PHASE 2

WEEK 1

DAY 3

ELBRUS/BOOTCAMP

План

1. AJAX
2. Fetch API
3. JSON

ELBRUS/BOOTCAMP

AJAX

Asynchronous Javascript and XML

ELBRUS/BOOTCAMP

Ajax и Fetch

Ajax — *подход* в веб-разработке для асинхронного обновления страниц.

Fetch API — одна из *реализаций* подхода Ajax.

Как было до AJAX

- Клик по ссылке → смена страницы целиком
- Отправка формы → полная перезагрузка страницы
- Сервер обрабатывает запрос → страница “висит”

ELBRUS/BOOTCAMP

Минусы обновления страницы

- Большой объём передаваемых данных
- Затраты на повторную отрисовку элементов
- Затраты на повторный запуск скриптов

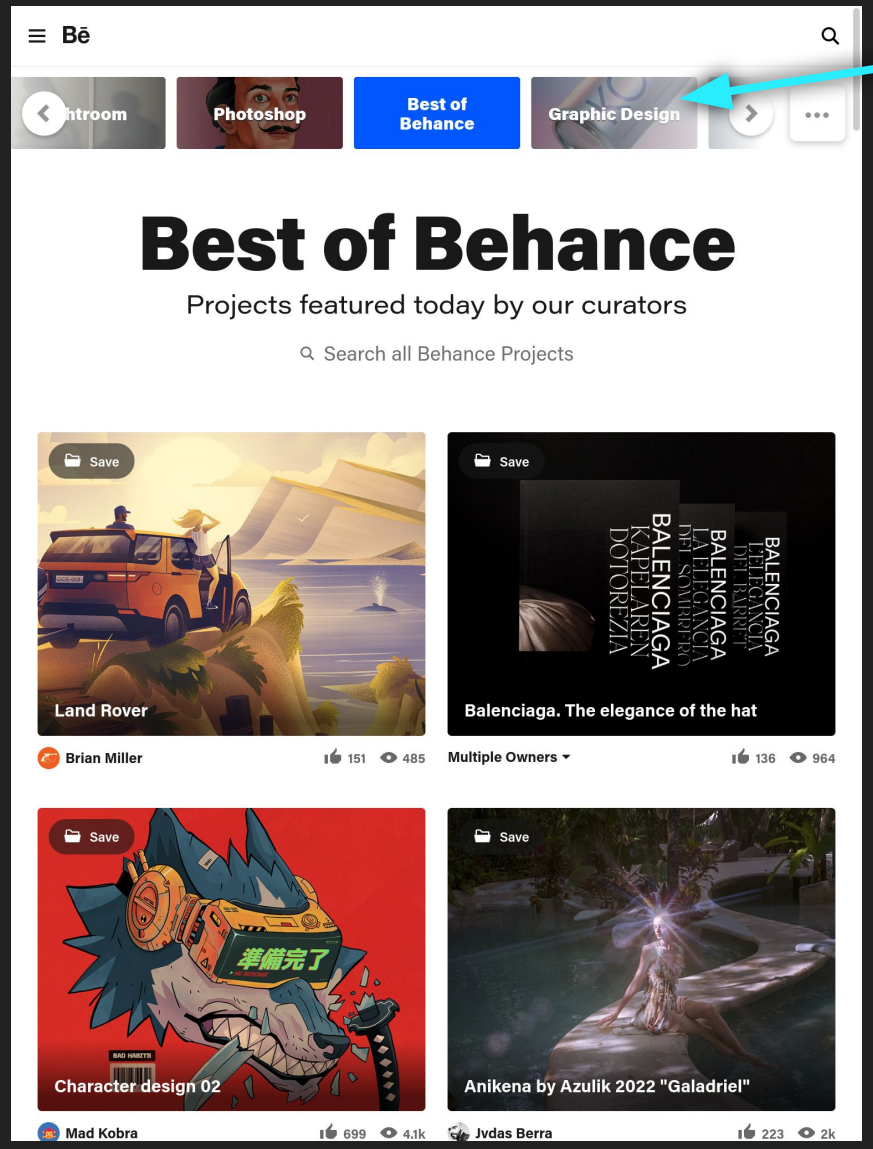
ELBRUS/BOOTCAMP

С использованием AJAX

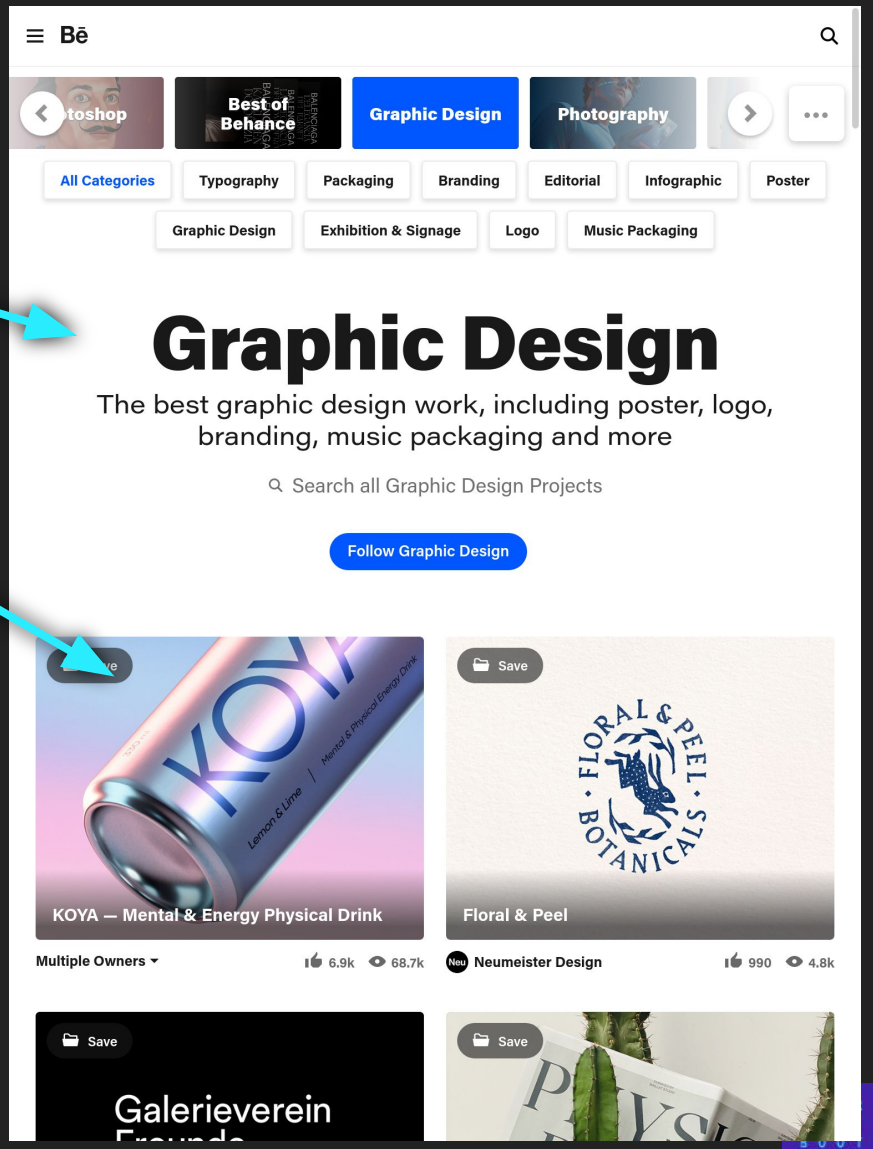
- Запросы на сервер асинхронные → страница не заблокирована
- Не нужно заново выполнять все скрипты
- Отрисовываем только то, что изменилось

ELBRUS/BOOTCAMP

AJAX в действии



1. Нажимаем на ссылку
2. Часть страницы обновилась
3. Полной перезагрузки не было



Fetch API

ELBRUS/BOOTCAMP

Fetch API

Браузерная технология, один из Web API.

Инструмент для реализации подхода AJAX.

https://developer.mozilla.org/ru/docs/Web/API/Fetch_API/Using_Fetch

ELBRUS/BOOTCAMP

Fetch: GET

```
// Отправить GET запрос по указанному URL
fetch('https://some-url.example');
```

Возвращает объект Promise, который выполняется в объект класса Response.

Fetch: POST

```
// Отправить POST запрос по указанному URL
fetch('https://some-url.example', {
 method: 'POST',
 headers: {
   'Content-Type': 'application/json',
 },
 body: JSON.stringify({
   myKey: 'my value',
 }),
});
```

Response

```
// Отправить GET запрос по указанному URL
const response = await fetch('https://some-url.example');

// Прочитать тело ответа в JSON-формате
const body = await response.json();
```

Методы чтения тела ответа (`json`, `text`, и т. д.) возвращают объект Promise.

Прочитать тело ответа можно лишь один раз!

ELBRUS/BOOTCAMP

JSON

JavaScript Object Notation

AJAX и JSON

Изначально AJAX задумывался для работы с XML.

Сейчас же повсеместно используется JSON.

JSON, example code

```
[
 {
   "id": 1,
   "name": "Leanne Graham",
   "username": "Bret",
   "email": "Sincere@april.biz",
   "address": {
     "street": "Kulas Light",
     "city": "Gwenborough",
     "geo": {
       "lat": "-37.3159",
       "lng": "81.1496"
     }
   },
   "phone": "1-770-736-8031 x56442",
 }
]
```

ELBRUS/BOOTCAMP

Submit Form, example code

```
form.addEventListener('submit', async (event) => {
 // отменяем действие формы по умолчанию
 event.preventDefault()

 // через деструктуризацию забираем имя input из формы (name="code")
 const { code } = event.target

 // метод fetch возвращает promise, создает запрос к серверу (по умолчанию
GET, если не указано явно)
 const response = await fetch(`http://localhost:3000/emo?smile=${code.value}`)

 // в зависимости от ответа сервера решаем как будем форматировать его на
клиенте, если это JSON -> JSON, если text -> text
const data = await response.text()

 // вывод ответа сервера в консоль браузера
 console.log(data)
})
```

One Event Listener, example code

```
// HTML
<dl id="todoList">
  <dt>Почистить зубы</dt>
  <dd><button class="done" type="button">Сделано</button></dd>
  <dt>Накормить кота</dt>
  <dd><button class="done" type="button">Сделано</button></dd>
  <dt>Купить молоко</dt>
  <dd><button class="done" type="button">Сделано</button></dd>
</dl>
```

```
// JS
document.getElementById('todoList').addEventListener('click', ({ target
}) => {
  if (target.classList.contains('done')) {
    // TODO: Fetch
  }
});
```

ELBRUS/ BOOT CAMP

Ссылки

- Fetch: https://learn.javascript.ru/fetch
- PreventDefault: https://learn.javascript.ru/default-browser-action

ELBRUS/BOOTCAMP