Java Script

NEXT

GENIUS space

План

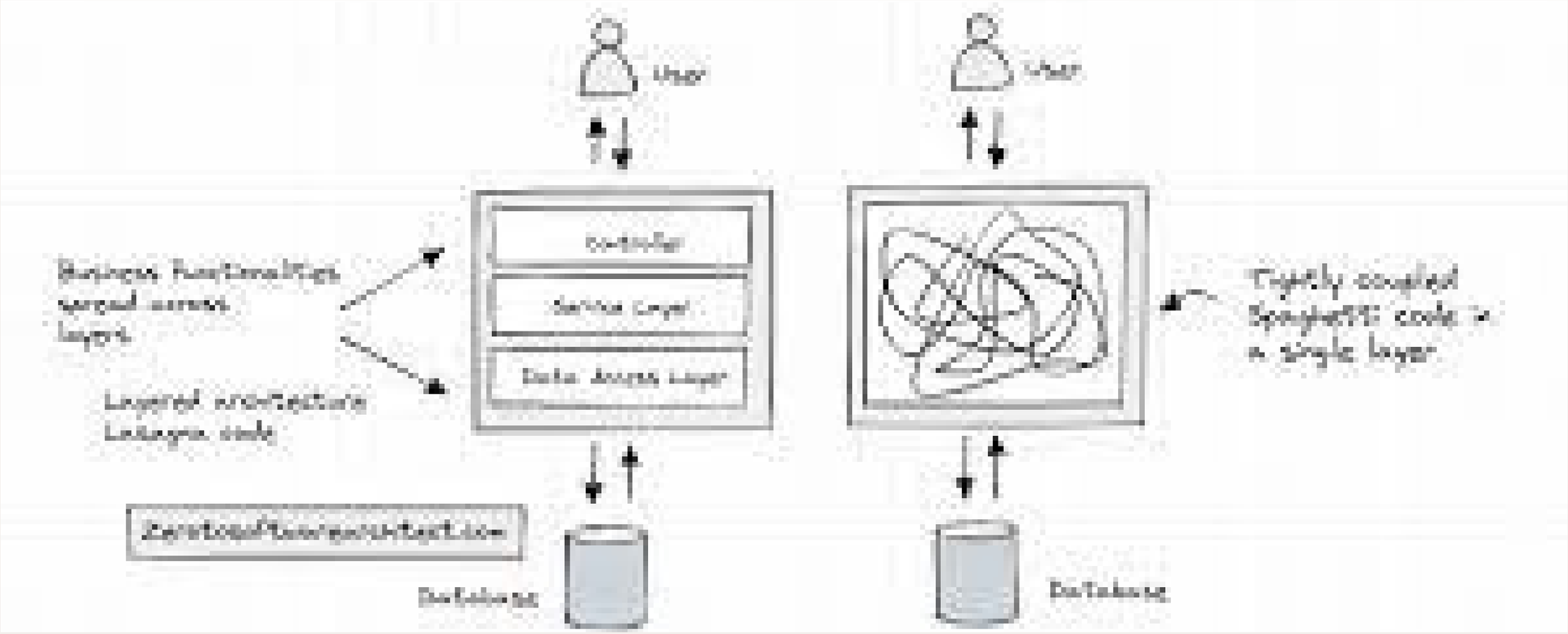
ООП

класи

об"єкти

інкапсуляція, абстракція, наслідування, поліморфізм

Процедурне програмування — набір не пов'язаних функцій і змінних для зберігання та обробки інформації. Цей підхід простий і прямолінійний, але при зростанні кодової бази результатом може бути те, що називається spaghetti code - тісно пов'язаний код.



ООП

Об'єктно-Орієнтоване Програмування (ООП) - один з методів програмування, який розглядає програму як множину «об'єктів», що взаємодіють між собою.

ООП представляє програмне забезпечення як сукупність взаємних об'єктів, а не просто набір команд чи функцій як у традиційному програмуванні.

У JavaScript є багато об'єктів які є вбудовані у ядро, так звані стандартні об'єкти JavaScript наприклад: String, Number, Date і т. д.

