Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |

|  |  |
| --- | --- |
| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |

**Лабораторная работа № 29**

**Инструкционно-технологическая карта**

Тема: Организация асинхронного кода через промисы.

Цель: Научиться разрабатывать алгоритмы и программы с использованием промисов.

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализовать алгоритм решения задачи.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

# Теоретические сведения

## Что такое Промисы?

Промис(Обещание) – это объект который содержит будущее значение асинхронной операции. Например, если запрашивать некоторые данные с сервера, промис обещает получить эти данные, которые можно будет использовать в будущем.

#### Состояния промисов

Промис в JavaScript, как и обещание в реальной жизни, имеет 3 состояния. Это может быть 1) нерешенный(в ожидании), 2) решенный/resolved (выполненный) или 3) отклоненный/rejected.

**Нерешенный или Ожидающий** – Промис ожидает, если результат не готов. То есть, ожидает завершение чего-либо(например, завершения асинхронной операции).

**Решенный или Выполненный** – Промис решен, если результат доступен. То есть, что-то завершило свое выполнение (например, асинхронная операция) и все прошло хорошо.

**Отклоненный** – Промиc отклонен, если произошла ошибка в процессе выполнения.

#### Синтаксис

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

  ...

});

Был создан новый промис, используя конструктор промисов, он принимает один аргумент, обратный вызов, также известный как исполнительная функция, которая принимает 2 обратных вызова, **resolve** и **reject**.

Исполнительная функция выполняется сразу же после создания промиса. Промис становится выполненным при помощи вызова **resolve**(), а отклоненным при помощи **reject**(). Например:

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

  if (allWentWell) {

    resolve('Все прошло отлично!');

  } else {

    reject('Что-то пошло не так');

  }

});

**resolve**() и **reject**() принимают один аргумент, который может быть строкой, числом, логическим выражением, массивом или объектом.

Другой пример:

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

  const randomNumber = Math.random();

  setTimeout(() => {

    if (randomNumber < .6) {

      resolve('Все прошло отлично!');

    } else {

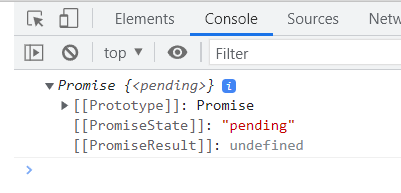
      reject('Что-то пошло не так');

    }

  }, 2000);

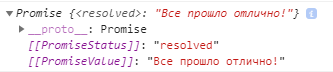
});

Когда промис был создан, он будет в состоянии ожидания и его значение будет **undefined**.

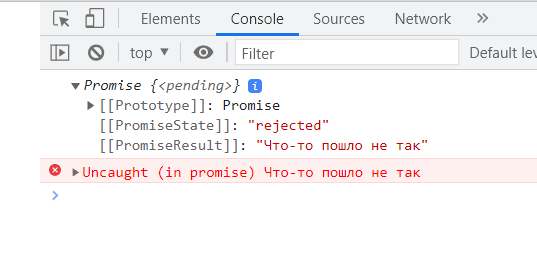


После 2 секунд таймер заканчивается, промис случайным образом либо выполняется, либо отклоняется, и его значением будет то, которое передано в функцию *resolve* или *reject*. Ниже пример двух случаев:

Успешное выполнение:



Отклонение промиса:



## Then и Catch

Всякий раз, когда создаётся объект «обещание», становятся доступны два метода: **then**() и **catch**(). Используя их, можно выполнить нужный код при успешном разрешении «обещания» (**resolve**(...)) или же код, обрабатывающий ситуацию с «ошибкой» (**reject**(...)).

# Контрольные вопросы

1. Что такое промисы?
2. Какие состояния могут принимать промисы?
3. Чем может быть аргумент, который принимают resolve и reject?
4. Каково будет значение промиса, когда он находится в состоянии ожидания?

# Задания

**Задание 1.**

Создайте функцию, которая принимает на вход два аргумента - URL и таймаут в миллисекундах. Функция должна возвращать промис, который выполнится через указанный таймаут и вернет данные, полученные по указанному URL.

**Задание 2.**

Напишите функцию, которая принимает на вход массив промисов и возвращает новый промис, который выполнится, когда все промисы в массиве будут выполнены. Результатом должен быть массив значений, возвращенных каждым из промисов.

**Задание 3.**

Создайте промис, который выполнится через случайное время (от 1 до 5 секунд) и вернет случайное число от 1 до 100. Затем напишите функцию, которая будет вызывать этот промис и выводить результат в консоль.

**Задание 4.**

Напишите функцию, которая принимает на вход массив объектов, у каждого из которых есть поле "url". Функция должна создавать промис для каждого URL и возвращать массив промисов. Когда все промисы будут выполнены, функция должна вернуть массив объектов, в которых к каждому URL добавлено новое поле "data" с данными, полученными по этому URL.

**Задание 5.**

Напишите функцию, которая принимает на вход массив промисов и таймаут в миллисекундах. Функция должна возвращать новый промис, который выполнится через указанный таймаут и вернет массив результатов выполнения каждого из переданных промисов. Если какой-то из промисов не выполнится за указанный таймаут, то он должен быть прерван и вместо результата вернуться ошибку.

# Литература

1. **Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.
2. **Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.
3. **Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич