Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информатика»

Н. В. Самовендюк

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ ИНТЕРНЕТ. FRONT-END

ПРАКТИКУМ
по выполнению лабораторных работ
по одноименной дисциплине для студентов
специальности 1-40 04 01 «Информатика
и технологии программирования»
дневной формы обучения

Гомель 2021

УДК 004.42(075.8)
ББК 32.973.22я73
С17

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 7 от 02.03.2020 г.*)

Рецензент: зав. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *К. С. Курочка*

Самовендюк, Н. В.

С17 Разработка приложений для Интернет. Front-End : практикум по выполнению лабораторн. работ по одноим. дисциплине для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. формы обучения / Н. В. Самовендюк. – Гомель : ГГТУ им. П. О. Сухого, 2021. – 108 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Представлены краткие теоретические сведения и задания к лабораторным работам, с помощью которых студенты знакомятся с разработкой приложений для интернет на стороне клиента. В качестве языка реализации рассматривается JavaScript. Рассмотрены средства языка JavaScript и методы их использования для решения конкретных задач, возникающих при разработке веб-приложений.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения.

УДК 004.42(075.8)
ББК 32.973.22я73

СОДЕРЖАНИЕ

Введение 4
Лабораторная работа № 1. Введение в JavaScript 5
Лабораторная работа № 2. Основные конструкции языка JavaScript 17
Лабораторная работа № 3. Использование функций в JavaScript .. 31
Лабораторная работа № 4. Работа с массивами в JavaScript..... 41
Лабораторная работа № 5. Работа со строками в JavaScript 49
Лабораторная работа № 6. Раьота с объектами..... 54
Лабораторная работа № 7. Объектная модель документа (DOM).. 66
Лабораторная работа № 8. Обработка и валидация форм с использованием JavaScript..... 79
Лабораторная работа № 9. Объектная модель браузера (BOM). Таймеры..... 86
Лабораторная работа № 10. Сериализация объектов. Библиотека jQuery 92
Лабораторная работа № 11. Взаимодействие с сервером. Технология AJAX..... 100
Литература 108

Введение

Дисциплина «Разработка приложений для Интернет» является одной из базовых при подготовке студентов по специальности 1-40 04 01 «Информатика и технологии программирования». Знания и умения, полученные при изучении дисциплины, необходимы для создания современных и эффективных веб-приложений.

Современные веб-приложение, как правило, разделяются на две части: клиент и сервер. Клиент представляет собой веб-страницу с кодом JavaScript. К серверным технологиям относятся PHP, Ruby, Node.js, ASP.NET и т.д., которые получают запрос от клиента, обрабатывают и отправляют в ответ результат обработки.

Практикум ориентирован на формирование у студентов основных понятий и принципов построения приложений для интернет на стороне клиента.

В качестве языка реализации рассматривается JavaScript.

В первой чати практикума рассматриваются: основы синтаксиса, управляющие конструкции, встроенные и пользовательские типы данных языка JavaScript. Представлены задания по работе с функциями, массивами и объектами, строками и регулярными выражениями.

Отдельное место в данном практикуме занимают темы, связанные с объектными моделями веб-документов и браузера. Акцентируется внимание на принципах событийно-управляемого программирования и событийной модели веб-страниц.

Рассматривается подход к построению интерактивных пользовательских интерфейсов веб-приложений на основе технологии AJAX.

Практикум полностью соответствует учебной программе по дисциплине «Разработка приложений для интернет».

Лабораторная работа № 1
Введение в JavaScript

Цель работы: изучить основные типы данных и операторы языка JavaScript.

Краткие теоретические сведения:

Для написания сценариев на языке JavaScript достаточно любого текстового редактора, но лучше использовать специализированные, такие как Sublime Text, Notepad++, или интегрированные среды разработки Visual Studio Code или WebStorm.

Программы на JavaScript могут быть вставлены в любое место HTML-документа с помощью тега <script>. Для вставки сценария или библиотеки из внешнего файла используется атрибут src, в котором указывается либо абсолютный путь до скрипта от корня сайта, либо полный URL-адрес. Например:

```
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
```

Сценарий JavaScript выполняется с использованием инструкций, которые представляют собой синтаксические конструкции и команды, которые выполняют действия. Обычно каждую инструкцию пишут на новой строке, чтобы код было легче читать, и заканчивают точкой с запятой.

```
alert('Hello');
alert('World');
```

Для пояснения программного кода используются комментарии. В JavaScript комментарии могут находиться в любом месте скрипта и могут быть однострочными и многострочными.

```
// Этот комментарий занимает всю строку
/*
Это - многострочный комментарий.
*/
```

Любое веб-приложение обычно обрабатывает какую-либо информацию. Для хранения информации в JavaScript используются переменные и константы.

