



Разработка веб-приложений. HTML5 и JavaScript

Практические работы

Оглавление

Оглавление	3
Упражнение 1. Введение в разработку веб-приложений	5
Раздел 1. Создание проекта в NetBeans	6
Раздел 2. Автоматическая проверка страницы (опционально, выполняется при наличии интернет подключения)	11
Раздел 3. Инструменты разработчика Chrome (опционально, выполняется при наличии интернет подключения)	14
Упражнение 2. JavaScript. Операторы, условия и циклы	19
Раздел 1. Калькулятор	20
Упражнение 3. JavaScript. Строки, массивы и объекты	27
Раздел 1. Форматирование вывода числа	28
Раздел 2. Регулярные выражения	30
Раздел 3. Хранение информации в массивах и объектах	32
Упражнение 4. Основы HTML5	34
Раздел 1. Создание страницы контактов	35
Раздел 2. Автоматическая проверка форм HTML5	39
Раздел 3. Объектная модель документа	42
Упражнение 5. Описание стилей документов с использованием CSS3	45
Раздел 1. Демонстрация CSS	46
Раздел 2. Отступы	47
Раздел 4. Поля элемента	50
Раздел 5. Псевдоклассы CSS	52
Раздел 6. Медиазапросы	54
Раздел 7. Анимация переходов	55
Раздел 8. Анимация	58
Раздел 9. Изменение стилей с помощью JavaScript	61
Упражнение 6. Графика и анимация HTML5	63
Раздел 1. Создание аналоговых часов с помощью HTML Canvas	64
Раздел 2. Захват и перенос элементов HTML Drag and Drop	70

Упражнение 7. JavaScript. Функции и объекты	74
Раздел 1. Функциональные выражения.....	75
Раздел 2. Передача функции в качестве параметра. Ключевое слово this.....	76
Раздел 3. Замыкания	78
Раздел 4. Разработка внешних модулей. Проблема счетчика	80
Раздел 5. Конструктор объекта	82
Упражнение 8. Передача и хранение данных веб-приложения	85
Раздел 1. Локализация сайта.....	86
Раздел 2. Работа с Cookie.....	90
Раздел 3. Работа с локальным хранилищем HTML5 Local Storage	92
Упражнение 9. Передача данных на сервер. Технологии AJAX и WebSocket	95
Раздел 1. Локализация сайта. Запрос данных с помощью AJAX.....	96
Раздел 2. Многопользовательский чат на WebSocket	99
Упражнение 10. Библиотека jQuery	104
Раздел 1. Крестики-нолики	105

Упражнение 1. Введение в разработку веб-приложений

О чем это упражнение:

В этой лабораторной работе Вы познакомитесь с инструментами разработки веб-приложений.

Что Вы должны будете сделать:

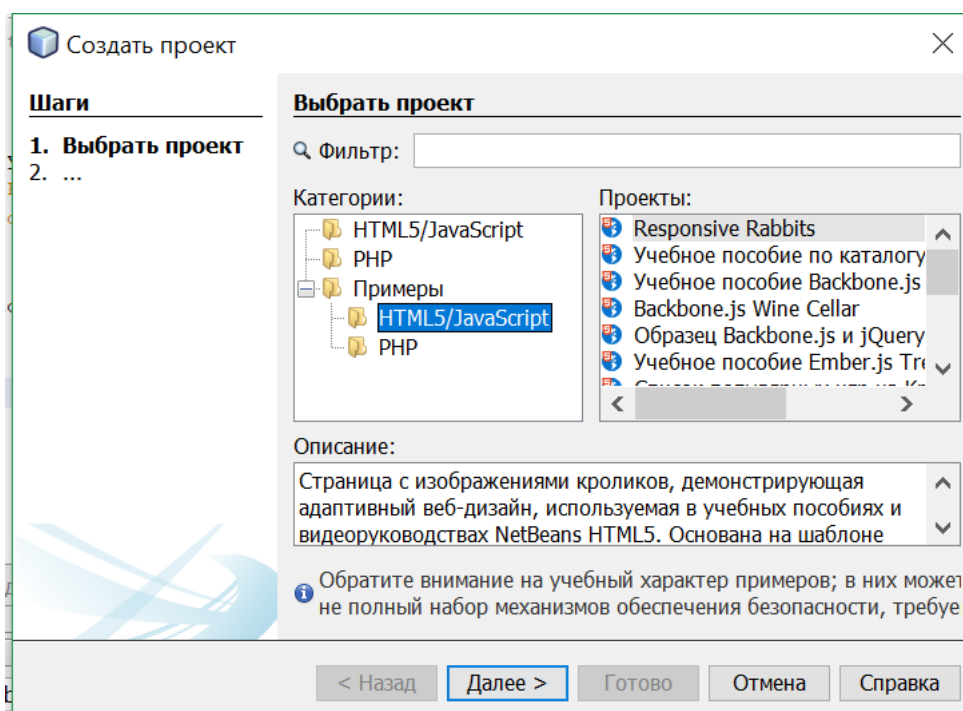
- Проверить код HTML5 и CSS сайта с помощью автоматических валидаторов
- Найти и устранить ошибку на сайте с помощью инструментов разработчика Google Chrome
- Познакомиться со средой разработки NetBeans

Раздел 1. Создание проекта в NetBeans

NetBeans IDE – полнофункциональная среда разработки веб-приложений на HTML5 и JavaScript.

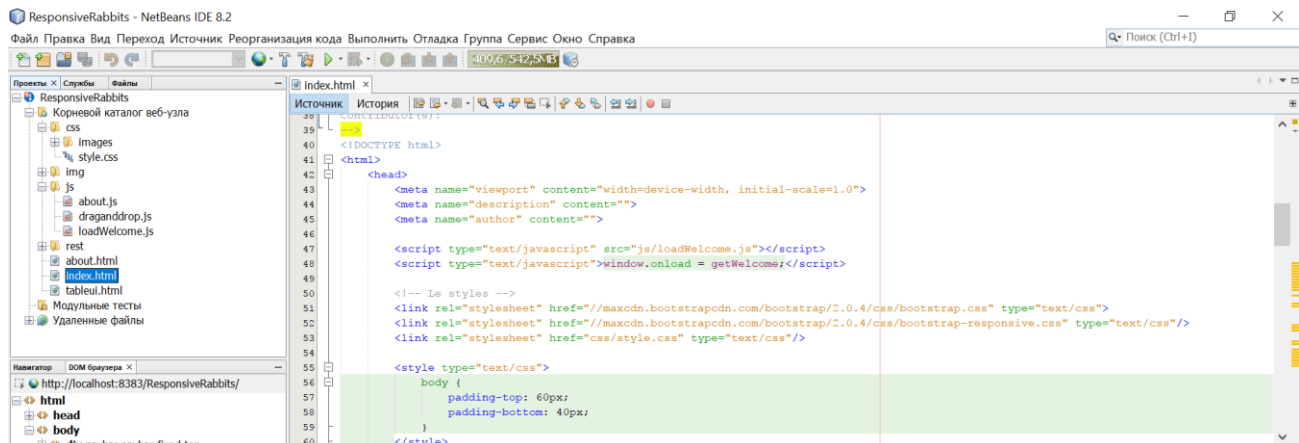
В этой лабораторной Вы познакомитесь с возможностями NetBeans на примере тестового проекта **ResponsiveRabbits**.

1. Реквизиты для доступа к виртуальной машине:
Учетная запись: **Администратор**
Пароль: **Java1Script**
2. Запустите NetBeans IDE, ярлык находится на рабочем столе
3. Выберите **Файл – Создать проект** или нажмите **Ctrl + Shift + N**
 - а. Выберите учебный пример **Responsive Rabbits**. Нажмите **Далее** и **Готово**.



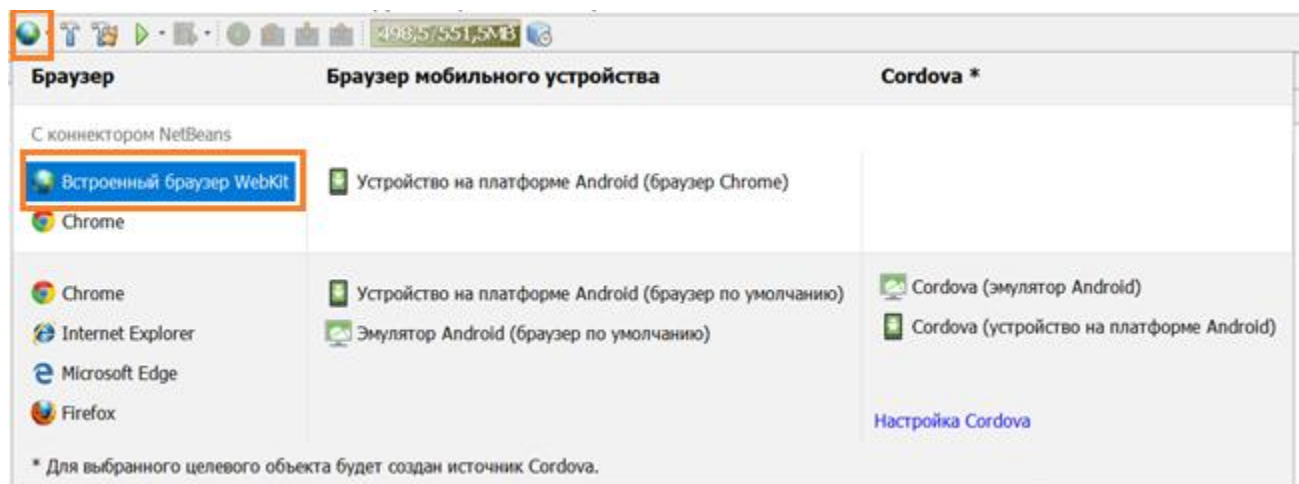
4. Изучите список файлов проекта
 - а. Файлы с расширением **html** описывают страницы сайта.
 - б. Файлы в каталоге **js** – скрипты JavaScript, которые позволяют сайту реагировать на действия пользователя
 - в. Файлы в каталоге **css** - стили CSS, описывающие вид элементов

- d. Файлы в каталоге **rest** – скрипты для отправки данных на сервер. В данном проекте нет серверной части, поэтому в каталоге хранятся скрипты-заглушки.



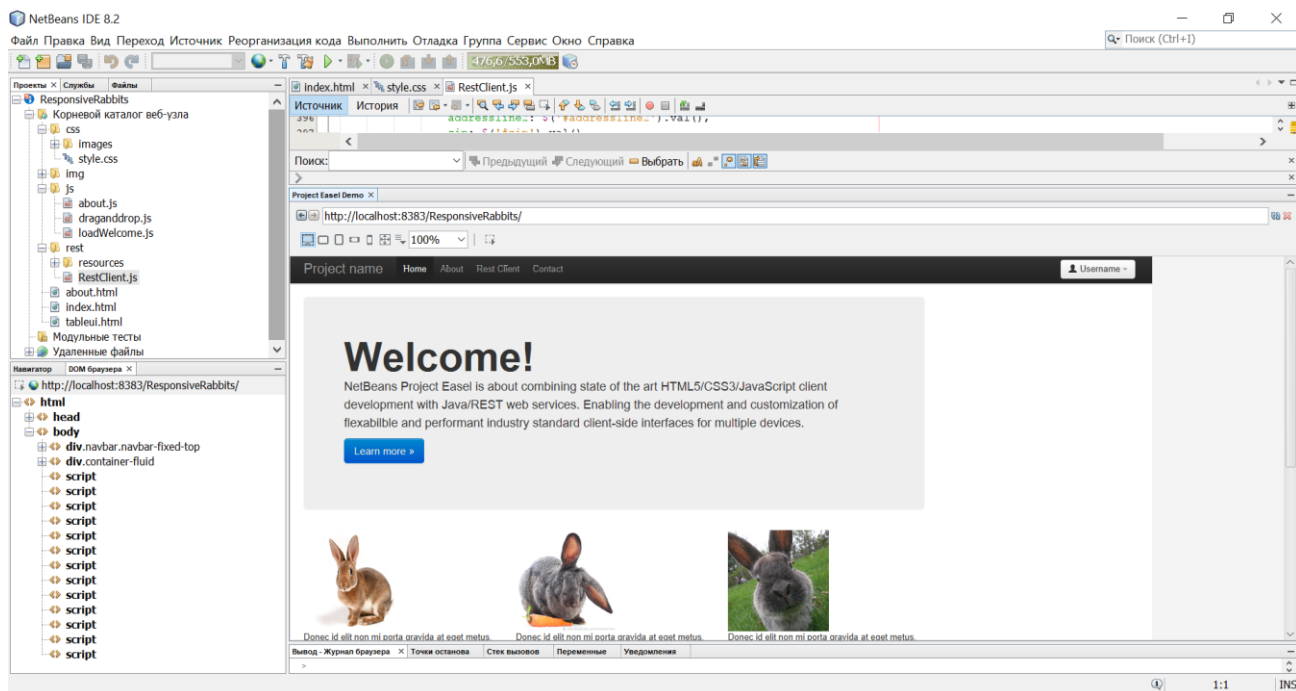
5. Запустите проект во встроенном браузере WebKit

- a. Выберите браузер WebKit для работы с проектом:
b. Нажмите **F6**




6. В нижней части экрана откроется вкладка с браузером, обрабатывающим страницу

- a. Поэкспериментируйте со страницей: проверьте работу переходов, анимации и пр.
b. Проверьте наличие сообщений об ошибках в консоли браузера

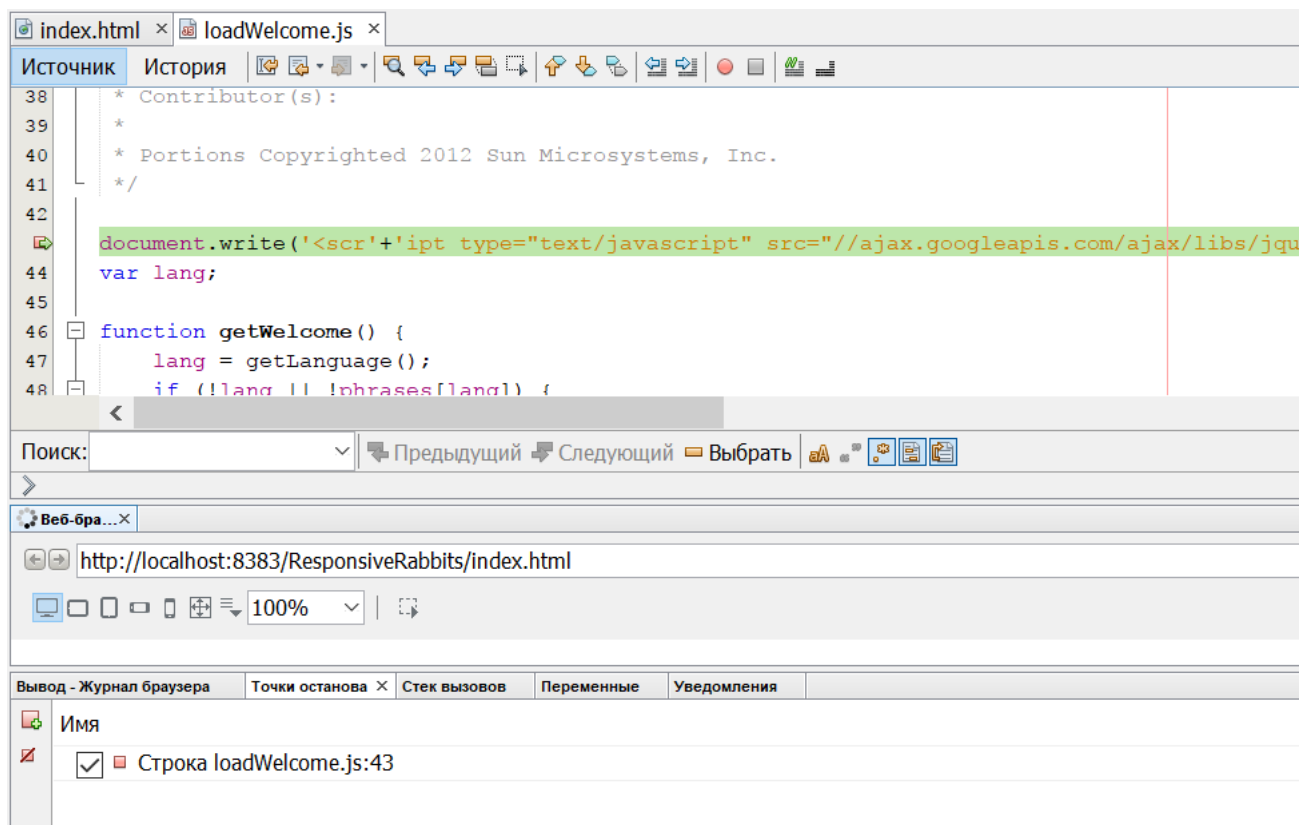


7. Одним из важных принципов веб-разработки является принцип отзывчивого (responsive) дизайна, при котором страница меняет вид в зависимости от ширины экрана устройства, отображающего ее.

а. Используя кнопки  встроенного браузера, проверьте, как будет отображаться эта страница на различных устройствах

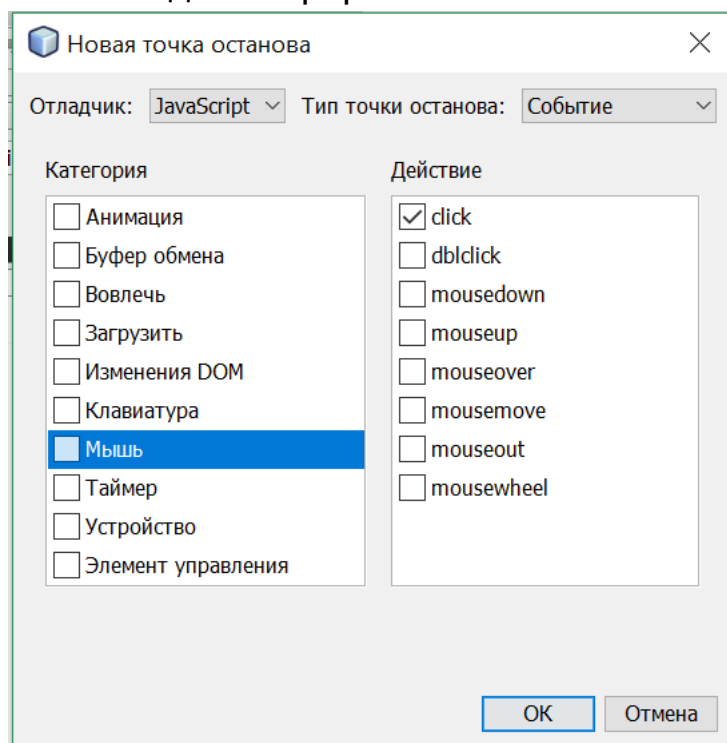
8. NetBeans имеет широкий функционал отладки веб-приложений. Один из базовых инструментов отладки – прерывания кода

а. Установите прерывание на строке 43 скрипта **loadWelcome.js** и обновите страницу

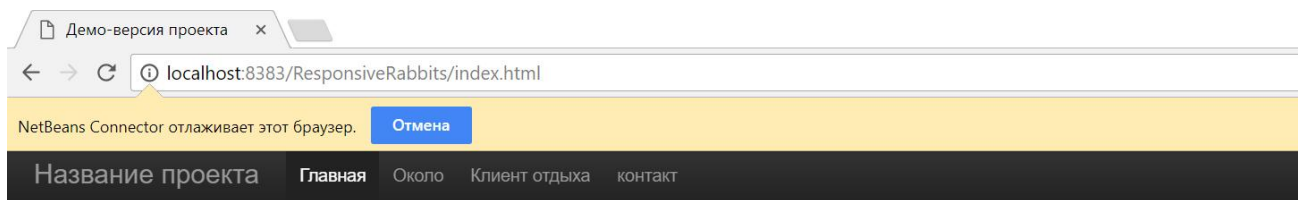


9. NetBeans также поддерживает прерывания по событию.

- Откройте **Отладка – Создать точку останова** или нажмите **Ctrl+Shift+F8**
- Создайте прерывание по нажатию левой кнопки мыши



10. NetBeans позволяет открывать сайт не только во встроенном браузере, но и во многих других, если в браузере установлено расширение NetBeans Connector.
- Запустите проект **ResponsiveRabbits** в браузере Google Chrome, используя интерфейс, который раньше использовался для указание браузера WebKit.
 - Обратите внимание на желтый баннер – он сообщает, что NetBeans отслеживает все изменения и события станицы
 - Нажмите любую кнопку и перейдите в NetBeans для анализа прерывания.



Добро пожаловать!

NetBeans Project Easel - это сочетание современной клиентской разработки HTML5 / CSS3 / JavaScript с веб-службами Java / REST. Обеспечение разработки и адаптации гибких и высокопроизводительных стандартных клиентских интерфейсов для нескольких устройств.

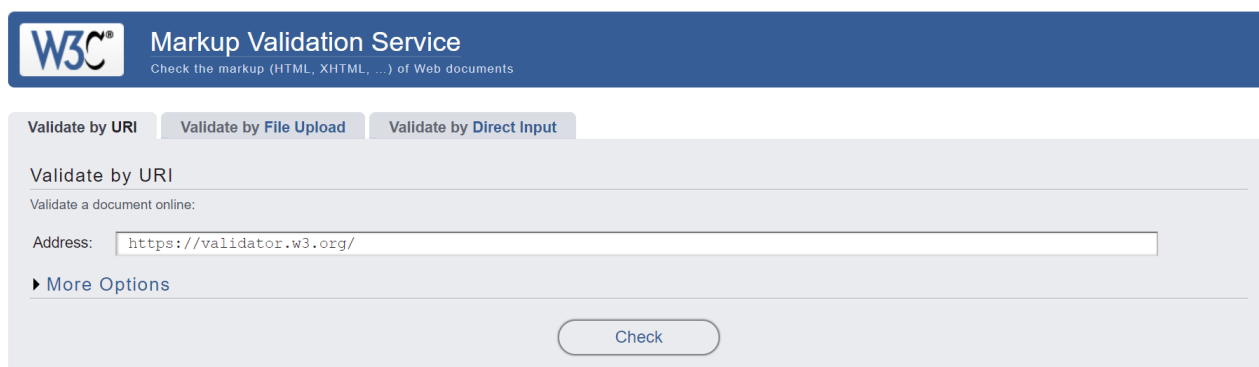
[больше информации »](#)



Раздел 2. Автоматическая проверка страницы (опционально, выполняется при наличии интернет подключения)

Сервис автоматической проверки позволяет отследить возможные ошибки HTML, CSS и пр. Несмотря на наличие ошибок и предупреждений, страница может отображаться некоторыми браузерами, однако во многих организациях принято правило обязательной проверки кода перед публикацией.

1. Сайт <https://validator.w3.org/> позволяет произвести автоматическую проверку страницы, доступной по указанному адресу.
 - а. Укажите адрес любой страницы, например страницы самого сайта <https://validator.w3.org/>



This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available. As an alternative you can also try our [non-DTD-based validator](#).

- b. Нажмите **Check**
 - c. Вы увидите несколько предупреждений об использовании устаревшего атрибута **type="text/javascript"** при подключении скриптов

Showing results for <https://validator.w3.org/>

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by address ▾

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

- Warning** The `type` attribute is unnecessary for JavaScript resources.
From line 475, column 1; to line 475, column 54
`<div><script type="text/javascript" src="scripts/combined"></scri`
- Warning** The `type` attribute is unnecessary for JavaScript resources.
From line 477, column 1; to line 477, column 74
`/script><script type="text/javascript" src="//www.w3.org/QA/Tools/w3c-include.js"></scri`

2. Поэкспериментируйте с включением опций

- Опция `source` показывает исходный код страницы
- Опция `outline` группирует информацию о заголовках различных типов
- Опция `Image` выводит информацию об изображениях и подписях





Image report

The `img` elements of the page are shown below categorized by their type of textual alternative. Please review that the images in each group match that group's definition.

Images with textual alternative

The following images have textual alternatives. Please review that the textual alternatives make sense considering the purpose of the image in the context of the page and that phrases like "Image of ..." are avoided.

Note that iconic images that are redundant with text next to them or purely decorative should have `alt=""` instead.

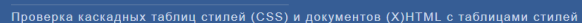
Image	Textual alternative	Location
	W3C Logo	From line 24, column 11; to line 24, column 89
	W3C	From line 440, column 3; to line 440, column 83
	Open-Source	From line 441, column 3; to line 441, column 172
	I heart Validator logo	From line 446, column 4; to line 446, column 145

Used the HTML parser. Externally specified character encoding was utf-8.
Total execution time 34 milliseconds.

3. Аналогичным образом проверьте любой сайт валидатором CSS

<http://jigsaw.w3.org/css-validator/>

- Проверьте сайт на соответствие различным стандартам
- Проверьте сайт на отображение в различных средах



Проверить набранный текст

Введите URI документа (HTML с CSS или только CSS) для проверки:

Расширения поставщика: По-умолчанию ▼

[Проверить](#)

Раздел 3. Инструменты разработчика Chrome (опционально, выполняется при наличии интернет подключения)

Браузер Google Chrome предоставляет широкий спектр инструментов для отладки сайта.

1. В браузере Google Chrome откройте страницу <https://googlechrome.github.io/devtools-samples/debug-js/get-started>
 - а. Этот сайт должен вычислять сумму двух чисел. Введите числа 5 и 6. Проверьте результат

Demo: Get Started Debugging JavaScript with Chrome DevTools

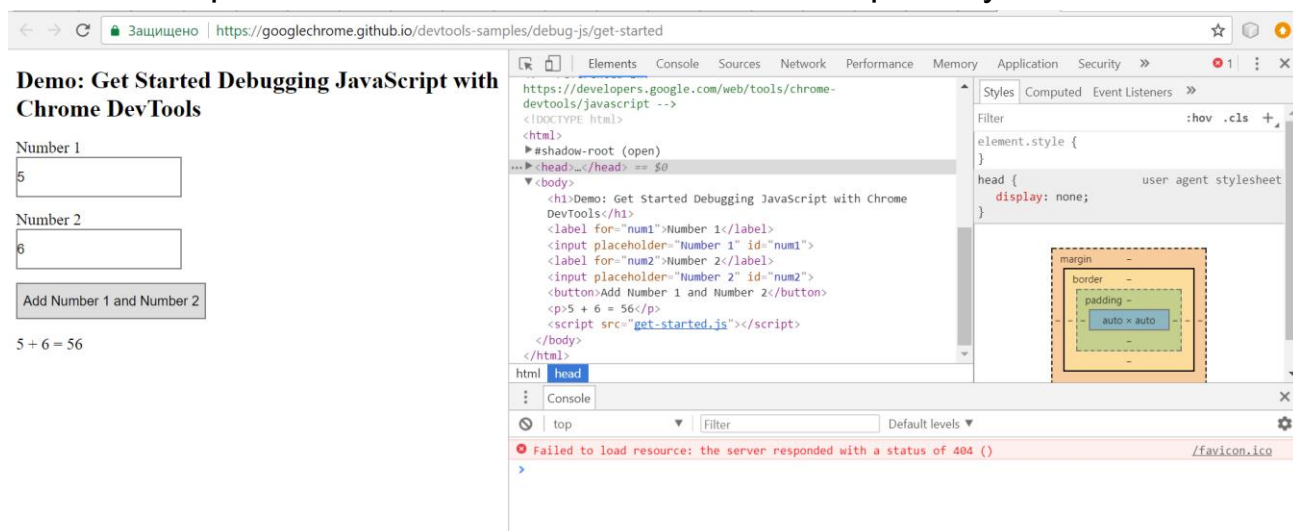
Number 1

Number 2

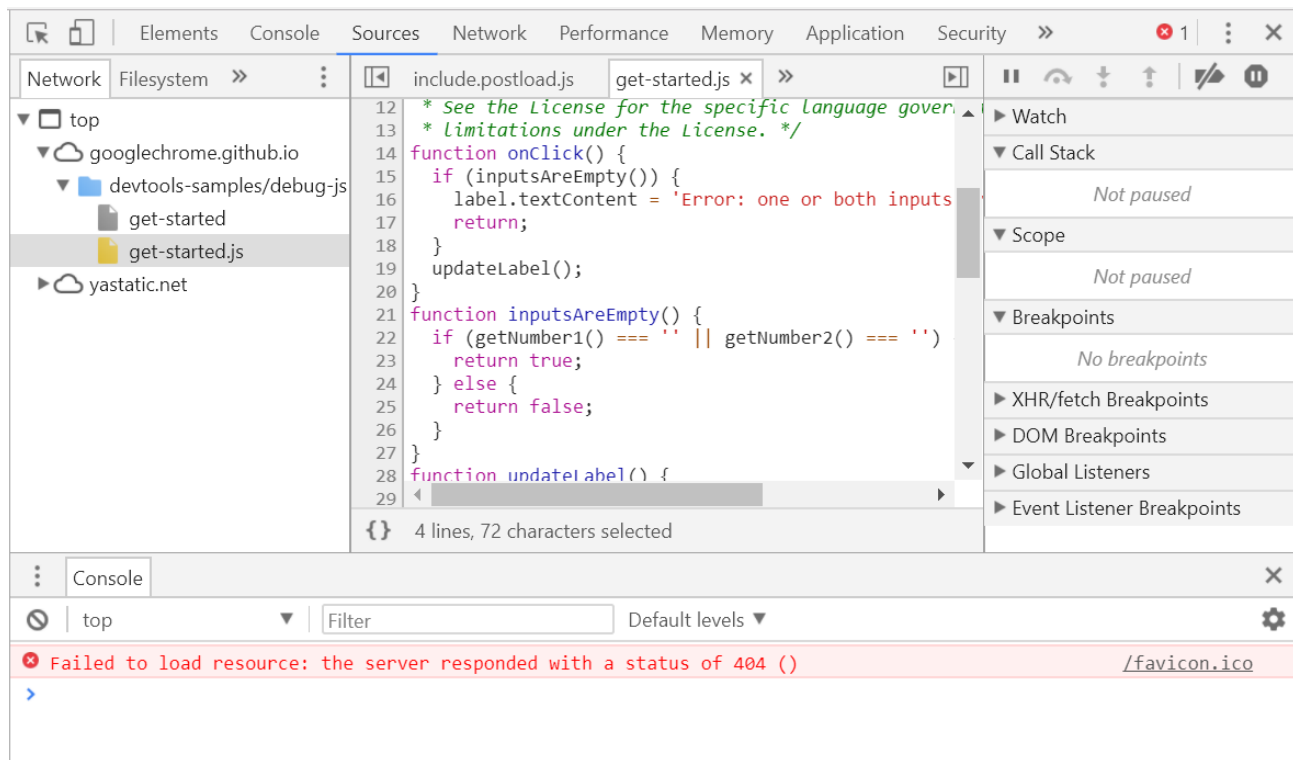
Add Number 1 and Number 2

5 + 6 = 56

2. Откройте инструменты разработчика, нажав **Ctrl + Shift + I**
 - а. Вы можете увидеть ошибку о неуспешной загрузке ресурса **favicon.ico** (изображение в описании вкладки). Эта ошибка не критична и не имеет отношения к неверной сумме

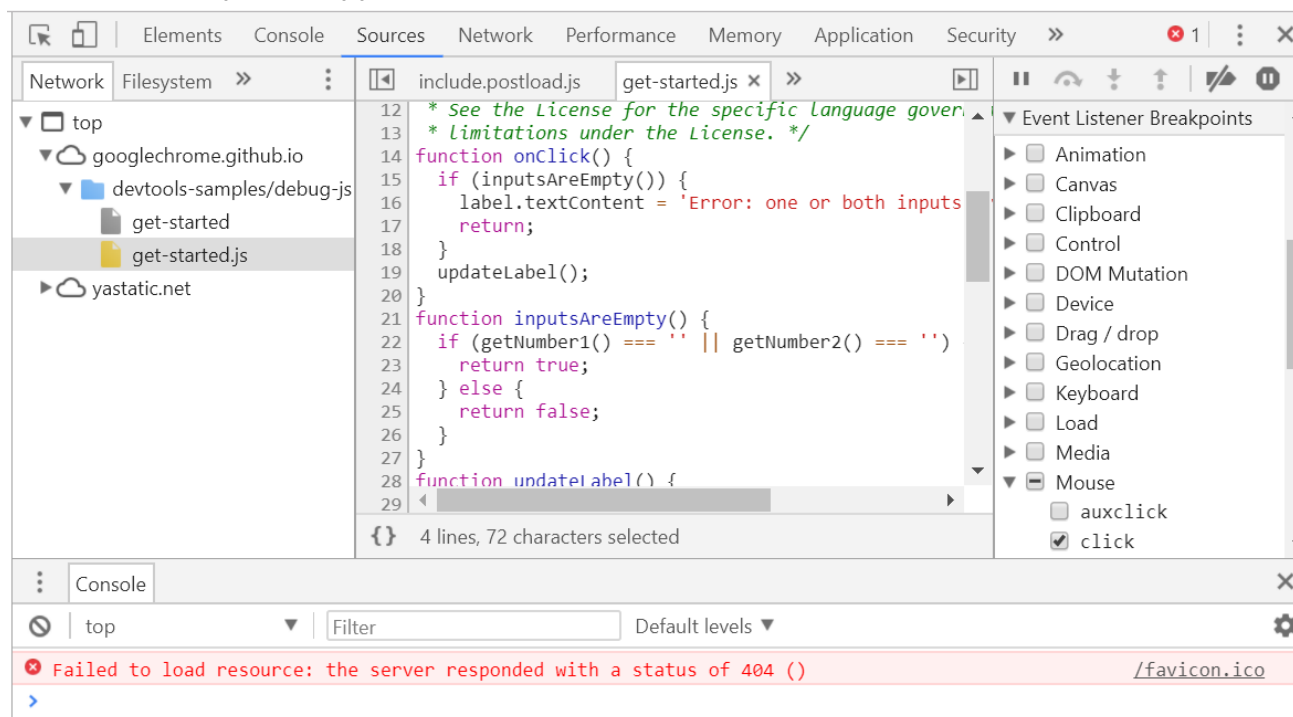


3. Разверните элемент **<body>** в коде HTML во вкладке **Elements**.
Определите идентификаторы первого и второго поля ввода
4. Откройте вкладку **Sources** для изучения кода JavaScript




