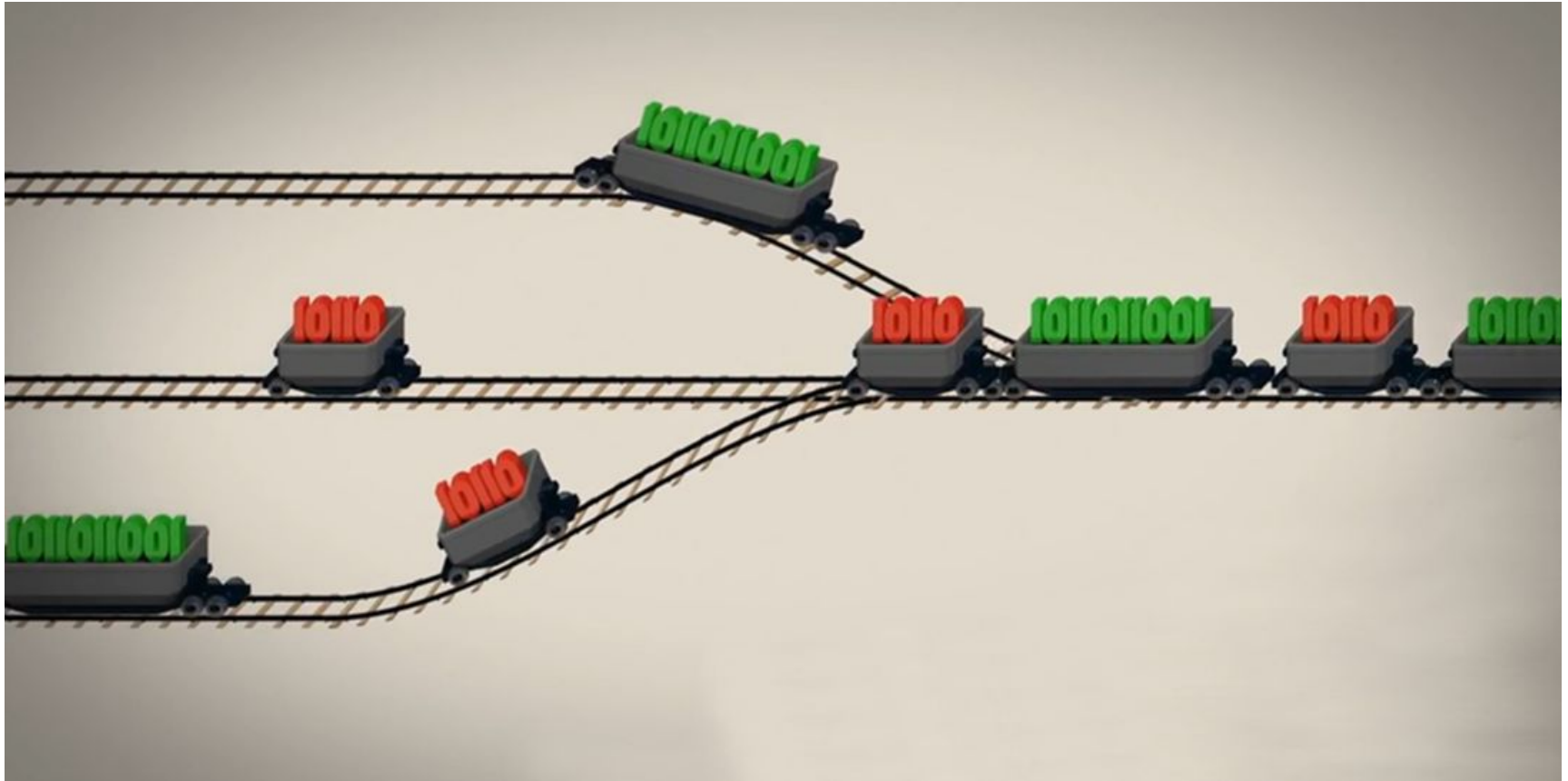


JS и AJAX

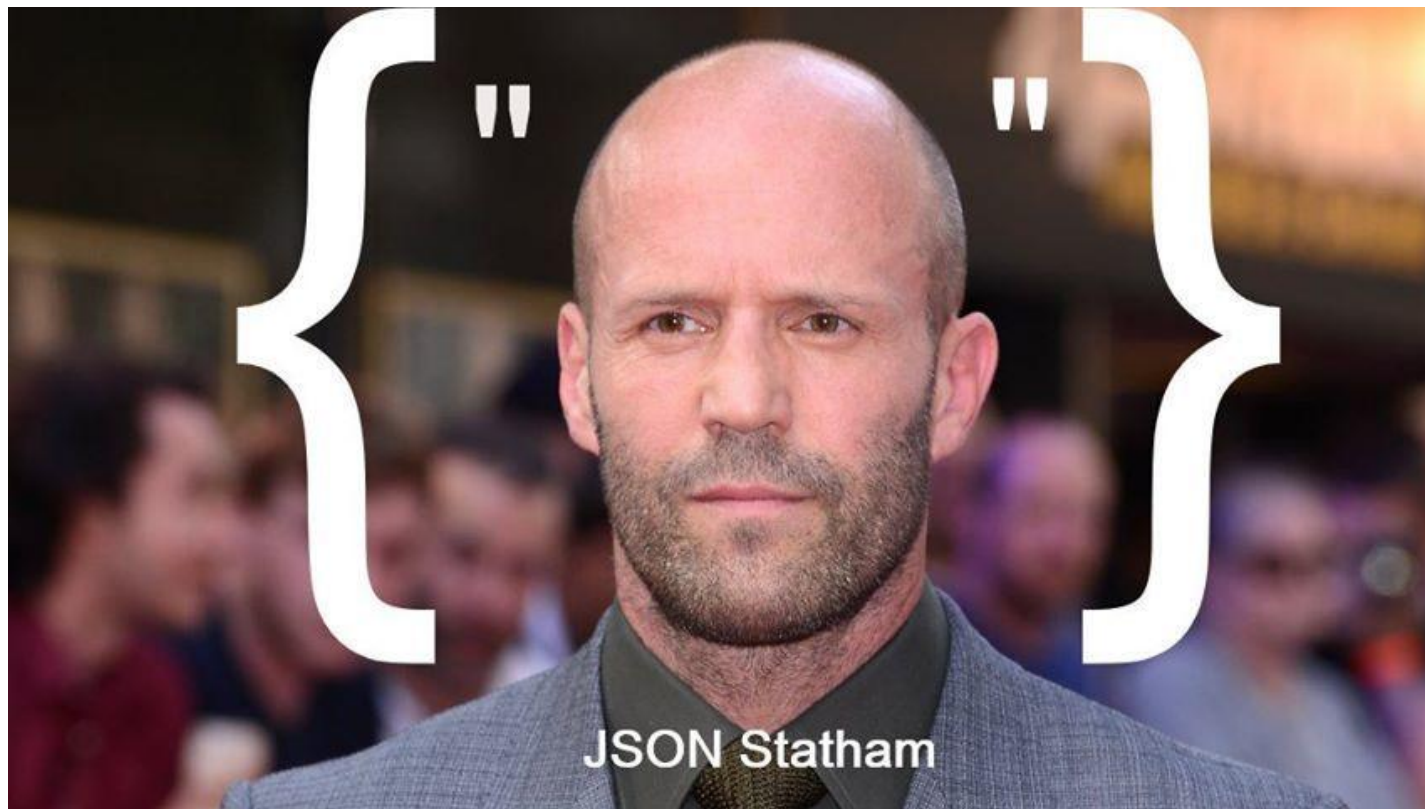
AJAX (***A**synchronous **J**avascript **A**nd **X**ml*)