Для создания переменной в JavaScript используйте ключевое слово let:

```
let message;
message = 'Hello';
```

На практике для краткости часто совмещают объявление переменной и запись данных в одну строку.

В старых скриптах вы также можете найти другое ключевое слово: var вместо let. Ключевое слово var – почти то же самое, что и let. Оно объявляет переменную, но немного по-другому, «устаревшим» способом.

Для объявления константной, то есть, неизменяемой переменной, используется ключевое слово const:

```
const PI = 3.1415;
```

При названии переменных надо учитывать, что JavaScript является регистрозависимым языком, то есть в следующем коде объявлены две разные переменные:

```
var myIncome;
var MyIncome;
```

Значения, которые заносятся в переменную с использованием оператора присваивания (=), могут быть одного из 8 основных типов:

- **number** для любых чисел: целочисленных или чисел с плавающей точкой, целочисленные значения ограничены диапазоном $\pm 2^{53}$.
- **bigint** для целых чисел произвольной длины.
- **string** для строк. Строка может содержать один или больше символов, нет отдельного символьного типа.
- **boolean** для true/false.
- **null** для неизвестных значений – отдельный тип, имеющий одно значение null.

- **undefined** для неприсвоенных значений – отдельный тип, имеющий одно значение undefined.
- **object** для более сложных структур данных.
- **symbol** для уникальных идентификаторов.

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

- Имеет две формы: typeof x или typeof(x).
- Возвращает строку с именем типа. Например, "string".
- Для null возвращается "object" – это ошибка в языке, на самом деле это не объект.

Часто значения, которые заносятся в переменную, являются результатом вычисления выражения. Выражение представляет из себя набор операндов и операторов, выполняющих действия над операндами.

JavaScript является наследником языка C. Поэтому набор операторов и управляющих конструкций в языке точно такой же как и в C, с которым вы уже знакомы.

Следует, однако, отметить, что язык JavaScript является интерпретируемым и слаботипизированным, в отличии от компилируемых. Тип переменной определяется по введенному значнению или результату вычиления выражения, поэтому в процессе выполнения сценария тип переменной может меняться. Следует внимательно следить за преобразованием типов.

Для преобразования строк в числа в JavaScript предусмотрены встроенные функции parseInt() и parseFloat(). Функция parseInt(строка, основание) преобразует указанную в параметре строку в целое число в системе счисления по указанному основанию (8, 10 или 16). Если основание не указано, то предполагается 10, то есть десятеричная система счисления. Функция parseFloat(строка) преобразует указанную строку в число с плавающей разделительной (десятичной, основание) точкой:

```
parseInt("3.14") // результат= 3
parseInt("Вася") // результат = NaN, то есть не является числом
parseInt("0xFF", 16) // результат = 255
parseFloat("3.l4") // результат= 3.14
parseFloat("-7.875") // результат = -7.875
```

В JavaScript предусмотрены довольно скудные средства для ввода и вывода данных. Это вполне оправданно, поскольку JavaScript создавался в первую очередь как язык сценариев для веб-страниц. Можно воспользоваться тремя стандартными методами глобального объекта window для ввода и вывода данных: alert(), prompt(), confirm().

Метод alert() позволяет выводить диалоговое окно с заданным сообщением и кнопкой ОК:

```
alert(“сообщение”);
```

Сообщение представляет собой данные любого типа: последовательность символов, заключенную в кавычки, число (в кавычках или без них), переменную или выражение.

Метод confirm() позволяет вывести диалоговое окно с сообщением и двумя кнопками – ОК и Отмена (Cancel). В отличие от метода alert этот метод возвращает логическую величину, значение которой зависит от того, на какой из двух кнопок щелкнул пользователь. Если он щелкнул на кнопке ОК то возвращается значение true (истина), если же он щелкнул на кнопке Отмена, то возвращается значение false (ложь). Синтаксис применения метода confirm имеет следующий вид:

```
let answer = confirm(“сообщение”);
```

Метод prompt позволяет вывести на экран диалоговое окно с сообщением, а также с текстовым полем, в которое пользователь может ввести данные. Кроме того, в этом окне предусмотрены две кнопки: ОК и Отмена (Cancel). Данный метод принимает два параметра: сообщение и значение, которое должно появиться в текстовом поле ввода данных по умолчанию. Если пользователь щелкнет на кнопке ОК, то метод вернет содержимое поля ввода данных, если он щелкнет на кнопке Отмена, то возвращается логическое значение false (ложь).

```
let input = prompt(“сообщение”,”начальное_значение”);
```

Практическая часть:

Задание 1. Работа с диалоговыми окнами.

1. В редакторе создайте файл z1_1.html и поместите в него следующий код:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Первая программа на JavaScript</title>
 </head>
 <body>
  <script>
   "use strict";
   alert("Мое первое диалоговое окно");
   const answer = confirm(
    "Хотите продолжить выполнения сценария на JavaScript"
   );
   if (answer) {
    const name = prompt("Введите ваше имя", "");
    alert(name + ", у вас уже начинает получаться!!!");
   } else alert("Жаль, можно было бы еще поработать!");
  </script>
 </body>
</html>
```

2. Откройте этот файл в браузере и просмотрите результат.

Задание 2. Вычисление арифметических выражений.

1. Создайте файл z1_2.html и поместите в него следующий код:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Вычисление арифметических выражений</title>
  </head>
  <body>
    <script>
      "use strict";
      const x = parseInt(prompt("Введите значение x", ""));
      let a = x * x - 7 * x + 10;
      let b = x * x - 8 * x + 12;
      let c = a / b;
      alert("c = " + c);
    </script>
  </body>
</html>
```

2. Просмотрите в браузере результаты работы скрипта.

Задание 3. Вычисление площади и периметра правильного n-угольника, описанного около окружности радиуса R.

1. Создайте файл z1_3.html и поместите в него следующий код:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Линейные алгоритмы</title>
  </head>
  <body>
    <script>
      "use strict";
      let n = parseInt(prompt("Введите количество углов n", ""));
      let r = parseInt(prompt("Введите радиус r", ""));
      let a = 2 * r * Math.tan(Math.PI / n);
      let p = a * n;
      let s = (1 / 2) * n * a * r;
```

```
        alert("Площадь = " + s.toFixed(2));
        alert("Периметр = " + p.toFixed(2));
      </script>
    </body>
  </html>
```

2. Просмотрите в браузере результаты работы скрипта.

Задание 4. Формирование динамических страниц с использованием метода write объекта document.

1. В текстовом редакторе создайте файл z1_4.html и поместите в него следующий код:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Динамически сформированная страница</title>
  </head>
  <body>
    <script>
      "use strict";
      document.write("Начало сформированной страницы");
      document.write("<h1> Заголовок первого уровня</h1>");
      document.write(
        '<p style="text-align:center;font-size:18px;color:red;">Изменение размера и цвета шрифта'
      );
      document.write("<p>Конец формирования страницы, содержащей сценарий");
    </script>
  </body>
</html>
```

2. Просмотрите в браузере результаты работы скрипта.

Индивидуальные задания

Во всех скриптах, в заголовке окна браузера должны быть ваши фамилия и имя!!!

Задание 1. Вы совершаете покупку в магазинах евроторга. Необходимо написать сценарий, в котором с использованием метода prompt вводится общая сумма покупки, вводится скидка в размере от 1 до 5 %. В окне документа должны быть отображены: общая стоимость покупки, скидка в рублях и стоимость покупки со скидкой. Вывод оформить как кассовый чек.

ООО "Евроторг"
СПАСИБО ЗА ПОКУПКУ!
Дата покупки: 12.09.2016 15:19

Наименование товара
1 = 900.00
Наименование товара
1 = 700.00

ИТОГО
1600.00 BYN
Ваша скидка: 5%
80.00 BYN
СУММА СО СКИДКОЙ
1520.00 BYN

Задание 2. Помогите школьнику выполнить домашнее задание: решить задачу (таблица 1) и посчитать значение переменной b (таблица 2).

Исходные данные вводятся с использованием метода prompt. При выводе информации предусмотреть форматирование документа, вывод текста задания, включая рисунок исходного выражения, и вывод информации о разработчике скрипта.

