Тема 8. Стрелочные функции. Область видимости и hoisting

хекслет колледж

Цель занятия:

Познакомить студентов со стрелочными функциями, углубить понимание областей видимости в JavaScript и сформировать базовое представление о механизме поднятия объявлений (hoisting).

Учебные вопросы:

1. Стрелочные функции (Arrow Functions)
2. Сокращённый синтаксис стрелочных функций
3. Hoisting (поднятие объявлений)

1. Стрелочные функции (Arrow Functions)

Стрелочные функции — это сокращённый способ записи функций в JavaScript.
Они были добавлены в язык в стандарте ES6 (ES2015) и используются для более компактной и читаемой записи кода.

Стрелочные функции не заменяют полностью обычные функции, а дополняют их.

Причины появления стрелочных функций:

- уменьшение объёма кода;
- повышение читаемости простых функций;
- удобство при работе с колбэками;
- более предсказуемое поведение в современных интерфейсах.

Базовый синтаксис:

Обычная функция:

```
function sum(a, b) {
    return a + b;
}
```

Эквивалент со стрелочной функцией:

```
const sum = (a, b) => {
    return a + b;
};
```

- **=>** — стрелка, заменяющая ключевое слово **function**;
- параметры указываются до стрелки;
- тело функции — после стрелки.

Пример стрелочной функции без параметров:

```
const greet = () => {
    console.log("Привет!");
};
greet();
```

Пример с одним параметром:

```
const sayHello = name => {
    console.log("Привет, " + name + "!");
};
sayHello("Анна");
```

Скобки можно опустить, если параметр один.

Отличие от обычных функций:

Обычная функция	Стрелочная функция
Использует function	Использует =>
Может быть объявлена отдельно	Обычно записывается в переменную
Более громоздкий синтаксис	Более компактный синтаксис

Практическое применение

Стрелочные функции часто используются:

- для простых вычислений;
- для кратких функций;
- в обработчиках и колбэках (позже).

Вывод:

- Стрелочные функции — это сокращённая форма записи функций.
- Они делают код короче и читаемее.
- Подходят для простых операций и часто используемого функционала.
- Не отменяют обычные функции, а используются вместе с ними.

2. Сокращённый синтаксис стрелочных функций

Стрелочные функции позволяют существенно сократить запись, если функция простая.
JavaScript предоставляет несколько правил сокращения синтаксиса.

Один параметр — без круглых скобок

Если у функции ровно один параметр, круглые скобки можно опустить:

```
const square = x => {
    return x * x;
};
```

Аналог обычной записи:

```
const square = (x) => {
    return x * x;
};
```

Однострочное тело функции

Если тело функции состоит из одной инструкции, фигурные скобки {} можно опустить:

```
const square = x => return x * x; // Ошибка
```

Правильная сокращённая запись:

```
const square = x => x * x;
```

Неявный return

Если фигурные скобки отсутствуют, результат выражения возвращается автоматически.

```
const sum = (a, b) => a + b;
console.log(sum(3, 4)); // 7
```

Эквивалент полной формы:

```
const sum = (a, b) => {
    return a + b;
};
```

Несколько параметров

При двух и более параметрах скобки обязательны:

```
const multiply = (a, b) => a * b;
```

Отсутствие параметров

Если параметров нет, используются пустые скобки:

```
const greet = () => "Привет!";
```

Многострочное тело функции

```
const checkNumber = x => {
    if (x > 0) {
        return "Положительное";
    }
    return "Неположительное";
};
```

Вывод:

- Стрелочные функции поддерживают короткую и полную форму записи.
- Неявный return работает только без фигурных скобок.
- Сокращённый синтаксис улучшает читаемость, если функция простая.
- Для сложной логики предпочтительнее использовать полную форму.

Стрелочные функции имеют ограничения, связанные с this и объектами.
Мы вернёмся к этому вопросу позже, когда будем изучать объекты, события и DOM.

3. Hoisting (поднятие объявлений)

Hoisting — это механизм JavaScript, при котором объявления переменных и функций «поднимаются» вверх своей области видимости на этапе компиляции.
На практике это значит, что вы можете использовать функцию или переменную до её фактического объявления (с некоторыми особенностями).

Hoisting функций

Функции, объявленные через function, полностью «поднимаются» вверх:

```
greet(); // Привет!

function greet() {
    console.log("Привет!");
}
```

Вызов функции до объявления работает;
Функция полностью доступна в области видимости.

Hoisting переменных

var

Переменные, объявленные через **var**, поднимаются, но не инициализируются:

```
console.log(x); // undefined
var x = 5;
console.log(x); // 5
```

Объявление **var x** поднимается наверх области видимости
Присвоение = 5 остаётся на месте
Из-за этого выводится **undefined**, а не ошибка

let и const

Переменные, объявленные через **let** и **const**, поднимаются, но нельзя использовать до объявления:

```
// console.log(y); // Ошибка
let y = 10;

// console.log(z); // Ошибка
const z = 20;
```

Этот период называется «временная мёртвая зона» (TDZ);
Попытка обратиться до объявления — ошибка.

Hoisting и стрелочные функции

Стрелочные функции, объявленные как переменные (**const** или **let**), не поднимаются как функции:

```
// greet(); // Ошибка
const greet = () => {
    console.log("Привет!");
};
```

Переменная **greet** подчиняется правилу **const** — нельзя использовать до объявления

Вывод:

- Hoisting — это поднятие объявлений переменных и функций на уровень их области видимости.
- function — полностью поднимается.
- var — поднимается без инициализации (undefined).
- let и const — поднимаются, но до объявления недоступны.
- Стрелочные функции, объявленные как переменные, ведут себя как let/const.
- Лучшей практикой является объявление переменных и функций перед использованием.

Итоги лекции:

- Стрелочные функции — это сокращённый синтаксис функций в JavaScript.
- Стрелочные функции удобны для коротких вычислений и передачи в другие функции (callback).
- Hoisting — механизм поднятия объявлений на уровень области видимости.
- Функции через function полностью поднимаются и доступны до объявления.
- let, const и стрелочные функции не доступны до объявления (временная мёртвая зона, TDZ).

Для безопасного и предсказуемого кода рекомендуется:

- объявлять переменные и функции до использования;
- использовать const и let вместо var;
- применять стрелочные функции там, где нужен компактный код

Контрольные вопросы:

- Что такое стрелочная функция и чем она отличается от обычной функции?
- Как записать стрелочную функцию с одним параметром и без фигурных скобок?
- Как работает неявный return в стрелочной функции?
- Можно ли использовать стрелочную функцию как конструктор? Почему?
- Что такое hoisting?
- Как ведут себя функции, объявленные через function, при hoisting?
- Как ведут себя переменные var при hoisting?
- Почему переменные let и const нельзя использовать до объявления?
- Как ведут себя стрелочные функции, объявленные через const или let, при hoisting?
- Какие рекомендации по использованию стрелочных функций и объявлению переменных можно дать для безопасного кода?

Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

@HEXLY.KZ