САМАРСКИЙ УНИВЕРСИТЕТ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

Лекция № 5

Цикл событий. Замыкания

2023

РУБРИКА: «ВОПРОСЫ ПО ЛЕКЦИИ № 4»

Предисловие о цикле событий

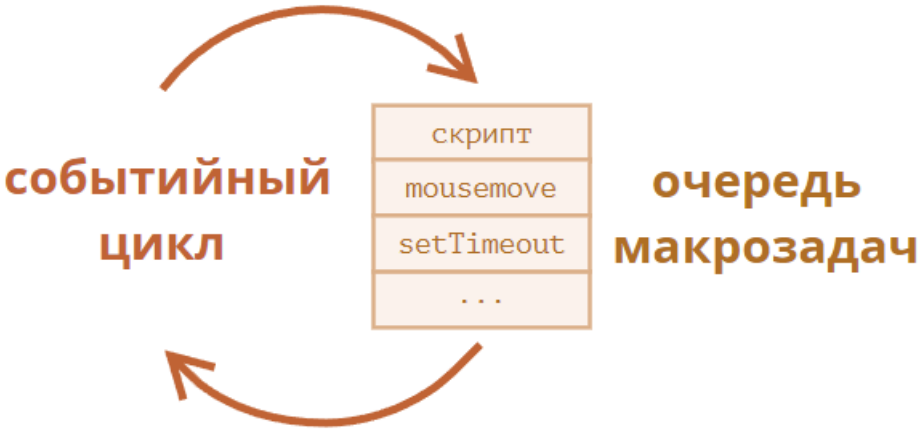
Что такое цикл событий?

Это бесконечно долгий процесс (до момента остановки выполнения программы), при котором на постоянной основе некоторый исполнитель ожидает задач, передаваемых в сценарии и ожидает новые

Зачем это нужно?

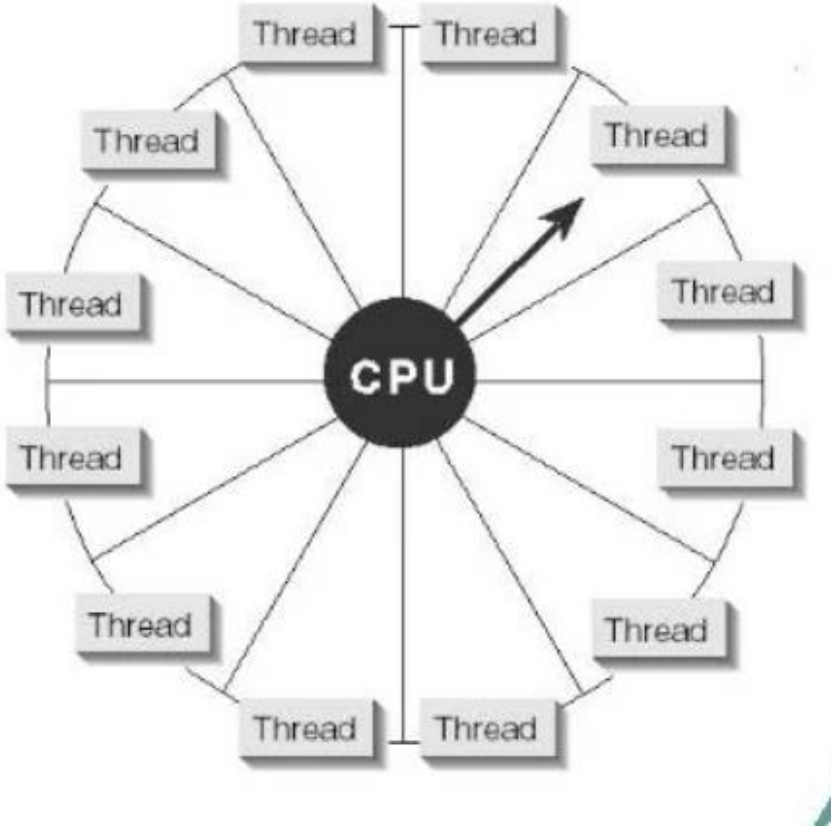
Для создания многопоточности в приложении
Отложенное выполнение/периодическое выполнение некоторых задач
Обработка событий с устройств ввода устройства
Обработка событий перевода режимов работы устройства

