УДК 004.43

ПРОМИСЫ И EVENT LOOP В JAVA SCRIPT

Мазанова Ю. А.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Воробей А.В. – магистр технических наук, ассистент кафедры ИПиЭ

Аннотация. В данной статье рассматривается концепция промисов в Java Script, цикл событий Event Loop, и модули в Java Script.

Ключевые слова: промисы, *Java Script*, *Event loop*, модули

Введение. Концепция промисов в *JavaScript* была введена для упрощения асинхронного программирования. Промисы – это объекты, которые представляют результат успешного или неудачного завершения асинхронной операции.

Вместо того, чтобы передавать функции обратного вызова в качестве аргументов, промисы позволяют записывать функции обратного вызова в объект промиса. Это делает код более читаемым и удобным для отладки [1].

Основная часть. Промисы имеют два состояния: *pending* и *settled*. Состояние *pending* означает, что операция все еще выполняется, а состояние *settled* означает, что операция завершена. Состояние *settled* может быть либо *fulfilled* (успешно завершено), либо *rejected* (завершено с ошибкой).

Промисы могут быть использованы для выполнения нескольких асинхронных операций одна за другой. Это достигается путем создания цепочки вызовов промисов. Функция then возвращает новый промис, который представляет завершение не только первоначального промиса, но и функций обратного вызова, переданных вами. Это позволяет вам выполнять две или более асинхронных операции одна за другой, причем каждая следующая начинается при успешном завершении предыдущей и использует результат ее выполнения [2].

Цикл событий (*event loop*) в *JavaScript* отвечает за выполнение кода, сбора и обработки событий и выполнения подзадач из очереди (*queued sub-tasks*). Эта модель весьма отличается от других языков программирования, таких как *C* и *Java*.

Когда вызывается функция, создаётся контекст выполнения (*Execution Context*). При вызове вложенной функции создаётся новый контекст, а старый сохраняется в специальной структуре данных - стеке вызовов (*Call Stack*).

Объекты размещаются в куче. Куча – это имя для обозначения неструктурированной области памяти.

Среда выполнения JavaScript содержит очередь задач. Эта очередь – список задач, подлежащих обработке. Каждая задача ассоциируется с некоторой функцией, которая будет вызвана, чтобы обработать эту задачу. Когда стек полностью освобождается, самая первая задача извлекается из очереди и обрабатывается. Обработка задачи состоит в вызове ассоциированной с ней функции с параметрами, записанными в этой задаче. Как обычно, вызов функции создаёт новый контекст выполнения и заносится в стек вызовов. Обработка задачи заканчивается, когда стек снова становится пустым. Следующая задача извлекается из очереди и начинается её обработка.

Цикл событий (*event loop*) в *JavaScript* отвечает за выполнение кода, сбора и обработки событий и выполнения подзадач из очереди (*queued sub-tasks*). Эта модель весьма отличается от других языков программирования, таких как *C* и *Java*.

В *JavaScript* нет модулей, которые имеются внутри цикла событий. Цикл событий – это механизм, который обрабатывает события и вызывает соответствующие функции обратного вызова.

Однако, в *JavaScript* есть модули, которые могут использоваться для организации кода и упрощения его поддержки. Модули позволяют разбить код на отдельные файлы, каждый из которых содержит определенную функциональность. Это упрощает чтение и понимание кода, а также уменьшает вероятность ошибок.

Вот некоторые из модулей, которые могут использоваться в *JavaScript*:

1. ***CommonJS***: это модульная система, которая используется в *Node.js*. Она позволяет экспортировать и импортировать модули с помощью функций *require* и *module.exports*.
2. ***AMD***: это модульная система, которая используется в браузерах. Она позволяет асинхронно загружать модули и определять зависимости между ними.
3. ***ES6***: это модульная система, которая была введена в *ECMAScript* 6. Она позволяет экспортировать и импортировать модули с помощью ключевых слов *import* и *export*.
4. ***UMD***: это универсальный модульный шаблон, который позволяет использовать модули в различных средах, включая браузеры и *Node.js* [3].

Заключение. Таким образом промисы представляют собой объекты, которые представляют результат успешного или неудачного завершения асинхронной операции и могут быть использованы для выполнения нескольких асинхронных операций одна за другой. Это достигается путем создания цепочки вызовов промисов.

Цикл событий (*event loop*) в *JavaScript* отвечает за выполнение кода, сбора и обработки событий.

Также в *JavaScript* есть модули, которые позволяют разбить код на отдельные файлы, каждый из которых содержит определенную функциональность. Это упрощает чтение и понимание кода, а также уменьшает вероятность ошибок.

Список литературы

1. Образовательный онлайн-форум «Хабр» [Электронный ресурс] / Образовательный онлайн-форум «Хабр» – Москва, 2017. – Режим доступа: https://habr.com/ru/articles/762618/ – Дата доступа: 22.01.2024

2. Образовательный онлайн-форум «Хабр» [Электронный ресурс] / Образовательный онлайн-форум «Хабр» – Москва, 2012. – Режим доступа: https://habr.com/ru/articles/143259/ – Дата доступа: 22.01.2024

3. Современный учебник JavaScript [Электронный ресурс] / Режим доступа: https://learn.javascript.ru/promise – Дата доступа: 22.01.2024

UDC 004.43

PROMISES AND EVENT LOOP IN JAVA SCRIPT

Mazanova J.A.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus (style T-institution)

Vorobey A.V. – master of technical sciences, assistant of the department of EPE

Annotation. This article covers the concept of promises in Java Script, the Event Loop, and modules in Java Script.

Keywords: Promises, Java Script, Event loop, modules.