Таблица 1

Вариант	Вычислить	Расчетные формулы
1	Площадь круга и длину окружности радиуса r	$S = \pi r^2 \quad l = 2\pi r$
2	Площадь и угол при основании равнобедренного треугольника с основанием a и высотой h	$S = \frac{a\,h}{2}$ $\alpha = \text{arctg}(2h/a)$
3	Площадь и периметр прямоугольника со сторонами a, b	$S = ab \quad P = 2(a+b)$
4	Скорость в конце пути и путь, пройденный за время t с ускорением a при $v_0 = 0$	$v = at \quad S = \frac{at^2}{2}$
5	Площадь и периметр квадрата со стороной a	$S = a^2 \quad P = 4a$
6	Объем и площадь боковой поверхности параллелепипеда со сторонами a, b, c	$V = abc \quad S = 2(a+b)c$
7	Площадь кольца с внешним радиусом R и внутренним r	$S = \pi(R^2 - r^2)$
8	Площадь боковой поверхности и объем цилиндра с радиусом основания r и высотой h	$S = 2\pi r\,h \quad V = \pi r^2 h$
9	Площадь и периметр прямоугольного треугольника с катетами a, b и гипотенузой c	$S = \frac{ab}{2}$ $P = a + b + c$
10	Объем и площадь поверхности куба со стороной a	$V = a^3 \quad S = 6a^2$
11	Периметр и площадь треугольника со сторонами a, b, c	$P = a + b + c = 2p$ $S = \sqrt{p(p-a)(p-b)(p-c)}$
12	Площадь основания и объем цилиндра с радиусом основания r и высотой h	$S = \pi r^2 \quad V = Sh$
13	Объем и площадь основания параллелепипеда со сторонами a, b, c	$V = abc \quad S = ab$
14	Площадь основания и объем конуса с радиусом основания r и высотой h	$S = \pi r^2 \quad V = \frac{Sh}{3}$
15	Гипотенузу и площадь прямоугольного треугольника с катетами a, b	$c = \sqrt{a^2 + b^2} \quad S = \frac{ab}{2}$
16	Высоту и площадь равнобедренной трапеции с основаниями a, b ($b > a$) и углом при большем основании α	$h = \frac{b-a}{2}\text{tg}\,\alpha$ $S = \frac{b+a}{2}h$
17	Площадь поверхности и объем шара радиуса R	$S = 4\pi R^2 \quad V = \frac{4}{3}\pi R^3$

Окончание табл. 1

Вариант	Вычислить	Расчетные формулы
18	Скорость в конце пути и путь, пройденный телом за время t с ускорением a и начальной скорости v_0	$v = v_0 + at$ $S = v_0 t + \frac{at^2}{2}$
19	Радиус круга, вписанного в треугольник со сторонами a, b, c	$r = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$ $p = \frac{a+b+c}{2}$
20	Кинетическую энергию тела массой m, движущегося со скоростью v	$E_k = \frac{mv^2}{2}$
21	Площадь прямоугольного треугольника с гипотенузой c и одним из катетов a	$S = \frac{ab}{2}$ $b = \sqrt{c^2 - a^2}$
22	Периметр и площадь прямоугольного треугольника с катетами a, b	$S = \frac{ab}{2}$ $P = a + b + \sqrt{a^2 + b^2}$
23	Высоту и площадь равнобедренного треугольника с основанием a и углом при основании	$h = \frac{a}{2}\operatorname{tg}\alpha$ $s = \frac{ah}{2}$
24	Радиус круга, описанного вокруг треугольника со сторонами a, b, c	$R = \frac{abc}{4\sqrt{p(p-a)(p-b)(p-c)}}$ $p = \frac{a+b+c}{2}$
25	Периметр и площадь параллелограмма со сторонами a, b и острым углом α	$S = ab\sin\alpha$ $P = 2(a+b)$
26	Площадь прямоугольной трапеции с основаниями a, b ($b > a$) и углом при большем основании α	$h = (b-a)\operatorname{tg}\alpha$ $s = \frac{b+a}{2}h$
27	Сопротивление проводника длиной l, площадью поперечного сечения S и удельным сопротивлением ρ	$R = \frac{\rho l}{S}$
28	Расстояние между точками с координатами $x1$, $y1$ и $x2$, $y2$	$l = \sqrt{(x2-x1)^2 + (y2-y1)^2}$
29	Путь, пройденный за время t со скоростью v	$S = vt$
30	Емкость плоского конденсатора C с площадью поверхности одной пластины S, расстоянием между пластинами d и диэлектрической проницаемостью материала ε	$C = \frac{\varepsilon S}{4\pi d}$

