19 Kasım 2023 Pazar

JS OOP Notes

1

- Summary:
- This section provides a high-level overview of object-oriented programming (OOP). It introduces the concept of objects as self-contained blocks of code that model real-world entities and explains how objects contain both data (properties) and behavior (methods). OOP aims to organize code, making it more flexible and easier to maintain compared to unstructured or "spaghetti code." The lecture emphasizes the importance of OOP in large-scale software engineering.

- The key points covered include:
  - **1. Definition of OOP:** OOP is a programming paradigm based on the concept of objects, which represent real-world entities and contain both data and behavior.
  - **2. Object Interaction:** Objects are used as building blocks in applications, interacting with each other through a public interface (API) that consists of accessible methods.
  - **3. OOP Goals:** OOP was developed to organize code, improve flexibility, and ease maintenance. It contrasts with spaghetti code, where code is disorganized and challenging to maintain.
  - **4. Comparison with Functional Programming:** The lecture acknowledges that OOP is not the only way to write organized code, mentioning functional programming as an alternative, which will be discussed later in the course.
  - **5. Class Concept:** Classes are introduced as blueprints in OOP, providing a way to create objects. A fictional example of a user class is used to illustrate the idea of creating instances (objects) from a class.
  - **6. Abstraction:** Abstraction involves ignoring or hiding unnecessary details to focus on the essential aspects of an implementation. It allows for a higher-level perspective and is crucial for creating understandable and maintainable code.
  - **7. Encapsulation:** Encapsulation involves keeping certain properties and methods private within a class, preventing external code from accessing them

directly. Public interfaces (APIs) are defined for external interactions, reducing the risk of bugs and ensuring code stability.

  - **8. Inheritance:** Inheritance is a mechanism where a child class inherits properties and methods from a parent class, promoting code reuse and creating a hierarchy of classes. An example involving user and admin classes is provided.
  - **9. Polymorphism:** Polymorphism allows a child class to override a method inherited from a parent class. Different implementations of a method can exist in child classes, providing flexibility and customization.

- The section concludes by mentioning that the next part of the course will delve into how object-oriented programming is implemented in JavaScript.

2

- Summary:

- In this part of the introduction to Object-Oriented Programming (OOP), the focus is on understanding how OOP works in JavaScript, emphasizing the use of prototypes. The key points covered include:

- **1. Review of Classical OOP Model**: The lecture begins with a review of the classical OOP model with classes and instances, which was covered in the previous lecture. A class is likened to a blueprint used to create instances, analogous to building houses from architectural plans.
- **2. Introduction to Prototypes in JavaScript:** JavaScript uses a different approach to OOP with something called prototypes. All objects in JavaScript are linked to a prototype object, and prototypal inheritance allows objects to access and use methods and properties defined on that prototype.
- **3. Delegation in Prototypal Inheritance**: Prototypal inheritance in JavaScript involves objects delegating behavior to the linked prototype object. The lecture introduces the concept of delegation and distinguishes it from the classical inheritance discussed in the previous lecture.
- **4. Practical Examples of Prototypal Inheritance**: The lecture provides practical examples of prototypal inheritance in JavaScript, citing instances such as using array methods like `map`. The methods are defined on the prototype (`array.prototype`), and objects inherit or delegate their behavior to these methods.

- **5. Implementing Object-Oriented Programming in JavaScript:** The lecture addresses questions about creating prototypes, linking objects to prototypes, and creating new objects without traditional classes. It introduces three methods: constructor functions, ES6 classes (a syntactic sugar over constructor functions), and `Object.create()`.
- **6. Three Ways of Implementing OOP in JavaScript:**
    - - Constructor Functions: Used to create objects programmatically and set the new object's prototype. Commonly used in JavaScript since its early days.
    - - ES6 Classes: Introduced in ES6, providing a more modern syntax for OOP. However, they are essentially a layer of abstraction over constructor functions.
    - - Object.create(): A straightforward way to link an object to a prototype, though less commonly used.
- **7. Continued Relevance of OOP Principles**: The lecture emphasizes that the four principles of OOP (abstraction, encapsulation, inheritance, and polymorphism) are still valid and important with prototypal inheritance. The upcoming sections will delve into implementing and using these principles in JavaScript.

- The lecturer assures that, although the terminology might be overwhelming, the practical application of prototypal inheritance will be clarified in subsequent lectures. The focus is on understanding how prototypal inheritance works in JavaScript and putting OOP into practice.