
Яндекс

Яндекс

Шаблонизация

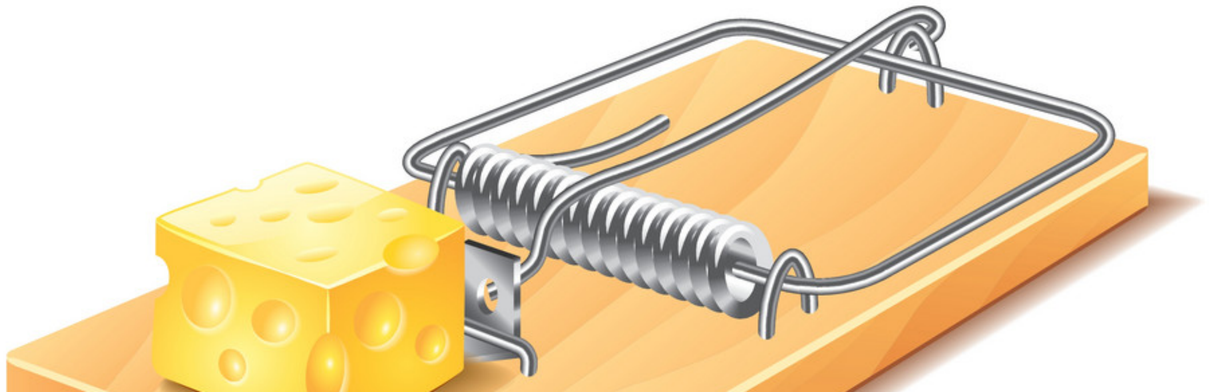
Главная задача фронтэнд разработчика

Главная задача фронтэнд разработчика

Какой бы сервис вы не делали, вы всегда работаете с данными, которые где-то хранятся. Иногда у вас, иногда - нет. Иногда в базе данных, иногда - в файлах, иногда - где-то еще.

Раньше довольно часто работу с данными и формирование того или иного интерфейса объединяли в одно приложение.

В чем минусы такого подхода?



Главная задача фронтэнд разработчика

- В случае наличия разных интерфейсов для одних и тех же данных (мобильное приложение, сайт и так далее) часть про данные придется писать для каждого приложения

- Если вы хотите шаблонизировать и на сервере, и на клиенте тоже, то вам придется писать разный код для одного и того же

Разделение фронтэнда и бекэнда

Сейчас все чаще приложения проектируются так, что бекэнд часть полностью отделена от фронтэнд части, это абсолютно разные не пересекающиеся проекты, более того, разработчики этих частей, казалось бы, одного сервиса, могут быть вообще не знакомы.