Таблица 2

Вариант	Вид функции	Вариант	Вид функции
1	$b=\frac{1+\cos^2(x+z)}{\lvert x^3-2y^2\rvert}$	16	$b=x+\frac{\sqrt[3]{zy}}{y+\cos x}$
2	$b=\frac{\ln^2\lvert z\rvert}{\sqrt[3]{\lvert x\rvert+\lvert y\rvert}}$	17	$b=\lg\left(\sqrt{e^{x-y}+x^{\lvert y\rvert}+z}\right)$
3	$b=\frac{y^3}{x+y^3\cos^2 z}$	18	$b=1+\frac{x^2+1}{3+y^2}+\sin 2z$
4	$b=\sqrt{x+\sqrt[4]{\lvert y\rvert}}+\cos^2 z$	19	$b=\lvert\cos x+\cos y\rvert+2\sin^2 z$
5	$b=\frac{\sqrt[3]{e^{\sin x}}\cos y}{z^2+1}$	20	$b=\frac{\ln(y^3)(z-x/2)}{2\cos^2 x}$
6	$b=z(\operatorname{tg} y-e^{-(x+3)})$	21	$b=\sqrt{10\left(\sqrt[3]{z}+x^{(y+2)}\right)}$
7	$b=\lvert x-y\rvert\left(\sin^2 z+\operatorname{tg} z\right)$	22	$b=(\sin z)^2+\lvert x+y\rvert$
8	$b=\sqrt{y+\sqrt[3]{x}}-1+2z$	23	$b=e^{2z}-\sqrt[3]{y\lvert x\rvert}$
9	$b=x\left(\operatorname{tg} z+\cos^2 y\right)$	24	$b=e^{(x-1)}+\sin y$
10	$b=e^{\lvert x-y\rvert}\left(\operatorname{tg}^2 z+1\right)^x$	25	$b=\sqrt{\lvert z\rvert e^{-(y+x/2)}}$
11	$b=\cos^2 z+\operatorname{tg} 2x+\lvert y\rvert$	26	$b=\frac{4y^2e^{2x}\sin^2 z}{3z^3+\ln x}$
12	$b=5\operatorname{tg} z-4y^2+\lvert xy\rvert$	27	$b=\frac{\sqrt{y\ln x}-zx^2}{1+\operatorname{tg}^2 x^2}x$
13	$b=(z-x)\frac{y-\ln z}{1+(y-x)^2}$	28	$b=\frac{\lg\left(y+\sqrt{z+x^2}\right)}{y+x^2}$
14	$b=y^z+\sqrt{\lvert x\rvert+\lvert y\rvert}$	29	$b=\frac{x^2+4}{\sin^2 z^2+x/2}y$
15	$b=\frac{\lg\left(\sqrt{x}+\sqrt{y}+2\right)}{\lvert 2z\rvert}$	30	$b=\frac{\sin x+\sqrt{\lvert z-y\rvert}}{y(x-2)+x^2}$

Для вычисления арифметического выражения используются методы объекта **Math**:

Методы	Описание
abs(x)	Возвращает абсолютное значения (модуль) числа x.
acos(x)	Возвращает арккосинус числа x в радианах.
asin(x)	Возвращает арксинус числа x в радианах.
atan(x)	Возвращает арктангенс числа x как численное значение между -PI/2 и PI/2.

ceil(x)	Округляет значение x до первого большего целого числа.
cos(x)	Возвращает косинус числа x (число x задается в радианах).
exp(x)	Возвращает значение E в степени x.
floor(x)	Округляет значение x до первого меньшего целого числа.
log(x)	Возвращает натуральный логарифм (с основанием E) x.
max(x1,x2,...xn)	Возвращает большее из чисел x1,x2,...xn.
min(x1,x2,...xn)	Возвращает меньшее из чисел x1,x2,...xn.
pow(x,y)	Возводит x в степень y и возвращает результат.
random()	Возвращает случайное число между 0 и 1 (например 0.6230522912910803).
round(x)	Округляет значение x до ближайшего целого числа.
sin(x)	Возвращает синус числа x (число x задается в радианах).
sqrt(x)	Возвращает квадратный корень x.
tan(x)	Возвращает тангенс угла.

Требование по содержанию отчета

В отчете должны быть отображены следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задания.
4. Листинг кода сценария с комментариями.
5. Результат выполнения скрипта.
6. Выводы.

Контрольные вопросы для защиты:

1. Какие виды диалоговых окон вы знаете?
2. Как ввести данные пользователя?
3. Как преобразовать строку в число?
4. Какие типы переменных используются в JavaScript?
5. Что означает значение **undefined**?
6. Как вывести сообщение в диалоговое окно?
7. Для чего используется ключевое слово **var**?
8. Для чего используется ключевое слово **let**?
9. Для чего используется ключевое слово **const**?
10. Для чего используется метод **write** объекта **document**?
11. Для чего используется встроенный объект **Math**?
12. Как записать сложное арифметическое выражение?

Лабораторная работа № 2
Основные конструкции языка JavaScript

Цель работы: изучить основные конструкции языка JavaScript.

Краткие теоретические сведения:

Условные конструкции

Условные конструкции позволяют выполнить те или иные действия в зависимости от определенных условий.

Оператор if

Оператор if относится к числу наиболее популярных. Оператор if применяется следующим образом:

```
if (условие) {
    [операторы]
}
```

В качестве условия может указываться любое логическое выражение. Если результат условия равен true, выполняются операторы, продолжается выполнение остального программного кода. Если условие возвращает false, операторы игнорируются.

