JS Урок 2

Стрелочные функции, области видимости, деструктуризация, spread и rest операторы, Promise, async/await

Функции. Стрелочные функции.

Функция - фрагмент кода, который можно вызывать из другого места программы.
Синтаксис функции js: function <name>(<params>) { <statements> }

Стрелочная функция - сокращенный синтаксис объявления функции. Не имеет собственного контекста (this).
Синтаксис стрелочной функции js: (<params>) => { <statements> } или (<params>) => <expression>

Области видимости

Область видимости – это текущий контекст в коде.
Существует глобальная и локальная области видимости. Локальная ограничивается блоком {}.

```
const string = "123";
if (true) {
    let num = 10;
    console.log(string); // "123"
    console.log(num); // 10
}
console.log(num); // ReferenceError: num is not defined
```

Деструктурирующее присваивание

Деструктурирующее присваивание – это специальный синтаксис, который позволяет нам «распаковать» массивы или объекты в кучу переменных, так как иногда они более удобны.

Деструктурирующее присваивание (массивы)

```
// у нас есть массив с именем и фамилией
let arr = ["Ilya", "Kantor"]

// деструктурирующее присваивание
// записывает firstName=arr[0], surname=arr[1]
let [firstName, surname] = arr;
console.log(firstName); // Ilya
console.log(surname);  // Kantor
```

Деструктурирующее присваивание (объекты)

```
let options = {
    title: "Menu",
    width: 100,
    height: 200
};
let {title, width, height} = options;
console.log(title);  // Menu
console.log(width);  // 100
console.log(height); // 200
```

Spread оператор (массивы)

Spread оператор используется для разделения коллекций на отдельные элементы.

```
const log = (a, b, c) => {  console.log(a, b, c); };
log(...['Spread', 'Rest', 'Operator']); // Spread Rest Operator


const arr = ['will', 'love'];
const data = ['You', ...arr, 'spread', 'operator'];
console.log(data); // ['You', 'will', 'love', 'spread', 'operator']
```

Spread оператор (объекты)

Также spread может разделять объекты.

```
var obj1 = { foo: 'bar', x: 42 };
var obj2 = { foo: 'baz', y: 13 };
var mergedObj = { ...obj1, ...obj2 };
// Object { foo: "baz", x: 42, y: 13 }
```

Rest оператор

Rest оператор используется для соединения отдельных значений в массив.

```
const log =(a, b, ...rest) => {
    console.log(a, b, rest);
};
log('Basic', 'rest', 'operator', 'usage'); // Basic rest ['operator', usage]
```

Promise

Объект **Promise** (промис) используется для отложенных и асинхронных вычислений.

Синтаксис
new Promise(function(resolve, reject) { ... }).then().then();

Promise может находиться в трёх состояниях:
ожидание (pending): начальное состояние, не исполнен и не отклонен.
исполнено (fulfilled): операция завершена успешно.
отклонено (rejected): операция завершена с ошибкой.

async/await

Когда вы объявляете функцию как асинхронную, через слово async, вы говорите, что данная функция возвращает Promise.
Каждая вещь которую вы ожидаете внутри этой функции, используя слово await, тоже возвращает Promise.

```
const do1 = async () => { console.log(1); }
const do2 = async () => { console.log(2); }

async () => {
    await do1();
    await do2();
}
```