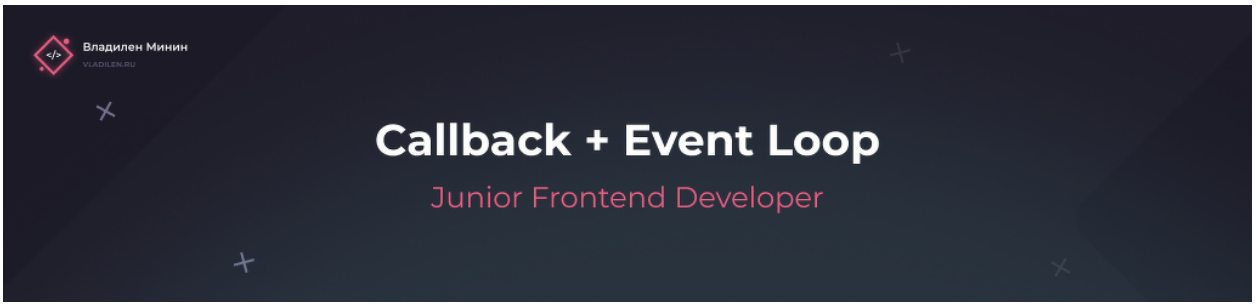

Цель урока:

Вы узнаете, что такое callback и как его использовать. Также будет разобрана популярная тема Event Loop (событийный цикл), на которую часто задают вопросы на собеседованиях.

Содержание урока:

1. Что такое callback и как его использовать в коде?
2. Event Loop. Что такое Call Stack?
3. Event Loop. Что такое Очередь Вызовов (Callback queue)?
4. Event Loop. Макрозадачи и микрозадачи. Очередь микрозадач.

Дополнительные материалы:

1. Асинхронность.Что такое Event Loop. JS SetTimeout 0: https://youtu.be/vIZs5tH-HGQ
2. Событийный цикл: микрозадачи и макрозадачи: https://learn.javascript.ru/event-loop

---

Задание #1

Напишите, в каком порядке будет выводится информация в консоль. Не вызывайте код, **попробуйте сами** написать правильный ответ **на листочке либо в заметках**.

```
setTimeout(() => {
 console.log('setTimeout');
}, 0);

const promise = new Promise((resolve) => {
 console.log('Promise');
 resolve();
});

promise.then(() => {
 console.log('Promise resolve');
});

console.log('End');
```

Задание #2

Напишите, в каком порядке будет выводится информация в консоль. Не вызывайте код, **попробуйте сами** написать правильный ответ **на листочке либо в заметках**.

```
function runCode() {
 console.log('before promise')
 return new Promise((resolve) => {
   setTimeout(() => {
     console.log('Zero Promise')

     resolve()
   }, 0)
 })
}

setTimeout(() => {
 console.log('Zero')
}, 0)

runCode().then(() => console.log('Zero Promise Invoked'))

console.log('One')
```

Задание #3

Напишите, в каком порядке будет выводится информация в консоль. Не вызывайте код, **попробуйте сами** написать правильный ответ **на листочке либо в заметках**.

Примечание: функция **getData** не выбрасывает ошибку.

```
const getData = (callback) => {
 fetch('https://jsonplaceholder.typicode.com/users/1')
   .then((response) => response.json())
   .then((user) => {
     console.log('Success');
     callback(user);
   })
   .catch((error) => {
     console.log(error);
   });
}

getData(() => {
 console.log('user received');

 const promise = new Promise((resolve) => {
   setTimeout(() => {
     resolve('promise resolved');
   }, 500);
```

```
    console.log('in promise');
    setTimeout(() => {
      console.log('last set timeout');
    }, 400);

  });

  promise.then((result) => {
    console.log(result);
  })
});

console.log('End')
```