Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |

|  |  |
| --- | --- |
| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |

**Лабораторная работа № 2**

**Инструкционно-технологическая карта**

Тема: Основные типы данных в JavaScript.

Цель: Изучить основные типы данных в JavaScript, научиться создавать переменные с необходимыми типами данных, выводить значение переменных.

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализовать алгоритм решения задачи.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

# Теоретические сведения

## Типы данных

Значение в JavaScript всегда относится к данным определённого типа. Например, это может быть строка или число.

Есть **восемь** основных типов данных в JavaScript.

Переменная в JavaScript может содержать любые данные. В один момент там может быть строка, а в другой – число:

// Не будет ошибкой

let message = "hello";

message = 123456;

Языки программирования, в которых такое возможно, называются «динамически типизированными». Это значит, что типы данных есть, но переменные не привязаны ни к одному из них.

#### Число

let n = 123;

n = 12.345;

Числовой тип данных (**number**) представляет как целочисленные значения, так и числа с плавающей точкой.

Существует множество операций для чисел, например, умножение \*, деление /, сложение +, вычитание - и так далее.

Кроме обычных чисел, существуют так называемые «специальные числовые значения», которые относятся к этому типу данных: **Infinity**, **-Infinity** и **NaN**.

**Infinity** представляет собой математическую бесконечность ∞. Это особое значение, которое больше любого числа.

Мы можем получить его в результате деления на ноль:

alert( 1 / 0 ); // Infinity

Или задать его явно:

alert( Infinity ); // Infinity

**NaN** означает вычислительную ошибку. Это результат неправильной или неопределённой математической операции, например:

alert( "не число" / 2 ); // NaN, такое деление является ошибкой

Значение **NaN** «прилипчиво». Любая операция с **NaN** возвращает **NaN**:

alert( "не число" / 2 + 5 ); // NaN

Если где-то в математическом выражении есть **NaN**, то результатом вычислений с его участием будет **NaN**.

**Математические операции – безопасны**

Математические операции в JavaScript «безопасны». Скрипт никогда не остановится с фатальной ошибкой (не «умрёт»). В худшем случае мы получим **NaN** как результат выполнения.

Специальные числовые значения относятся к типу «число». Конечно, это не числа в привычном значении этого слова.

#### BigInt

В JavaScript тип «number» не может содержать числа больше, чем (253-1) (т. е. 9007199254740991), или меньше, чем -(253-1) для отрицательных чисел. Это техническое ограничение вызвано их внутренним представлением.

Для большинства случаев этого достаточно. Но иногда нам нужны действительно гигантские числа, например, в криптографии или при использовании метки времени («timestamp») с микросекундами.

Тип **BigInt** был добавлен в JavaScript, чтобы дать возможность работать с целыми числами произвольной длины.

Чтобы создать значение типа **BigInt**, необходимо добавить **n** в конец числового литерала:

// символ "n" в конце означает, что это BigInt

const bigInt = 1234567890123456789012345678901234567890n;

#### Строка

Строка (**string**) в JavaScript должна быть заключена в кавычки.

let str = "Привет";

let str2 = 'Одинарные кавычки тоже подойдут';

let phrase = `Обратные кавычки позволяют встраивать переменные ${str}`;

В JavaScript существует три типа кавычек.

1. Двойные кавычки: "Привет".
2. Одинарные кавычки: 'Привет'.
3. Обратные кавычки: `Привет`.

Двойные или одинарные кавычки являются «простыми», между ними нет разницы в JavaScript.

Обратные же кавычки имеют расширенную функциональность. Они позволяют нам встраивать выражения в строку, заключая их в ${…}. Например:

let name = "Иван";

// Вставим переменную

alert( `Привет, ${name}!` ); // Привет, Иван!

// Вставим выражение

alert( `результат: ${1 + 2}` ); // результат: 3

Выражение внутри ${…} вычисляется, и его результат становится частью строки. Мы можем положить туда всё, что угодно: переменную name, или выражение 1 + 2, или что-то более сложное.

Обратите внимание, что это можно делать только в обратных кавычках. Другие кавычки не имеют такой функциональности встраивания!

alert( "результат: ${1 + 2}" ); // результат: ${1 + 2} (двойные кавычки ничего не делают)

#### Булевый (логический) тип

Булевый тип (**boolean**) может принимать только два значения: **true** (истина) и **false** (ложь).

Такой тип, как правило, используется для хранения значений да/нет: **true** значит «да, правильно», а **false** значит «нет, не правильно».

Например:

let nameFieldChecked = true; // да, поле отмечено

let ageFieldChecked = false; // нет, поле не отмечено

Булевые значения также могут быть результатом сравнений:

let isGreater = 4 > 1;

alert( isGreater ); // true (результатом сравнения будет "да")

#### Значение «null»

Специальное значение null не относится ни к одному из типов, описанных выше.

Оно формирует отдельный тип, который содержит только значение null:

let age = null;

В JavaScript null не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках.

