lec 4 summary:

Here is the summary of the topics you provided, presented in a similar, structured quick-reference format.

💻 OOP, Objects, Calling, and JSON: Quick Reference

1. 🧱 OOP (Object-Oriented Programming)

OOP is a programming paradigm based on the concept of **"objects,"** which can contain data (properties) and code (methods).

Concept	Description
Object	A fundamental unit containing state (data) and behavior (methods).
Class	A blueprint or template for creating objects (used in modern JS/ES6).
Encapsulation	Bundling data (properties) and the methods that operate on that data into a single unit (the object).
Inheritance	Mechanism where a class can inherit properties and methods from another class.
Polymorphism	The ability of a method to do different things depending on the object it is acting upon (e.g., method overriding).

---

2. 👤 User-Defined Anonymous Objects (Object Literals)

This is the most common and simplest way to create a single, custom object in JavaScript without using a formal class or constructor function.

Concept	Syntax / Example	Details
Creation	`let person = { firstName: "Ali", age: 30, sayHi: function() { ... } };`	Defined using curly braces `{}`.
Properties	`person.firstName` or `person["age"]`	Key-value pairs. Keys are strings (or Symbols), values can be any data type.
Methods	`person.sayHi()`	Properties whose value is a function.
Deletion	`delete person.age;`	Removes a property from the object.

---

3. 📞 Calling Methods and Functions

The way a function or method is called directly influences the value of the `this` keyword inside that function.

Calling Way	Syntax / Example	this Keyword Value
Simple/Direct	`functionName();`	**Global Object** (or `undefined` in Strict Mode).
Method	`object.methodName();`	**The containing Object** (`object`).
Constructor	`new ClassName();`	**The newly created Object** (instance).
Indirect (Explicit)	`functionName.call(obj, ...args);` `functionName.apply(obj, [args]);`	**The first argument passed** (`obj`). Allows you to manually set `this` .

4. 🔑 The `this` Keyword

The value of `this` is not fixed; it is determined at **runtime** based on **how** a function is executed.

- **In a regular function:** `this` points to the **global object** (e.g., `window` or `global`).
- **In an object method:** `this` points to the **object** that owns the method.
- **In a class method:** `this` points to the **instance** of the class.
- **In an Arrow Function:** Arrow functions do **not** have their own `this` . They inherit `this` from the surrounding (lexical) scope. This behavior is key for callbacks and maintaining context.

5. 📦 JSON (JavaScript Object Notation)

JSON is a lightweight data-interchange format that is language-independent but uses conventions familiar to programmers of the C-family of languages (including JavaScript).

Concept	Description
Purpose	Used for transmitting data between a server and web application.
Format	Data is stored in **key/value pairs**, similar to JavaScript object literals. **Keys must be strings** (enclosed in double quotes).
Data Types	Supports string, number, object, array, boolean, and `null` . **Does not** support functions, dates, or `undefined` .

Serialization	`JSON.stringify(jsObject)` **:** Converts a JavaScript object into a **JSON string** (for sending data).
Parsing	`JSON.parse(jsonString)` **:** Converts a JSON string back into a **JavaScript object** (for receiving data).