Кожен стандартний об'єкт являється екземпляром об'єкта Object, наслідуючи його властивості і методи.

У JavaScript об'єкти це сукупність методів і властивостей.

Властивість об'єкта це зміна яка належить об'єкту і містить значення певного типу даних.

У JavaScript властивості є динамічними, що означає що їх можна додавати і видаляти з об'єкта.

Метод об'єкта це функція яка належить об'єкту і виконує якусь певну дію для об'єкта.

У JavaScript спочатку не було класів але у ECMAScript 6 появилися.

Клас у JavaScript є шаблоном об'єкту і описує його властивості і методи. Описавши class створюють об'єкт за допомогою оператора new.

У JavaScript є наступні способи створення об'єкта:

Object()

```
var ob = new Object();
ob.property="тест";
alert( ob.property );
```

Object.create()

```
var ob = Object.create(null, {property:{value:"тест"}});
alert( ob.property );
```

літеральний спосіб

```
var ob = {property:"тест"};
alert( ob.property );
```

функція-конструктор

```
function myOb(){
this.property="тест";
}
var ob = new myOb();
alert( ob.property );
```

class

```
class myOb{
 constructor(){
  this.property="тест";
  }
}
var ob = new myOb();
alert( ob.property );
```

На відмінно від інших мов програмування у JavaScript немає приватних властивостей, тобто властивість яка доступна лишу для об'єкта і не доступна для створеного екземпляра об'єкта.

Основні поняття ОПП

Уявіть, що ми проектуємо телефон. У нього буде екран, кнопки, зарядка і т.д. Телефон повинен мати можливість дзвонити, виходити в інтернет, фотографувати, тощо.

Клас

Ми описуємо всі запчастини з яких складається телефон, як ці запчастини взаємодіють між собою і що повинен зробити користувач, щоб телефон дзвонив, виходив в інтернет та інше.

Результатом роботи буде деякий ескіз (шаблон, схема). Ми щойно розробили те, що в ООП називається клас.

Клас - спосіб опису сутності, що визначає стан і поведінку, що залежить від цього стану, а також правила для взаємодії з цією сутністю (контракт).

У нашому випадку клас описує сутність – телефон. Властивістю класу будуть камера, модем, дисплей і т.д. Методами класу дзвонити, фотографувати і т.п.

Об'єкт

Ми спроектували креслення і схеми, за ними телефони сходять з конвеєра . Кожен з них точно повторює креслення, всі системи взаємодіють саме так, як ми спроектували, але кожен телефон є унікальним. Всі вони мають індивідуальний номер телефону , телефони відрізняються кольором, малюнком на панелі і тд. Ці телефони є екземплярами класу.

Об'єкт (примірник) - це окремий представник класу, що має конкретний стан і поведінку, що повністю визначається класом. Це те, що створено за кресленням, тобто за описом із класу.

Говорячи простою мовою, об'єкт має конкретні значення властивостей та методи, що працюють із цими властивостями на основі правил, заданих у класі. У даному прикладі, якщо клас це якийсь абстрактний телефон на кресленні, то об'єкт — це конкретний телефон, яким ви користуєтесь

Інтерфейс

Коли ми підходимо до автомата з кавою або сідаємо за кермо автомобіля, існує певний набір елементів керування, з якими ми можемо взаємодіяти.
Інтерфейс – це набір властивостей та методів класу, доступних для використання під час роботи з екземпляром.
Насправді, інтерфейс специфікує клас, чітко визначаючи всі можливі дії над ним.
Хороший приклад інтерфейсу - панель приладів автомобіля, яка дозволяє викликати такі методи як збільшення швидкості, гальмування, поворот, перемикання передач, включення фар і т. п.
При описі інтерфейсу класу дуже важливо дотриматися балансу між гнучкістю і простотою. Клас із простим інтерфейсом буде легко використовувати, але існуватимуть завдання, які за допомогою нього вирішити буде не під силу. Якщо інтерфейс буде гнучким, то швидше за все, він складатиметься з досить складних методів з великою кількістю параметрів, які дозволятимуть робити дуже багато, але його використання буде пов'язане з великими складнощами та ризиком зробити помилку, щось переплутавши.

ООП побудовано на чотирьох основних поняттях: інкапсуляція, абстракція, наслідування та поліморфізм.

Інкапсуляція

Внутрішні процеси роботи телефону досить складні. Але всі ці дії приховані від користувача і дозволяють йому дзвонити і писати, не замислюючись, що відбувається всередині. Саме приховування внутрішніх процесів, що відбуваються в телефоні, дозволяє ефективно використовувати його навіть новачкам. Це і є інкапсуляція.

Інкапсуляція (encapsulation) - це властивість системи, що дозволяє об'єднати дані та методи, що працюють з ними, у класі та приховати деталі реалізації від користувача.

Інкапсуляція пов'язана з поняттям інтерфейсу класу. Все те, що не входить до інтерфейсу, інкапсулюється (приховано) у класі. Користувач може працювати з усім функціоналом через інтерфейс, не замислюючись про те, як реалізований функціонал.

Абстракція

Коли ми телефонуємо ми не замислюватиметься про хімічний склад фарби панелі, особливості взаємодії внутрішніх деталей тощо. Однак це не заважає нам використовувати весь функціонал

Абстрагування (abstraction) — це спосіб виділити набір значних показників об'єкта, крім розгляду незначні.

Наслідування

Уявімо що інженерам поставили завдання розробки та випуску модельного ряду сучасніших сматфонів. При цьому вони вже мають стару модель, яка відмінно зарекомендувала себе. Очевидно, що інженери не проектуватимуть новий телефон з нуля, а взявши за основу попереднє покоління, внесуть низку конструктивних змін.

Усі модифікації матимуть більшість властивостей колишньої моделі. При цьому кожна з моделей реалізуватиме деяку нову функціональність або конструктивну особливість. У даному випадку ми маємо справу з успадкуванням.

Спадкування (inheritance) - це властивість системи, що дозволяє описати новий клас на основі вже існуючого, з частково або повністю функціоналом, що запозичується. Клас, від якого виробляється успадкування, називається базовим, батьківським чи суперкласом. Новий клас називається нащадком, спадкоємцем чи похідним класом.

Якщо взяти приклад із життя, у нас є HTML-елементи. У наступних модулях ми дізнаємося, що всі вони представлені об'єктами. У них є загальні властивості та методи для керування станом. Замість того, щоб додавати загальні властивості на кожен тип елемента, ми можемо описати їх у класі HTMLElement, і ті елементи, яким потрібен подібний інтерфейс, можуть успадковувати HTMLElement. Спадкування допомагає зменшити обсяг коду, що повторюється.

Поліморфізм

Було б не дуже зручно, якби всі телефони функціонували по різному і власник андроїду не міг зателефонувати з айосу чи навпаки.

Вся справа в тому, що основні елементи використання телефону мають одну й ту саму конструкцію та принцип дії. По суті, можна сказати, що всі сматфони мають той самий інтерфейс, а користувач, абстрагуючись від сутності смартфону, працює саме з цим інтерфейсом. Незалежно від того, яким чином буде реалізовуватись дзвінок, інтерфейс залишиться тим самим.

Поліморфізм (polymorphism) - це властивість системи дозволяє використовувати об'єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта. Дозволяє перевизначати у класах спадкоємців реалізації методів базового класу.

Додаткові матеріали

- http://xn--80adth0aefm3i.xn--j1amh/%D0%9E%D0%9E%D0%9F
- https://www.pcmag.com/encyclopedia/term/spaghetti-code
- https://scaleyourapp.com/spaghetti-code-explained/
- https://uk.javascript.info/object
- https://uk.javascript.info/class
- https://uk.javascript.info/class-inheritance
- https://uk.javascript.info/static-properties-methods
- https://uk.javascript.info/private-protected-properties-methods
- https://uk.javascript.info/extend-natives

Домашнє завдання

Створити телефонну книгу

- створити початковий клас Abonent, де зберігатимуться ім*я і номер
- створити set який прийматиме телефон і номер
- створити get який виводитиме данні про абонента
- створити три різних юзери
- вивести данні