# Блок 4. Функциональное программирование на JavaScript

## Занятие 2. Иммутабельность. Написание и тестирование чистых функций.

Давайте изменим наш код таким образом, чтобы везде использовались только иммутабельные преобразования и чистые функции.

Вот пример использования иммутабельного варианта функции removeEmployee:

**function** *removeEmployee*(employees, id) {  
 **return** employees.**filter**(e=>e.**id**!==id);  
}

Создайте файл employees/service.pure.js и поместите туда иммутабельный removeEmployee.

Как вы видите, выглядит эта функция значительно лаконичнее, зависит только от входных данных, а в качестве результата возвращает только итоговое значение.

Теперь вам нужно самостоятельно переписать ряд функций из service.js, изменяющих DATA так, чтобы они были чистыми иммутабельными функциями:

* addEmployee
* addPhone
* setDateOfBirth
* setEmployeeManager

**View как чистая функция.**

Как мы рассматривали в лекционной части, в паттерне MVC используют разделение программы на Model, View и Controller. При этом данные модели по возможности должны быть неизменяемыми, а методы View и контроллера – чистыми функциями. Чтобы реализовать эту функциональность, реализуем View как чистую функцию.

Начнем со списка сотрудников. Сейчас генерация этого списка управляется функцией showEmployees – достаточно монструозным методом, который занимается манипуляцией DOM. Проблема с манипуляциями DOM в том, что эти операции не являются чистыми и приводят к постоянному изменению состояния (в данном случае – HTML). Когда код становится большим и сложным, управлять мутациями DOM становится крайне затруднительно. Поэтому в качестве альтернативы можно использовать View как чистую функцию:

Данные 🡪 функция View 🡪 HTML

Одна функция View привязана к одному месту на странице – например, showEmployees генерирует код для списка сотрудников. Результирующий HTML заменяет предыдущий список. То есть вместо постепенного создания мы используем полную замену. Хотя такая манипуляция и менее эффективна, она гораздо проще с точки зрения написания кода и позволяет создавать и поддерживать достаточно большой и сложный код. Вопрос неэффективности работы при полной подмене HTML решается за счет использования виртуальной DOM. Но проблемы с производительностью могут возникнуть только в случае работы с очень большими HTML-фрагментами. Итак, наша функция showEmployees, которая сейчас манипулирует DOM, будет использовать новую чистую функцию внутри себя– showEmployeesView, которую нужно написать.

showEmployees не является чистой, но она тривиальна, и вызывает чистую функцию showEmployeesView. Та, в свою очередь, решает простую задачу: принимает employees и возвращает HTML со списком employees:

employees 🡪 HTML

Попробуйте самостоятельно переписать метод addEmployeeUI так, чтобы он стал чистой функцией.