Конспекты

Лекция 5

Здравствуйте, рад приветствовать на 5 лекции

СЛАЙД 3

Page Object Model - это шаблон проектирования, который стал популярным в автоматизации тестирования для улучшения обслуживания тестирования и **уменьшения дублирования кода**.

Это популярный паттерн, который является де-факто стандартом в автоматизации тестирования веб-продуктов. Основная идея состоит в том, чтобы **разделить логику тестов от реализации**.

Объект страницы - это объектно-ориентированный класс, который служит интерфейсом для страницы

Затем тесты используют методы этого класса объектов страницы всякий раз, когда им нужно взаимодействовать с пользовательским интерфейсом этой страницы.

Каждую веб-страницу проекта можно описать в виде объекта класса. Взаимодействие пользователя описываются в методах класса, а в тестах остается **только бизнес-логика**.

Данный подход помогает избежать проблем с тестами при **изменении верстки веб-приложения**. Вам необходимо поправить только класс, описывающий страницу, а не делать правки в десятках тестов, в которых используется какой-то популярный элемент.

Основное преимущество заключается в том, что если пользовательский интерфейс изменяется для страницы, сами тесты изменять не нужно, только код внутри объекта страницы должен измениться, **а не во всех тестах**.
Впоследствии все изменения для поддержки этого нового пользовательского интерфейса расположены в одном месте.

1) Избегаем дублирования кода
2) Если что-то меняется на странице, меняем это в одном месте, а не во всех тестах
3) Тесты и бизнес-логика отделены от более низкоуровневого взаимодействия со страницей
4) Такие тесты выглядят понятнее

РАЗБИРАЮ ПРИМЕР ГОТОВОГО СКРИПТА НЕ РО

СЛАЙД 4

СЛАЙД 5

СЛАЙД 6

Объект — это совокупность связанных данных и/или функциональных возможностей. Обычно состоят из нескольких переменных и функций, которые называются свойства и методы, если они находятся внутри объектов.

У нас сейчас внутри объекта есть некоторые данные и функционал, и теперь можно получить доступ к ним с помощью некоторого лёгкого и простого синтаксиса!

Объект состоит из нескольких элементов, каждый из которых имеет своё название (пример `name` и `age` выше), и значение (пример `['Bob', 'Smith']` и `32`). Каждая пара название/значение должны быть разделены запятой, а название и значение в каждом случае разделяются двоеточием. Синтаксис всегда следует этому образцу:

Значение члена объекта может быть чем угодно

Вы, вероятно, задаетесь вопросом, что такое "this"? Ключевое слово `this`, ссылается на текущий объект, внутри которого пишется код — поэтому в нашем случае `this` равен объекту `person`.

У объектов есть определение себя (self, this) и поведение, наследуемое от чертежа, т.е. класса (классовое наследование) или других объектов (прототипное наследование)

Объектно-ориентированное программирование (ООП) — это парадигма программирования, которая использует абстракции, чтобы создавать модели, основанные на объектах реального мира.

ООП представляет программное обеспечение как совокупность взаимодействующих объектов, а не набор функций или просто список команд (как в традиционном представлении).

Так как ООП настоятельно подчеркивает модульность, объектно-ориентированный код проще в разработке и проще для понимания впоследствии

СЛАЙД 7

После сохранения и обновления, попробуйте ввести что-нибудь следующее в консоль JavaScript браузера:

СЛАЙД 8

В классовом ООП классы являются чертежами для объектов. Объекты (или экземпляры) создаются на основе классов. Существует конструктор, который используется для создания экземпляра класса с заданными свойствами.

Здесь при помощи ключевого слова `class` из ES6 мы создаём класс `Person` со свойствами `firstName` и `lastName`, которые хранятся в `this`. Значения свойств задаются в конструкторе, а доступ к ним осуществляется в методе `getFullName()`.

Мы создаём экземпляр класса `Person` с именем `person` с помощью ключевого слова `new`:

Затем используйте вызов `new MyClass()` для создания нового объекта со всеми перечисленными методами.
При этом автоматически вызывается метод `constructor()`, в нём мы можем инициализировать объект.

Для расширения класса мы можем создать другой класс. Расширим класс `Person` с помощью класса `User`. `User` — это `Person` с почтой и паролем:

СЛАЙД 9

СЛАЙД 10

Ознакомьтесь факультативно с материалами. Т.к. тема уж слишком **обширна**, я не имею право вскользь касаться каких-то понятий. За эту тему лучше браться основательно, поэтому стоит подождать, что вам расскажут мои коллеги в лекциях, отдельно посвящённых этому вопросу.

Пока мы увидим на примерах в ходе работы, как наследуются классы (**Наследование** — способ сказать, что эти объекты похожи на другие за исключением некоторых деталей. Наследование позволяет ускорить разработку за счёт повторного использования кода.), как работает **this** и т.д. Пока этого понимания будет достаточно, главное – запомнить **синтаксис**.

Частое использование приведёт к тому, что вы интуитивно начнёте понимать нюансы работы, а потом как раз подкрепите теорией.

Но если не поймёте сами и сразу, то не стоит переживать, это обычная ситуация. В таком случае дождитесь следующих лекций.

Ресурсы.

СЛАЙД 11

ES — это просто сокращение для ECMAScript. Каждое издание ECMAScript получает аббревиатуру ES с последующим его номером. Всего существует 8 версий ECMAScript.
Для того, чтобы писать код, используя преимущества и более воспринимаемый синтаксис новых версий ES, можно использовать бэйбэл в качестве компилятора. Если коротко, то Babel - это набор инструментов, который в основном используется для преобразования кода ECMAScript 2015+ в обратно совместимую версию JavaScript в современных и старых браузерах или средах. Поэтому нужно сделать следующее:

НАПРИМЕР, ИМПОРТ

Устанавливаем библиотеки

Версия 5 WebdriverIO была разработана с учетом поддержки шаблонов объектов страниц. Внедрив принцип «элементы как граждан первого класса», теперь можно создавать большие тестовые наборы, используя этот шаблон.

Для создания объектов страницы не требуется никаких дополнительных пакетов. Оказывается, чистые современные классы предоставляют все необходимые функции, которые нам нужны:

Наследование между страницами и инкапсуляция методов и свойств

Цель использования объектов страницы - абстрагировать любую информацию страницы от реальных тестов. В идеале вы должны хранить все селекторы или специальные инструкции, которые являются уникальными для

определенной страницы, в объекте страницы, чтобы вы все равно могли запустить свой тест после того, как полностью переработали свою страницу.

Во-первых, нам нужен объект главной страницы, который мы вызываем `Page`. Он будет содержать общие селекторы или методы, от которых будут наследоваться все объекты страницы.

НАЧИНАЮ ПИСАТЬ КОД НА РО

СЛАЙДЫ 12, 13, 14, 15

СЛАЙД 16

ДЗ. Следующая лекция – 23.03 в 17:00. Так же, как и сегодня, – удалённый формат. Это будет **НЕ ВОРКШОП**