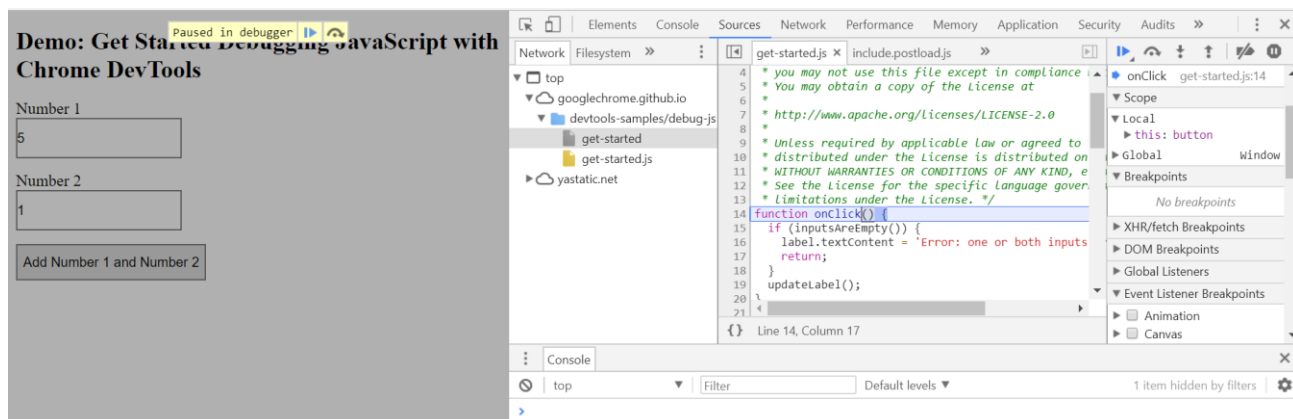
5. Для определения проблемного участка кода можно воспользоваться прерываниями.


- Известно, что сумма вычисляется после нажатия на кнопку
- Можно установить прерывание по нажатию на левую кнопку мыши
- В правой (или нижней, в зависимости от разрешения экрана) секции найдите **Event Listener BreakPoints – Mouse – click**




6. Введите два числа и нажмите кнопку.

- а. Нажимайте на синий треугольник  (**Resume Script Execution F8**) до тех пор, пока не попадете на функцию **onClick()**. Если вы проскочили эту функцию, то можно нажать на кнопку еще раз.

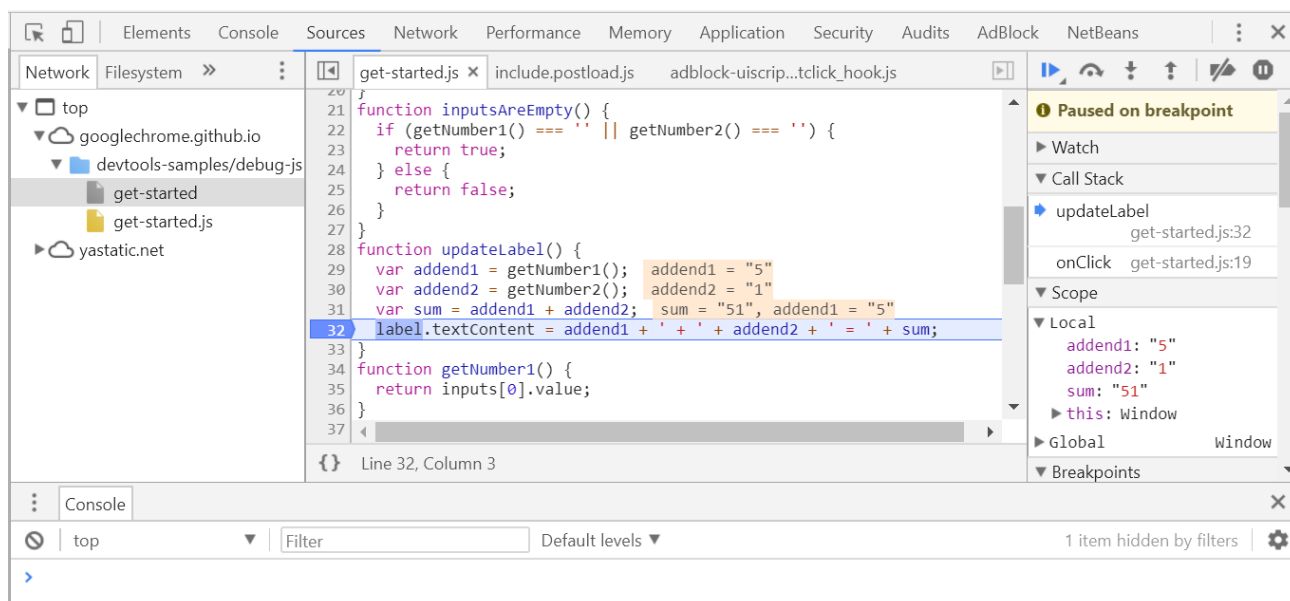


7. Далее выполняйте код построчно с помощью кнопки  (**Step into next function call**).

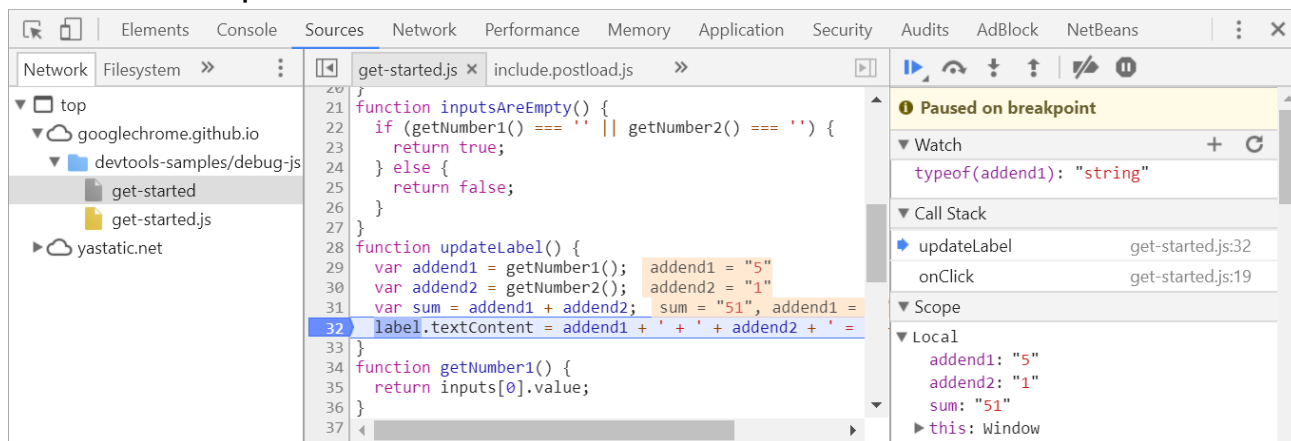
- а. Кнопка  (**Step over next function call**) позволяет не переходить к построчному выполнению функций, в работе которых нет проблем (например функция **inputsAreEmpty()**)

8. С помощью построчного выполнения кода можно найти строку кода, вычисляющую сумму.

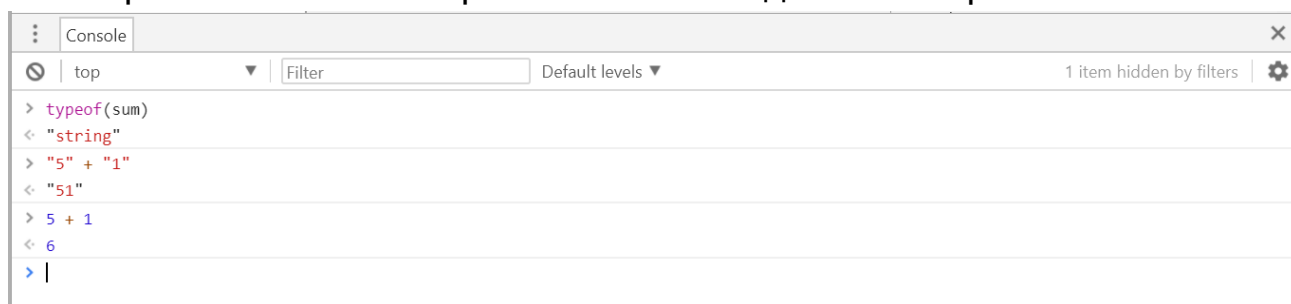
- а. Установите прерывание на эту строку для быстрого перехода к ней



9. Обратите внимание на область **Scope – Local**, которая показывает текущее значение переменных
10. Область **Watch** позволяет Вам наблюдать за значением любых переменных и выражений
 - а. Например, добавьте наблюдение за значением типа переменной **addend1**



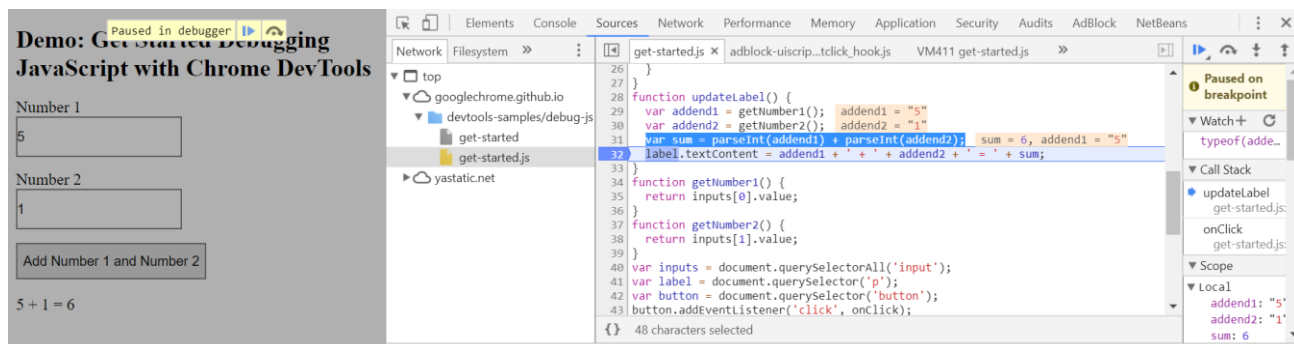
11. Консоль в нижней части экрана позволяет Вам в реальном времени выполнять различные команды JavaScript



12. Итого: проблема заключается в том, что вместо арифметического сложения используется сложение строк.
13. Измените строку вычисления сложения на строку

```
var sum = parseInt(addend1) + parseInt(addend2);
```

14. Нажмите **Ctrl+S** и еще раз запустите страницу. Проверьте, что теперь результат сложения верный
 - а. Обновлен только код, сохраненный на стороне клиента.
 - б. Для полного исправления ошибки необходимо изменить код на сервере.



Упражнение 2. JavaScript.

Операторы, условия и циклы

О чем это упражнение:

В этой лабораторной работе Вы напишете простой код на JavaScript и протестируете его.

Что Вы должны будете сделать:

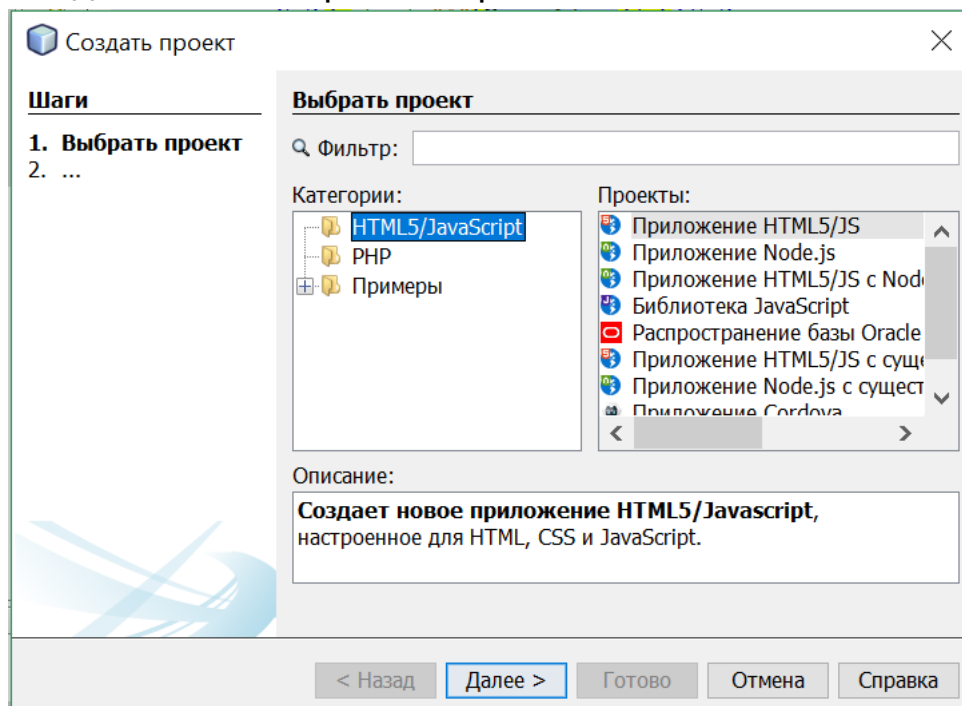
- Разработать скрипт-калькулятор
- Установить защиту от данных неверного формата
- Применить циклы для вычисления сложных операций

Раздел 1. Калькулятор

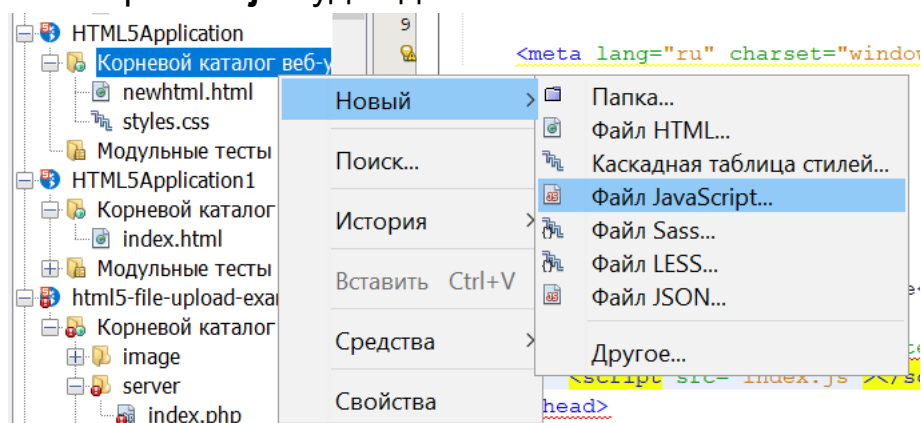
Создайте скрипт-калькулятор, который

- Запрашивает значения операторов у пользователя с помощью метода **prompt()**
- Выводит результат различных математических функций с помощью **alert()**.

1. Создайте новый проект «Приложение HTML5/JS» в NetBeans



2. Добавьте в проект новый файл типа JavaScript. Назовите его **calc**. Расширение **.js** будет добавлено автоматически.



3. Внутри секции **<head>** страницы **index.html** измените название “TODO supply a title” на “**JavaScript Calculator**”

4. Подключите скрипт **calc.js** к HTML-файлу. Внутри секции <head> добавьте строку с тегом script:

```
<head>
  <title> JavaScript Calculator </title>

  <script src="calc.js"></script>

</head>
```

5. Дальнейший код пишите в файле **calc.js**
6. Объявите переменные **x** и **y**. Запросите их значение у пользователя

```
var x, y;
x = prompt ('Введите X', 0);
y = prompt ('Введите Y', 0);
```

7. Объявите переменные для хранения суммы, разницы, частного и остатка от деления. Вычислите их значение.

```
var sum, sub, mod, div;
```

```
sum = x + y;
sub = x - y;
div = x / y;
mod = x % y;
```

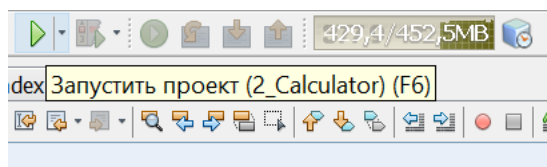
8. Объявите переменную **result** и запишите в нее текст, который Вы хотите выдать пользователю. Выдайте значение переменной пользователю с помощью вызова **alert()**

```
var result;
result = "Результаты математических операций: \n";
```

```
result += x + ' + ' + y + ' = ' + sum + '\n';
result += x + ' - ' + y + ' = ' + sub + '\n';
result += x + ' / ' + y + ' = ' + div + '\n';
result += x + ' % ' + y + ' = ' + mod + '\n';
```

```
alert (result);
```

9. Запустите код, нажав **F6** или нажав на зеленый треугольник.



10. Проверьте работу вашего кода. Нет ли логических ошибок в работе скрипта? Обратите внимание на вычисленную сумму.

Результаты математических операций:

$3 + 4 = 34$

$3 - 4 = -1$

$3 / 4 = 0.75$

$3 \% 4 = 3$

11. Причина некорректного поведения заключается в том, что переменные **x** и **y** – строки. Поэтому вместо арифметической суммы оператор **+** выдал результат конкатенации строк. Прочие операторы для строк производят неявное преобразование в числа. Укажите явное преобразование переменных в числовой формат после их ввода пользователем. После этого повторите запуск.

```
x = parseFloat(x);
```

```
y = parseFloat(y);
```

Результаты математических операций:

$3 + 4 = 7$

$3 - 4 = -1$

$3 / 4 = 0.75$

$3 \% 4 = 3$

12. Протестируйте код на более сложных примерах, например отрицательных числах и дробных. Убедитесь, что результаты корректные.

Результаты математических операций:

$1 + 3 = 4$

$1 - 3 = -2$

$1 / 3 = 0.3333333333333333$

$1 \% 3 = 1$

13. Измените формат вывода результата, например, скорректируйте количество отображаемых знаков после запятой.

```
result = "Результаты математических операций: \n";
result += x + ' + ' + y + ' = ' + sum.toFixed(2) + '\n';
result += x + ' - ' + y + ' = ' + sub.toFixed(2) + '\n';
result += x + ' / ' + y + ' = ' + div.toFixed(2) + '\n';
result += x + ' % ' + y + ' = ' + mod.toFixed(2) + '\n';
```

Результаты математических операций:

$1 + 3 = 4.00$

$1 - 3 = -2.00$

$1 / 3 = 0.33$

$1 \% 3 = 1.00$

14. Протестируйте работу вашего кода в случае, когда пользователь указал строки, а не числа.

Результаты математических операций:

$\text{NaN} + \text{NaN} = \text{NaN}$

$\text{NaN} - \text{NaN} = \text{NaN}$

$\text{NaN} / \text{NaN} = \text{NaN}$

$\text{NaN} \% \text{NaN} = \text{NaN}$

15. Для того чтобы защитить код от нечислового значения **NaN** (Not a Number) используйте функцию **isNaN()**.

- а. Если дополнительно Вы хотите проверять на бесконечность (**Infinite**, **-Infinite**) используйте функцию **isFinite()**

```
if (isNaN(x) || isNaN(y))
{
    alert ('Ожидаются числа');
} else
```

```
{  
    // Вычисление математических функций  
}
```

16. Протестируйте работу вашего кода в случае, когда **Y == 0**.

```
Результаты математических операций:  
4 + 0 = 4.00  
4 - 0 = 4.00  
4 / 0 = Infinity  
4 % 0 = NaN
```

17. Добавьте проверку на числовое значение перед выдачей пользователю значений частного и остатка

```
if (isFinite(div)) {  
    result += x + ' / ' + y + ' = ' + div.toFixed(2) + '\n';  
}  
if (isFinite(mod)) {  
    result += x + ' % ' + y + ' = ' + mod.toFixed(2) + '\n';  
}
```

18. Сейчас Ваш код должен запрашивать у пользователя значение и завершаться, если введенное значение неверно. Измените код таким образом, чтобы он запрашивал у пользователя значение переменной до тех пор, пока пользователь не введет число или не нажмет кнопку **Отмена** (или **ESC**). Ниже приведен пример для **x**, аналогично нужно сделать и для **y**.

```
do  
{  
    x = prompt ('Введите X', 0);  
} while (isNaN(x) && x != null)
```

19. Предполагая, что **Y** положительное целое число, реализуйте операцию возведения **X** в степень **Y** с помощью цикла **for**.

```
var pow = 1;  
for (i = 0; i < y; i++)  
{
```



```
    pow *= x;
  }
  result += x + ' ** ' + y + ' = ' + pow.toFixed(2) + '\n';
```

20. Опционально. Добавьте проверку на то, что **Y** целое положительное число.

а. Проверку на целое число можно реализовать с помощью битового оператора сложения по модулю.

```
if (((y ^ 0) === y) && (y > 0))
{
  // Вычислить результат возведения в степень x** y
}
```

21. Опционально. Добавьте в скрипт операции перевода чисел в двоичный код и выполните над ними побитовую операцию **И** и **ИЛИ**

```
var and, or;
```

```
and = x & y;
or = x | y;
```

```
result += dec2bin(x) + ' & ' + dec2bin(y) +
  ' = ' + and + '\n';
```

```
result += dec2bin(x) + ' | ' + dec2bin(y) +
  ' = ' + or + '\n';
```

```
function dec2bin(dec){
  return (dec >>> 0).toString(2);
}
```

Результаты математических операций:

$5 + -5 = 0.00$

$5 - -5 = 10.00$

$5 / -5 = -1.00$

$5 \% -5 = 0.00$

$101 \& 111111111111111111111111111111011 = 1$

$101 | 111111111111111111111111111111011 = -1$

22. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\2_Calculator

Упражнение 3. JavaScript. Строки, массивы и объекты

О чем это упражнение:

В этой лабораторной Вы научитесь писать более сложный код на JavaScript с использованием функций, регулярных выражений, массивов и объектов.

Что Вы должны будете сделать:

- Создать функцию, форматирующую вывод числа
- С помощью регулярных выражений найти все числа в строке
- Написать скрипт, использующий массивы и объекты для хранения информации

Раздел 1. Форматирование вывода числа

Напишите функцию, которая будет принимать в качестве параметра число **number=123456789** и выдавать строку “**123 456 789**”

Протестируйте свое решение на следующих примерах:

- Числа ‘12’, ‘1 234’, ‘12 345’
- Отрицательные числа ‘-123 456 789’
- Дробные числа ‘1 234.56’, ‘1 234.56 789’

Попробуйте сначала разработать функцию самостоятельно.

1. Создайте новый проект.
 - a. Измените заголовок
 - b. Добавьте новый файл JavaScript и подключите его в проект
2. Создайте функцию **formatNumber**, принимающую параметр **number** и возвращающую строку **result**

```
function formatNumber(number) {  
    var result = '';  
    return result;  
}
```

3. Преобразуйте число **number** в строку. Добавьте в функцию цикл, проходящий строку справа налево и отрезающий по три символа.

```
var num = number + '';  
for (var i = num.length - 3; i >= 0; i -= 3) {  
    result = ' ' + num.substring(i, i + 3) + result;  
}
```

4. Остаток строки добавьте к строке **result**.

```
result = num.slice(0, num.length % 3) + result;
```

5. Сейчас Ваша функция должна корректно форматировать натуральные числа. Для того, чтобы обрабатывать отрицательные числа можно воспользоваться следующим подходом:

- a. Если **number** отрицательное (например -1234), то вычислить модуль (абсолютное значение) number: 1234
- b. Вызвать функцию форматирования полученного числа. Получим “1 234”
- c. Выдать результат форматирования, добавив в начало знак минус: “-1 234”

Добавьте в начало функции следующий код:

```
if (number < 0) {  
    result = formatNumber(Math.abs(number));  
    return '-' + result;  
}
```

6. Аналогичный принцип можно использовать для дробных чисел:

- a. Выявить дробные числа (содержат точку)
- b. Отдельно отформатировать дробную и целую часть
- c. Выдать результат, объединив их точкой

```
var indexPoint = num.indexOf('.');  
  
var fraction, real;  
if (indexPoint != -1 ) {  
    fraction = formatNumber(num.substring(indexPoint+1));  
    real = formatNumber (num.substring(0,indexPoint));  
    result = real + '.' + fraction;  
    return result;  
}
```

7. Готовое решение Вы можете найти в

C:\Users\Администратор\Documents\solutions\3_NumberFormat

8. Также для решения этой задачи можно воспользоваться встроенной функцией **number.toLocaleString()**

Раздел 2. Регулярные выражения

С помощью регулярного выражения найдите все числа в строке.

Например, для строки “1 123 123.456 0.456 -123 12abc3 123.
.123” должны быть найдены числа 1 123 123.456 0.456 -123

1. Продолжайте разработку в том же проекте
2. Для простоты отладки создайте строку статично.

```
var str = “1 12 123.46 0.46 -123 12ab3 123. .123”
```

3. Напишите регулярное выражение, которое выбирает из строки все натуральные числа.

```
var numregex = /\d+/igm  
console.log (str.match (numregex));
```

4. Измените регулярное выражение так, чтобы были выбраны в т.ч. отрицательные числа

```
var numregex = /-?\d+/igm  
console.log(str.match (numregex));
```

5. Измените регулярное выражение так, чтобы были выбраны в т.ч. дробные числа (символ разделитель точка).

- а. Помните о необходимости экранировать специальные символы (в т.ч. точку) в регулярных выражениях

```
var numregex = /-?\d+(\.\d+)?/igm  
console.log(str.match (numregex));
```

6. Обратите внимание, что сейчас регулярное выражение находит дополнительно числа 12, 3 и 123, которые являются частью нечисловых строк. Измените регулярное выражение таким образом, чтобы оно осуществляло поиск по целым словам.

- а. В данном случае использовать шаблон **\b** (обозначающий конец или начало слова) не удастся, т.к. точка считается символом, разделяющим слова

- b. Можно указать более сложную последовательность: слово может начинаться либо с начала строки **^**, либо с пробельного символа **\s** и заканчиваться либо концом строки **\$**, либо пробельным символом.

```
var numregex = /^(^|\s)-?\d+(\.\d+)?(\s|$)/igm  
str.match (numregex);
```

7. Опционально. Используйте функцию **formatNumber()** и регулярные выражения для того, чтобы отформатировать все числа в строке, введенной пользователем.

- a. Используйте следующий формат вызова метода **string.replace (regexp, replace_function)**

```
var str = prompt ('Введите строку с числами');  
  
var numregex = /^(^|\s)-?\d+(\.\d+)?(\s|$)/igm  
str = str.replace (numregex, formatNumber);  
  
alert (str);
```

Раздел 3. Хранение информации в массивах и объектах

Напишите скрипт, запрашивающий у пользователя информацию о городах и обрабатывающий ее.

- Информация о городах должна храниться в массиве
- По каждому городу должна храниться следующая информация: название, страна, количество населения
- Скрипт должен находить город с самым большим населением и выдавать информацию о нем пользователю.

1. Создайте новый проект
2. Создайте массив городов **arrayTown** и запросите у пользователя информацию по каждому городу.

- a. Информацию о городе сохраните в виде объекта
- b. Информация о населении должна храниться в виде числа

```
var arrayTown = [];  
  
while (confirm('Вы хотите ввести город?'))  
{  
  
    var town = {};  
  
    town.townName = prompt ('Укажите название:');  
  
    town.state = prompt ('Укажите страну:');  
  
    town.population = prompt('Укажите население:');  
  
    town.population = parseInt(town.population);  
  
    arrayTown.push(town);  
  
}
```

3. Опционально. Добавьте проверку, на то, что введенное пользователем количество населения можно преобразовать в число
4. Напишите функцию сравнения двух городов
 - a. Функция должна выдавать положительное значение, если население города А больше города Б, и отрицательное в ином случае


```
function compareTowns (townA, townB)
{
    return townA.population - townB.population;
}
```

5. Отсортируйте массив с помощью этой функции

```
arrayTown.sort(compareTowns);
```

6. Выдайте пользователю информацию о самом населенном городе
а. Массив отсортирован по возрастанию

```
alert ('Самый населенный город ' +
    arrayTown[arrayTown.length - 1 ].townName +
    ' с населением в ' +
    arrayTown[arrayTown.length - 1 ].population + ' человек');
```

7. Опционально. Добавьте форматирование выводимых чисел
8. Готовое решение Вы можете найти в

C:\Users\Администратор\Documents\solutions\3_Towns

Упражнение 4. Основы HTML5

О чем это упражнение:

В этой лабораторной работе Вы создадите простые страницы на языке HTML5 и научитесь изменять содержимое страницы с помощью JavaScript

Что Вы должны будете сделать:

- Создать страницу HTML со списком контактов
- Создать форму HTML с автоматической проверкой введенных значений
- Создать страницу HTML с динамически изменяемым содержимым

Раздел 1. Создание страницы контактов

Разработайте страницу контактов компании. По каждому сотруднику в описании должна присутствовать следующая информация:

- Имя
- Фотография
- Должность
- Контактная информация (телефон, почта, номер офиса)
- Ссылка на персональную страницу

Пример вывода информации о сотруднике

Мария Иванова



Руководитель отдела продаж

Телефон: 123-456-789

Офис: 102

Почта: m.ivanova@company.com

1. Создайте новый проект «Приложение HTML5/JS» в NetBeans
2. В блоке **<body>** удалите строку **<div>TODO write content</div>**
3. Добавьте заголовок страницы

`<h1> Контакты компании </h1>`

4. Добавьте заголовок третьего уровня с именем сотрудника

`<h3>Мария Иванова </h3>`

Добавьте изображение на страницу.

- а. Изображения Вы можете найти в
C:\Users\Администратор\Documents\4_HTML5_Contacts\pix
- б. Создайте в проекте папку **pix**. Скопируйте в нее все изображения из указанного выше каталога.
- в. Укажите подпись, которая будет отображаться, если изображение недоступно

```

```

5. Информацию о сотруднике выведите списком

```
<ul>
  <li>Руководитель отдела продаж</li>
  <li>Телефон: 123-456-789</li>
  <li>Офис: 102 </li>
  <li>Почта: m.ivanova@company.com</li>
</ul>
```

6. Нажмите **F6** и проверьте результат в браузере

Мария Иванова



- Руководитель отдела продаж
- Телефон: 123-456-789
- Офис: 102
- Почта: m.ivanova@company.com

7. Выделите должность, сделав ее заголовком четвертого уровня

```
<li><h4>Руководитель отдела продаж</h4></li>
```

8. Выделите слова «Телефон», «Почта» и «Офис» жирным

```
<li><b>Телефон:</b> 123-456-789</li>
<li><b>Офис:</b> 102 </li>
<li><b>Почта:</b> m.ivanova@company.com</li>
```

9. Добавьте к имени сотрудника ссылку, ведущую на персональную страницу

а. Вместо реальной ссылки можно использовать заглушку

```
<h3><a href="#"> Мария Иванова </a></h3>
```

10. Нажмите **F6** и проверьте результат в браузере

Мария Иванова



- **Руководитель отдела продаж**
- **Телефон:** 123-456-789
- **Офис:** 102
- **Почта:** m.ivanova@company.com

11. Опционально. Добавьте информацию о прочих сотрудниках компании. Для разделения информации используйте тег

```
<br clear="all">
```

12. Опционально. Измените стиль отображения

- Добавьте границу изображению
- Выровняйте изображения по левому краю
- Отключите маркировку у списка

```
<style type="text/css">
    ul {
        list-style-type: none
    }
    img {
        margin-right: 20px;
        float:left;
        border: 1px solid;
```

```
}  
</style>
```

13. Нажмите F6 и проверьте результат в браузере

Мария Иванова



Руководитель отдела продаж

Телефон: 123-456-789

Офис: 102

Почта: m.ivanova@company.com

9. Готовое решение Вы можете найти в

C:\Users\Администратор\Documents\solutions\4_HTML5_Contacts

Раздел 2. Автоматическая проверка форм HTML5

Создайте страницу «Регистрационная анкета» с автоматической проверкой вводимых значений.

- Поле «Имя» обязательно. Также необходим автоматический перевод фокуса на это поле.
- Почта и сайт должны проверяться согласно шаблонам
- Поле идентификатора должно содержать одну цифру и три буквы латинского алфавита (например, 1ABC).
- Возраст должен быть числом от 18 до 75
- Страна, цвет и даты должны выбираться из предварительно подготовленных списков

Регистрационная анкета

Имя:

Почта:

Сайт:

Идентификатор:

Страна:

Возраст:

Любимый цвет:

Дата рождения :

Неделя:

Месяц:

Время:

Дата и время:

1. Создайте новый проект
2. Установите название в блоке **<head>**

```
<title>HTML5 Validation</title>
```

3. В блоке **<body>** добавьте заголовок страницы

```
<h1>Регистрационная анкета</h1>
```

4. Создайте форму

- a. Серверная часть разработана не будет, поэтому в поле action укажите пустую строку
- b. Дальнейшие элементы создавайте внутри формы

```
<form action="" method="post">
```

```
</form>
```

5. Поле «Имя» обязательно. Также необходим автоматический перевод фокуса на это поле.

Имя:

```
<input type="text" autofocus id="name" name="name" required>
```

6. Почта и сайт должны проверяться согласно шаблонам

Почта:

```
<input type="email" id="email" name="email" required placeholder="test@mail.ru">
```

Сайт:

```
<input type="url" id="website" name="website" placeholder="http://testurl.com">
```

7. Поле идентификатора должно содержать одну цифру и три буквы латинского алфавита (например, 1ABC). Необходимо вывести пользователю информацию о шаблоне.

```
<abbr title="Цифра и три буквы латинского алфавита">Идентификатор:</abbr>
```

```
<input type="text" id="ID" required name="ID" pattern="[0-9][A-Z]{3}" placeholder="1ABC">
```


8. Страна должна выбираться из выпадающего списка

Страна:

```
<input list="Countries"      placeholder="Country Name">
<datalist id="Countries">
  <option value="Россия">
  <option value="Казахстан">
  <option value="Украина">
  <option value="Белоруссия">
  <option value="Латвия">
</datalist>
```

9. Возраст – число от 18 до 75

```
<abbr title="Число от 18 до 75">Возраст:</abbr>
<input type="number" id="Age" name="Age" required
min="18" max="75">
```

10. Цвет и даты должны выбираться из списков.

```
Любимый цвет: <input type="color" id="Color" name="Color">
Дата рождения: <input type="date" id="Date" name="Date">
Неделя: <input type="week" id="Week" name="Week">
Месяц: <input type="month" id="Month" name="Month">
Время: <input type="time" id="Time" name="Time">
Дата и время: <input type="datetime-local" id="DateTime"
name="DateTime">
```

10. Готовое решение Вы можете найти в

C:\Users\Администратор\Documents\solutions\4_HTML5_Validation

Раздел 3. Объектная модель документа

Сделайте страницу, которая позволяет ввести информацию о городах.

- Используйте ранее разработанный скрипт для обработки информации о городах.
 - По каждому городу должна храниться информация: название, страна, количество населения.
 - Информация о занесенных городах должна отображаться пользователю в виде таблицы.
1. Создайте новый проект, укажите название и заголовок страницы.
 2. Создайте элементы для ввода информации о городе

Название:

Страна:

Население:

3. Добавьте кнопку, нажатие которой должно приводить к запуску функции
 - a. Задача функции: сохранить введенную информацию в массив **arrayTown**
 - b. Введенную информацию можно получить через свойство **value** элементов **<input>**

```
<input type="button" value="Сохранить город"
onclick=clickButton()>
```

```
<script>
```

```
var arrayTown = [];
```

```
function clickButton ()
```

```
{
```

```
    var town  = {};
```

```
    town.townName = document.getElementById('townName').value;
```

```
town.state = document.getElementById('state').value;

town.population =
document.getElementById('population').value;

town.population = parseInt(town.population);

arrayTown.push(town);
}
</script>
```

4. Добавьте таблицу (пока пустую)

```
<table id='townTable'></table>
```

5. Напишите функцию, выводящую данные из массива arrayTown в таблицу. При каждом вызове функции необходимо:

- a. Очищать таблицу
- b. Записывать в таблицу заголовков и данные отсортированного массива arrayTown

```
function showTowns ()
{
    arrayTown.sort(compareTowns);
    arrayTown.reverse();

    document.getElementById('townTable').innerHTML = '<tr>' +
        '<th>Название</th>' +
        '<th>Страна</th>' +
        '<th>Население</th>' +
        '</tr>';

    for (i = 0; i < arrayTown.length; i++ )
    {
        var str="";
        str += '<tr>';
        str += '<td>' + arrayTown[i].townName + '</td>';
        str += '<td>' + arrayTown[i].state + '</td>';
        str += '<td>' + arrayTown[i].population + '</td>';
    }
}
```

```
        str += '</tr>';  
        document.getElementById('townTable').innerHTML += str;  
    }  
}
```

6. Добавьте вызов функции **showTowns()** из функции **clickButton()**

7. Опционально. Добавьте описание стилей элементов в блок **<head>**

```
<style>  
    h1 {  
        font-size: 1.5em  
    }  
    input, button {  
        min-width: 72px;  
        min-height: 36px;  
        border: 1px solid grey;  
    }  
    label, input, button {  
        display: block;  
    }  
    input {  
        margin-bottom: 1em;  
    }  
    table, th, td {  
        border: 1px solid black;  
        text-align: center;  
        width: 500px;  
        height: 36px;  
    }  
</style>
```

8. Готовое решение Вы можете найти в

C:\Users\Администратор\Documents\solutions\4_HTML5_Towns

Упражнение 5. Описание стилей документов с использованием CSS3

О чем это упражнение:

В этой лабораторной работе Вы изучите CSS стили, как на примере готовых демонстрационных проектов, так и самостоятельно создавая новые проекты.

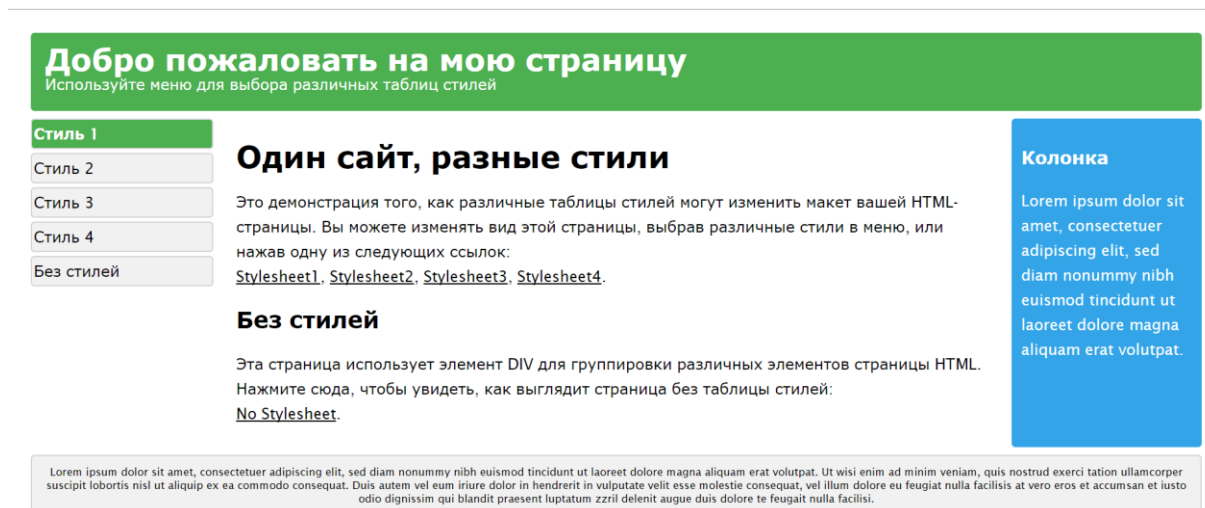
Что Вы должны будете сделать:

- Изучить возможности CSS с помощью демонстрационного проекта
- Создать HTML документ и подключить к нему внешнюю таблицу стилей
- Создать проект, демонстрирующий работу CSS отступов
- Создать проект, демонстрирующий работу CSS полей
- Создать проект, демонстрирующий работу псевдоклассов на примере ссылок
- Создать проект, демонстрирующий работу медиазапросов. Изучить способы применения медиазапросов
- Создать проект, демонстрирующий работу CSS переходов. Изучить различные аспекты CSS переходов
- Создать проект, демонстрирующий работу CSS анимации. Изучить различные аспекты CSS анимации
- Создать проект, демонстрирующий различные способы изменения CSS стилей с помощью JavaScript

Раздел 1. Демонстрация CSS

Откройте готовый проект, чтобы ознакомиться с возможностями CSS.

1. Откройте в браузере готовый проект CSS_Demo.html, находящийся в директории **C:\Users\Администратор\Documents\5_CSS**



2. Попробуйте переключаться между различными таблицами стилей с помощью кнопок на странице.
3. Откройте исходный код документа и найдите там секции описания CSS. Изучите их.

Раздел 2. Отступы

Создайте проект, демонстрирующий использование свойства **margin**.

1. Создайте новый проект **CSS_Margin_Bottom**
2. В index.html этого проекта, в секции **<body>** опишите блочный элемент без надписи класса **wrapper**, а также два дополнительных элемента внутри него: с классами **section1** и **section2**:

```
<div class="wrapper">  
    <div class="section1">Секция 1</div>  
    <div class="section2">Секция 2</div>  
</div>
```

3. Создайте таблицу стилей **style1.css** в директории с файлом проекта.
4. Привяжите к документу внешнюю таблицу стилей с названием **style1.css**, находящуюся в той же директории, что и файл проекта:

```
<link rel="stylesheet" type="text/css"  
href="./style1.css">
```

5. Внутри таблицы стилей задайте следующие параметры класса **section1**: высота 30 пикс., нижний отступ элемента 10 пикс.

```
.section1 {  
    height: 30px;  
    margin-bottom: 10px;  
}
```

6. Для класса **section2** задайте высоту в 30 пикс.:

```
.section2 {  
    height: 30px;  
}
```

7. Для класса **wrapper** задайте следующие параметры: сплошные черные границы толщиной в 1 пикс., ширина 200 пикс. И серый цвет заливки.

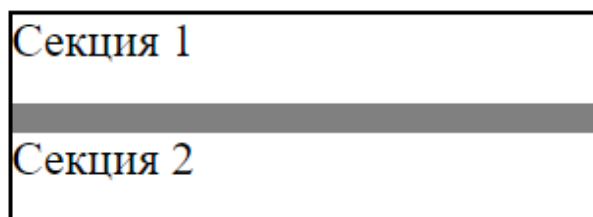
```
.wrapper {
```

```
border: 1px solid black;  
width: 200px;  
background-color: gray;  
}
```

8. Для тега **div** задайте ширину 200 пикс. и белый цвет заливки:

```
div {  
    width: 200px;  
    background-color: white;  
}
```

9. Откройте свой проект в браузере. Он должен выглядеть следующим образом:



10. Объясните, откуда появилась серая полоса в середине элемента.

11. Готовое решение Вы можете найти в каталоге
C:\Users\Администратор\Documents\solutions\5_CSS
CSS_Margin_Bottom.html

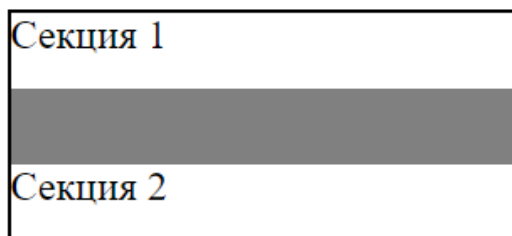
1. Измените параметр **margin-bottom** класса **section1** на 30 пикс.

2. Добавьте параметр **margin-top** равный 10 пикс. классу **section2**:

```
margin-top: 10px;
```

3. Протестируйте код. Обратите внимание, что серая полоса в середине имеет высоту 30 пикс, а не 40.

- а. Если соседние элементы имеют пересекающиеся отступы, то итоговое поле между элементами не будет равняться сумме двух отступов.



Раздел 4. Поля элемента

Согласно спецификации CSS, ширина блока складывается из ширины контента (**width**), значений отступов (**margin**), полей (**padding**) и границ (**border**). Аналогично обстоит и с высотой блока. Свойство **box-sizing** позволяет изменить этот алгоритм, чтобы свойства **width** и **height** задавали размеры не контента, а размеры блока. Значение **border-box** параметра **box-sizing** означает, что свойства **width** и **height** включают в себя значения полей и границ, но не отступов (**margin**). Создайте новый проект, демонстрирующий использование свойства **padding** и **box-sizing**.

1. Создайте новый проект **CSS_Padding** «Приложение HTML5/JS».
2. Задайте в секции **<style>** для класса **ex1** тега **div** ширину равную 300 пикс., светло-синий фон, отступ по всем сторонам элемента в 25 пикс. и параметр **box-sizing**, равный значению **border-box**:

```
div.ex1 {  
    width: 300px;  
    background-color: yellow;  
}
```

3. Задайте в секции **<style>** для класса **ex2** тега **div** ширину равную 300 пикс. и желтый фоновый цвет.

```
div.ex2 {  
    width: 300px;  
    padding: 25px;  
    box-sizing: border-box;  
    background-color: lightblue;  
}
```

4. В секции **<body>** опишите заголовок с текстом «Поля и ширина элемента»

```
<h2> Поля и ширина элемента </h2>
```

5. В секции **<body>** опишите тег **div** класса **ex1**.

```
<div class="ex1"> Ширина этого элемента 300px.</div>
```

6. Опишите тег **div** класса **ex2**.

```
<div class="ex2"> Ширина этого элемента также 300px, несмотря  
на поля по 50px, из-за свойства box-sizing: border-box </div>
```

7. Откройте проект в браузере. Он должен выглядеть следующим образом:

Поля и ширина элемента

Ширина этого элемента 300px

Ширина этого элемента также
300px, несмотря на поля по 50px,
из-за свойства box-sizing: border-box

8. Убедитесь в том, что ширина элементов соответствует значению параметра **box-sizing**.
9. Готовое решение Вы можете найти в
**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_Padding.html**

Раздел 5. Псевдоклассы CSS

Создайте проект, демонстрирующий использование псевдоклассов CSS на примере ссылок.

1. Создайте новый проект **CSS_Link** «Приложение HTML5/JS»
2. В секции **<style>** с помощью псевдокласса **link** опишите цвет непосещенной ссылки тега **a** красным:

```
a:link {  
    color: red;  
}
```

3. В секции **<style>** с помощью псевдокласса **visited** опишите цвет посещенной ссылки тега **a** зеленым:

```
a:visited {  
    color: green;  
}
```

4. Далее с помощью псевдокласса **hover** задайте, чтобы цвет ссылки тега **a**, на которую наведен курсор, менял цвет на ярко-розовый
а. Обратите внимание, что псевдокласс **hover** должен быть описан после псевдоклассов **link** и **visited**

```
a: hover {  
    color: hotpink;  
}
```

5. Далее с помощью псевдокласса **active** задайте, чтобы цвет ссылки тега **a** в момент нажатия становился синим
а. Обратите внимание, что псевдокласс **active** должен быть описан после псевдокласса **hover**.

```
a:active {  
    color: blue;  
}
```

6. Добавьте описание ссылки, ведущей на текущий документ:

```
<a href="CSS_Link.html" target="_blank">Ссылка</a>
```

7. Откройте проект в браузере. Цвет ссылки должен варьироваться в соответствии с логикой работы стилей:

- а. Убедитесь в правильной логике отображения документа:
проверьте работу всех псевдоклассов.

8. Готовое решение Вы можете найти в

**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_Link.html**

Раздел 6. Медиазапросы

Создайте проект, демонстрирующий использование медиазапросов для масштабирования элементов.

1. Создайте новый проект **CSS_Resize** «Приложение HTML5/JS»
2. В секции **<style>** опишите фоновый цвет тега **body** как розовый:

```
body {  
    background-color: pink;  
}
```

3. Создайте медиазапрос к размеру экрана, который бы изменял цвет тега **body** на светло-зеленый, если минимальная ширина экрана становится равной или более 1000 пикс.:

```
@media screen and (min-width: 1000px) {  
    body {  
        background-color: lightgreen;  
    }  
}
```

4. Откройте проект в браузере. Попробуйте изменять размер окна браузера и убедитесь в корректной работе документа.
5. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\5_CSS\CSS_Resize.html
6. Откройте готовый файл
C:\Users\Администратор\Documents\5_CSS\CSS_Menu_Resize.html
 - а. Эта страница демонстрирует работу переноса элементов меню в случае изменения размера экрана.
 - б. Изучите исходный код файла и протестируйте его работу.

Раздел 7. Анимация переходов

Создайте проект, демонстрирующий изменение ширины и высоты элементов с помощью CSS переходов.

1. Создайте новый проект **CSS_Transition** «Приложение HTML5/JS»
2. В секции **<style>** задайте для тега **div** следующие параметры: ширина и высота 100 пикс., красный фоновый цвет, двухсекундный переход ширины и четырехсекундный переход высоты.