Цикл событий :: Общая схема работы



Задачи из очереди исполняются по правилу «первым пришёл – первым ушёл». Когда браузер заканчивает выполнение скрипта, он обрабатывает событие **mousemove**, затем выполняет обработчик, заданный **setTimeout**, и так далее

Цикл событий :: Концепт параллельности вычислений



Цикл событий :: Особенности

Рендеринг (отрисовка страницы) никогда не происходит во время выполнения задачи движком. Не имеет значения, сколь долго выполняется задача. Изменения в DOM отрисовываются только после того, как задача выполнена.

Если задача выполняется очень долго, то браузер не может выполнять другие задачи, обрабатывать пользовательские события, поэтому спустя некоторое время браузер предлагает превентивно завершить долго выполняющуюся задачу. Такое возможно, когда в скрипте много сложных вычислений или ошибка, ведущая к бесконечному циклу

Цикл событий :: Макрозадачи и микрозадачи

Макрозадачи:

- Когда загружается внешний скрипт <script src="...">, то задача – это выполнение этого скрипта.
- Когда пользователь двигает мышь, задача – сгенерировать событие mousemove и выполнить его обработчики.
- Когда истечёт таймер, установленный с помощью setTimeout(func, ...), задача – это выполнение функции func.
- Другие события взаимодействия и временные события...

Микрозадачи:

Микрозадачи приходят только из кода. Обычно они создаются промисами: выполнение обработчика **.then/catch/finally** становится микрозадачей.
Микрозадачи также используются «под капотом» await, т.к. это форма обработки промиса.

ВАЖНО!

Сразу после каждой макрозадачи движок исполняет все задачи из очереди микрозадач перед тем, как выполнить следующую макрозадачу или отобразить изменения на странице, или сделать что-то ещё.

События браузера

Браузерные события — это такие **события**, которые происходящие во время взаимодействия пользователя с веб-страницей. Они дают компьютеру возможность отслеживания действия и реагировать на них.

События браузера :: Основные события

События мыши:

click – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).
contextmenu – происходит, когда кликнули на элемент правой кнопкой мыши.
mouseover / **mouseout** – когда мышь наводится на / покидает элемент.
mousedown / **mouseup** – когда нажали / отжали кнопку мыши на элементе.
mousemove – при движении мыши.
События на элементах управления:

submit – пользователь отправил форму <form>.
focus – пользователь фокусируется на элементе, например нажимает на <input>.
Клавиатурные события:

keydown и **keyup** – когда пользователь нажимает / отпускает клавишу.
События документа:

DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.
CSS events:

transitionend – когда CSS-анимация завершена.

Замыкания :: Определение

Замыкание — это комбинация функции и лексического окружения, в котором эта функция была определена. Другими словами, замыкание даёт доступ к набору внешней функции из внутренней функции. В JavaScript замыкания создаются каждый раз при создании функции.

Замыкания :: Примеры

ПРИМЕР № 1

```
function makeFunc() {
  var name = "Hello, World!";

  function displayName() {
    alert(name);
  }

  return displayName;
};

var myFunc = makeFunc();
myFunc();
```

ПРИМЕР № 2

```
function init(): void {
    let name: string = “Hello, World";
    function displayName(): string {
        alert (name);
    }
    displayName();
}
init();
```

ПРИМЕР № 3

```
function makeMul(x): function {
  return function(y): number {
    return x * y;
  };
};

let mul8: function = makeMul(8);
let mul9: function = makeMul(9);

console.log(mul8(7));
console.log(mul9(4));
```

РУБРИКА: «ВОПРОСЫ СЛУШАТЕЛЕЙ»

КОНЕЦ ЛЕКЦИИ № 5
СПАСИБО ЗА ВНИМАНИЕ