Типы данных и переменные

Типы данных в JavaScript

Тип данных:

- Множество значений и операций на этих значениях;
- Определенный тип данных хранимый в ячейках памяти компьютера

JavaScript слабо типизированый язык

- Позволяет выполнять большинство операций над данными без обьявления их типа
- Значения имеют тип, переменные нет
- Переменные могут хранить любые типы данных
- Все переменные определяются с помошью предикаторов (ключевых слов предшествующих обьявленной переменной) `var`, `let` или `const`

```
var count = 5; // переменая хранит целочисленный тип
count = 'привет'; // теперь хранит строку

var name = 'Курс JavaScript'; // переменная хранит строку

let mark = 5.25 // переменная хранит число с плавающей точкой
mark = true; // теперь хранит булево значение

const MAX_COUNT = 250; // константа хранит число
MAX_COUNT = 0; // ошибка переназначения значения константы
```

Переменные в JavaScript

Переменная это: Поименованная область в памяти, которая может быть изменена в процессе выполнения программы

Переменные позволяют:

- Хранить информацию
- Извлекать информацию
- Манипулировать информацией

```
let counter = 5;
```

Числа в JavaScript

- Все числа в JavaScript хранятся в виде чисел с плавающей запятой двойной точности
- В соответствии с **IEEE-754** стандартом
  - Могут быть представлены как объект с типом `Number`

```
let value = 5;
value = 3.14159;
value = new Number(100); // Number { 100 }
value = value + 1; // 101
let biggestNum = Number.MAX_VALUE;
```

Изменения численных типов

- Перевод `floating-point` в `integer` числа

```
let valueDouble = 8.75;
let valueInt = valueDouble | 0; // 8
```

- Конвертация в `integer` с помощью округления

```
let valueDouble = 8.75;
let roundedInt = (valueDouble + 0.5) | 0; // 9
```

- Конвертация `string` в `integer`

```
let str = '1234';
let i = str | 0 + 1; // 1235
```

Целочисленный тип

- является 64 битным
- Целые значения могут содержать цифры от `` -9007199254740992 `до` `9007199254740992`
- Их их базовым типом является число с плавающей точкой (** IEEE-754 **)

```
let studentsCount = 5;
let maxInteger = 9007199254740992;
let minInteger = -9007199254740992;
let a = 5, b = 3;
let sum = a + b; // 8
let div = a / 0; // Infinity
```

Числа с плавающей точкой

- Является диапазонов всех действительных чисел
- Могут быть заданы с точностью до определенного количества знаков после точки
- Могут вести себя ненормально при вычислениях

```
let PI = Math.PI; // 3.141592653589793
let minValue = Number.MIN_VALUE; // 5e-324
let maxValue = Number.MAX_VALUE; // 1.79e+308
let div0 = PI / 0; // Infinity
let divMinus0 = -PI / 0; // -Infinity
let unknown = div0 / divMinus0; // NaN
```

ненормальность чисел с плавающей точкой

- Иногда могут наблюдаться отклонения при использовании чисел с плавающей точкой
- Сравнение чисел с плавающей точкой не может быть выполнена непосредственно с применением операторов эквивалентности (`==` и `===`)

```
```

let a = 0.1;
let b = 0.2;
let sum = 0.3;
let equal = (a+b === sum); // false!!!
console.log('a+b = '+ (a + b) + ', sum = ' +
sum + ', sum == a+b? is ' + equal);

```
### Тип булев (логический)
- Содержит **два возможных состояния**:
  - `true` and `false`
- Используется в логических выражениях

```js
let a = 1;
let b = 2;
let greaterAB = (a > b);
console.log(greaterAB);  // false

let equalA1 = (a === 1);
console.log(equalA1);    // true

console.log((a !== b) && (b > 0));
```

Строковый тип

- Представлен в виде **последовательности символов**
- Строки должны быть заключены в кавычки:
  - **Оба** `'` , `"` работают корректно
  - **ES6** так же включает `` ` `` (ticks) для перевода в строчный тип
- Строки могут склеиваться
  - Используется оператор `+`

```js
let s = 'Добро пожаловать в JavaScript';
let name = 'Вася' + ' ' + 'Пупкин';
let greeting = `${s}, ${name}`;

console.log(greeting); // Добро пожаловать в JavaScript, Вася Пупкин
```

- пример конкатенации

