***JavaScript - Day -1: Introduction to Browser & web***

***20/09/2023 - Wednesday - 9:00 AM : 11:30 AM***

***DAY 1 ACTIVITY:***

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

|  |  |
| --- | --- |
| **HTTP1.1** | **HTTP2** |
| 1. In HTTP/1.1, multiple resources (e.g., images, scripts, stylesheets) are fetched over separate connections, and each connection can only handle one request at a time. | 1. HTTP/2 introduces multiplexing, allowing multiple requests and responses to be interleaved over a single connection. |
| 2. This can lead to a phenomenon called head-of-line blocking, where one slow-loading resource delays the loading of subsequent resources. | 2. This eliminates the need for multiple connections and minimizes head-of-line blocking, resulting in faster page loads. |
| 3. Headers, which contain metadata about the request and response, are sent as plain text and are not compressed. | 3. HTTP/2 employs header compression, reducing the size of headers sent over the connection. This reduces bandwidth usage and speeds up page loading times. |
| 4. HTTP/1.1 doesn't support server push. The server can't proactively send additional resources to the client before they are requested. | 4. HTTP/2 allows the server to push resources to the client without waiting for the client to request them. |
| 5. HTTP/1.1 doesn't have built-in prioritization mechanisms. Resources are fetched in the order they are requested. | 5. HTTP/2 supports resource prioritization, allowing developers to specify which resources are more critical. |
| 6. HTTP/1.1 uses a text-based protocol, which can be human-readable but less efficient for machines to parse. | 6. HTTP/2 uses a binary protocol, which is more efficient for both servers and clients to process. While not human-readable, it's faster to parse and reduces latency. |
| 7. While HTTPS is supported in HTTP/1.1, it's not mandatory. Many websites still use HTTP without encryption. | 7. HTTP/2 encourages the use of TLS encryption (HTTPS) and, in some implementations, requires it. This enhances security and privacy on the web. |
| 8. It works on the textual format. | 8. It works on the binary protocol. |
| 9. There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | 9. It allows multiplexing so one TCP connection is required for multiple requests. |
| 10. It uses requests resource Inlining for use getting multiple pages | 10. It uses PUSH frame by server that collects all multiple pages |
| 11. It compresses data by itself. | 11. It uses HPACK for data compression. |

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

**Objects in JavaScript:**

JavaScript is an object-oriented language. Almost everything in JavaScript is an object or can be represented as an object. Objects are collections of key-value pairs, where each key is a string and each value can be any data type, including other objects, functions, or primitive values like numbers and strings.

Here's a simple example of an object in JavaScript:

var person = {

firstName: 'John',

lastName: 'Doe',

age: 30,

};

In this example, person is an object with three properties: firstName, lastName, and age, each associated with a value.

**Internal Representation of Objects:**

* Under the hood, JavaScript engines use various techniques to represent objects efficiently. While the exact implementation details may vary among JavaScript engines (such as V8 in Chrome or SpiderMonkey in Firefox), here are some common aspects of how objects are internally represented:
* Property Names as Strings: Object properties are stored as strings in JavaScript. When you access a property, the JavaScript engine looks up the corresponding string key to retrieve the associated value.
* Hash Maps or Hash Tables: JavaScript engines typically use hash maps or hash tables to store object properties. These data structures allow for fast lookups, insertions, and deletions, making property access efficient.
* Hidden Classes and Shape Optimization: JavaScript engines employ optimization techniques like hidden classes (V8) or shapes (SpiderMonkey) to improve object property access speed. These optimizations help engines quickly determine the memory layout of objects.
* Prototypes and Inheritance: Objects in JavaScript often have a prototype chain, which allows them to inherit properties and methods from their prototype objects. This inheritance is implemented through a chain of hidden references.

**######### THANK YOU #########**