**Difference between HTTP1.1 vs HTTP2**

HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the internet. It is the protocol that enables browsers and web servers to communicate and exchange information. Two versions of the protocol that are widely used today are HTTP/1.1 and HTTP/2. In this blog post, we will explore the main differences between these two versions of the protocol and how they affect web development.

* HTTP/1.1 was first defined in 1999 and is the most widely used version of the protocol. It is a text-based protocol that uses a request-response model to exchange information between browsers and servers. In this model, the browser sends a request to the server, and the server sends back a response. Each request and response is sent over a separate connection, which means that a browser can only send one request at a time.
* HTTP/2, on the other hand, is a binary protocol that was first defined in 2015. It was designed to address some of the limitations of HTTP/1.1, such as the need for multiple connections to load a single webpage. HTTP/2 uses a single connection for multiple requests and responses, which is known as multiplexing. This allows for faster and more efficient communication between browsers and servers.
* One of the key differences between HTTP/1.1 and HTTP/2 is the way they handle requests and responses. HTTP/1.1 uses a pipeline model, where requests are sent one after the other, and responses are received in the same order. HTTP/2, on the other hand, uses a multiplexing model, where requests and responses can be sent and received in any order. This allows for faster communication and better utilization of network resources.
* Another important difference between these two versions of the protocol is the way they handle headers. In HTTP/1.1, headers are sent with each request and response, and they can become quite large, which increases the amount of data that needs to be transferred. HTTP/2, on the other hand, uses a mechanism called HPACK to compress headers, which reduces the amount of data that needs to be transferred.
* Finally, HTTP/2 also supports server push, which allows the server to push resources to the browser without the browser requesting them. This can improve the loading time of web pages by eliminating the need for the browser to request resources that it knows it will need.
* In conclusion, HTTP/2 is the newer and more efficient version of the protocol that addresses some of the limitations of HTTP/1.1. It uses a single connection for multiple requests and responses, which is known as multiplexing, and it also compresses headers and supports server push. While, HTTP/1.1 is widely used and supported but it has some limitations and it's less efficient than HTTP/2.
* As a web developer, it is important to be familiar with the differences between these two versions of the protocol and how they affect web development. With the use of HTTP/2, web developers can create faster and more efficient websites, which can improve the user experience.

**Objects and its internal representation in javaScript**

* JavaScript is a powerful programming language that allows developers to create complex and dynamic web applications. One of the most important concepts in JavaScript is the object, which is used to store and organize data. In this blog post, we will explore the internal representation of objects in JavaScript and how they are used to store and manipulate data. In JavaScript, an object is a collection of key-value pairs. Each key is a string or a symbol, and the value can be any data type, including another object. This allows objects to be nested and used to create complex data structures. Objects are created using the object literal notation, which is denoted by curly braces {}. For example, the following code creates an object with two key-value pairs:

|  |  |
| --- | --- |
| |  | | --- | | let person = {  name: "John Doe",  age: 30  }; | |

* The internal representation of an object in JavaScript is called an object map. It is a hash table that stores the key-value pairs of the object. The keys in an object map are called property names, and they are used to access the values stored in the object. For example, the following code accesses the value of the "name" property of the "person" object:

|  |
| --- |
| console.log(person.name); // Outputs "John Doe" |

* Properties can also be accessed using the square bracket notation, which allows you to use variables or expressions as property names. For example, the following code uses a variable to access the value of the "name" property:

|  |
| --- |
| let prop = "name";  console.log(person[prop]); // Outputs "John Doe" |

* In addition to storing data, objects can also contain methods, which are functions that can be called on an object. Methods are defined as properties of an object, with the value being a function. For example, the following code defines a "greet" method on the "person" object:

|  |
| --- |
| let person = {  name: "John Doe",  age: 30,  greet: function() {  console.log("Hello, my name is " + this.name);  }  };  person.greet(); // Outputs "Hello, my name is John Doe" |

* In conclusion, objects in JavaScript are a powerful and versatile data structure that allows developers to organize and manipulate data in a structured way. The internal representation of an object is an object map, which is a hash table that stores the key-value pairs of the object. Understanding the internal representation of objects is essential for understanding how to work with objects in JavaScript.