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s, height 4s;  
}
```

3. Добавьте описание новых размеров тега **div** (ширина и высота 300 пикс.) для псевдокласса наведения курсора на элемент:

```
div:hover {  
  width: 300px;  
  height: 300px;  
}
```

4. В секции **<body>** вставьте следующие описания и тег **div**:

```
<h1>Свойство transition</h1>
```

```
<p>Наведите на элемент <b>div</b> для активации свойства  
<b>transition:</b></p>
```

```
<div></div>
```

5. Откройте проект в браузере.

Свойство transition

Наведите на элемент **div** для активации свойства **transition**:



6. Протестируйте работу переходов CSS.
7. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\5_CSS\CSS_Transition.html
8. Измените код таким образом, чтобы переход был отложен на 1 секунду

```
div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    transition: width 2s, height 4s;  
    transition-delay: 1s;  
}
```

9. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\5_CSS\CSS_Transition_Delay.html
10. Протестируйте работу различных временных функций: **linear, ease, ease-in, ease-out, ease-in-out**

```
div {transition-timing-function: linear;}
```

11. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents \5_CSS\CSS_Transition_Speed.html

12. Добавьте к имеющемуся переходу поворот блока на 180 градусов

```
div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    transition: width 2s, height 2s, transform 2s;  
}
```



```
div:hover {  
    width: 300px;  
    height: 300px;  
    transform: rotate(180deg);  
}
```

13. Готовое решение Вы можете найти в
**C:\Users\Администратор\Documents\5_CSS\
CSS_Transition_Transformation.html**

Раздел 8. Анимация

Создайте проект, демонстрирующий изменение цвета элемента с помощью CSS анимации.

1. Создайте новый проект CSS_Animation «Приложение HTML5/JS»
2. В секции **<style>** задайте следующие параметры тега **div**, указав анимацию **example**:

```
div {  
    width: 100px;  
    height: 100px;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
}
```

3. Задайте анимацию **example** с изменением фонового цвета от красного к желтому:

```
@keyframes example {  
    from {background-color: red;}  
    to {background-color: yellow;}  
}
```

4. Добавьте тег **div**:

```
<div></div>
```

5. Убедитесь в корректной работе анимации.
 - а. Обратите внимание, что по окончании анимации элемент возвращается к исходному стилю.
6. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\5_CSS\CSS_Animation.html
7. Измените описание анимации, указав промежуточные цвета

```
@keyframes example {  
  0%   {background-color: red;}  
  25%  {background-color: yellow;}  
  50%  {background-color: blue;}  
  100% {background-color: green;}  
}
```

8. Готовое решение Вы можете найти в
**C:\Users\Администратор\Documents\5_CSS\
CSS_Animation_%.html**

9. Добавьте анимацию перемещения объекта

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  position: relative;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

```
@keyframes example {  
  0%   {background-color:red; left:0px; top:0px;}  
  25%  {background-color:yellow; left:200px; top:0px;}  
  50%  {background-color:blue; left:200px; top:200px;}  
  75%  {background-color:green; left:0px; top:200px;}  
  100% {background-color:red; left:0px; top:0px;}  
}
```

10. Готовое решение Вы можете найти в
**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_Animation_Direction.html**
11. Добавьте задержку начала анимации 2 секунды и укажите, что повторять анимацию нужно бесконечно

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;
```

```
position: relative;
animation-name: example;
animation-duration: 4s;
animation-delay: 2s;
animation-iteration-count: infinite;
}
```

12. Готовое решение Вы можете найти в
**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_Animation_Delay_Count.html**
13. Откройте в браузере готовый проект, находящийся в
директории Проверьте работу различных временных функций в
анимации.

```
div {animation-timing-function: linear;}
```

14. Готовое решение Вы можете найти в
**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_Animation_Speed.html**

Раздел 9. Изменение стилей с помощью JavaScript

Создайте проект, демонстрирующий изменение стилей CSS с помощью HTML DOM.

1. Создайте новый проект **CSS_JS_Script** «Приложение HTML5/JS»
2. В секции <body> опишите два одинаковых параграфа с разными идентификаторами:

```
<p id="p1">Hello World!</p>
```

```
<p id="p2">Hello World!</p>
```

3. Далее опишите скрипт, изменяющий стиль параграфа со вторым идентификатором:

```
<script>
```

```
    document.getElementById("p2").style.color = "blue";
```

```
    document.getElementById("p2").style.fontFamily = "Arial";
```

```
    document.getElementById("p2").style.fontSize = "larger";
```

```
</script>
```

4. Готовое решение Вы можете найти в

**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_JS_Script.html**

5. Измените код таким образом, чтобы параграф изменялся после нажатия кнопки

```
<button type="button" onclick=changeStyles() > Нажми!</button>
```

```
<script>
```

```
function changeStyles() {
```

```
    document.getElementById("p2").style.color = "blue";
```

```
    document.getElementById("p2").style.fontFamily = "Arial";
```

```
    document.getElementById("p2").style.fontSize = "larger";
```

```
}
```

```
</script>
```

6. Готовое решение Вы можете найти в

**C:\Users\Администратор\Documents\solutions\5_CSS\
CSS_JS_Event.html**

7. Добавьте две кнопки, при нажатии на которые скрывается и отображается параграф **p1**

```
<input type="button" value="Скрыть" onclick=
"document.getElementById('p1').style.visibility='hidden'">
<input type="button" value="Показать" onclick=
"document.getElementById('p1').style.visibility='visible'">
```

8. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\5_CSS\CSS_JS_Event_Hide.html

Упражнение 6. Графика и анимация HTML5

О чем это упражнение:

В этой лабораторной работе Вы создадите анимированные страницы HTML с использованием элемента HTML Canvas и с использованием технологии Drag and Drop.

Что Вы должны будете сделать:

- Создать страницу с анимированными аналоговыми часами
- Создать страницу с анимацией на основе HTML5 Drag and Drop

Раздел 1. Создание аналоговых часов с помощью HTML Canvas

С помощью элементов рисования HTML Canvas создайте сайт с аналоговыми часами.

- Время на часах должно показывать текущее значение.
- Каждую секунду стрелки должны поворачиваться, показывая новое время
- Оформите код в виде отдельных функций.



Помимо функций рисования, разобранных в теоретической части Вам понадобятся функции:

ctx.translate(X,Y) – переопределение центра координат из позиции (0,0) в позицию (X,Y)

ctx.rotate(ang) – поворот осей на угол **ang** относительно центра координат

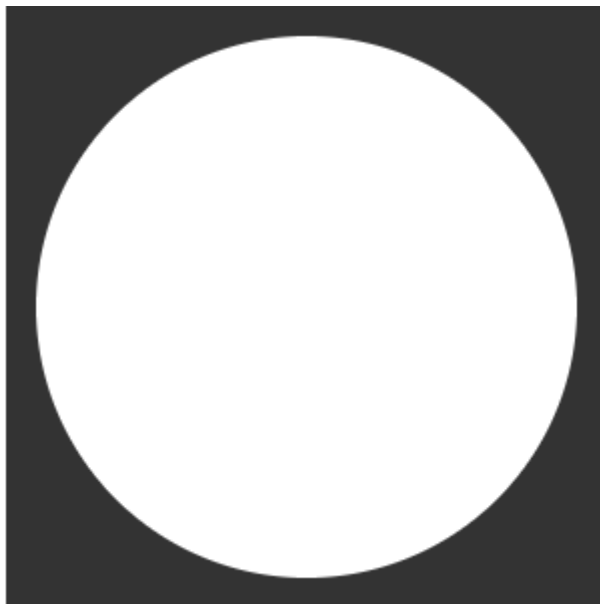
1. Создайте элемент HTML Canvas размером 400x400

```
<canvas id="canvas" width="400" height="400" style="background-color:#333"></canvas>
```

2. Нарисуйте заготовку под циферблат: белый круг в центре области рисования


```
<script>
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
drawFace(ctx, radius);

function drawFace(ctx, radius) {
    ctx.beginPath();
    ctx.arc(0, 0, radius, 0 , 2*Math.PI);
    ctx.fillStyle = "white";
    ctx.fill();
}
</script>
```



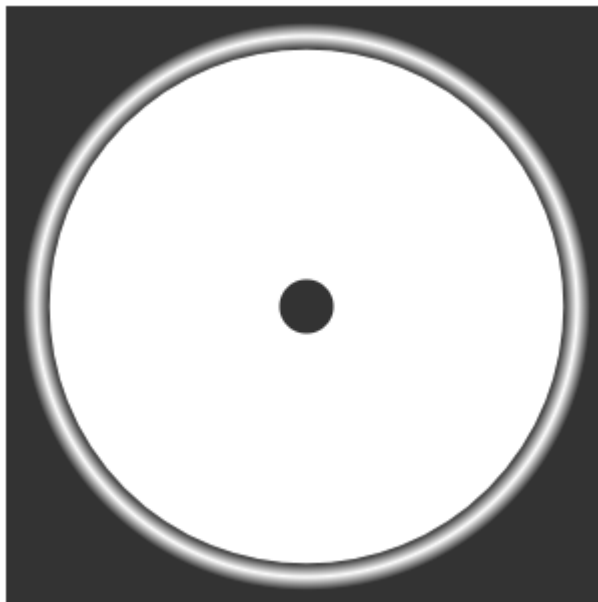
3. Добавьте обод с градиентной заливкой

```
function drawFace(ctx, radius) {
    ...
    var grad;
    grad =
ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
    grad.addColorStop(0, '#333');
    grad.addColorStop(0.5, 'white');
    grad.addColorStop(1, '#333');
```

```
    ctx.strokeStyle = grad;  
    ctx.lineWidth = radius*0.1;  
    ctx.stroke();  
}
```

4. Добавьте сердцевину

```
function drawFace(ctx, radius) {  
    ...  
    ctx.beginPath();  
    ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);  
    ctx.fillStyle = '#333';  
    ctx.fill();  
}
```

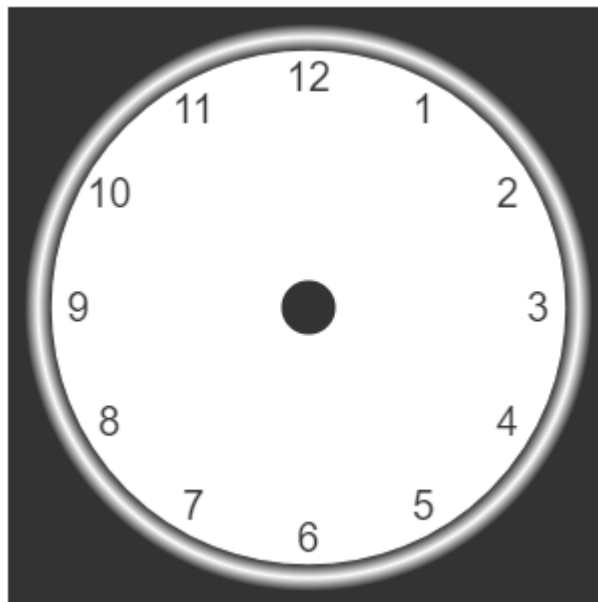


5. Добавьте цифры к циферблату

- Для определения координат цифры нужно совершить поворот относительно центра часов на $\text{num} * 1/12 * 2 * \text{Math.PI}$ и перейти по оси X на $0,85 * \text{radius}$
- Чтобы цифры отображались вертикально, необходимо совершить обратный поворот относительно центра цифры

```
function drawNumbers(ctx, radius) {  
    var ang;  
    var num;  
    ctx.font = radius*0.15 + "px arial";
```

```
ctx.textBaseline="middle";
ctx.textAlign="center";
for(num= 1; num < 13; num++){
    ang = num * Math.PI / 6;
    ctx.rotate(ang);
    ctx.translate(0, -radius*0.85);
    ctx.rotate(-ang);
    ctx.fillText(num.toString(), 0, 0);
    ctx.rotate(ang);
    ctx.translate(0, radius*0.85);
    ctx.rotate(-ang);
}
```



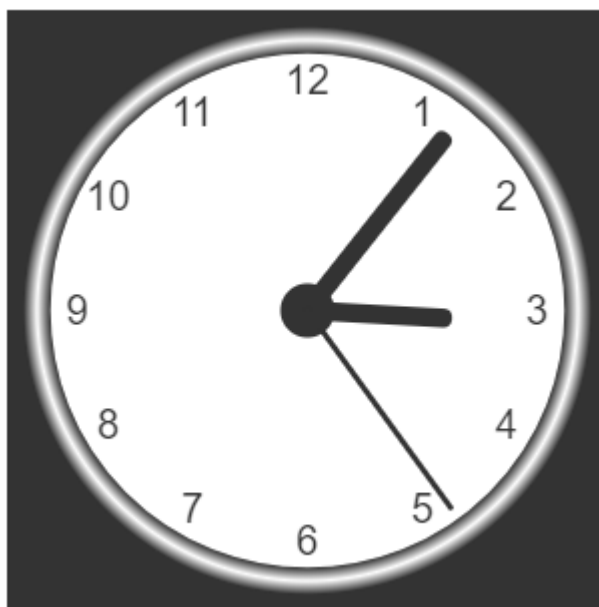
6. Добавьте функцию рисования стрелок
- а. Длину, ширину и угол поворота передайте параметрами

```
function drawHand(ctx, pos, length, width) {
    ctx.beginPath();
    ctx.lineWidth = width;
    ctx.lineCap = "round";
    ctx.moveTo(0,0);
    ctx.rotate(pos);
    ctx.lineTo(0, -length);
    ctx.stroke();
}
```

```
    ctx.rotate(-pos);
}
```

7. Добавьте функцию отображения текущего времени
 - а. Вычислите углы положения часовой, минутной и секундной стрелки и передайте их в функцию **drawHand()**
 - б. Не забудьте о промежуточных положениях минутной и часовой стрелки

```
function drawTime(ctx, radius){
    var now = new Date();
    var hour = now.getHours();
    var minute = now.getMinutes();
    var second = now.getSeconds();
    //hour
    hour=hour%12;
    hour=(hour*Math.PI/6) + (minute*Math.PI/(6*60))
+ (second*Math.PI/(360*60));
    drawHand(ctx, hour, radius*0.5, radius*0.07);
    //minute
    minute=(minute*Math.PI/30)+(second*Math.PI/(30*60));
    drawHand(ctx, minute, radius*0.8, radius*0.07);
    // second
    second=(second*Math.PI/30);
    drawHand(ctx, second, radius*0.9, radius*0.02);
}
```



8. Запустите обновление состояния часов раз в секунду. Для этого сделайте дополнительную функцию **drawClock()** и запустите ее периодическое выполнение

```
function drawClock() {  
    drawFace(ctx, radius);  
    drawNumbers(ctx, radius);  
    drawTime(ctx, radius);  
}  
setInterval(drawClock, 1000);
```

9. Готовый код Вы можете найти в
C:\Users\Администратор\Documents\solutions\6_HTML_Clock

Раздел 2. Захват и перенос элементов HTML Drag and Drop

Измените код страницы таким образом, чтобы добавить анимацию при перемещении элементов в корзину.



1. Создайте проект с готовым исходным кодом. Код Вы можете найти в каталоге

C:\Users\Администратор\Documents\6_DragAndDrop\index.html

2. Изучите файлы проекта. Файл **Index.html** описывает статичную страницу с изображениями документов и корзины

```
<html>
<head>
  <link rel="stylesheet" href="example.css">
</head>
<body>
```

```
<div class="container">
  
  
  
  
</div>
<div class="trashempty"></div>
</body>
</html>
```

3. В стилях CSS описано два стиля **trashempty** – пустая корзина для бумаг и **trashfull** – полная корзина.
4. Добавьте функции-обработчики событий **ondragstart**, **ondragover**, **ondrop**. Добавьте идентификаторы к документам.

```
<div class="trashempty" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>

...
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
}
</script>
```

5. Измените функцию **drop()** таким образом, чтобы она изменяла стиль корзины в состояние «полная корзина»

```
function drop(ev) {
```

```
...
    ev.target.className="trashfull";
}
```

6. Измените функцию **drop()** таким образом, чтобы перенесенный документ удалялся

```
function drop(ev) {
    ...
    var data = ev.dataTransfer.getData("text");
    var child = document.getElementById(data);
    var parent = child.parentNode;
    parent.removeChild(child);
}
```

7. Опционально. Добавьте возможность извлекать документы из корзины, путем захвата корзины и перетаскивания ее в левую область экрана.

```
<div id='cont1' class="container" ondrop="restore(event)"
ondragover="allowDrop(event)">
```

8. Создаваемому документу необходимо установить:
- а. Уникальный идентификатор (например, **Date.getTime()**)
 - б. Изображение и стиль
 - в. Обработку события захвата

```
function restore(ev) {
    ev.preventDefault();
    var img = document.createElement('img');
    img.src = "images/document.png";
    img.id = new Date().getTime();
    img.className='document';
    img.draggable=true;
    img.addEventListener("dragstart", drag);
    document.getElementById('cont1').appendChild(img);
}
```


9. Опционально. Добавьте счетчик удаленных и восстановленных документов. Изменяйте стиль корзины в соответствии с показаниями счетчика
10. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\6_DragAndDrop

Упражнение 7. JavaScript. Функции и объекты

О чем это упражнение:

В этой лабораторной работе Вы познакомитесь с различными способами определения функций, разберете Работу замыканий и научитесь разрабатывать внешние модули

Что Вы должны будете сделать:

- Используя различные способы определения функций создать страницу с возможностью изменения размера текста
- Написать внешний модуль, реализующий код счетчика
- Создать конструктор объекта-счетчика

Раздел 1. Функциональные выражения

JavaScript позволяет создавать функции по время выполнения кода.