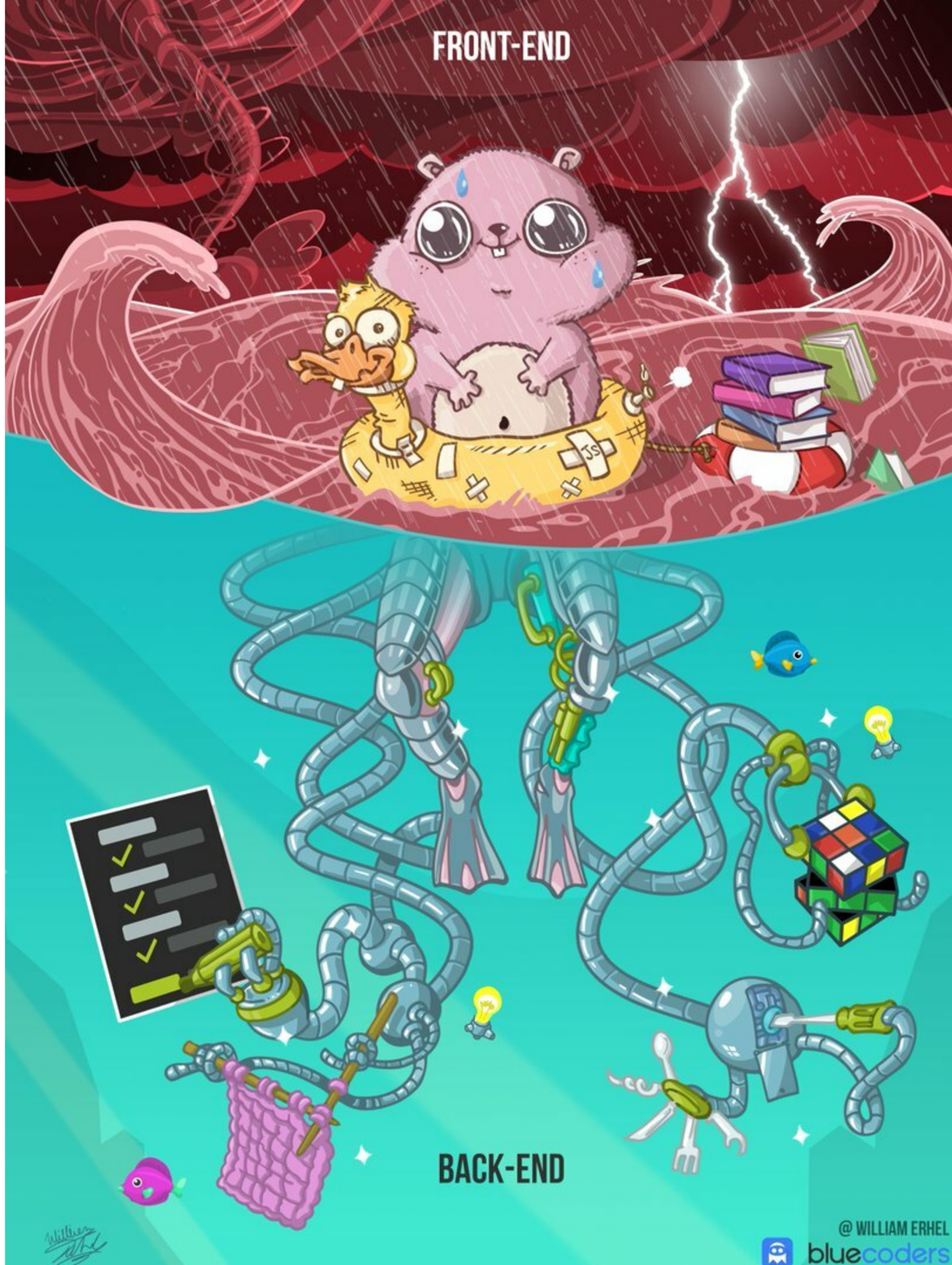
В итоге бекэнд предоставляет данные по http-ручкам (если по-простому - то с помощью ссылок, url), запрашивая данные из которых, вы получаете в ответ текст, обычно, в формате JSON

JSON - несложный, интуитивно понятный формат, очень похожий на то, как в JS объявляются объекты. Он очень распространен и поддержан во многих языках из коробки

Разделение фронтэнда и бекэнда

Плюсы такой архитектуры, помимо нивелирования недостатков:

- **Не толкаемся локтями в проекте**
- **Раскатив такое API наружу, может получить комьюнити разработчиков, которое пользуется нашим API**

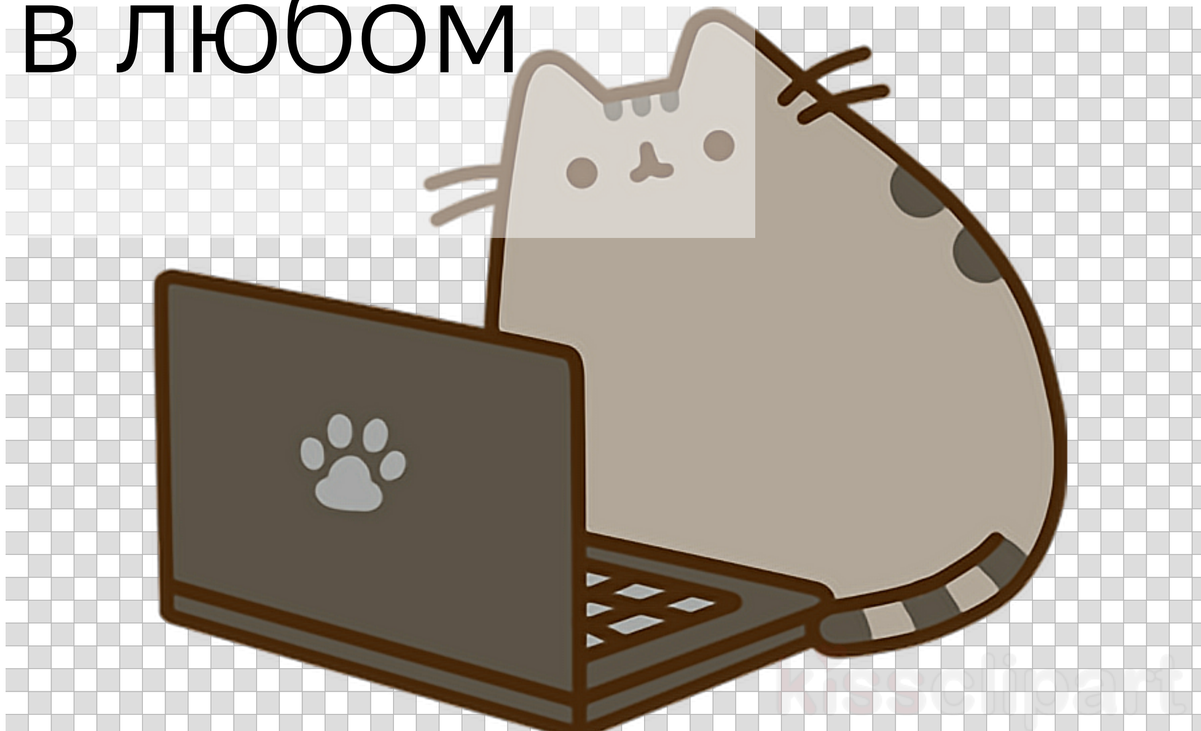


Пример несложного запроса в публичный API

Главная задача фронтэнд разработчика

- Итого, вы всегда будете работать с API - внешним или внутренним
- От API вы будете получать строку в формате JSON - как получать, сможете узнать на занятии про AJAX
- Строку вы всегда преобразуете в объект или что-то другое с помощью JSON.parse
- Теперь ваша задача сварить из этого объекта верстку

