Асинхронный JS

Кирилл Талецкий

TeachMeSkills
7 сентября 2023

Синхронный JS

Синхронный JS

- JS является однопоточным языком программирования
- Для нас это означает, что любой последующий вызов выполняется только после завершения предыдущего

```
function third()
{ console.log('3') }

function second()
{ console.log('2') }

function first()
{ console.log('1') }

first();
second();
third();
```

Что такое очередь и стек?

Синхронный JS
Стек вызовов

- Когда у нас появляются вложенные функции, JS немедленно переходит к их вызову
- Предыдущие вызовы запоминаются в стеке вызовов (call stack)

```
function fifth() {
  console.log('5');
}

function fourth() {
  fifth();
  console.log('4');
}

function third() {
  fourth();
  console.log('3');
}

function second() {
  third();
  console.log('2');
}

function first() {
  second();
  console.log('1');
}

first();
```

Синхронный JS
Стек вызовов - рекурсия

- В случае с рекурсией, происходит то же самое, только вложенный вызов - это вызов функцией самой себя

- Если допустить ошибку в *базовом условии* (выход из рекурсии), то стек вызовов может оказаться переполнен.

```
function pow(x, n) {
  if (n == 1) {
    return x;
  } else {
    return x * pow(x, n - 1);
  }
}

console.log(pow(2, 4));
```

Асинхронный JS

Асинхронный JS
Отложенные вызовы

- window.setTimeout(cb, ms) – вызовет функцию cb() через время ms
- window.setInterval(cb, ms) – будет вызывать функцию cb() через интервалы, равные ms

```
window.setTimeout(() => {
  console.log('отложенный вызов');
}, 3000);

window.setInterval(() => {
  console.log('вызов по таймеру');
}, 5000);
```

Отмена отложенных вызовов

```
const deferredCallTimeout = setTimeout(() => {
  console.log('отложенный вызов');
}, 2000);

setTimeout(() => {
  console.log('отмена!');
  clearTimeout(deferredCallTimeout);
}, 1500);
```

Event Loop

Как работают отложенные вызовы в JS

- setTimeout складывает колбэк в специальную очередь — очередь задач
- Очередь задач (Task queue) иногда называется очередью макрозадач
- Браузер по мере очищение стека забирает колбэки из очереди и выполняет их

```
const callback = () => {
  console.log('отложенный вызов');
};


window.setTimeout(callback, 3000);
// Добавит callback в очередь задач через
3000 мс
```

Event Loop

Откладываем выполнение кода на одну итерацию

- Можно использовать setTimeout(cb, ms) с нулевой задержкой, чтобы выполнить код только после того, как *стек вызовов* будет очищен
- Используется для вызовов, которые должны быть исполнены как можно скорее, после завершения исполнения текущего кода
- Также полезно для разбиения тяжёлой задачи на более маленькие части

```
window.setTimeout(() => {console.log('lol')});

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

Асинхронный JS:
События DOM

Обработчики событий

- Обработчики событий попадают в очередь задач в тот момент, когда событие сработало

```
const handler = () => {
  console.log('click');
};

document.body.onClick(handler);

/**
 * handler() попадёт в очередь задач когда
 * пользователь кликнет в пределах body
 */
```