Создайте страницу с возможностью изменения размера текста при перемещении ползунка.

Обычный текст

Заголовок 1

Заголовок 2

12  26

1. Создайте элементы HTML: текст, два заголовка и ползунок

```
<p>Обычный текст</p>
<h1>Заголовок 1 </h1>
<h2>Заголовок 2</h2>
12<input type="range" min="12" max="26" step="1" id="zoom" >26
```

2. Добавьте обработку события input для ползунка. Изменяйте размер шрифта элемента **<body>** в соответствии со значением на ползунке.
 - а. Стили по умолчанию предполагают, что размер текста заголовков установлен относительно размера шрифта основного текста.
Заклучите этот скрипт в элемент `<script>` на вашей странице

```
document.getElementById('zoom').oninput = function () {
    document.body.style.fontSize =
        document.getElementById('zoom').value + 'px';
};
```

3. В данном случае было использовано анонимное функциональное выражение.

Раздел 2. Передача функции в качестве параметра.

Ключевое слово *this*

Многие встроенные методы JavaScript принимают функции в качестве параметра.

Добавьте пользователю возможность вводить размер шрифта напрямую. Решите задачу, используя метод **addEventListener** для добавления реакции на событие.

Обычный текст

Заголовок 1

Заголовок 2

12  26

Введите размер:

1. Добавьте элемент HTML

Введите размер:

```
<input type="number" min="12" max="26" id="num">
```

2. Добавьте обработку события ввода параметра в новое поле.

Изменяйте размер шрифта элемента **<body>** в соответствии со значением, указанным в этом поле.

```
document.getElementById('num').addEventListener ('input',  
    function () {  
        document.body.style.fontSize =  
            document.getElementById('num').value + 'px';  
    });
```

3. В данном случае также было использовано анонимное функциональное выражение.
4. Для того чтобы избежать дублирования кода, вынесите определение функции в отдельную строку
 - a. Обратите внимание, что размер шрифта должен браться с активного элемента
 - b. Каждая функция имеет ключевое слово `this`, которое соответствует объекту, от которого запущена данная функция.

```
var sizer = function () {  
    document.body.style.fontSize =  
        this.value + 'px';  
};  
  
document.getElementById('zoom').oninput = sizer;  
document.getElementById('num').addEventListener('change', si  
zer);
```

5. Протестируйте код.
 - a. Обратите внимание, что в объекты передается сама функция, а не результат ее выполнения, т.е. отсутствуют скобки после названия

```
// Неправильная передача  
document.getElementById('zoom').oninput = sizer ()
```

Раздел 3. Замыкания

Функцию можно не только передавать в другую функцию в качестве параметра, но и возвращать в качестве результата работы – это называется замыканием.

Добавьте три ссылки, при нажатии на которые устанавливается фиксированный размер текста.

Обычный текст

Заголовок 1

Заголовок 2

12  26

Введите размер:

12 14 16

1. Добавьте элементы HTML

```
<a href="#" id="size-12">12</a>  
<a href="#" id="size-14">14</a>  
<a href="#" id="size-16">16</a>
```

2. Создайте функцию **makeSizer()**

```
function makeSizer(size) {  
    return function() {  
        document.body.style.fontSize = size + 'px';  
    };  
}
```

3. Добавьте обработку события **click** для ссылок

```
document.getElementById('size-12').onclick = makeSizer(12);  
document.getElementById('size-14').onclick = makeSizer(14);  
document.getElementById('size-16').onclick = makeSizer(16);
```

4. Протестируйте код

- a. Обратите внимание, что в этом случае (так же как и в предыдущем разделе) объектам передается функция
- b. В отличие от предыдущего раздела, функция генерируется динамически в зависимости от переданного размера

Раздел 4. Разработка внешних модулей. Проблема счетчика

Подключение внешних модулей в проект позволяет повторно использовать однажды разработанный код.

Разработайте внешний модуль, реализующий код счетчика.

- Начальная позиция счетчика – 0
 - Доступ к счетчику возможен только через функцию `increment()`, увеличивающую значение на 1.
1. Добавьте новый файл JavaScript в проект. Назовите его **counter.js**
 2. Подключите скрипт в проект, добавив в блок **<head>** строку

```
<script src="counter.js"></script>
```

3. В скрипте **counter.js** реализуйте код счетчика

```
var counter = 0;

function increment (){
    counter++;
    return counter;
}
```

4. В файле **index.html** добавьте код, использующий счетчик

```
<script>
    increment(); increment();
    alert (increment ());      // 3
</script>
```

5. Проверьте работу кода. Код должен выдавать значение счетчика 3.
6. Код использует глобальную переменную **counter**. Это может привести к конфликтам переменных модулей и основного кода. Отладка подобных проблем может занимать существенное время.
 - а. Добавьте в основной код переменную с именем **counter**

```
<script>
    increment(); increment();
    var counter = 10;
    alert (increment ());      // 11
```



```
        alert (counter);                // 11
    </script>
```

7. Проверьте работу кода. Код должен выдавать значение счетчика 11, т.к. основной код переопределил значение переменной-счетчика.
 - а. Обратите внимание, что из основного кода совершенно неочевидно, что переменная **counter** изменена.
8. Измените код внешнего модуля, так чтобы код основного модуля не могу менять значение счетчика каким-либо образом, кроме обращения к функции **increment()**.
 - а. Используйте замыкания и немедленно вызываемые функции

```
var increment = (function () {
    var counter = 0;
    return function () {
        counter++;
        return counter;
    };
})();
```

9. Проверьте работу кода. Теперь первый вывод показывает значение счетчика – 3, а значение переменной **counter** в основном коде – 10.
 - а. Таким образом, использование замыканий позволяет защитить внутренние переменные внешних модулей от изменения из основного потока кода.

```
<script>
    increment(); increment();
    var counter = 10;
    alert (increment ());        // 3
    alert (counter);             // 10
</script>
```

Раздел 5. Конструктор объекта

Расширьте функционал счетчика.

- Добавьте функцию **decrement ()**, уменьшающую значение счетчика на единицу
- Добавьте возможность чтения (но не записи) текущего значения счетчика из основного потока кода.
- Значение счетчика, как и ранее, должно быть защищено от прямого изменения из основного потока кода.

Добавьте пользователю возможность создавать сколько угодно независимых счетчиков.

1. Измените код внешнего модуля таким образом, чтобы функция возвращала не одну функцию, а несколько функций в составе объекта.

```
var Counter = (function () {  
    var counter = 0;  
    return {  
        increment: function () {counter++;  
            return counter;},  
        decrement: function () {counter--;  
            return counter;},  
        getValue: function () {  
            return counter;}  
    };  
})();
```

2. В основном модуле используйте следующий код

```
<script>  
    Counter.increment(); Counter.increment();  
    var counter = 10;  
    alert (Counter.getValue ());           // 2  
    alert (counter);                       // 10  
</script>
```

3. Сейчас внешний модуль присваивает счетчик в переменную **Counter**. Оставьте во внешнем модуле только описание функции.
 - а. До сих пор Вы использовали анонимные функциональные выражения. Сейчас Вы напишете объявление функции.

- b. Добавьте имя функции, чтобы иметь возможность вызывать ее.

```
function Counter () {  
    var counter = 0;  
    return {  
        increment: function () {  
            counter++;  
            return counter;  
        },  
        decrement: function () {  
            counter--;  
            return counter;  
        },  
        getValue: function () {  
            return counter;  
        }  
    };  
};
```

4. В основном коде создайте несколько объектов-счетчиков, используя функцию **Counter** в качестве конструктора.

```
<script>  
    var c1 = new Counter ();  
    var c2 = new Counter ()  
    c1.increment(); c1.increment();  
    c2.decrement ();  
    alert (c1.getValue ());           // 2  
    alert (c2.getValue ());           // -1  
</script>
```

5. Ключевое слово **this** указывает на объект, связанный с функцией. Все свойства и методы функции, доступные из внешнего кода принадлежат объекту **this**.

- а. Упростите код внешнего модуля, добавив к объекту **this** функции и свойства, которые должны быть доступны из основного кода.

```
function Counter () {  
    var counter = 0;  
  
    this.increment = function () {  
        counter++;  
    };  
};
```

```
        return counter;
    };
    this.decrement = function () {
        counter--;
        return counter;
    };
    this.getValue = function () { return counter;};
};
```

Упражнение 8. Передача и хранение данных веб-приложения

О чем это упражнение:

В этой лабораторной работе Вы изучите различные способы сохранения информации веб-приложения.

Что Вы должны будете сделать:

- Разработать сайт с поддержкой локализации
- Разработать скрипт, распознающий пользователя сайта
- Разработать сайт с корзиной покупок

Раздел 1. Локализация сайта

Разработайте сайт с формой, текст которой меняется в зависимости от выбранного пользователем языка.

HTML5 Internationalization Example



The screenshot shows a web form titled "Application Form" with a language selection bar at the top right containing flags for Russian, French, German, and UK English. The form fields are: "Name" (text input), "Company" (text input), "Telephone No." (text input), and "Gender" (dropdown menu with "Male" selected).

Пример локализации HTML5



The screenshot shows the same web form localized into Russian. The title is "Форма" and the language bar contains the same flags. The form fields are: "Имя" (text input), "Компания" (text input), "Телефон" (text input), and "Пол" (dropdown menu with "Мужской" selected).

Для хранения дополнительных атрибутов, например, язык элемента HTML5 в данном примере используются дополнительные атрибуты.

Для присвоения дополнительных атрибутов тексту используется тег ****.

1. Создайте проект «Приложение HTML5\JS» на основе исходных кодов

C:\Users\Администратор\Documents\8_JSON_Internationalization

2. Изучите файлы проекта. Файл **index.html** – описывает веб-страницу с формой и изображениями флагов. Обратите внимание, что все флаги входят в класс **language**, а все локализуемые элементы имеют атрибут **data-langkey**

```

```

```
<span data-langkey="Name">Name</span><input type="text">
```

3. Из файла **index.html** подключены два скрипта. Скрипт **connection.js** отвечает за отправку запроса и получение данных, этот скрипт использует технологию AJAX, с которой Вы познакомитесь в следующей главе.
4. Скрипт **html5-internationalization.js** должен содержать следующие функции:
 - a. Функция **switchLanguage()**, вызываемая по нажатию на изображение флага. Функция вызывает функцию **request(file)**, где **file** – имя запрашиваемого файла с локализацией.
 - b. Функция **processLangDocument()**, обрабатывающая полученный документ с данными локализации. Эта функция должна изменять текст элементов HTML с атрибутом **data-lang**.
5. Каталог **json** содержит документы с данными локализации. Ниже содержимое файла **russia.json**

```
{  
  "Name": "Имя",  
  "Company": "Компания",  
  "Email Address": "Почтовый адрес",  
  "Application Form": "Форма",  
  "Telephone No.": "Телефон",  
  "Gender": "Пол",  
  "Male": "Мужской",  
  "Female": "Женский",  
  "Prefer not to say": "Предпочитаю не указывать",  
  "HTML5 Internationalization Example": "Пример  
локализации HTML5"  
}
```

6. Разработайте функцию **switchLanguage()**

```
function switchLanguage(language){  
    request ("i18n/" + language + ".json");  
}
```

7. Добейтесь того, чтобы при нажатии на изображения флагов вызывалась функция **switchLanguage()**. Передавайте при нажатии в **switchLanguage()** значение атрибута **data-lang**.

```
var languages =  
document.getElementsByClassName('language');  
  
for (index=0; index < languages.length; index++)  
{  
    languages[index].addEventListener('click', function(){  
  
        switchLanguage(this.dataset.lang);  
    });  
}
```

8. Разработайте функцию **processLangDocument()**. Эта функция получает строку **responseText** – строка в формате JSON. Необходимо преобразовать эту строку в объект JavaScript.

```
function processLangDocument(responseText){  
    var langDocument = {};  
    langDocument = JSON.parse(responseText);  
    ...  
}
```

9. Далее внутри функции **processLangDocument()** необходимо подготовить перечень всех локализуемых элементов

```
var tags = document.querySelectorAll('span,img,a,label,  
li,option, h1,h2,h3,h4,h5,h6');
```

10. У каждого локализуемого элемента есть атрибут **data-langkey**, указывающий, какие данные находятся в этом элементе. Значения атрибута **data-langkey** совпадают с именами свойств объекта

JSON. Замените содержимое элементов HTML на значения свойств объекта JSON.

Отдельно обработайте ситуацию отсутствия подходящего значения (в этом случае должен остаться английский текст).

```
for (index=0; index < tags.length; index++)
{
    var key = tags[index].dataset.langkey;
    if(langDocument[key])
        tags[index].innerText = langDocument[key];
}
```

11. Опционально. Добавьте новое поле с текстом «Email Address» и атрибутом **data-langkey=' Email Address'**. Проверьте, что локализация для этого поля работает.
12. Готовое решение Вы можете найти в **C:\Users\Администратор\Documents\solutions\8_JSON_Internationalization**
13. Опционально. Добавьте локализацию данной страницы на еще один язык.
 - a. Создайте файл **mylang.json** с данными локализации
 - b. Создайте еще одно изображение флага с атрибутом **data-lang='mylang'**

Раздел 2. Работа с Cookie

Напишите сайт, который запоминает имя пользователя и выдает персонализированное приветствие.

- При открытии страницы проверить сохранено ли в данных куки имя пользователя, если нет – запросить имя
- Если имя пользователя известно, выдать «Добро пожаловать снова, <имя пользователя>»

1. Напишите функцию для сохранения куки. Не забудьте обновить время устаревания куки

```
function setCookie(cname,cvalue,exdays) {  
    var d = new Date();  
    d.setTime(d.getTime() + (exdays*24*60*60*1000));  
    var expires = "expires=" + d.toGMTString();  
    document.cookie = cname + "=" + cvalue + ";" + expires  
+ ";path=/";  
}
```

2. Напишите функцию извлечения информации из куки. Помните, что данные в куки хранятся в кодированном виде. Формат куки «имя=значение; имя=значение»

```
function getCookie(cname) {  
    var name = cname + "=";  
    var decodedCookie =  
decodeURIComponent(document.cookie);  
    var ca = decodedCookie.split(';');  
    for(var i = 0; i < ca.length; i++) {  
        var c = ca[i].trim();  
        if (c.indexOf(name) == 0) {  
            alert (c);  
            return c.substring(name.length, c.length);  
        }  
    }  
    return "";  
}
```

3. Разработайте функцию **checkCookie()**, проверяющую сохранено ли имя пользователя в куки

```
function checkCookie() {  
    var user=getCookie("username");
```

```
    if (user != "") {  
        alert("Добро пожаловать снова, " + user);  
    } else {  
        user = prompt("Пожалуйста, представьтесь: ", "");  
        if (user != "" && user != null) {  
            setCookie("username", user, 30);  
        }  
    }  
}
```

4. Установите, что функция **checkCookie()** должна запускаться автоматически при загрузке документа

```
<body onload="checkCookie()">
```

5. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\8_Cookie

Раздел 3. Работа с локальным хранилищем HTML5 Local Storage

Разработайте сайт с корзиной для покупок.