Конструкция if..else

Иногда одной конструкции if оказывается недостаточно. Часто требуется зарезервировать набор операторов, которые будут выполняться в случае, когда условное выражение возвращает false. Это можно сделать, добавив еще один блок непосредственно после блока if:

```
if (условие) {
    [операторы]
} else {
    [операторы]
}
```

Конструкция switch..case

Конструкция switch..case является альтернативой использованию конструкции if..else if..else и позволяет обработать сразу несколько условий.

После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора case. И если совпадение будет найдено, то будет выполняться определенный блок case.

В конце каждого блока case ставится оператор break, чтобы избежать выполнения других блоков. Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default:

```
let income = 300;
switch(income){

   case 100 :
      console.log("Доход равен 100");
      break;
   case 200 :
      console.log("Доход равен 200");
      break;
   case 300 :
      console.log("Доход равен 300");
      break;
   default:
      console.log("Доход неизвестной величины");
      break;
}
```

Тернарная операция

Тернарная операция состоит из трех операндов и имеет следующее определение:

[первый операнд - условие] ? [второй операнд] : [третий операнд].

В зависимости от условия тернарная операция возвращает второй или третий операнд: если условие равно true, то возвращается второй операнд; если условие равно false, то третий.

Циклы

Циклы позволяют в зависимости от определенных условий выполнять некоторое действие множество раз. В JavaScript имеются следующие виды циклов:

- **for;**
- **for..in;**
- **for..of;**
- **while;**
- **do..while.**

Цикл for

Цикл for имеет следующее определение:

```
for ([инициализация счетчика]; [условие]; [изменение счетчика]){
   // действия
}
```

Цикл for..in

Цикл for..in предназначен для перебора массивов и объектов. Его формальное определение:

```
for (индекс in объект) {
   // действия
}
```

Цикл for...of

Цикл for...of похож на цикл for...in и предназначен для перебора коллекций, например, массивов:

```
for (индекс of массив) {
   // действия
}
```

Цикл while

Цикл while выполняется до тех пор, пока некоторое условие истинно. Его определение:

```
while (условие) {
```

```
    // действия
}
```

Цикл do..while

В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется. Его определение:

```
do {
    // действия
} while(условие)
```

Операторы continue и break

Иногда бывает необходимо выйти из цикла до его завершения. В этом случае мы можем воспользоваться оператором break. Если нам надо просто пропустить итерацию, но не выходить из цикла, мы можем применять оператор continue.

Практическая часть:

Операторы условного перехода

Задание 1

Создайте файл **z2_1.html**, содержащий следующий скрипт, демонстрирующий возможности использования условного оператора **IF**:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Использование оператора IF</title>
  </head>
  <body>
    <script>
      "use strict";
```

```
        document.write(
          "<p style='text-align:center'>Использование оператора
<strong>IF</strong></p>"
        );
        let age = 24;
        if (age < 18)
          document.write("Вы слишком молоды для просмотра этого
сайта"); // ничего не выводится
        age = 10;
        if (age < 18)
          document.write("Вы слишком молоды для просмотра этого
сайта"); // выводится сообщение

        document.write(
          "<p style='text-align:center'>Использование оператора
<strong>IF</strong> с блоком <strong>ELSE</strong></p>"
        );
        let s = "плохо";
        if (s == "хорошо") document.write("Я в хорошем настроении!");
        else document.write("Мне ", s);

        document.write(
          "<p style='text-align:center'>Использование оператора
<strong>IF</strong> с блоками <strong>ELSE IF</strong></p>"
        );
        if (s == "хорошо") {
          document.write("Я в хорошем настроении!");
        } else if (s == "плохо") {
          document.write("Не отчаивайтесь!");
        } else {
          document.write("Непонятно, просто", s);
        }
      </script>
    </body>
  </html>
```

Задание 2

Создайте файл z2_2.html, содержащий следующий скрипт, демонстрирующий нахождения большего среди трех чисел с использованием условного оператора IF и условной операции:

```
<script>
```

```
        "use strict";
        //Пример нахождения большего среди трех чисел
        document.write(
          "<p style='text-align:center'>Пример определения большего числа</p>"
        );
        let a = 1,
        let b = 22,
        let c = 3;
        if (a > b && a > c) document.write("Наибольшее среди трех чисел - a=", a);
        else if (b > c) document.write("Наибольшее среди трех чисел - b=", b);
        else document.write("Наибольшее среди трех чисел - c=", c);
        //Использование тернарной операции
        let max = a > b ? a : b;
        max = max > c ? max : c;
        document.write("<p>Значение наибольшее среди трех чисел - ", max);
    </script>
```

Задание 3

Создайте файл z2_3.html, содержащий скрипт, демонстрирующий возможности использования оператора SWITCH:

```
    <script>
      "use strict";
      let age = parseInt(prompt("Введите ваш возраст"));
      let b = age % 10;
      let s;
      switch (b) {
        case 1:
          s = " год";
          break;
        case 2:
        case 3:
        case 4:
          s = " года";
```

```
      break;
    default:
      s = " лет";
  }
  if (age > 10 && age < 20) s = "лет";
  document.write("Вам - ", age, s);
</script>
```

Операторы цикла

Задание 4

Создайте файл z2_4.html, содержащий скрипт, демонстрирующий использование операторов цикла for, while, do while:

```
<script>
  "use strict";
  document.write("<p>Использование оператора цикла <b>FOR</b></p>");
  // Возведение числа x в степень
  let x = parseInt(prompt("Введите число x")); // заданное число
  let y = parseInt(prompt("Введите степень y")); // степень, в которую надо возвести число
  let z = x; // результат
  for (let i = 2; i <= Math.abs(y); i++) z = z * x;
  z = y > 0 ? z : 1 / z;
  document.write("Число ", x, " в степени ", y, " равно ", z);

  document.write("<p>Использование оператора цикла <b>WHILE</b></p>");
  let z1 = x;
  let i = 2;
  while (i <= Math.abs(y)) {
   z1 = z1 * x;
   i++;
  }
  z1 = y > 0 ? z1 : 1 / z1;
  document.write("Число ", x, " в степени ", y, " равно ", z1);
```

```
        document.write("<p>Использование оператора цикла <b>DO WHILE</b></p>");
        let z2 = x;
        i = 2;
        do {
          z2 = z2 * x;
          i++;
        } while (i <= Math.abs(y));
        z2 = y > 0 ? z2 : 1 / z2;
        document.write("Число ", x, " в степени ", y, " равно ", z2);
      </script>
```

Задание 5

Создайте файл z2_6.html, содержащий скрипт, демонстрирующий использование оператора цикла for in:

```
      <script>
        "use strict";
        let propertyInfo = "";
        for (let propertyName in document) {
          propertyInfo = propertyName + " — " + document[propertyName];
          document.write(propertyInfo + "<br>");
        }
      </script>
```

Задание 6

Создайте файл z2_7.html, содержащий скрипт, демонстрирующий использование оператора цикла while в игре с пользователем:

```
      <script>
        "use strict";
        let guess = Math.random() * 100; // генерация случайного числа
        guess = Math.floor(guess); // отбрасывание дробной части
        let f = true;
        let number;
```

