Паттерны проектирования очень важны для веб-разработчиков, и мы можем писать лучший код, освоив их. В этой статье я буду использовать TypeScript, чтобы представить Паттерн Наблюдатель и Паттерн Издатель-Подписчик.

Паттерн Наблюдатель

Паттерн Наблюдатель (Observer) широко используется в веб-приложениях - MutationObserver, IntersectionObserver, PerformanceObserver, ResizeObserver, ReportingObserver. Все эти API можно рассмотреть как примеры Паттерна Наблюдатель. Кроме того, этот паттерн также используется для мониторинга событий и реагирования на мутации данных (например, при изменении данных и автоматическом обновлении страницы).

Паттерн Наблюдатель определяет отношение «один ко многим», и этим позволяет нескольким объектам-наблюдателям одновременно отслеживать наблюдаемый субъект. При изменении состояния наблюдаемого субъекта все объекты-наблюдатели будут уведомлены о таком изменении, чтобы они в свою очередь могли автоматически обновить свое состояние.

В Паттерне Наблюдатель есть две основные роли: Субъект и Наблюдатель.

На диаграмме выше в качестве Субъекта выступает моя статья (Article), а Наблюдателями являются Chris1993 и Bytefish. Паттерн Наблюдатель поддерживает простейшую связь в режиме широковещательной передачи, и поэтому все наблюдатели автоматически уведомляются о публикации новой статьи.

Чтобы лучше понять дальнейший код, давайте сначала изучим соответствующую ему диаграмму UML:

На приведенном выше рисунке мы с помощью механизмов интерфейса определяем непосредственно интерфейс Observer и интерфейс Subject соответственно, которые используются для описания объектов Observer и Subject.

Затем мы определяем классы ConcreteObserver и ConcreteSubject, в которых реализуются вышеуказанные интерфейсы:

Давайте проверим работу функций в наших классах на примере вот таких команд:

В результате выполнения вышеуказанного кода, терминал выведет такой результат:

В настоящее время я пишу на две основные тематики - JavaScript и TypeScript. Поэтому, если я захочу опубликовать новую статью, то об этом необходимо уведомить только читателей, интересующихся JavaScript, или только читателей, интересующихся TypeScript. Если мы используем Паттерн Наблюдатель, нам следует создать два разных Субъекта. Но с другой стороны можно поменять решение и использовать паттерн Издатель-Подписчик.

Паттерн Издатель-Подписчик

В архитектуре программного обеспечения Издатель-Подписчик — это парадигма обмена сообщениями, в которой отправители сообщений (называемые издателями) не отправляют сообщения напрямую конкретным получателям (называемые подписчиками). Вместо этого опубликованные сообщения группируются по разным категориям и отправляются к разным подписчикам. Аналогичным образом подписчики могут интересоваться одной или нескольким категориями сообщений и получать только такие интересующие их сообщения, не зная о существовании издателей.

В паттерне Издатель-Подписчик есть три основные роли: Издатели, Каналы и Подписчики.

На приведенном выше рисунке Издатель — это Bytefer, тема A и тема B в Каналах соответствуют теме JavaScript и теме TypeScript соответственно, а Подписчики — Chris1993, Bytefish и т. д.

Давайте реализуем класс EventEmitter на основе паттерна Издатель-Подписчик:

После определения класса EventEmitter мы можем использовать его следующим образом:

Если запустить на исполнение вышеприведенный код то терминал выведет следующий результат: Received: Observer pattern.

В событийно-ориентированной архитектуре паттерн Издатель-Подписчик играет важную роль. Конкретная реализация этого паттерна может использоваться в качестве шины событий для реализации обмена сообщениями между различными компонентами или модулями в одной системе. Для популярной архитектуры подключаемых модулей паттерн Издатель-Подписчик можно использовать для реализации обмена сообщениями между различными подключаемыми модулями.

Надеюсь, после прочтения этой статьи у вас появилось определенное представление о паттерне Наблюдателя и паттерне Издатель-Подписчик. Если у вас есть какие-либо вопросы, пожалуйста, не стесняйтесь написать мне сообщение. В дальнейшем я продолжу знакомить вас с другими паттернами, и если вам это интересно, можете подписаться на меня в Medium или Twitter.