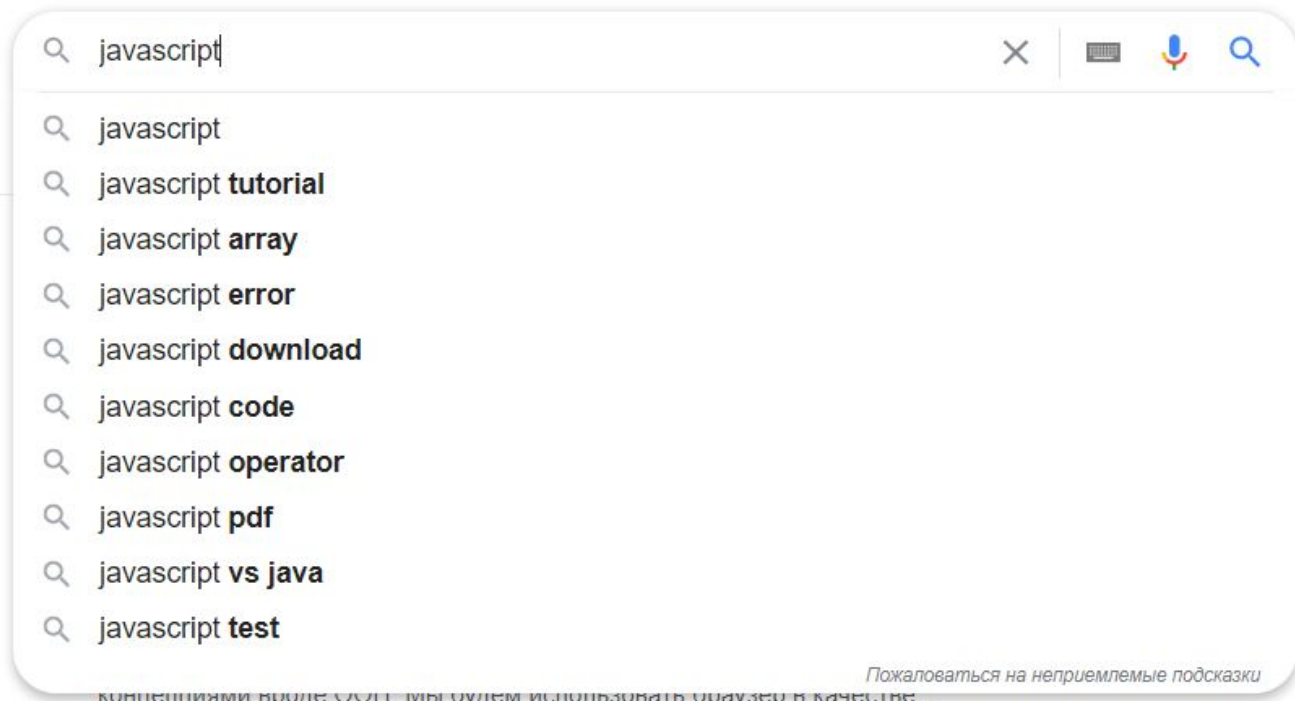
# Синхронность и асинхронность процессов



JSON (ДжЭйсон или ДжисОн)



# Пример использования | Живой список



# Пример использования | Корзина



Бestseller

**Чистый код. Создание, анализ и рефакторинг. Библиотека программиста | Мартин Роберт К.**

★★★★★ 52 отзыва [В избранное](#) [||](#) [Сравнить](#) [→](#) [Поделиться](#)

Тип книги:

 Печатная книга (6)

Другие издания:



544 Р



Нет в наличии



Нет в наличии



Нет в наличии



Нет в наличии

Нашли дешевле?


**544 Р**

Б 27 баллов при оплате [Ozon.Card](#)

**Добавить в корзину**

Автор ..... [Мартин Роберт К.](#)

# Пример использования | Форма



## Создайте аккаунт Google

Имя

Фамилия

Имя пользователя


@gmail.com

Можно использовать буквы латинского алфавита, цифры и точки.

