**1.HTTP 1 vs. HTTP 1.1 vs. HTTP 2: A Detailed Analysis**

This article presents a detailed analysis of the differences between major HTTP versions such as HTTP 1, HTTP 1.1, and HTTP2.

**HTTP 1.1 vs. HTTP 2: Key Differences Between the Two HTTP Versions**

Comparisons are common, and it is nothing different in the cybersecurity world. One such technology is hypertext transfer protocol (HTTP). This is why there is a constant tug-of-war of**HTTP 2 vs. HTTP 1**. But even before HTTP 1.1 or HTTP 2 came into existence, there was HTTP. However, as the Internet evolved and became more complex, the need for development came with it.

What limited the existing HTTP is the lack of support for persistent connections, thus resulting in increased latency and decreased performance. Due to all these reasons, HTTP 1.1 came into development and became the standard for HTTP communication on the web.

So, to understand **HTTP 2 vs. HTTP 1.1** better, let us first get into the details of each. Let’s start with HTTP 1.1.

**What Is HTTP 1.1?**

Officially standardized in 1999, HTTP 1.1 is the most recent version of HTTP. Just like its predecessor, this application protocol transfers data between web servers and clients. However, what makes it different is the incorporation of PUT, DELETE, and OPTIONS, and it includes features like HTTP pipelining and chunked transfer encoding. Due to all these features, HTTP 1.1 provides better performance and more flexibility than HTTP 1.

**Benefits of HTTP 1.1**

There are several benefits of HTTP 1.1 compared to the earlier version of HTTP 1.0.

* **Persistent Connections**: One of the most significant benefits of HTTP 1.1 is the support for persistent connections, allowing multiple requests to be sent over the same connection. This reduces the overhead of establishing new connections for each request and results in improved performance.
* **Improved Caching**: With several features that improve caching, such as the ability to send conditional GET requests and support for caching negotiation, clients can cache responses and avoid unnecessary network traffic.
* **More Efficient Request and Response Handling**: With PUT, DELETE, and OPTIONS, fine-grained control over resources on the server is facilitated. It also supports HTTP pipelining and chunked transfer encoding, thus reducing the latency of sending large payloads.

As technology became more evolved, HTTP 2 was developed. It addressed some of the limitations and inefficiencies of HTTP 1.1.

Moving forward with **HTTP 1.1 vs. 2.0**, let’s now discuss HTTP 2.

**What Is HTTP 2?**

HTTP 2 is the second major version of the HTTP network protocol used for transmitting data over the Internet. It was developed to improve the performance and efficiency of web applications by reducing the amount of data sent over the wire. This reduces the number of round trips required to load a webpage and allows for a more efficient application of resources.

But why is HTTP 2.0 vs. 1.1 a debated topic, and what makes the former more efficient than its previous version?

**Why HTTP 2 Replaced HTTP 1.1**

As the technology developed and the Internet evolved, HTTP 1.1 soon became outdated. This was the reason why [HTTP 2 was developed](https://en.wikipedia.org/wiki/HTTP/2_Server_Push). However, some other reasons include the following:

* **Request Multiplexing**

In HTTP 1.1, each request and response was sent over a separate connection. For each request, a new connection had to be established, resulting in a high number of round trips required to load a webpage. HTTP 2 came with multiplexing of requests.

* **Header Compression**

HTTP 1.1 did not have built-in support for header compression, so headers were sent in plain text, which meant that they took up a significant amount of bandwidth. As a result, while sending large numbers of headers or when sending headers over slow connections, problems were faced.

These are the major reasons why HTTP 1.1 was replaced. However, there are other differences between the two versions. So first, let us compare both of them.

**HTTP 2.0 vs. 1.1**

These are some of the parameters based on which we will compare both the HTTP versions.

**● Predicting Resource Requests**

In HTTP/1.1, the client-server initiate all requests for resources such as images, stylesheets, and scripts. The server can only respond to requests that it receives. This means that the client must first request the HTML of a web page, parse it, and then make additional requests for any additional resources it needs to render the page. This results in delayed page load times.

Contrary to this, HTTP 2 allows for server push, so the server proactively pushes resources to the client without the client needing to request them. This speeds up page load times as the client starts processing and rendering resources as soon as they are received.

**● Buffer Overflow**

In HTTP 1.1, a buffer overflow can occur when a client sends a request with a header that is larger than the server’s buffer size. This can cause the server to crash or become unresponsive. To prevent buffer overflow, servers typically have a maximum buffer size for incoming requests and reject any requests exceeding this limit.

HTTP 2, on the other hand, uses a more sophisticated approach to prevent buffer overflow. It uses a flow control mechanism that allows the server to send data to the client in small chunks rather than sending all the data at once.

**● Multiplexing**

HTTP 2 supports multiplexing, allowing multiple requests and responses to be sent over a single connection simultaneously. This helps to reduce the latency and increase the overall performance of the connection. In contrast, HTTP 1.1 uses a separate connection for each request and response, resulting in increased latency and reduced performance.

**● Binary Protocol**

HTTP 1.1 uses plain text to encode and transmit data. Though it is easy for humans to read and understand the data, it can be less efficient than a binary protocol.

