1. *Write a blog on Difference between HTTP 1.1 vs HTTP2*

**HTTP/1.1: The Old Standard**

HTTP/1.1 has been the workhorse of the internet for over a decade. It's the protocol responsible for delivering web pages and their resources to your browser when you enter a URL. However, as web applications grew in complexity, HTTP/1.1 started to show its limitations.

**Head-of-Line Blocking**: In HTTP/1.1, each resource request is made sequentially. If a resource takes time to load, it blocks subsequent requests, slowing down the entire page load process. This is known as "***head-of-line blocking***."

**High Latency**: HTTP/1.1 requires multiple connections for parallel resource retrieval, which increases latency due to the overhead of establishing and maintaining these connections.

**Redundant Headers**: Each request in HTTP/1.1 includes redundant headers, which consume bandwidth and slow down communication.

**HTTP/2: The Modern Solution**

HTTP/2 was introduced as an answer to the limitations of HTTP/1.1. It was designed to improve web performance by addressing these issues and introducing several innovative features:

**Multiplexing**: HTTP/2 allows multiple requests and responses to be multiplexed over a single TCP connection. This means that multiple resources can be sent and received in parallel, eliminating head-of-line blocking and reducing latency.

**Header Compression**: HTTP/2 uses header compression, reducing the overhead caused by redundant headers in HTTP/1.1. This leads to faster and more efficient communication between clients and servers.

**Server Push:** One of the most significant features of HTTP/2 is server push. With this capability, a server can proactively send resources to the client before the client requests them. This can significantly improve page load times by reducing the need for additional requests.

**Binary Protocol:** HTTP/2 uses a binary protocol instead of the plain text format used in HTTP/1.1. This makes it more efficient to parse and less error-prone.

**Prioritization:** HTTP/2 introduces the concept of stream prioritization, allowing the client to specify the importance of different resources. This enables better control over resource loading and ensures that critical resources are delivered first.

*2. Write a blog about objects and its internal representation in Javascript*

**Anatomy of an Object**

At a high level, JavaScript objects are collections of key-value pairs, where the keys are strings (or Symbols) and the values can be any data type, including other objects, functions, and primitive values like numbers and strings.

**Sample:**

const person = {

firstName: "John",

lastName: "Doe",

age: 30,

isEmployed: true,

};

Each key in the object is also known as a property, and its corresponding value is referred to as the property's value.

## **Under the Hood: Object Representation**

To understand how objects are represented internally, we need to delve into two crucial components: properties and prototypes.

### **1. Properties**

Properties in JavaScript objects are stored as a combination of two separate data structures:

* **Property Descriptor**: This data structure defines the attributes of a property, such as whether it's writable, enumerable, or configurable.
* **Property Value**: This is the actual value associated with the property.

### 

### 

### 

### **2. Prototypes**

JavaScript objects can also have a prototype, which is another object. If a property is not found on the current object, JavaScript looks for it on the prototype object. This mechanism is known as the prototype chain, and it allows for property inheritance.

**Sample:**

const parent = {name: "Parent" };

const child = Object.create(parent);

child.age = 10;

console.log(child.name); // “Parent”

**In this example, child inherits the name property from its prototype (parent).**

## 

## **Object Creation**

JavaScript provides several ways to create objects:

* **Object Literal**: As shown earlier, you can create an object using curly braces {}.
* **Constructor Functions:** You can define object blueprints (constructors) and create instances using the new keyword.

**Sample:**

function Person(firstName, lastName)

{

this.firstName = firstName;

this.lastName = lastName;

}

const john = new Person("John", "Doe");

**Object.create():** You can create objects with a specified prototype using Object.create().

Sample:

const parent = { name: "Parent" };

const child = Object.create(parent);

## **Object Methods**

JavaScript objects come with built-in methods to manipulate their properties, including:

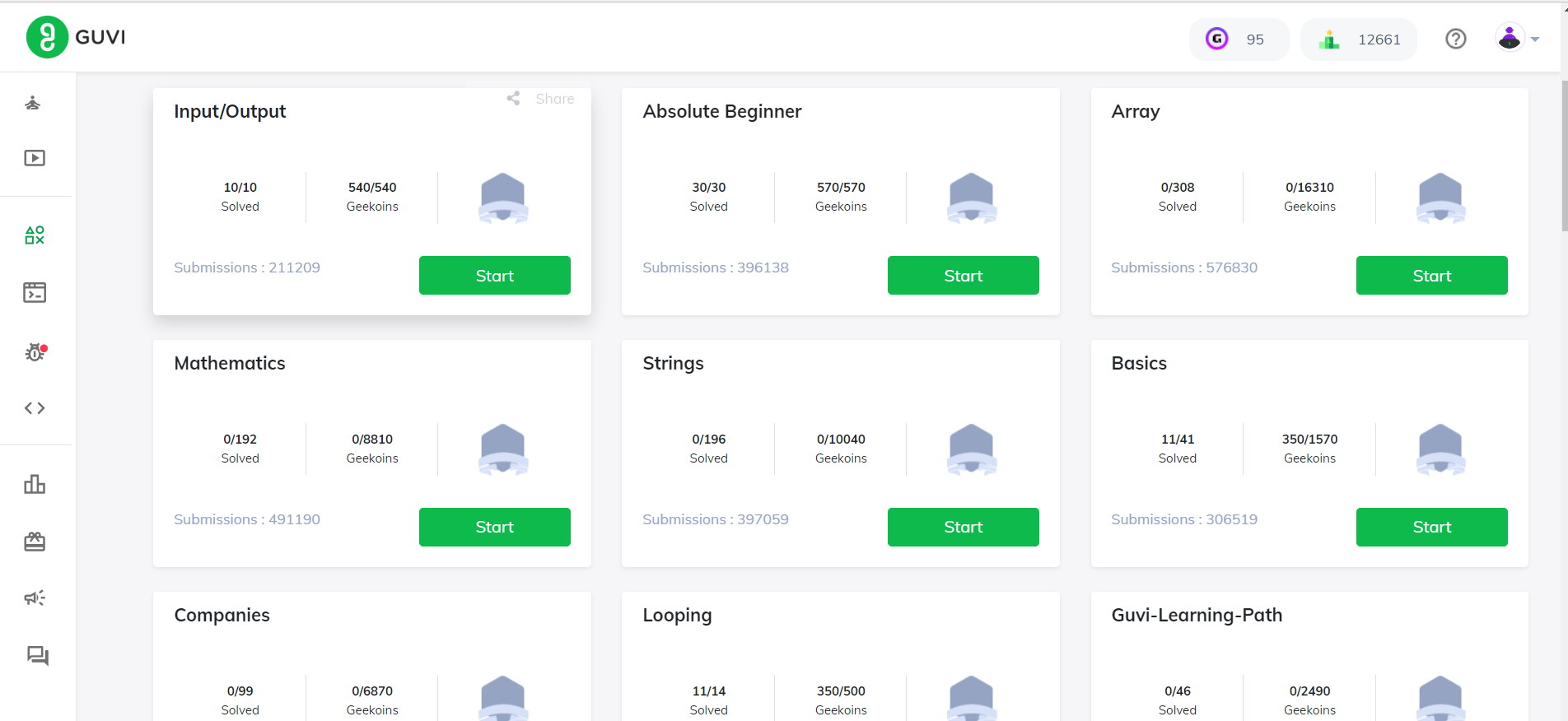
* **Object.keys():** Returns an array of an object's own enumerable property names.
* **Object.values()**: Returns an array of an object's own enumerable property values.
* **Object.entries():** Returns an array of an object's own enumerable property [key, value] pairs.

## **Conclusion**

JavaScript objects are the building blocks of the language, allowing developers to structure and manage data effectively. Their internal representation involves property descriptors, values, and prototypes, which combine to provide a versatile and powerful tool for organizing and accessing data.

Understanding how objects work under the hood is crucial for writing efficient and maintainable JavaScript code. Whether you're creating simple data structures or complex object-oriented systems, a solid grasp of objects and their internal representation is essential for becoming a proficient JavaScript developer.

**codekata practice**



**Read about IP address, port, HTTP methods, MAC address**

**1. IP Address (Internet Protocol Address):**

* An IP address is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.
* IP addresses are used to identify and locate devices on a network, allowing data to be routed between them.
* There are two main types of IP addresses: IPv4 (32-bit address) and IPv6 (128-bit address), with IPv6 being introduced to address the exhaustion of IPv4 addresses.

**2. Port:**

* A port is a 16-bit number used to identify specific processes or services running on a device within a network.
* Ports are essential for allowing multiple services to coexist on a single device, as they help route incoming data packets to the appropriate application or service.
* Ports are divided into three ranges: well-known ports (0-1023), registered ports (1024-49151), and dynamic or private ports (49152-65535).

**3. HTTP Methods (Hypertext Transfer Protocol Methods):**

* HTTP methods, also known as HTTP verbs, are used to indicate the desired action to be performed on a resource identified by a URL in the context of the HTTP protocol.
* Common HTTP methods include:
  + **GET:** Used to request data from a server.
  + **POST:** Used to send data to be processed to a server.
  + **PUT**: Used to update a resource or create a new resource if it doesn't exist.
  + **DELETE**: Used to request the removal of a resource.
  + **PATCH**: Used to apply partial modifications to a resource.
  + **HEAD:** Similar to GET but only requests metadata about a resource, not the resource itself.
  + **OPTIONS:** Used to retrieve information about the communication options available for a resource.
* These methods are crucial for enabling different interactions between clients (usually web browsers) and servers over the World Wide Web.

**4. MAC Address (Media Access Control Address):**

* A MAC address is a unique hardware address assigned to every network interface controller (NIC) or network adapter in a device.
* Unlike IP addresses, which can change as devices connect to different networks, MAC addresses are typically permanent and hardcoded into the device's hardware.
* MAC addresses are used for local network communication at the data link layer (Layer 2) of the OSI model and help devices within the same local network segment identify each other.