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 № 4**

**Инструкционно-технологическая карта**

Тема: Консоль разработчика. Преобразование типов в JavaScript.

Цель: Научиться использовать консоль разработчика. Научиться преобразовывать типы в JavaScript.

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

Код уязвим для ошибок. И, скорее всего, будут совершены ошибки в коде… Впрочем, будучи откровеннее: явно будут совершены ошибки в коде.

Но по умолчанию в браузере ошибки не видны. То есть, если что-то пойдёт не так, это останется незамеченным, что именно сломалось, и не сможет починиться.

Для решения задач такого рода в браузер встроены так называемые «Инструменты разработки» (Developer tools или сокращённо – devtools).

Chrome и Firefox снискали любовь подавляющего большинства программистов во многом благодаря своим отменным инструментам разработчика. Остальные браузеры, хотя и оснащены подобными инструментами, но все же зачастую находятся в роли догоняющих и по качеству, и по количеству свойств и особенностей. В общем, почти у всех программистов есть свой «любимый» браузер. Другие используются только для отлова и исправления специфичных «браузерно-зависимых» ошибок.

Для начала знакомства с этими мощными инструментами, нужно узнать, как их открывать, смотреть ошибки и запускать команды JavaScript.

## Google Chrome

Создать страницу с данным кодом:

<!DOCTYPE HTML>

<html>

<head>

    <meta charset="utf-8">

</head>

<body>

    На этой странице есть ошибка в скрипте.

    <script>

        lalala

    </script>

</body>

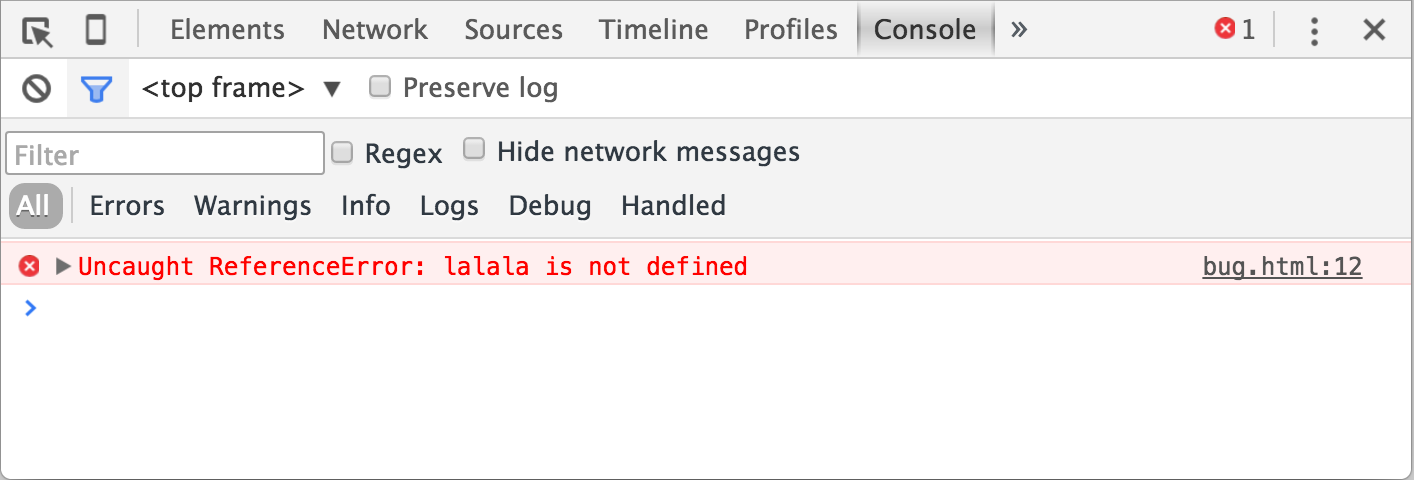
</html>

В её JavaScript-коде закралась ошибка. Она не видна обычному посетителю, поэтому можно найти её при помощи инструментов разработки.

Открывается нажатием на **F12**.

По умолчанию в инструментах разработчика откроется вкладка **Console** (консоль).

Она выглядит приблизительно следующим образом:



Точный внешний вид инструментов разработки зависит от используемой версии **Chrome**. Время от времени некоторые детали изменяются, но в целом внешний вид остаётся примерно похожим на предыдущие версии.

В консоли можно увидеть сообщение об ошибке, отрисованное красным цветом. В этом случае скрипт содержит неизвестную команду «lalala».

Справа присутствует ссылка на исходный код bug.html:12 с номером строки кода, в которой эта ошибка и произошла.

Под сообщением об ошибке находится синий символ «**>»**. Он обозначает командную строку, в ней можно редактировать и запускать JavaScript-команды. Для их запуска нужно нажать Enter.

Обычно при нажатии Enter введённая строка кода сразу выполняется.

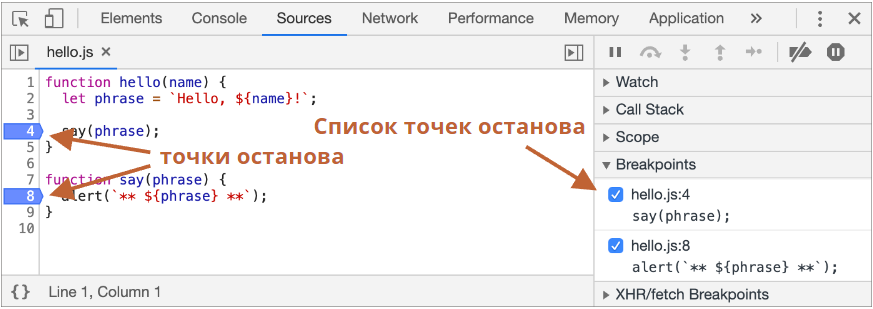
Чтобы перенести строку, нужно нажать **Shift**+**Enter**. Так можно вводить более длинный JS-код.

## Точки останова (breakpoints)

Точка останова – это участок кода, где отладчик автоматически приостановит исполнение JavaScript.

Пока исполнение поставлено «на паузу», можно просмотреть текущие значения переменных, выполнить команды в консоли, другими словами, выполнить отладку кода.

В правой части графического интерфейса можно увидеть список точек останова. А когда таких точек выставлено много, да ещё и в разных файлах, этот список поможет эффективно ими управлять:



## Firefox, Edge и другие

Инструменты разработчика в большинстве браузеров открываются при нажатии на F12.

Их внешний вид и принципы работы мало чем отличаются. Разобравшись с инструментами в одном браузере, можно без труда работать с ними и в другом.

## Преобразование типов

Чаще всего операторы и функции автоматически приводят переданные им значения к нужному типу.

Например, **alert** автоматически преобразует любое значение к строке. Математические операторы преобразуют значения к числам.

Есть также случаи, когда нужно преобразовать значение в ожидаемый тип.

## Строковое преобразование

Строковое преобразование происходит, когда требуется представление чего-либо в виде строки.

Например, **alert(value)** преобразует значение к строке.

Также можно использовать функцию **String(value)**, чтобы преобразовать значение к строке:

let value = true;

alert(typeof value); // boolean

value = String(value); // теперь value это строка "true"

alert(typeof value); // string

Преобразование происходит очевидным образом. **false** становится **"false"**, **null** становится **"null"** и т.п.

## Численное преобразование

Численное преобразование происходит в математических функциях и выражениях.

Например, когда операция деления / применяется не к числу:

alert( "6" / "2" ); // 3, строки преобразуются в числа

Можно использовать функцию **Number(value)**, чтобы преобразовать **value** к числу:

let str = "123";

alert(typeof str); // string

let num = Number(str); // становится числом 123

alert(typeof num); // number

Явное преобразование часто применяется, когда ожидается получить число из строкового контекста, например из текстовых полей форм.

Если строка не может быть явно приведена к числу, то результатом преобразования будет **NaN**. Например:

let age = Number("Любая строка вместо числа");

alert(age); // NaN, преобразование не удалось

Правила численного преобразования:

|  |  |
| --- | --- |
| **Значение** | **Преобразуется в…** |
| undefined | NaN |
| null | 0 |
| true / false | 1 / 0 |
| string | Пробельные символы по краям обрезаются. Далее, если остаётся пустая строка, то получается 0, иначе из непустой строки «считывается» число. При ошибке результат NaN. |

Примеры:

alert( Number("   123   ") ); // 123

alert( Number("123z") );      // NaN (ошибка чтения числа на месте символа "z")

alert( Number(true) );        // 1

alert( Number(false) );       // 0

Учитывая, что **null** и **undefined** ведут себя по-разному. Так, **null** становится **нулём**, тогда как **undefined** приводится к **NaN**.

## Логическое преобразование

Логическое преобразование самое простое. Оно происходит в логических операциях, но также может быть выполнено явно с помощью функции **Boolean(value)**.

Правило преобразования:

Значения, которые интуитивно «пустые», вроде 0, пустой строки, **null**, **undefined** и **NaN**, становятся **false**.

Все остальные значения становятся **true**.

Например:

alert(Boolean(1)); // true

alert(Boolean(0)); // false

alert(Boolean("Привет!")); // true

alert(Boolean("")); // false

# Контрольные вопросы

1. При помощи какой клавиши можно попасть в окно разработчика в браузерах?
2. В каком случае необходимо выполнять строковое преобразование?
3. Что такое точка останова?
4. Что будет возвращено, если при численном преобразовании, если оно невозможно?
5. Что при логическом преобразовании, приводит к **false**?

# Задания

**Задание 1.**

Дана строковая переменная a со значением '10' и строковая переменная b со значением '20'. Сложить данные переменные как числа.

**Задание 2.**

Даны две переменные с числами. Сложить значения этих переменных как строки и вывести результат на экран.

**Задание 3.**

Дана переменная с числом. Найти количество цифр в этом числе, используя функции преобразования.

**Задание 4.**

Даны две переменные с числами. Найти количество цифр в одном и во втором числе, результаты сложить и вывести на экран.

**Задание 5.**

Экспериментальным путём при помощи консоли выясните значения данных выражений.

true + false

8 / "2"

"number" + 5 + 1

5 + 1 + "number"

7 && 2

2 && 7

null + 1

undefined + 1

"five" + + "two"

'true' == true

false == 'false'

null == ''

!!"false" == !!"true"

"4" - 3

"4px" - 3

0 || "0" && 1

**Задание 6.**

Произведите отладку написанного вами кода при помощи точек останова.

# 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