[Использовать текущий адрес электронной почты](#)

Пароль


Подтвердить



Пароль должен содержать не менее восьми знаков, включать буквы, цифры и специальные символы

[Войти](#)

Далее



Один аккаунт – для всех сервисов Google.

# Стрелочные функции (arrow function)

```
let sum = (a, b) => a + b;
```

/\* Более короткая форма для:

```
let sum = function(a, b) {
```

```
    return a + b;
```

```
};
```

```
*/
```

```
console.log(sum(1, 2));
```



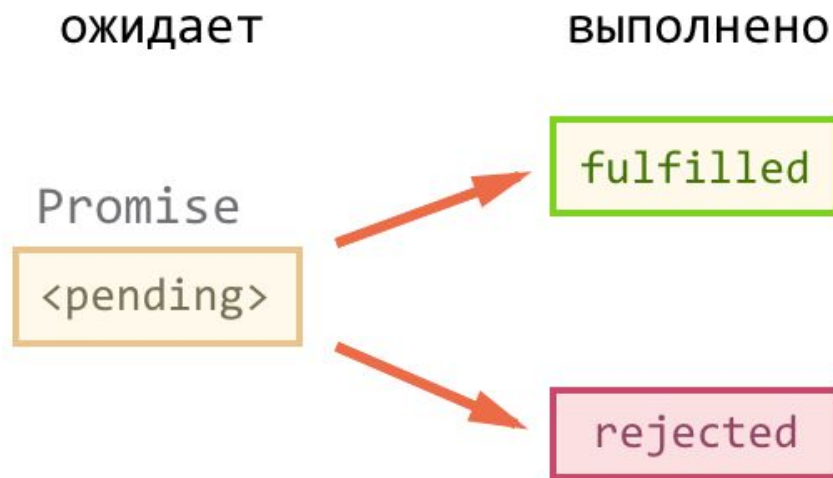
# XMLHttpRequest (*XHR*)

```
const xhr = new XMLHttpRequest();  
xhr.open(METHOD, URL);  
xhr.send([body]);
```

# События объекта XMLHttpRequest

```
xhr.onload = () => {  
    alert(`Загружено: ${xhr.status} ${xhr.response}`);  
};  
  
xhr.onerror = () => {  
    alert(`Ошибка соединения`);  
};  
  
xhr.onprogress = (event) => {  
    // event.loaded - количество загруженных байт  
    // event.total - количество байт всего (только если lengthComputable равно true)  
    alert(`Загружено ${event.loaded} из ${event.total}`);  
};
```

# Promise (Промис)



# Создание промиса

```
const promise = new Promise((resolve, reject) => {  
  setTimeout(() => {  
    if (Math.random() > 0.5) {  
      resolve('result');  
    } else {  
      reject('o_o');  
    }  
  }, 1000)  
})
```

# Обработка промиса

```
promise
  .then(
    value => console.log('Fulfilled: ', value),
    error => console.log('Rejected: ', error)
  );

promise
  .catch(error => console.log('Rejected: ', error));
```

# Fetch

```
const promise = fetch(url, [options])
```

# Response

```
const response = await fetch(url);  
if (response.ok) { // если HTTP-статус в диапазоне 200-299  
    const json = await response.json();  
} else {  
    alert("Ошибка HTTP: " + response.status);  
}
```

# async / await

```
async function doFetch() {  
  const response = await fetch(url);  
  if (response.ok) {  
    const json = await response.json();  
    console.log(json)  
  }  
}
```

```
const responsePromise = fetch(url);  
responsePromise.then(response => {  
  if (response.ok) {  
    const jsonPromise = response.json();  
    jsonPromise.then(json =>  
      console.log(json)  
    )  
  }  
})
```