**Understanding the Differences Between HTTP/1.1 and HTTP/2**

The Hypertext Transfer Protocol (HTTP) is the foundation of any data exchange on the Web, and it is a protocol used for transmitting hypertext. Since its inception, HTTP has undergone several iterations, with HTTP/1.1 being the most widely adopted version until the introduction of HTTP/2. This blog aims to highlight the key differences between HTTP/1.1 and HTTP/2, outlining the improvements and benefits that HTTP/2 brings to the table.

**1. Multiplexing**

HTTP/1.1:

- In HTTP/1.1, each request-response cycle is limited to a single connection. This means that multiple requests from the same client require separate connections, leading to an inefficient use of resources and potential bottlenecks due to connection limits.

HTTP/2:

- HTTP/2 supports multiplexing, allowing multiple requests and responses to be sent over a single TCP connection simultaneously. This significantly reduces latency and improves the efficiency of data transmission.

**2. Header Compression**

HTTP/1.1:

- Headers in HTTP/1.1 are sent as plain text and often contain redundant information, which increases the overhead of each request and response. Each request carries the full header set, leading to inefficiency.

HTTP/2:

- HTTP/2 introduces HPACK compression for headers, reducing the size of header frames and thus decreasing the overall amount of data transferred. This compression helps to reduce latency and improve load times.

**3. Server Push**

HTTP/1.1:

- HTTP/1.1 does not natively support server push. Each resource must be requested individually by the client, which can lead to slower page load times as the browser waits for required assets.

HTTP/2:

- HTTP/2 allows servers to push resources to the client proactively before they are requested. This feature can significantly speed up page loads by preemptively sending necessary resources such as CSS, JavaScript, and images.

**4. Stream Prioritization**

HTTP/1.1:

- In HTTP/1.1, all requests are treated equally, which can lead to inefficiencies, especially when critical resources are delayed due to less important ones being processed first.

HTTP/2:

- HTTP/2 allows clients and servers to assign priority levels to streams. This means that critical resources can be delivered more quickly, improving overall performance and user experience.

**5. Binary Protocol**

HTTP/1.1:

- HTTP/1.1 uses a text-based protocol, which is more human-readable but less efficient for machines to parse.

HTTP/2:

- HTTP/2 uses a binary protocol, making it more efficient for machines to parse and less prone to errors. This change improves both performance and security.

**Understanding Objects and Their Internal Representation in JavaScript**

JavaScript is a versatile language widely used for web development. One of the core concepts in JavaScript is the object, which allows developers to create complex data structures. This blog will delve into what objects are in JavaScript, how they are internally represented, and how they function.

**1. Definition and Creation**

Objects in JavaScript:

- An object in JavaScript is a collection of key-value pairs where the keys are strings (or symbols) and the values can be any type of data, including other objects or functions.

Creation of Objects:

- Objects can be created in several ways, including object literals, constructor functions, the `Object.create()` method, and ES6 classes.

**2. Properties and Methods**

Properties:

- Properties are the values associated with an object. They can be added, modified, or deleted dynamically.

Methods:

- Methods are functions associated with an object. They define behaviors that the object can perform.

**3. Internal Representation**

Prototype Chain:

- JavaScript objects have a prototype, which is also an object. This prototype object can have its own prototype, forming a prototype chain. This chain is used for property and method inheritance.

Hidden Class:

- JavaScript engines use hidden classes and inline caching to optimize property access. A hidden class is an internal representation used to store information about an object's properties and their memory layout.

Property Attributes:

- Each property in a JavaScript object has attributes: `writable`, `enumerable`, and `configurable`. These attributes control how the property can be manipulated.

**4. Performance Considerations**

Avoid Property Deletion:

- Deleting properties can make objects less predictable for the JavaScript engine's optimization techniques.

Consistent Property Order:

- Accessing properties in a consistent order can improve performance due to better optimization by the JavaScript engine.

Use Methods over Accessors:

- Directly accessing properties repeatedly can be less efficient than using methods, especially if the method can cache results or optimize access patterns.