- Имена товаров и количество запросите у пользователя
- Выводите данные в виде таблицы
- Сохраняйте данные в HTML5 Local Storage

Введите товары и их количество

Товар:

Количество:

Список покупок

Имя	Значение
Морковь	10
Огурцы	5
Помидоры	3

* Удалить все товары

1. Создайте проект «Приложение HTML5\JS» на основе исходных кодов **C:\Users\Администратор\Documents\8_HTMLStorage**
2. В файле **index.html** описаны элементы страницы.
 - a. При загрузке страницы запускается функция **doShowAll ()**, отображающая таблицу с товарами.
 - b. При нажатии на кнопку “Сохранить” запускается функция **SaveItem()**, сохраняющая товар в HTML5 LocalStorage
 - c. При нажатии на кнопку “Удалить” запускается функция **RemoveItem()**, удаляющая товар из HTML5 LocalStorage
 - d. При нажатии на кнопку “Очистить” запускается функция **ClearAll()**, очищающая содержимое HTML5 LocalStorage
3. Сами функции находятся в файле **Storage.js**. Ваша задача – разработать их.
4. Напишите функцию **SaveItem()**

```
function SaveItem() {  
  
    var name = document.forms.ShoppingList.name.value;  
    var data = document.forms.ShoppingList.data.value;  
    localStorage.setItem(name, data);  
    doShowAll();  
}
```

5. Напишите функцию **RemoveItem()**

```
function RemoveItem() {  
    var name = document.forms.ShoppingList.name.value;  
    document.forms.ShoppingList.data.value =  
localStorage.removeItem(name);  
    doShowAll();  
}
```

6. Напишите функцию **ClearAll()**

```
function ClearAll() {  
    localStorage.clear();  
    doShowAll();  
}
```

7. Функция **doShowAll ()** должна проверять браузер на совместимость и выводить содержимое HTML5 Local Storage в формате таблицы. Функция частично готова. Напишите код для формирования таблицы.

```
for (i = 0; i <= localStorage.length - 1; i++) {  
    key = localStorage.key(i);  
    list += "<tr><td>" + key + "</td>\n<td>"  
        + localStorage.getItem(key)  
        + "</td></tr>\n";  
}  
document.getElementById('list').innerHTML = list;
```

8. Проверьте работу локального хранилища, открыв ту же страницу в другой вкладке. Вы должны увидеть, что введенные товары сохранились.
9. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\8_HTMLStorage

Упражнение 9. Передача данных на сервер. Технологии AJAX и WebSocket

О чем это упражнение:

В этой лабораторной работе Вы научитесь отправлять запрос на сервер и обрабатывать полученные данные, а также разработаете многопользовательский чат.

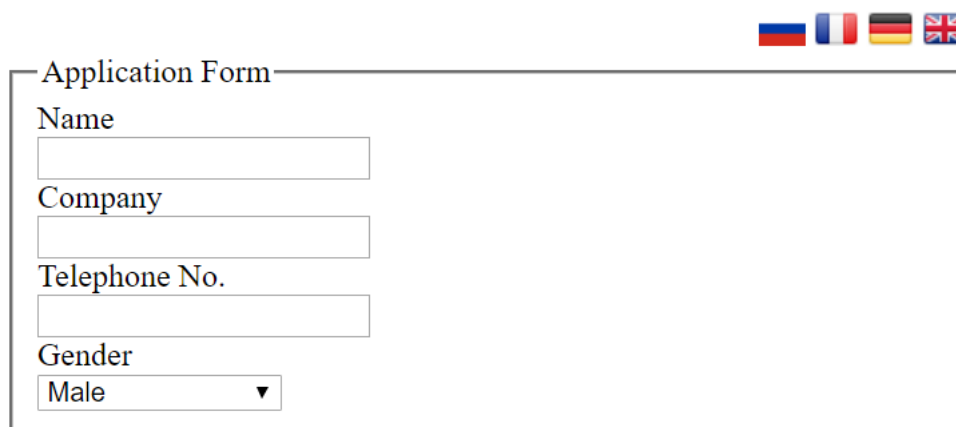
Что Вы должны будете сделать:

- Разработать модуль передачи данных для сайта с поддержкой локализации
- Разработать многопользовательский чат на основе технологии WebSocket

Раздел 1. Локализация сайта. Запрос данных с помощью AJAX

Повторно рассмотрим проект страницы с формой, текст которой меняется в зависимости от выбранного языка. В этом разделе разработайте модуль передачи данных на сервер

HTML5 Internationalization Example



The screenshot shows a web form titled "Application Form" in English. At the top right, there are four flags: Russian, French, German, and British. The form contains the following fields: "Name" (text input), "Company" (text input), "Telephone No." (text input), and "Gender" (a dropdown menu with "Male" selected).

Пример локализации HTML5



The screenshot shows the same web form as above, but localized into Russian. The title is "Форма". The flags remain the same. The form fields are: "Имя" (text input), "Компания" (text input), "Телефон" (text input), and "Пол" (a dropdown menu with "Мужской" selected).

1. Создайте проект «Приложение HTML5\JS» на основе исходных кодов

C:\Users\Администратор\Documents\9_AJAX_Internationalization

2. Изучите файлы проекта. Файл **index.html** – описывает веб-страницу с формой и изображениями флагов.

3. Из файла **index.html** подключены два скрипта. Скрипт **connection.js** отвечает за отправку запроса и получение данных. В данном разделе Вы должны разработать этот скрипт.
4. Скрипт **html5-internationalization.js** содержит следующие функции:
 - a. Функция **switchLanguage()**, вызываемая по нажатию на изображение флага. Функция вызывает функцию **request (file)** из скрипта **connection.js**, где **file** – имя запрашиваемого файла с локализацией.
 - b. Функция **processLangDocument()**, обрабатывающая полученный документ с данными локализации. Эта функция должна вызываться при получении ответа от сервера.
5. Каталог **json** содержит документы с данными локализации.
6. Разработайте скрипт **connection.js**
7. Создайте объект соединения AJAX

```
var xhttp = new XMLHttpRequest();
```

8. Установите обработку на успешное получение сообщения от сервера.
 - a. Помните, что **xhttp.readyState** хранит текущее состояние полученного сообщения (полностью полученное сообщение: состояние 4).
 - b. Код состояния HTTP хранится в **xhttp.status** (успешно полученное сообщение – 200)
 - c. При успешном получении сообщения вызовите функцию **processLang()** и передайте ей полученное сообщение **xhttp.responseText** на обработку

```
xhttp.onreadystatechange = function(){  
    if (this.readyState === 4 && this.status === 200) {  
        processLangDocument(this.responseText);  
    }  
};
```

9. Разработайте функцию **request (file)**, запрашивающую содержимое ресурса, URI которого передается в качестве параметра.
 - a. Используйте метод HTTP GET
 - b. Формат запрашиваемых данных **"application/json"**

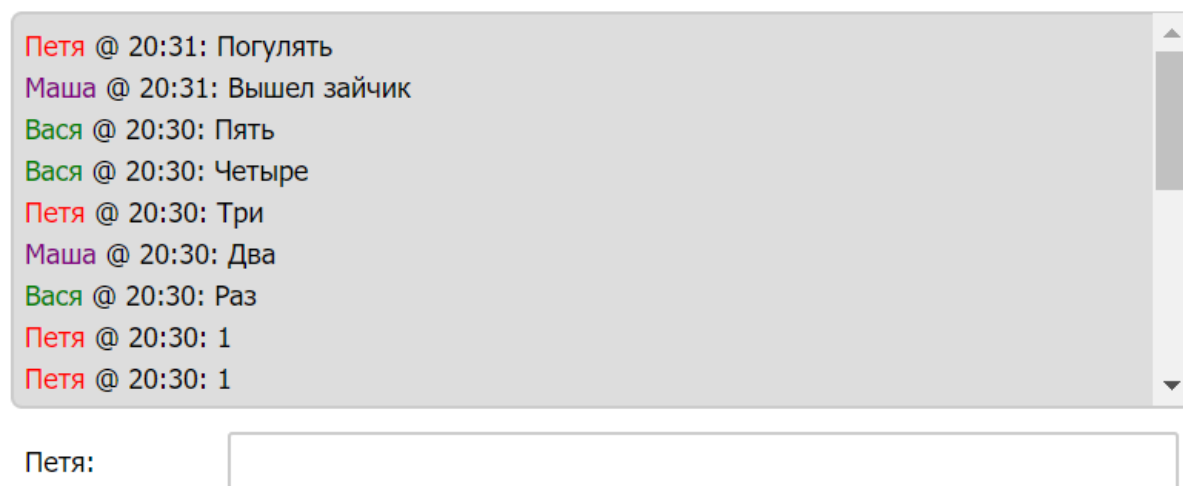
```
function request (file)
{
    xhttp.open("GET", file, true);
    xhttp.setRequestHeader("Content-type",
        "application/json");
    xhttp.send();
};
```

14. Готовое решение Вы можете найти в
C:\Users\Администратор\Documents\solutions\9_AJAX_Internationalization

Раздел 2. Многопользовательский чат на WebSocket

Напишите чат, используя технологию WebSocket для подключения к серверу.

- Клиент подключается к серверу WebSocket и отправляет на сервер имя пользователя
- Сервер выделяет пользователю цвет и отправляет его пользователю
- Сервер отправляет пользователю 100 последних сообщений (с указанием времени, автора и цвета автора).



1. Создайте проект «Приложение Node.js» на основе исходных кодов **C:\Users\Администратор\Documents\9_WebSocket_Chat**
2. Файл **index.html** содержит описание элементов веб-страницы.
Активные элементы:
 - а. Поле **content**, в которое выводятся сообщения, полученные от сервера
 - б. Текстовое поле **input**, в котором пользователь набирает сообщение
 - в. Поле **status**, отображающее статус либо имя пользователя
3. Скрипт **chat-frontend.js** содержит логику работы клиента. Вам необходимо доработать этот скрипт
 - а. Открывает подключение к серверу
 - б. Обрабатывает получаемые сообщения
 - в. Выводит сообщения в поле **content** с помощью функции **addMessage()**
 - г. Отправляет сообщения на сервер

4. Скрипт **chat-server.js** содержит логику работы сервера

5. Откройте подключение к серверу **ws://127.0.0.1:1337**

```
var connection = new WebSocket('ws://127.0.0.1:1337');
```

6. Если соединение установлено успешно, запросите имя пользователя

```
connection.onopen = function () {  
    input.removeAttribute ('disabled');  
    status.innerHTML = 'Укажите имя:';  
};
```

7. Если соединение не удалось установить, выдайте ошибку

```
connection.onerror = function (error) {  
    content.innerHTML = '<p>Проблемы с соединением </p>';  
};
```

8. Обработайте сообщения, приходящие от сервера. Сервер отправляет сообщения в формате JSON.

```
connection.onmessage = function (message) {  
    var json = JSON.parse(message.data);  
    ...  
}
```

9. Сервер отправляет сообщения трех видов (для подробной информации посмотрите код сервера).

а. Сервер отправляет цвет подключившемуся пользователю

Формат сообщения: **{'type':'color', 'data':color}**

При получении этого сообщения в поле статус нужно указать имя пользователя и цвет, а также разблокировать поле **input**

```
if (json.type === 'color') {  
    myColor = json.data;  
    status.innerHTML = myName + ': '  
    status.setAttribute('style','color:' + myColor);  
    input.removeAttribute ('disabled');  
}
```

- b. Сервер рассылает всем пользователям новое сообщение

Формат сообщения:

```
{'type': 'message',  
  'data': {  
    time: time,  
    text: htmlEntities,  
    author: userName,  
    color: userColor  
  }  
}
```

При получении сообщения необходимо вывести его в поле **content**.

```
else if (json.type === 'message') {  
    input.removeAttribute ('disabled');  
    addMessage(json.data.author, json.data.text,  
               json.data.color, new Date(json.data.time));  
}
```

- c. Сервер отправляет новому пользователю всю историю сообщений.

Формат сообщения: **{'type':history, 'data': array_of_messages}**
}

При получении сообщения необходимо вывести его в поле **content**.

```
else if (json.type === 'history') {  
    for (var i=0; i < json.data.length; i++) {  
        addMessage(json.data[i].author,  
                    json.data[i].text,  
                    json.data[i].color,  
                    new Date(json.data[i].time));  
    }  
}
```

d. Если получено иное сообщение – выдать ошибку

```
else {  
    console.log('Неверный формат сообщения: ', json);  
}
```

10. Отправка сообщений производится при нажатии клавиши **Enter**. Добавьте в функцию обработки события задачу отправки сообщения.

```
connection.send(msg);
```

11. Запустите сервер, нажав **F6**. Если возникает сообщение об ошибке, скорее всего это связано с тем, что в проекте для запуска выбран не тот файл. Скорректировать это можно в свойствах проекта: **Запустить → Файл запуска -> указать chat-server.js**
12. Откройте несколько вкладок со страницей index.html и протестируйте ваш чат.
13. Готовое решение Вы можете найти в

C:\Users\Администратор\Documents\solutions\9_WebSocket_Chat

Упражнение 10. Библиотека jQuery

О чем это упражнение:

В этой лабораторной работе Вы напишете сайт с игрой в крестики-нолики с применением библиотеки JQuery.

Что Вы должны будете сделать:

- Разработать логику игры в крестики-нолики

Раздел 1. Крестики-нолики

Разработайте игру в крестики-нолики на поле 3x3 для двух пользователей.

- Для упрощения задачи предполагается, что оба пользователя подключаются от одного клиента, т.е. серверную часть и передачу данных на сервер реализовывать не нужно.
- Если ни один игрок не собрал выигрышную комбинацию и все клетки поля заняты – пользователи должны получить сообщение о ничьей.
- После окончания раунда игру следует начать заново
- Для определения победителя предлагается использовать следующую схему:
 - Сопоставим каждой клетке поля некий индикатор, степень двойки

1	2	4
8	16	32
64	128	256

- Учет клеток, отмеченных игроком, будем вести как сумму индикаторов клетки. Так, если игрок поставил “X” во все клетки первой строки, его балл будет равен 7.
- Известно, что выигрышные комбинации: 7, 56, 448, 73, 146, 292, 273, 84
- При игре, описанной иллюстрацией ниже по тексту, балл X – 23, балл O – 328

X	X	X
O	X	
O		O

- Если представить балл игрока как двоичное число, а выигрышную комбинацию как двоичную маску, то при применении маски к баллу игрока можно будет проверить, совпадает ли текущий расклад с одной из выигрышных комбинаций.
- Таким образом, условие **(wins[i] & score) === wins[i]** позволяет определить, победил ли текущий игрок.

1. Создайте проект «Приложение HTML5\JS» на основе исходных кодов **C:\Users\Администратор\Documents\10_jQuery_TicTacToe**
 - а. Файл **index.html** описывает пустую таблицу из девяти клеток
 - б. Файл **styles.css** описывает стили отображения таблицы
 - в. Файл **index.js** должен описывать логику игры. В скрипте описаны глобальные переменные
 - i. **wins** – массив выигрышных комбинаций
 - ii. **score** – текущий балл игроков
 - iii. **move** – номер текущего хода
2. Добавьте дополнительный атрибут-индикатор к каждой ячейке таблицы
 - а. Используйте дополнительные индикаторы **data-**

```
<table id='board'>
  <tr>
    <td data-indicator=1></td>
    <td data-indicator=2></td>
    <td data-indicator=4></td>
  </tr>
  <tr>
    <td data-indicator=8></td>
```

```
        <td data-indicator=16></td>
        <td data-indicator=32></td>
    </tr>
    <tr>
        <td data-indicator=64></td>
        <td data-indicator=128></td>
        <td data-indicator=256></td>
    </tr>
</table>
```

3. Установите событие по нажатию кнопки мыши на элементах **td** и **tr** таблицы с идентификатором **board**

```
$("#board tr td").click(function() {
    ...
});
```

4. При нажатии необходимо определить, пуста ли клетка
- Если ячейка пуста, то установить в нее символ текущего игрока (нечетные ходы – крестик, четные – нолик)
 - Символ устанавливается добавлением текста в ячейку

```
if ($(this).text() === "") {
    turn = (move%2)? "X":"O";
    $(this).append(turn);
    ...
}
```

5. Далее необходимо вычислить текущий рейтинг

```
score[turn] += parseInt(this.dataset.indicator);
```

6. Если игрок собрал игрок выигрышную комбинацию, выдать сообщение о победе и начать новую игру

```
if (checkForWinner(score[turn])) {  
    $('p').text(turn + " wins!");  
    startNewGame();  
    return;  
}
```

7. Если ни один из игроков не собрал выигрышную комбинацию и ходов больше не осталось, выдать сообщение о ничьей и начать новую игру. Иначе перейти к следующему ходу

```
if (move === 9) {  
    $('p').text("Ничья");  
    startNewGame();  
    return;  
} else {  
    move++;  
    $('p').text ('Ходит ' + (move%2)? "X":"O")):  
}
```

8. Реализуйте функцию **checkForWinner()**. Функция принимает балл текущего игрока в качестве параметра и выдает **true**, если игрок победил и **false** в противном случае

```
function checkForWinner(score) {  
    for (var i = 0; i < wins.length; i += 1) {  
        if ((wins[i] & score) === wins[i]) {  
            return true;  
        }  
    }  
    return false;  
}
```

9. Реализуйте функцию **startNewGame ()**

- а. Функция должны сбрасывать значения всех компонентов в начальное значение

```
function startNewGame (){  
    $("#board tr td").text("");  
    score.X = 0; score.O =0;  
    move = 1;  
}
```

10. Готовое решение Вы можете найти в

**C:\Users\Администратор\Documents\solutions\10_jQuery_TicTac
Toe**