1. To read:

a. https://stackoverflow.com/questions/1517582/what-is-the-difference-between-statically-ty

ped-and-dynamically-typed-languages

b. https://stackoverflow.com/questions/17253545/scripting-language-vs-programming-langu

age

c. https://cs.lmu.edu/~ray/notes/paradigms/

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

3. Write a blog about objects and its internal representation in Javascript

4. codekata practice

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

Answers :

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

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.

Before diving into the main differences between HTTP/1.1 and HTTP/2, we will see the both at high-level.

**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. Note that not all the resources are returned to the client in the first call for data. The requests and responses will go back and forth between the server and client until the web browser has received all the resources necessary to render the contents of the HTML page on your screen.

**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. 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. As opposed to HTTP/1.1, which keeps all requests and responses in plain text format, HTTP/2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers. An application-level API would still create messages in the conventional HTTP formats, but the underlying layer would then convert these messages into binary. This ensures that web applications created before HTTP/2 can continue functioning as normal when interacting with the new protocol.

**Important differences between HTTP/1.1 and HTTP/2**

**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 can 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.

**Prioritization:**

In the context of web performance, prioritization refers to the order in which pieces of content are loaded. Suppose a user visits a news website and navigates to an article. Should the photo at the top of the article load first? Should the text of the article load first? Should the banner ads load first? Prioritization affects a webpage's load time. For example, certain resources, like large JavaScript files, may block the rest of the page from loading if they must load first. More of the page can load at once if these render-blocking resources load last. In addition, the order in which these page resources load affects how the user perceives page load time. If only behind-the-scenes content (like a CSS file) or content the user can't see immediately (like banner ads at the bottom of the page) loads first, the user will think the page is not loading at all. If the content that's most important to the user loads first, such as the image at the top of the page, then the user will perceive the page as loading faster.

In HTTP/2, developers have hands-on, detailed control over prioritization. This allows them to maximize perceived and actual page load speed to a degree that was not possible in HTTP/1.1. HTTP/2 offers a feature called weighted prioritization. This allows developers to decide which page resources will load first, every time. In HTTP/2, when a client makes a request for a webpage, the server sends several streams of data to the client at once, instead of sending one thing after another. This method of data delivery is known as multiplexing. Developers can assign each of these data streams a different weighted value, and the value tells the client which data stream to render first.

**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.

**3. Write a blog about objects and its internal representation in JavaScript**

Objects are one of the non-primitive datatypes of JavaScript. It is a collection of key-value pairs. The value could be another object or it could be an Array. Hence the objects are bit complex in nature, it can store multiple data. We can store functions as value in the object by assigning a key to it. In this case we refer to the function as a method.

In JS the objects are implemented as ‘pass by reference’ that means, if we store any object in a variable, we are not actually storing the object in the variable but the memory location of the object in the heap. If we are copying the value of the variable to another variable, we are copying the address of the object (i.e.) both the objects will have the same address and will point to the same object in the memory. Thus, we can manipulate the object’s data using any variable.

In JS we can create objects using object literal ({ }) or using new Object() or using Object.create(), any which ways, we will get the address of a new object stored in memory.

We can access to Object’s properties or methods using dot notation (.) or bracket notation ([]). Bracket notation is useful if we are not aware of the property or method name and if the name is stored in another variable or if the name comes from a webservice, we can make use of bracket notation.

Example of Key-value pair:

Age : “30”

Example of Object:

{

Name: “vasu”,

Age: “30”

}

Example of creating objects:

1.Using object literal

var person = { }

person.Name = “vasu”

person.Age = “30”

2. Using object literal and assigning the properties on the fly

var person2 = {

Name: “Deva”,

Age: “35”

}

3.Using new Object()

var person3 = new Object()

person3.Name = “Krishna”

person3.Age = “37”

Example of Accessing Object’s data

var person2 = {

Name: “Deva”,

Age: “35”

}

console.log(person2.Name) // prints Deva

var ageProp = “Age”

console.log(person2[ageProp]) // prints 35