**HTTP2 Vs. HTTP1 or HTTP1 Vs. HTTP2 – Let’s Understand The Two Protocols**

HTTP/1.1 has been around for more than a decade. With Google’s SPDY leading the way in 2015, the IETF (Internet Engineering Task Force) gave us HTTP/2, which introduces several features to reduce page load times. Let’s compare HTTP2 Vs. HTTP1.1 in detail.

HTTP/2 achieves faster webpage loading without performance optimizations that require extensive human efforts in terms of development. It significantly reduces the complexities that had crept into HTTP/1.1 and gives us a robust protocol which, though not without its flaws, will perhaps stand the test of time. Before making this leap forward, let’s trace our steps back to when the internet was in its infancy to understand how the different versions evolved into the current form.

**The Beginnings of HTTP & The Internet**

Our story begins in 1969, with a program called Advanced Research Projects Agency Network (ARPANET). ARPANET used packet switching and allowed multiple computers to communicate with each other on a single network. However, this was just a by-product. The original intention behind ARPANET was to design a time-sharing system that allowed research institutes to share their computer resources for effective utilization of processing power.

**Objects in JavaScript:**

In JavaScript, an object is a composite data type that allows you to group together data and functions into a single entity. Unlike other primitive data types, such as numbers or strings, objects can be complex and dynamic, mirroring the structure of real-world entities. They are essential for organizing and managing code in a way that is both intuitive and scalable.

**Internal Representation:**

Internally, JavaScript objects are represented as key-value pairs. Each key is a string or symbol (the property name), and its associated value can be any valid JavaScript value, including other objects or functions. The internal structure of an object is often implemented as a hash table or a similar data structure to ensure efficient access and manipulation.

**For example, in the person object created above:**

**name, age,** and **profession** are keys.

**'John Doe', 25,** and **'Software** **Developer'** are their respective values.

Accessing and Modifying Properties:

You can access and modify the properties of an object using dot notation or square bracket notation.

**Object Methods:**

Objects can also contain functions, referred to as methods. These methods can perform actions related to the object or manipulate its internal state. In the person object, sayHello is a method that logs a greeting to the console.

**Conclusion:**

Understanding the internal representation of objects in JavaScript is crucial for developing efficient and maintainable code. Objects provide a powerful and flexible way to model complex structures in your programs, making them an indispensable part of the JavaScript language. Whether you are working with front-end frameworks like React or building server-side applications with Node.js, a solid grasp of JavaScript objects will undoubtedly enhance your programming prowess.