

Тема 2. Переменные и типы данных. Преобразование типов.



хекслет колледж

Цель занятия:

Познакомиться с понятием переменной и типами данных.

Учебные вопросы:

1. Понятие переменной в JavaScript.
Способы объявления переменных.
2. Основные типы данных в JavaScript
3. Преобразование типов данных

1. Понятие переменной в JavaScript. Способы объявления переменных.

Переменная — это именованная область памяти, предназначенная для хранения данных, к которым программа может обращаться и изменять их в процессе выполнения.

1. Введение

В JavaScript есть три ключевых слова для объявления переменных:

1. С помощью `let`

- Современный способ объявления.
- Можно не присваивать значение.
- Позволяет изменять значение переменной.

```
let x = 5;  
x = 10; // можно изменить
```

2. С помощью `const`

- Используется для объявления констант (значение нельзя переназначить).
- Обязательно нужно присвоить значение при объявлении.

```
const PI = 3.14;  
PI = 3.1415; // ❌ ошибка, значение нельзя изменить
```

3. С помощью `var`

- «Старый» способ объявления переменных.
- Может приводить к ошибкам, поэтому в современном коде почти **не используется**.

```
var y = 20;  
y = 30; // можно изменить
```

Резюме

- `let` — для изменяемых переменных (рекомендуется в большинстве случаев).
- `const` — для значений, которые не должны изменяться.
- `var` — устаревший способ, применять не рекомендуется.

Правила именования переменных.

Основные правила:

- Допустимые символы
- латинские буквы a-z, A-Z
- цифры 0-9 (но не в начале)
- символы _ и \$



Примеры:

```
let userName;  
let $price;  
let _value1;
```



Ошибки:

```
let 1abc;      // нельзя начинать с цифры  
let my-name;   // дефис недопустим
```

Чувствительность к регистру.

user, User и USER — это разные переменные.

```
let user = "Иван";
let User = "Пётр";
```

Запрещённые имена.

Нельзя использовать зарезервированные слова JavaScript:

```
let let = 5; // ✗ Ошибка
```

Соглашения по стилю (best practices)

camelCase для переменных и функций:

```
let userName = "Иван";
let totalPrice = 100;
```

Константы в UPPER_CASE (если значение известно заранее и не меняется):

```
const PI = 3.14159;
const API_URL = "https://example.com";
```

Имена должны быть осмысленными — отражать содержание переменной, а не абстрактные a, b, c (кроме временных случаев в циклах)

Советы по выбору имён:

- Избегать абстрактных слов: data, info, item, thing.
- Не злоупотреблять односимвольными именами (x, y, z), кроме математики или счётчиков цикла.
- Следовать стилю camelCase в JS.
- Давать имена так, чтобы без комментариев было понятно назначение переменной.

// Хорошо ✓

```
let userName = "Иван";           // отражает суть – имя пользователя
let age = 25;                   // понятно, что это возраст
let isLoggedIn = true;          // булево значение, читается как вопрос
let maxSpeed = 120;             // логично для предела скорости
const PI = 3.14;                // математическая константа
```

// Плохо ✗

```
let a = "Иван";                 // непонятно, что за "а"
let data = 25;                  // слишком абстрактно
let temp = true;                // "temp" что? временный? температура?
let SPEED = 120;                // переменная выглядит как константа
const P = 3.14;                 // слишком кратко, теряется смысл
let имяПользователя = "Оля";   // кириллица – работает, но не рекомендуется
let user-name = "Иван";         // ✗ дефис – синтаксическая ошибка
```

Итог:

- Имя переменной должно быть корректным (по синтаксису) и понятным (по смыслу).
- Соблюдение правил именования облегчает командную работу и делает код более читаемым.

2. Основные типы данных в JavaScript

В JavaScript все данные имеют **тип**. Тип данных определяет:

- какие значения может принимать переменная;
- какие операции можно выполнять с этими значениями.

JavaScript является **динамически типизированным языком** — тип данных определяется автоматически во время выполнения программы и может меняться.

Примитивные типы данных — это базовые, «простые» типы, которые не являются объектами и хранятся по значению, а не по **ссылке**.

В JavaScript существует 7 примитивных типов:

1. **number** — числовой тип (целые и дробные числа).

```
let age = 25;  
let price = 19.99;  
let infinity = Infinity; // бесконечность  
let notANumber = NaN; // результат ошибочной операции
```

NaN (Not a Number) — это специальное числовое значение в JavaScript, которое означает «результат вычисления не является допустимым числом».

NaN относится к типу **number**, даже несмотря на своё название.