```js
let firstName = 'Вася';
let lastName = 'Пупкин';
console.log('Привет, ' + firstName + '!');

let fullName = firstName + ' ' + lastName;
console.log('Ваше полное имя ' + fullName);
```

- Строки в JavaScript храняться в кодировке UTF
```

```
let asSalamuAlaykum = 'السلام عليكم';
alert(asSalamuAlaykum);

let кирилица = 'Това е на кирилица!';
alert(кирилица);

let leafJapanese = '葉';
alert(leafJapanese);
```

- Конвертация строки в число может быть произведена с помошью функций `parseInt` и `parseFloat`:

```
let numberString = '123'
console.log(parseInt(numberString)); // выводит 123
let floatString = '12.3';
console.log(parseFloat(floatString)); // выводит 12.3
```

- `parseInt` и `parseFloat` отсекают все нечисловые символы

```
let str = '123Hello';
console.log(parseInt(str)); // prints 123
```

- `parseInt` и `parseFloat` могут использоваться, но они медленные и демонстрируют странное поведение.
- Операцию перевода строки в число можно сделать с помошью следующих подходов:
  - С помошью округления:

```
'123.3' | 0 -> returns 123
```

- Приведением типа:

```
Number('123.3') -> returns 123.3
'123.3' * 1 -> returns 123.3
+'123.3' -> returns 123.3
```

Undefined (неопределенные) and Null (нулевые) значения

JavaScript имеет специальное значение `undefined`, оно означает что переменная не определена (ее не существует в текущем контексте), `undefined` не одно и тоже что и `null`, который представляет нулевое значение.

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

x = 5;
console.log(x); // 5

x = undefined;
console.log(x); // undefined

x = null;
console.log(x); // null
```

Проверка типа переменных

Переменная может быть проверена в процессе выполнения программы

```
let x = 5;
console.log(typeof x); // number
console.log(x); // 5

x = new Number(5);
console.log(typeof x); // object
console.log(x); // Number {}

x = null;
console.log(typeof x); // object

x = undefined;
console.log(typeof x); // undefined
```

Объявление и использование переменных

Когда мы декларируем переменную мы

- Специфизируем **имя** (идентификатор переменней)
- Можем заполнить **инициализирующим значением**

```
<var | let | const> <identifier> [= <initialization>];
```

```
let emptyVariable;
var height = 200;
let width = 300;
const depth = 250;
```

Правила именования идентификаторов

- Буквы (Unicode)
- Цифры [`0` - `9`]
- Нижнее подчеркивание '`_`'
- Доллар '`$`'
- Не могут начинаться с цифр, и быть зарезервированными JavaScript словами
- Переменные и названия функций правильно именовать стилем `camelCase`

Идентификатор может быть описательными именем, рекомендуется применять латинскую кодировку, JavaScript регистрозависимый язык

- *корректное именование:*

```
let New = 2; // здесь N заглавная
let _2Pac; // этот идентификатор начинается _
let поздравление = 'Hello'; // Юникод
// Часто применяемые:
let greeting = 'Hello';
let n = 100; // Неинформативно
let numberOfClients = 100; // Информативно
// Через чур информативно:
let numberOfPrivateClientOfTheFirm = 100;
```

- *некорректное именование*:

```
let new;    // new это ключевое слово
let 2Pac;   // Не может начинаться с цифр
```

Присвоение значений переменным

- Присвоение значений переменным производится с помошью оператора `=`
- Присвоение может юыть каскадным, когда следующая переменная приравнивается к предыдущей
- Переменные декларированные `const` не могут быть переопределены дргуим значением

```
let firstValue = 5;
let secondValue;
let thirdValue;

// Используем уже декларированную переменную:
secondValue = firstValue;

// Каскадное присваивание

thirdValue = firstValue = 3; // Avoid this!
```

Инициализация переменных

- Знавение должно быть проинициализировано до использования переменной
- Может инициализироваться уже проинициализированной переменной
- Неициализированные переменные неопределены `undefined`

```
let heightInMeters = 1.74;

let greeting = 'Hello World!';
let message = greeting;

const parsedNumber = parseInt('1239') + 1;
```

Локальные и глобальные переменные

- Локальные переменные могут быть определены с помшью ключевых слов `var`, `let` или `const`
  - `var` - время жизни переменной определено в пространстве текущей функции или глобальном пространстве
  - `let` - время жизни переменной в текущем пространстве
  - `const` - такое же как `let`, но не может быть переопределено
- глобальные переменные
  - Определяются **без ключевых слов**
  - Плохая практика **не делайте так!**

```
let a = 5; // а в локальной области видимости
a = 'блаблабла'; // так же упонимается здесь
```

```
a = undefined;
a = 5; // то же что и window.a
```