# Difference Between HTTP/1.1 and HTTP/2

## Introduction

The Hypertext Transfer Protocol (HTTP) is the foundation of any data exchange on the Web and a protocol used for transmitting hypertext requests and information on the internet. HTTP has evolved from its initial version to HTTP/1.1 and then to HTTP/2, each bringing improvements and optimizations.

## Key Differences Between HTTP/1.1 and HTTP/2

|  |  |  |
| --- | --- | --- |
| Feature | HTTP/1.1 | HTTP/2 |
| Protocol Type | Text-based | Binary |
| Multiplexing | No (Head of line blocking) | Yes (Multiple requests in a single connection) |
| Data Compression | Yes (by itself) | Yes (Uses HPACK) |
| Server Push | No | Yes (PUSH frame by server) |
| Connection Management | Multiple connections for multiple requests | Single connection for multiple requests |
| Request Prioritization | No | Yes |
| Header Compression | Limited | Yes (HPACK) |
| Security | No inherent encryption | Better support for encryption (TLS) |
| Latency | Higher due to multiple connections | Lower due to multiplexing |

## Detailed Comparison

### Protocol Type

HTTP/1.1: Works on a textual format, meaning it transmits data in a plain text form.  
HTTP/2: Works on a binary protocol, which is more efficient in terms of parsing and transmitting data.

### Multiplexing

HTTP/1.1: Suffered from head-of-line blocking where one slow request can delay all other requests.  
HTTP/2: Allows multiplexing, meaning multiple requests and responses can be in flight at the same time over a single TCP connection.

### Data Compression

HTTP/1.1: Compresses data by itself but has limited capabilities.  
HTTP/2: Uses HPACK for header compression, which is more efficient and reduces overhead.

### Server Push

HTTP/1.1: Does not support server push.  
HTTP/2: Introduces server push, allowing servers to send resources proactively to the client.

### Connection Management

HTTP/1.1: Requires multiple connections for multiple requests, increasing latency.  
HTTP/2: Uses a single connection for multiple requests, significantly improving performance.

### Request Prioritization

HTTP/1.1: Lacks request prioritization.  
HTTP/2: Supports request prioritization, enabling more important resources to be delivered faster.

## Conclusion

HTTP/2 provides significant improvements over HTTP/1.1 in terms of performance, efficiency, and security. These enhancements make HTTP/2 a better choice for modern web applications.

# Objects and Their Internal Representation in JavaScript

## Introduction

Objects in JavaScript are a fundamental part of the language and are used to store collections of data and more complex entities. Understanding objects and their internal representation is crucial for effective JavaScript programming.

## What is an Object?

An object is an unordered collection of related data and functions, known as properties and methods. Properties are variables that belong to the object, while methods are functions that belong to the object.

## Internal Representation of Objects

JavaScript objects are dynamic collections of properties, where each property is a key-value pair.

## Key Concepts

|  |  |
| --- | --- |
| Concept | Description |
| Property | A key-value pair where the key is a string (or Symbol) and the value can be any data type, including other objects and functions. |
| Method | A function that is a property of an object. |
| Prototype | Each JavaScript object has a prototype. A prototype is also an object. All JavaScript objects inherit their properties and methods from their prototype. |
| Constructor Function | A special type of function used for creating objects. |

## Example of an Object

```javascript  
let person = {  
 firstName: "John",  
 lastName: "Doe",  
 age: 30,  
 fullName: function() {  
 return this.firstName + " " + this.lastName;  
 }  
};  
```

## Properties and Methods

|  |  |  |
| --- | --- | --- |
| Key | Value | Description |
| firstName | "John" | Property holding a string value |
| lastName | "Doe" | Property holding a string value |
| age | 30 | Property holding a number value |
| fullName | function() | Method that returns the full name |

## Prototype Chain

Every object in JavaScript has a prototype. When trying to access a property or method, JavaScript first looks at the object itself. If it doesn't find the property or method, it looks at the object's prototype, and so on, up the prototype chain.

## Creating Objects

1. \*\*Object Literals\*\*: The simplest way to create an object.  
  
```javascript  
let car = {  
 make: "Toyota",  
 model: "Corolla",  
 year: 2021  
};  
```  
  
2. \*\*Using a Constructor Function\*\*:  
  
```javascript  
function Car(make, model, year) {  
 this.make = make;  
 this.model = model;  
 this.year = year;  
}  
let myCar = new Car("Toyota", "Corolla", 2021);  
```  
  
3. \*\*Using the `Object.create` Method\*\*:  
  
```javascript  
let animal = {  
 type: "Mammal",  
 displayType: function() {  
 console.log(this.type);  
 }  
};  
let cat = Object.create(animal);  
cat.type = "Cat";  
cat.displayType(); // Outputs: Cat  
```

## Conclusion

Understanding objects and their internal representation is essential for mastering JavaScript. Objects allow you to model real-world entities and provide the foundation for creating more complex data structures and functionality in your programs.