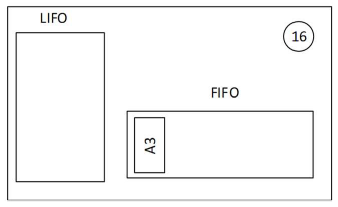
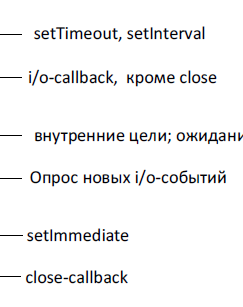
**Задание 7**

Про Event loop.



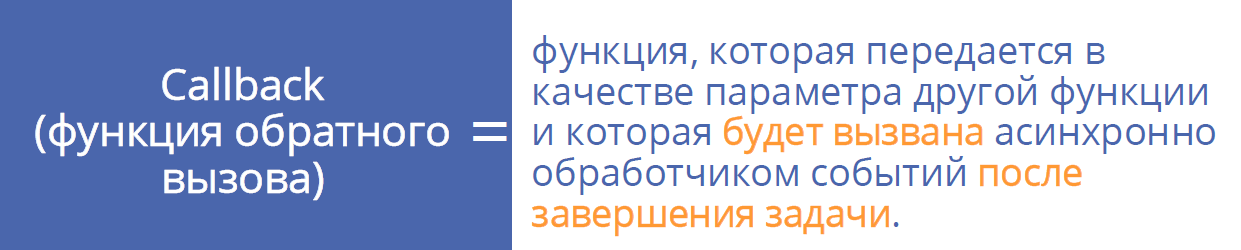
В LIFO помещаются макрозадачи (по одной задаче за один проход цикла):

, которые работают по принципу стека, т.е.

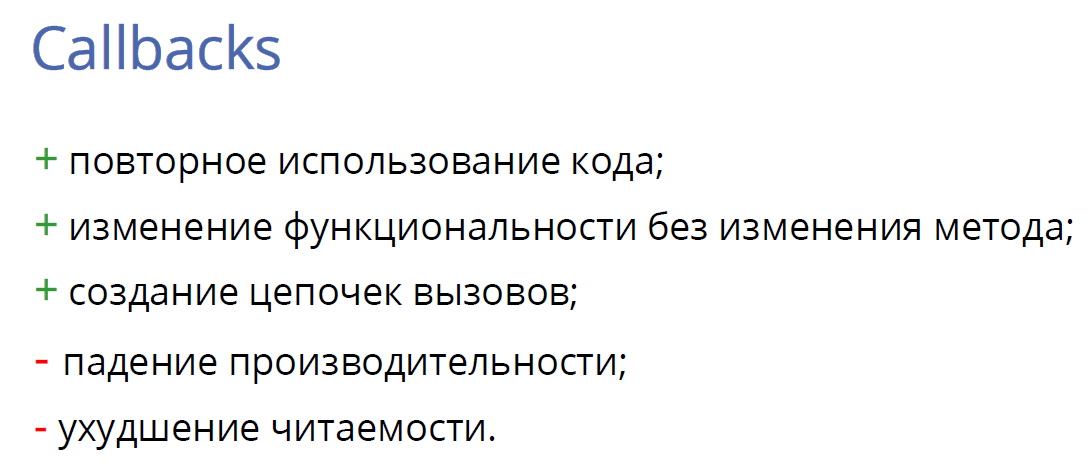
Вначале main, затем должны setTimeout (но там временная задержка, поэтому не добавляются), затем два промиса добавляются в очередь (FIFO), далее добавляется вызов функции f2. После вызова функции f2 отчищается стек и вызываются микрозадачи из очереди. В самом конце выполняются

setTimeout(f1, 50);  
setTimeout(f2, 30);

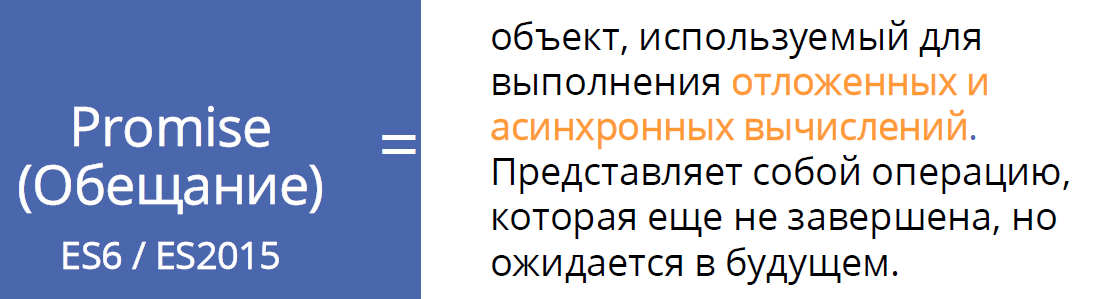
1. Что такое **callback**?



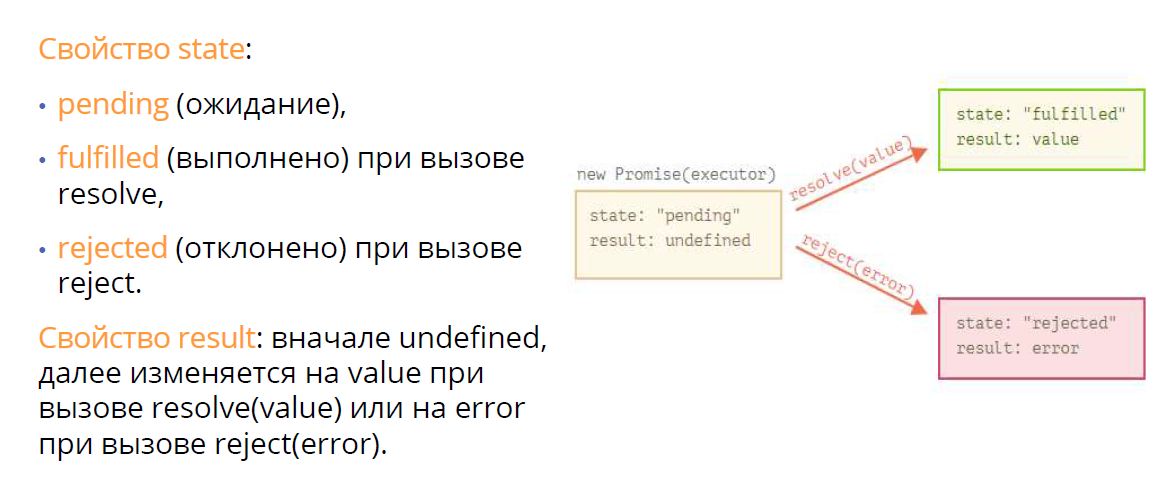
1. В чем минусы использования коллбэков? Какие есть способы их решения?



1. Что такое **Promise** и как он работает?



1. В каких **состояниях** может находиться Promise?



1. Как изменить состояние Promise?

Для изменения состояния Promise вы можете использовать функции `resolve` и `reject`, переданные в функцию-исполнитель промиса. Вызовите функцию `resolve` для успешного завершения промиса и передайте ей соответствующее значение. Вызовите функцию `reject` для завершения промиса с ошибкой и передайте ей соответствующую ошибку.

1. Как изменить значение Promise?

Значение промиса нельзя изменять после его создания. Промис возвращает значение только один раз, при его разрешении (`resolve`) или отклонении (`reject`).