Это просто специальное значение, которое представляет собой «ничего», «пусто» или «значение неизвестно».

В приведённом выше коде указано, что значение переменной age неизвестно.

#### Значение «undefined»

Специальное значение undefined также стоит особняком. Оно формирует тип из самого себя так же, как и null.

Оно означает, что «значение не было присвоено».

Если переменная объявлена, но ей не присвоено никакого значения, то её значением будет undefined:

let age;

alert(age); // выведет "undefined"

Технически мы можем присвоить значение undefined любой переменной:

let age = 123;

// изменяем значение на undefined

age = undefined;

alert(age); // "undefined"

…Но так делать не рекомендуется. Обычно null используется для присвоения переменной «пустого» или «неизвестного» значения, а undefined – для проверок, была ли переменная назначена.

#### Объекты и символы

Тип object (объект) – особенный.

Все остальные типы называются «примитивными», потому что их значениями могут быть только простые значения (будь то строка, или число, или что-то ещё). В объектах же хранят коллекции данных или более сложные структуры.

#### Оператор typeof

Оператор typeof возвращает тип аргумента. Это полезно, когда мы хотим обрабатывать значения различных типов по-разному или просто хотим сделать проверку.

У него есть две синтаксические формы:

1. Синтаксис оператора: typeof x.
2. Синтаксис функции: typeof(x).

Другими словами, он работает со скобками или без скобок. Результат одинаковый.

Вызов **typeof** **x** возвращает строку с именем типа:

typeof undefined // "undefined"

typeof 0 // "number"

typeof 10n // "bigint"

typeof true // "boolean"

typeof "foo" // "string"

typeof Symbol("id") // "symbol"

typeof Math // "object"  (1)

typeof null // "object"  (2)

typeof alert // "function"  (3)

Последние три строки нуждаются в пояснении:

Math – это встроенный объект, который предоставляет математические операции и константы.

Результатом вызова typeof null является "object". Это официально признанная ошибка в typeof, ведущая начало с времён создания JavaScript и сохранённая для совместимости. Конечно, null не является объектом. Это специальное значение с отдельным типом.

Вызов typeof alert возвращает "function", потому что alert является функцией. Функции относятся к объектному типу. Но typeof обрабатывает их особым образом, возвращая "function". Так тоже повелось от создания JavaScript. Формально это неверно, но может быть удобным на практике.

## Итого

В JavaScript есть 8 основных типов.

**number** для любых чисел: целочисленных или чисел с плавающей точкой; целочисленные значения ограничены диапазоном ±(253-1).

**bigint** для целых чисел произвольной длины.

**string** для строк. Строка может содержать ноль или больше символов, нет отдельного символьного типа.

**boolean** для **true/false**.

**null** для неизвестных значений – отдельный тип, имеющий одно значение **null**.

**undefined** для неприсвоенных значений – отдельный тип, имеющий одно значение **undefined**.

**object** для более сложных структур данных.

**symbol** для уникальных идентификаторов.

Оператор **typeof** позволяет нам увидеть, какой тип данных сохранён в переменной.

Имеет две формы: **typeof x** или **typeof(x)**.

Возвращает строку с именем типа. Например, "string".

Для **null** возвращается **"object"** – это ошибка в языке, на самом деле это не объект.

# Контрольные вопросы

1. Перечислите и опишите типы данных JavaScript.
2. Какой оператор позволяет увидеть, какой тип данных сохранён в переменной?
3. Какой тип данных нужен для хранения одного символа?
4. Что означает «Динамическая типизация»?
5. Какими символами можно обозначать начало и конец строки?
6. Какой особенный функционал возможен в знаке обратной кавычки (**`**)?
7. В каком случае переменной объявляется тип **undefined**?

# Задания

**Задание 1.**

Создайте веб-страницу с html-скелетом и тегом *script*:

<html><head></head>

<body>

<!-- Сценарий -->

<script>

    // для кода JavaScript

</script>

<!-- Конец сценария -->

</body></html>

В коде для javascript инициализируйте переменную для присваивания ей значения – имени, которое введет пользователь в модальное окно *prompt()*:

let name = prompt("Введите Ваше имя");

В результате данного присваивания в переменной name сохранится значение, которое введет пользователь в модальное окно.

Выведите сохраненное значение с помощью метода *document.write()* с несколькими параметрами, разделенными через запятую:

document.write("Привет, ", name," !");

Протестируйте сценарий в браузере.

**Задание 2**. Определите экспериментально тип данных и значение следующих выражений:

1. 1/2;
2. 12/0;
3. true + false;
4. 12 / "6";
5. "number" + 15 + 3;
6. 15 + 3 + "number";
7. [1] > null;
8. "foo" + + "bar";
9. 'true' == true;
10. false == 'false';
11. null == '';
12. !!"false" == !!"true";
13. [‘x’] == ‘x’.

# Литература

**Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.

**Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.

**Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич