**Assignment: 1 (Day-1 Introduction to Browser & Web)**

**Question: 2 Objects and it is Internal Representation in JavaScript**

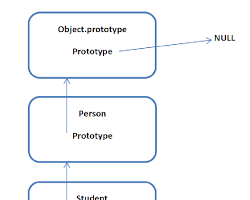
**Objects in JavaScript:**

* Fundamental building blocks: They represent real-world entities or abstract concepts, holding collections of related data and behaviors.
* Key-value pairs: Properties (keys) store values of various data types, including primitives, other objects, or functions (methods).
* Unordered: Property order doesn't matter.
* Reference types: Variables assigned objects hold a reference (pointer) to the object's memory location, not a direct copy of its values.

**Internal Representation:**

1. Prototype Chain:

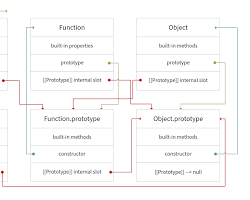
* Each object inherits properties and methods from its prototype, forming a chain-like structure.
* Visual example:

[](https://dimosbanis.net/software-thoughts/object-oriented-javascript-the-prototype-chain-and-inheritance/)

JavaScript object prototype chain

2. Internal Slots:

* Objects are stored in memory as a hidden class with a dictionary-like structure for properties.
* Components:
  + Header: Contains object type, size, and prototype links.
  + Slots: Hold property names and values efficiently.
* Visual example:

[](https://www.telerik.com/blogs/understand-prototypes-prototypal-inheritance-javascript)

JavaScript object internal slots

3. Property Table:

* Efficiently maps property names to values, especially for frequent property access.

4. Hidden Classes:

* Used by JavaScript engines to optimize object creation and property access, minimizing memory usage and improving performance.

Key Points:

* Mutable: Object properties can be added, modified, or removed after creation.
* Methods: Functions defined within objects, representing actions or behaviors.
* Prototypes: Enable object inheritance and code reusability.
* Internal representation details can vary across JavaScript engines.

Understanding these concepts is crucial for:

* Efficient object manipulation
* Memory optimization
* Avoiding common pitfalls like accidental modification of shared objects
* Debugging complex object-oriented code

Additional Insights:

* Object destructuring: A concise way to extract multiple object properties into separate variables.
* Spread operator: Useful for copying or merging objects.
* Object.keys(), Object.values(), Object.entries(): Methods for iterating over object properties.