Frontender[1.0] JavaScript - Типы данных. Data types

YouTube	https://youtu.be/yS04L7bZ3Zk
Telegram	https://t.me/Dmitry_Kolotilshikov
Github	https://github.com/DmitryKolotilshikov/
Boosty	https://boosty.to/dmitry_ko
# Номер урока	4

💡 *В JavaScript есть 8 основных типов данных*

number, bigint, string, boolean, null, undefined, symbol, object

https://learn.javascript.ru/types

Когда говорят, что **null** в JavaScript является объектом, это ведет к путанице из-за исторической особенности языка. Вот как можно объяснить это простыми словами:

В начале развития JavaScript была ошибка в реализации, из-за которой typeof **null** возвращал "**object**". Это случилось потому, что во внутреннем представлении типов данных используются теги типов, и **null** был неправильно помечен тегом, предназначенным для объектов.

В действительности **null** — это специальное значение, которое представляет собой "ничего", "пусто" или "значение не известно". Это не объект и не ссылка на объект. Оно представляет отсутствие какой-либо ценности и отличается от undefined, которое обозначает отсутствие значения у переменной.

Разработчики JavaScript решили не исправлять этот баг, поскольку он уже широко использовался в коде, написанном на ранних версиях языка, и его исправление могло бы привести к массовой поломке существующего кода. Поэтому до сих пор typeof **null** возвращает "object", но это больше традиция и особенность, чем правильное представление о типе **null**.

💡 **Function type**

JavaScript функции играют очень важную роль, поэтому они имеют свой собственный тип — **function**. Проще говоря, это нужно для того, чтобы различать функции от других типов данных, например, чисел, строк, объектов и так далее. Это важно, поскольку функции в JavaScript могут быть использованы почти как любой другой объект: их можно присваивать переменным, передавать как аргументы другим функциям или даже возвращать из функций как результат.

Тип **function** помогает нам понимать, что конкретная переменная или выражение может быть "вызвано", то есть выполнено как код. Это подразумевает, что функция содержит блок инструкций, которые будут выполнены, когда функция будет вызвана. Тип **function** также позволяет функциям иметь свойства и методы, как у объектов, что делает их очень мощными и гибкими инструментами программирования.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js"></script>
    <title>Data types</title>
</head>
<body>
</body>
</html>
```

```javascript
// есть 8 основных типов данных
// number, bigint, string, boolean, null, undefined, symbol, object

// Number
const age = 28;
console.log(typeof age);

// String
const firstName = 'Dmitry Ko';
console.log(typeof firstName);

// bigint
const bigNumber = 123124121242142222324324 2n;
console.log(typeof bigNumber);

// boolean
const isActive = true;
const isConfirmed = false;
console.log(typeof isActive);

// null
const city = null; // ничего, пусто
console.log(typeof city); // Для null возвращается "object" – это ошибка в языке, на самом де
ле это не объект.

// undefined
let isLoading;
console.log(isLoading); // undefined
console.log(typeof isLoading); // undefined
isLoading = false;
```

```
// symbol
const id = Symbol('123'); // для уникальных идентификаторов в объектах
console.log(typeof id);

// Ссылочные типы | Reference Types
// object
const person = {
    name: 'Alex',
    age: 24,
}
console.log(person);
console.log(typeof person);

const numArr = [1,2,3,4,5];
console.log(numArr);
console.log(typeof numArr);

function info() {
    return 'Hello World!';
}
console.log(typeof info) // function

// в JavaScript по сути всё объект
console.log(Number.prototype);
console.log(Object.getOwnPropertyNames(Number.prototype));
```