(‘M3’, ‘BMW’);  
let car2 = new Vehicle(‘GT’, ‘Ford’)

console.log(car1.name); -> Output: M3  
console.log(car2.name); -> Output: GT

**Object Constructor** **:**Another way to create objects in JavaScript involves using the “Object” constructor. The Object constructor creates an object wrapper for the given value. This, used in conjunction with the “new” keyword allows us to initialize new objects.  
Example :  
const school = new Object();  
school.name = ‘Vivekanada school’;  
school.location = ‘Delhi’;  
school.established = 1971;

school.displayInfo = function(){  
console.log(${school.name} was established   
in ${school.established} at ${school.location});  
}

school.displayInfo();  
In this example,above are not well suited to programs that require the creation of multiple objects of the same kind, as it would involve repeatedly writing the above lines of code for each such object.

**Constructors:** Constructors in JavaScript, like in most other OOP languages, provides a template for creation of objects. In other words, it defines a set of properties and methods that would be common to all objects initialized using the constructor.

Having more than one function in a class with the name of constructor() results in an error.  
**Prototypes :** Another way to create objects involves using prototypes. Every JavaScript function has a prototype object property by default(it is empty by default). Methods or properties may be attached to this property. A detailed description of prototypes is beyond the scope of this introduction to objects.

However you may familiarize yourself with the basic syntax used as below:  
let obj = Object.create(prototype\_object, propertiesObject)  
// the second propertiesObject argument is optional  
An example of making use of the Object.create() method is:  
let footballers = {  
position: “Striker”  
}

let footballer1 = Object.create(footballers);

// Output : Striker   
console.log(footballer1.position);  
**From above example:**  
In the above example footballers served as a prototype for creating the object “footballer1”.All objects created in this way inherits all properties and methods from its prototype objects. Prototypes can have prototypes and those can have prototypes and so on. This is referred to as prototype chaining in JavaScript. This chain terminates with the Object.prototype which is the default prototype fallback for all objects. Javascript objects, by default, inherit properties and methods from Object.prototype but these may easily be overridden. It is also interesting to note that the default prototype is not always Object.prototype.For example Strings and Arrays have their own default prototypes — String.prototype and Array.prototype respectively.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***HTTP2.0:***

1->It allows same TCP connection for multiple parallel requests.

2->It uses header compression to reduce the overhead caused by TCP’s slow-start mechanism.

3->This provides encrypted connections and increasing user and application security.

4->It multiplexes streams within a single TCP connection, and will have to implement flow control in a different manner

5->In this version uses HPACK compression to decrease the average size of the header.

***HTTP/1.1:***

1->In this version connections are not encrypted and application security was less compared to http2.

2->The other problem with HTTP/1.1 is the duplication of data across requests.

3->In this version image assets keep loading for a longer time one after another to complete the full image.

4->It relies on the transport layer to avoid buffer overflow, each new TCP connection requires a separate flow control mechanism.

5->It uses formats like gzip to compress the data transferred in the messages.