2. **bigint** — целые числа произвольной длины (добавлен в ES2020).

```
let big = 1234567890123456789012345678901234567890n;
```

3. **string** — строки (текстовые данные).
Можно использовать "...", '...', или
шаблонные строки `...`.

```
let name = "Иван";
let message = `Привет, ${name}!`;
```

4. **boolean** — логический тип: принимает значения true или false.

```
let isStudent = true;  
let hasAccess = false;
```

5. **`undefined`** — значение «не присвоено».
Переменная объявлена, но не
инициализирована.

```
let x;  
console.log(x); // undefined
```

6. **null** - «пустое значение», отсутствие объекта. Используется программистом явно.

```
let car = null; // машины нет
```

7. **symbol** - уникальный примитивный тип данных, который используется для создания уникальных идентификаторов.

```
let id = Symbol("id");
let anotherId = Symbol("id");
console.log(id === anotherId); // false
```

Типизация в JavaScript.

В JS переменная не привязана жёстко к какому-то типу.

Одну и ту же переменную можно использовать для хранения разных значений:

```
let x = 5;          // число
x = "Hello";       // строка
x = true;          // логическое значение
```

Примитивные типы данных в JavaScript:

Тип	Назначение	Пример
<code>number</code>	Представление чисел (целых и дробных)	<code>let x = 42;</code>
<code>bigint</code>	Очень большие целые числа	<code>let big = 9007199254740991n;</code>
<code>string</code>	Текстовые данные (строки символов)	<code>let name = "Alice";</code>
<code>boolean</code>	Логические значения (<code>true / false</code>)	<code>let isAdmin = false;</code>
<code>undefined</code>	Значение по умолчанию для неинициализированной переменной	<code>let x;</code>
<code>null</code>	«Пустое» значение, отсутствие объекта	<code>let user = null;</code>
<code>symbol</code>	Уникальный идентификатор	<code>let id = Symbol("id");</code>

Оператор проверки типов.

В JavaScript используется оператор **typeof** для определения типа данных переменной или выражения.

Он возвращает строку с названием типа.

Синтаксис:

typeof выражение

Примеры:

```
typeof 42;          // "number"
typeof "Привет";   // "string"
typeof true;        // "boolean"
typeof undefined;   // "undefined"
typeof null;        // "object"  (историческая ошибка JS)
typeof Symbol("id"); // "symbol"
typeof 123n;        // "bigint"
```

3. Преобразование типов данных

Преобразование типов — это процесс изменения значения одного типа данных в другой.

В JavaScript существует:

- **неявное** (автоматическое) преобразование,
- **явное** (программируемое) преобразование.

Неявное (автоматическое) преобразование типов

JavaScript может автоматически менять тип значения в ходе выполнения выражений.

Пример:

```
let result = "5" + 2;  
console.log(result); // "52"
```

Число **2** автоматически преобразуется в строку.
Оператор **+** со строкой выполняет конкатенацию.

Другой пример:

```
let result = "5" - 2;  
console.log(result); // 3
```

Строка "5" преобразуется в число.

Оператор - выполняет математическое действие.

Явное преобразование типов

Явное преобразование выполняется по инициативе разработчика.

Преобразование в число

```
Number("10");    // 10  
Number("abc");  // NaN
```

Используется, когда:

- данные приходят в виде строки (например, из формы),
- нужно выполнить расчёты.

Преобразование в строку

```
String(25);           // "25"  
String(true);         // "true"
```

Используется для:

- вывода данных,
- формирования сообщений.

Преобразование в логический тип

```
Boolean(1);          // true  
Boolean(0);          // false  
Boolean("");         // false  
Boolean("text");     // true
```

Чаще всего применяется:

- в условиях,
- для проверки наличия значения.

Итог:

- JavaScript может автоматически преобразовывать типы данных.
- Явное преобразование делает код более понятным и надёжным.

Контрольные вопросы:

Что такое переменная и для чего она используется в программе?

Какие способы объявления переменных существуют в JavaScript?

В чём разница между `let` и `const`?

В каких случаях не рекомендуется использовать `var`?

Какие требования предъявляются к именам переменных?

Перечислите основные примитивные типы данных в JavaScript.

В чём разница между значениями `null` и `undefined`?

Почему JavaScript называют динамически типизированным языком?

Для чего используется оператор `typeof`?

Что такое преобразование типов данных?

В чём разница между неявным и явным преобразованием типов?

Как преобразовать строку в число явно?

Что такое `Nan` и в каком случае оно может возникнуть?

Домашнее задание:

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

хекслет колледж

@HEXLY.KZ