Это и есть ваша главная задача, которая будет присутствовать в любом проекте



Способы генерации верстки

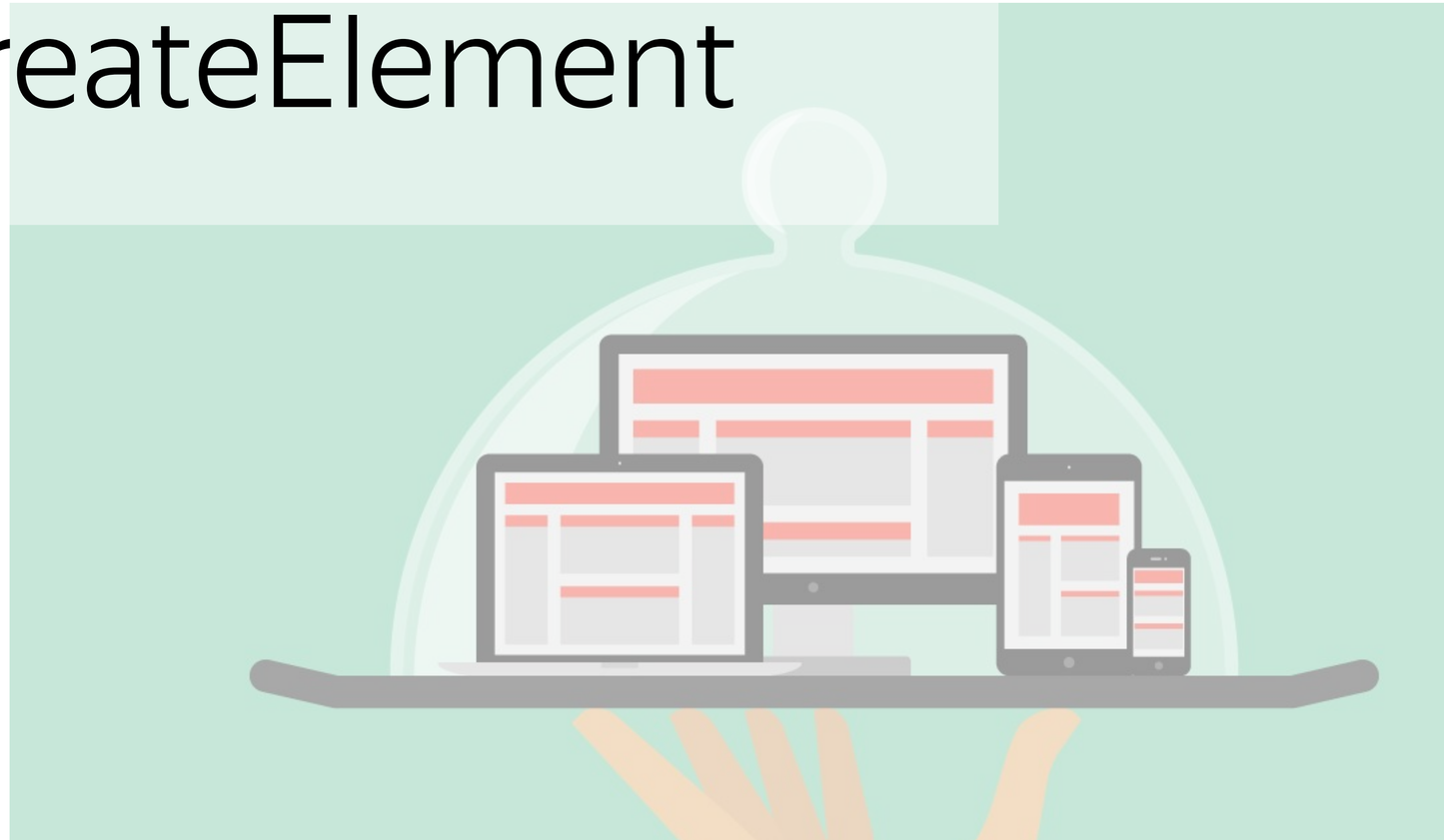
Просто клеить строки и innerHtml

innerHtml

- Просто и быстро
- Кроссплатформенно
- Благодаря древовидной структуре HTML видна структура несложного сниппета
- Незащищены от опечаток и ошибок вложенности
- В сложном сниппете html расплывется его структура
- XSS-угрозы

Методы создания верстки в браузере

document.createElement



document.createElement

- Создает объект-узел, с именем тега равным переданной строке. Например document.createElement('button') создаст кнопку

- Помимо имени тега, создаваемый элемент по функциональности будет тоже правильным, то есть - кнопкой

- Это полноценный элемент, можно менять у него контент, добавлять события и т.п.

element.appendChild

element.appendChild

- Добавляет аргумент в конец списка детей element
- Аргументом должен быть объект узел, либо «нечто похожее», например, строка - не прокатит
- Если аргументом будет объект-узел, присутствующий в документе - то этот узел будет перемещен

element.insertBefore

element.insertBefore(node, child)

- Добавляет аргумент node в список детей element, перед существующим ребенком child
- Поначалу может показаться неудобным
- Тем не менее, на пару с appendChild позволяют вставить блок в любое место дерева документа
- Также, как и appendChild, телепортирует блок, если node присутствует на странице

Примеры

Вставить элемент в начало родителя:

```
parent.insertBefore(element, parent.FirstChild)
```

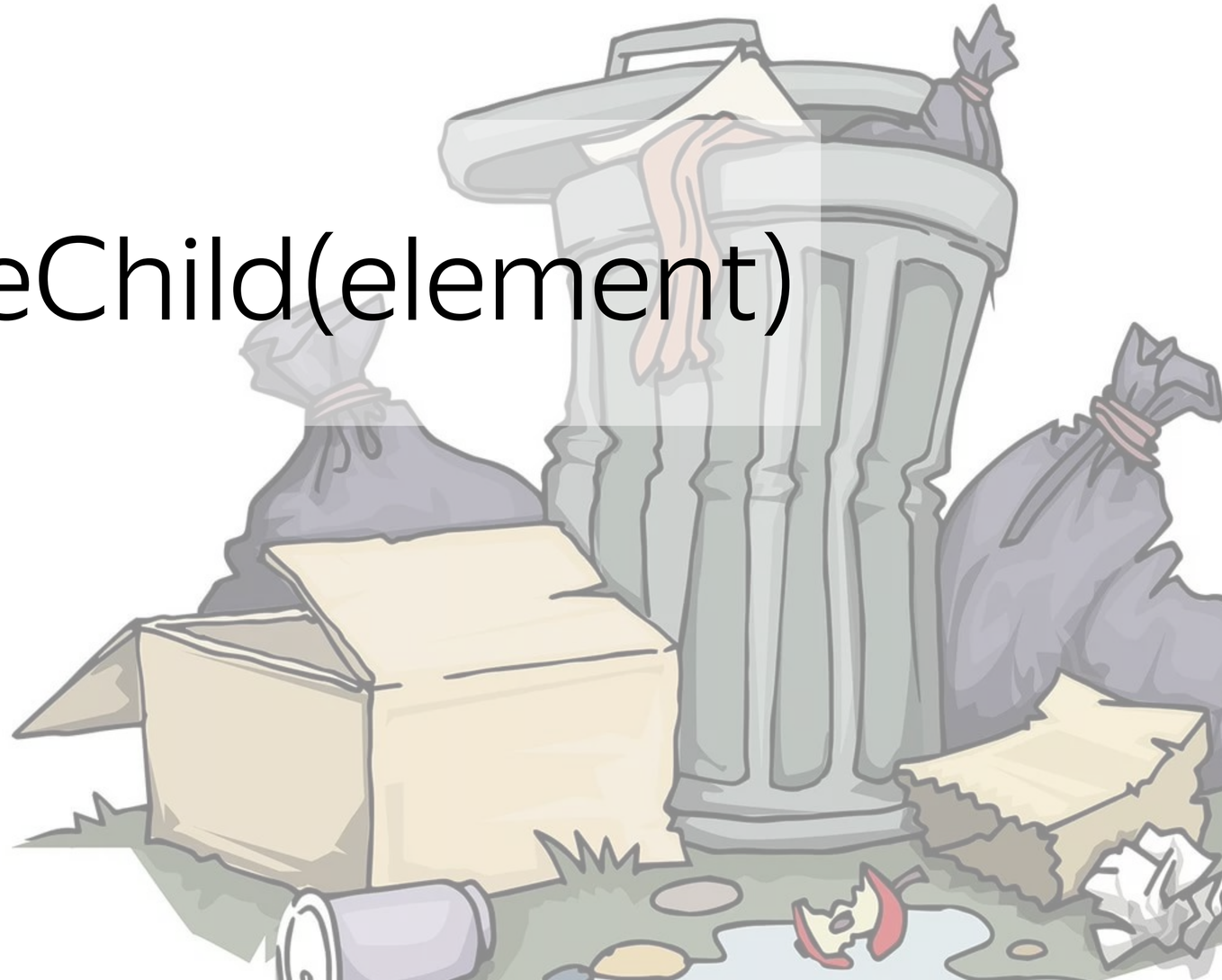
Вставить элемент до текущего элемента:

```
currentElement.parentElement.insertBefore(
    element, currentElement
)
```

Вставить элемент после текущего элемента:

```
currentElement.parentElement.insertBefore(
    element, currentElement.nextSibling
)
```

parent.removeChild(element)



element.cloneNode(true)

document.createDocumentFragment();

document.createTextNode('');

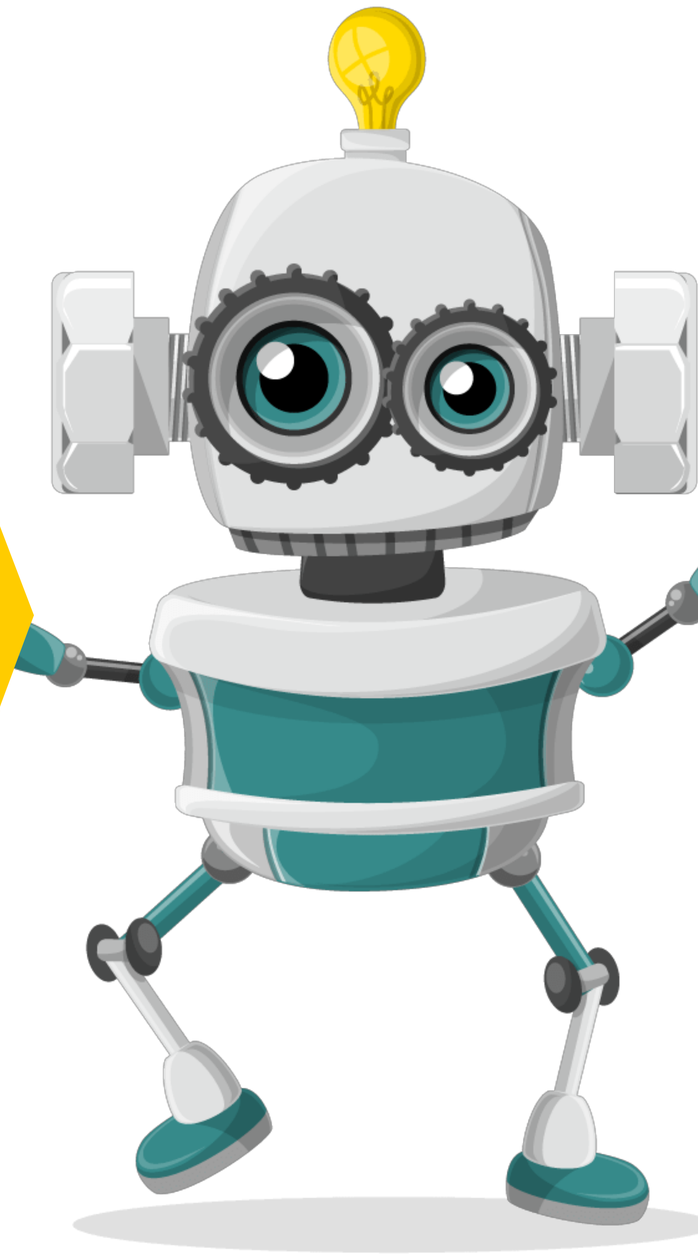
Шаблонизируем средствами браузера

Браузерная шаблонизация

- Никаких XSS
- Не может быть ошибок вложенности
- Быстрее innerHTML

- Очень, очень много кода
- Код плохо структурирован, невозможно понять как сниппет будет выглядеть в интерфейсе.
- Работает только в браузере

Пишем небольшой шаблонизатор



Яндекс

Спасибо

Шлейко Александр

Разработчик интерфейсов

 dusty@yandex-team.ru

 @dustyo_O