Если вам нужно изменить значение промиса в последующем, вам следует использовать возврат нового промиса с новым значением. Например:

```javascript

const myPromise = new Promise((resolve, reject) => {

// Логика обработки промиса

if (условие) {

resolve("Новое значение промиса");

} else {

resolve("Исходное значение промиса");

}

});

myPromise.then(result => {

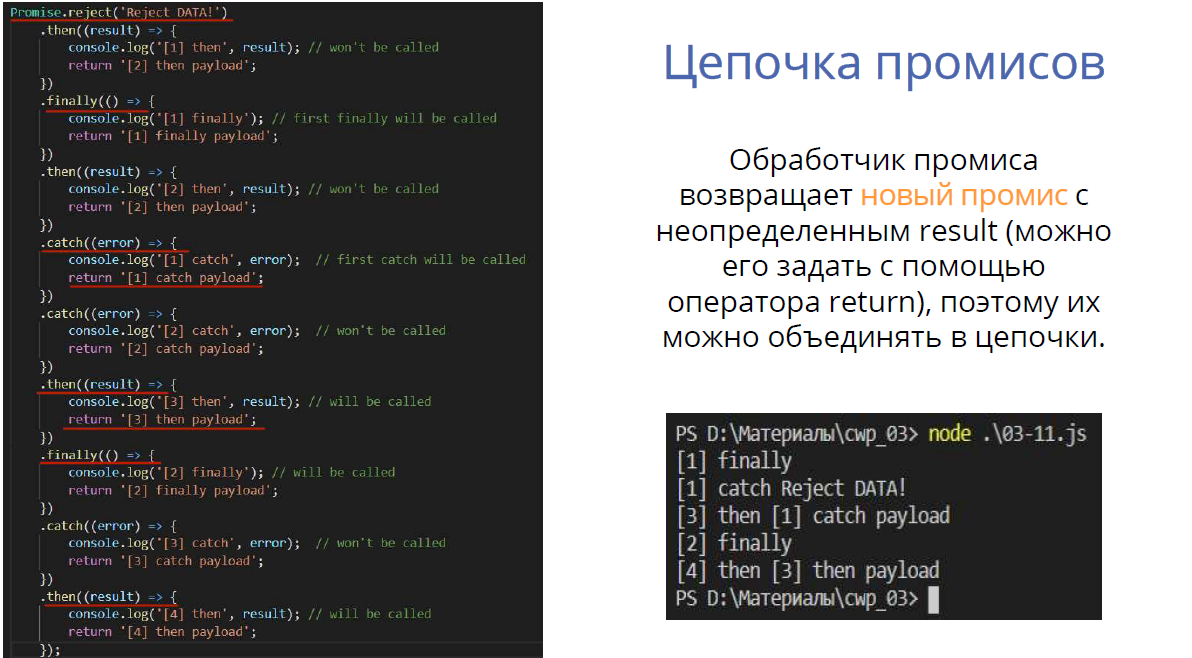
console.log(result); // Выводит "Новое значение промиса" или "Исходное значение промиса"

});

```

В этом примере, если `условие` истинно, промис разрешается с новым значением "Новое значение промиса". В противном случае, промис разрешается с исходным значением "Исходное значение промиса".

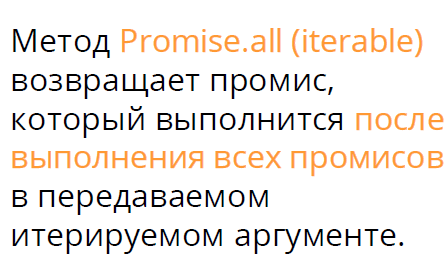
1. Что такое **цепочки промисов** и как они работает?

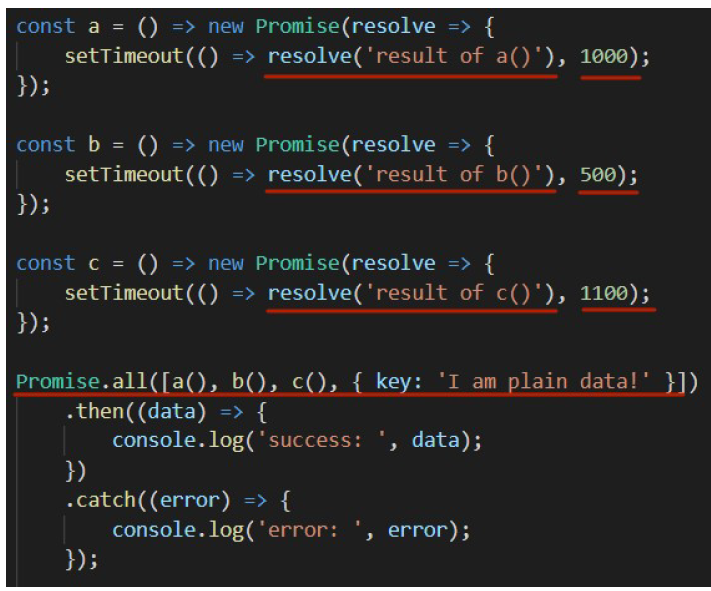


1. Назовите два способа обработки ошибок в Promise.

Try/catch и then/catch у promise

1. Для чего нужен метод **Promise.all()**?





1. В чем отличия методов **Promise.race()** и **Promise.any()**?

Может показаться, что Promise.any() делает то же самое, что и Promise.race(), однако эти функции отличаются в отношении того, как они работают с промисами, которые завершились с ошибкой. Promise.race() возвращает первый завершенный промис, вне зависимости завершился он с ошибкой или нет. А Promise.any() возвращает первый успешно завершенный промис (если такой имеется). Если же все промисы завершились с ошибкой, то генерируется исключение типа **AggregateError**.

1. Что такое **async/await**?