HTTP 2 uses a series of binary codes to encode and transmit data rather than plain text. Binary protocols are generally more efficient than text-based protocols because they can transmit data more compactly.

|  |  |  |
| --- | --- | --- |
|  | **HTTP 1.1** | **HTTP 2** |
| **Development** | It was developed in the year 1997 | It was developed in the year 2015. |
| **Compression** | It compresses data by itself. | It uses HPACK for data compression. |
| **Binary Protocol** | A text-based protocol uses plain text to encode and transmit data. | It works on the binary protocol as a series of binary codes encode and transmit data rather than plain text. |
| **Security** | The client sends a request to a server, and the server sends a response back to the client. | A different underlying [protocol called Secure](https://cheapsslweb.com/blog/an-ultimate-guide-on-secure-shell-protocol) Remote Protocol 2 (SRP2) establishes a secure connection between a client and a server. |
| **Multiplexing** | A separate connection is established for each request and response, which can add overhead and latency to the communication process. | It allows multiplexing so multiple requests and responses can be sent over a single connection. |
| **Buffer Overflow** | HTTP 1.1 cannot handle buffer overflow vulnerabilities due to the lack of sufficient measures. | HTTP 2 includes measures to prevent buffer overflow vulnerabilities. |
| **Performance** | HTTP 1.1 does not include any in-built features, so the performance it delivers is less efficient. | HTTP 2 is designed to be more efficient and performant than HTTP 1.1. This is because HTTP 2 includes several features like multiplexing, binary protocol and header compression. |

**Which Browser Supports HTTP 2 and HTTP 1?**

Most modern web browsers support HTTP 1.1 and  HTTP 2, including [Google Chrome](https://cheapsslweb.com/blog/how-to-fix-err-ssl-version-interference-error-in-chrome), Mozilla Firefox, Apple Safari, Microsoft Edge, and Opera.

However, the HTTP version might vary based on the version of the browser, and some older versions may not support it. So it all comes down to the operating system, as well as the specific configuration of the browser.

**How to Move to HTTP 2**

To move to HTTP 2, you can do the following:

* Upgrade your web server software to support HTTP 2. Some of the most popular web servers, such as Apache and Nginx, have built-in support for this HTTP version.
* Obtain an SSL certificate for your website. HTTP 2 requires the use of SSL to encrypt the connection between the browser and the server.
* Update your website’s code to use the new features. This includes using the new multiplexing feature to send multiple requests and responses over a single connection and using the new server push feature to send resources to the browser without waiting for a request.
* Test your website to ensure that it is fully compatible with HTTP 2. It includes testing for any issues with browser compatibility and any issues with the performance of your website.
* Finally, configure your server to use HTTP 2 and redirect all HTTP 1.1 requests to it.

This is how you can move from HTTP 1.1 to HTTP 2.

**Concluding Words**

HTTP is the backbone of the internet and enables the creation and sharing of various types of information and resources. Without HTTP, the internet would not be able to function in the way that it does today. However, with time HTTP versions kept changing.

We hope the **HTTP 2.0 vs. 1.1** comparison helps you understand the HTTP versions.

**2.Objects and its internal representation in javascript**

Objects are important data types in javascript. Objects are different than primitive datatypes (i.e. number, string, boolean, etc.). Primitive data types contain one value but Objects can hold many values in form of Key: value pair. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

Every object has some property associated with some value. These values can be accessed using these properties associated with them.

var myCar = new Object();

myCar.make = 'Suzuki';

myCar.model = 'Altros';

myCar.year = 1978;

myCar.wheels = 2;

After creating myCar object, the value inside the object can be accessed using keys.

i.e.

*myCar.year*

Output: 1978

These values can be accessed using brackets notation also.

*myCar[year]*

Output: 1978

The syntax for adding a property to an object is :

ObjectName.ObjectProperty = propertyValue;

The syntax for deleting a property from an object is:

delete ObjectName.ObjectProperty;

The syntax to access a property from an object is:

objectName.property

//or

objectName["property”]

//or

objectName[expression]

So, conclusion and simple definition for Java Script properties is “Properties are the values associated with a JavaScript object”.

**Object methods**

An object method is an object property containing a function definition.

i.e.,

Let’s assume to start the car there will be a mechanical functionality.

function(){return ignition.on}

and so similar is to stop/brake/headlights on & off, etc.  
So, conclusion and simple definition for Java Script Object methods is “Methods are actions that can be performed on objects.”

**Create JavaScript Object with Object Literal**

One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces as shown below

let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};

**Create JavaScript Object with Constructor**

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below

function Vehicle(name, maker) {

this.name = name;

this.maker = maker;

}

let car1 = new Vehicle(’Fiesta’, 'Ford’);

let car2 = new Vehicle(’Santa Fe’, 'Hyundai’)

console.log(car1.name); //Output: Fiesta

console.log(car2.name); //Output: Santa Fe

**Using the JavaScript Keyword new**

The following example also creates a new JavaScript object with four properties:

var person = new Object();

person.firstName = “John”;

person.lastName = “Doe”;

person.age = 50;

person.eyeColor = “blue”;