```
let count = 0;
let answer = confirm("Сыграем?");
if (answer) {
  while (f) {
    number = parseInt(prompt("введите число"));
    count++;
    if (number == guess) {
      alert("Вы угадали\n количество попыток: " + count);
      f = false;
    } else if (number > guess) {
      alert("Число меньше, попробуйте еще раз");
    } else {
      alert("Число больше, попробуйте еще раз");
    }
  }
 } else alert("Жаль.\nМожет быть в другой раз?");
</script>
```

Индивидуальные задания

Во всех скриптах, в заголовке окна браузера должны быть ваши фамилия и имя!!!

Задание 1

Пусть в скрипте lab2-1.html задана переменная lang, которая может принимать значения «ru», «en», «fr» или «de», введенные пользователем. Используя операторы if-else-elseif обеспечьте вывод на экран полного названия языка (русский, английский и т.д.) в зависимости от того, что присвоено переменной lang. Обязательно предусмотреть случай неверного задания значения переменной lang - тогда должна выводиться надпись «Язык неизвестен».

Задание 2

Сделайте тоже самое, что в задании 3, но используя оператор switch (скрипт lab2-2.html). Обработку неправильного ввода осуществить с использованием конструкции **try..catch.**

Задание 3

В соответствии со своим вариантом необходимо написать Java-скрипт (lab2-3.html) для вычисления значения функции y. При выводе информации предусмотреть форматирование документа, вывод текста задания на лабораторную работу, включая рисунок исходной функции, и вывод информации о разработчике скрипта.

Вариант	Вид функции
1	$y=\begin{cases}1/x, & \text{если } x\ge -5,\ x\ne 0 \quad (1)\\ x^2, & \text{если } x\le -10 \quad (2)\\ \sqrt{\lvert x+1\rvert} & \text{в остальных случаях} \quad (3)\end{cases}$
2	$y=\begin{cases}x^2, & \text{если } x\le 0,\ x\ne -10 \quad (1)\\ \sqrt{x+1}, & \text{если } x>1 \quad (2)\\ 1/x & \text{в остальных случаях} \quad (3)\end{cases}$
3	$y=\begin{cases}x+e^{2x}, & \text{если } x\le 0,\ \ x\ne -1 \quad (1)\\ \cos^2 x, & \text{если } 0<x\le 3{,}14 \quad (2)\\ x & \text{в остальных случаях} \quad (3)\end{cases}$
4	$y=\begin{cases}x^3, & \text{если } x>5,\ \ x\ne 20 \quad (1)\\ x^2, & \text{если } -5\le x\le 5 \quad (2)\\ \lg\lvert x\rvert & \text{в остальных случаях} \quad (3)\end{cases}$
5	$y=\begin{cases}\sqrt{x}, & \text{если } x\ge 100,\ x\ne 105 \quad (1)\\ \sqrt[3]{x}, & \text{если } x=20 \text{ или } x=40 \quad (2)\\ x^2+1 & \text{в остальных случаях} \quad (3)\end{cases}$
6	$y=\begin{cases}\sqrt{x-1}, & \text{если } x\ge 10,\ x\ne 20 \quad (1)\\ 1/x+e^{2x}, & \text{если } x<0 \quad (2)\\ \ln(x+1) & \text{в остальных случаях} \quad (3)\end{cases}$

Вариант	Вид функции
7	$y=\begin{cases}8x+1, & \text{если } x\ge 5,\ x\ne 9 \quad (1)\\ x^2+\lvert x\rvert, & \text{если } x\le 1 \quad (2)\\ x^3+\sqrt{x} & \text{в остальных случаях} \quad (3)\end{cases}$
8	$y=\begin{cases}1-3x, & \text{если } x>0,\ x\ne 8 \quad (1)\\ x^2-\sin x, & \text{если } x\le -1 \quad (2)\\ \cos x & \text{в остальных случаях} \quad (3)\end{cases}$
9	$y=\begin{cases}x^3+1, & \text{если } x\ge 8,\ x\ne 10 \quad (1)\\ 2x^2+\sqrt[3]{\lvert x\rvert}, & \text{если } x\le 1 \quad (2)\\ \sqrt{x} & \text{в остальных случаях} \quad (3)\end{cases}$
10	$y=\begin{cases}2x^2, & \text{если } x>0,\ x\ne 3 \quad (1)\\ \sqrt{x^2+1} & \text{если } x\le -2 \quad (2)\\ \lvert x+5\rvert & \text{в ост случаях} \quad (3)\end{cases}$
11	$y=\begin{cases}\sqrt{\lvert 2x-x^2-1\rvert}, & \text{если } x\le -1,\ x\ne -4 \quad (1)\\ \ln(x+3), & \text{если } x>0 \quad (2)\\ x/2 & \text{в ост. случаях} \quad (3)\end{cases}$
12	$y=\begin{cases}\sqrt{x-1}, & \text{если } x\ge 10, x\ne 20 \quad (1)\\ 1/x+e^{2x}, & \text{если } x<0 \quad (2)\\ \ln(x+1) & \text{в остальных случаях} \quad (3)\end{cases}$
13	$y=\begin{cases}x/3, & \text{если } -3\le x\le 3 \quad (1)\\ \lg(x^2+1), & \text{если } x<-3 \text{ или } x=4 \quad (2)\\ \sqrt{x^3-2} & \text{в ост случаях} \quad (3)\end{cases}$

Вариант	Вид функции
14	$y=\begin{cases}\lvert x^3+4\rvert, & \text{если } x\le -1 \text{ или } x=0 \quad (1)\\ \sqrt{x/2}, & \text{если } x\ge 8 \quad (2)\\ x^3 & \text{в ост. случаях} \quad (3)\end{cases}$
15	$y=\begin{cases}\sqrt{x+1}, & \text{если} x\ge 8, x\ne 10 \quad (1)\\ 0{,}6x, & \text{если} 0<x<8 \quad (2)\\ \lg\lvert x\rvert+3 & \text{в ост случаях} \quad (3)\end{cases}$

Задание 4

В скрипте lab2-4.html, используя вложенные циклы for, отобразите на экране таблицу 10×10, в ячейках которой идут числа от 1 до 100.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

При этом красным цветом выделены «треугольные» числа, т.е. числа вида n(n+1)/2 (где n=1,2,… 14). Все другие числа черного цвета. Массивы не использовать! Ширина рамки таблицы равна 1, отступ содержимого ячеек от границы равен 5.

Задание 5

В скрипте lab2-5.html вычислить и вывести на экран значения функции, используя стандартные функции объекта Math и с помощью разложения функции в ряд Тейлора. При написании скрипта воспользоваться оператором цикла do .. while.

Вариант	Разложение функции в ряд Тейлора
1.	$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+...\right)$ $\lvert x\rvert>1$
2.	$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}-...$ $\lvert x\rvert<\infty$
3.	$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ ...$ $\lvert x\rvert<\infty$
4.	$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}-\ ...$ $-1<x\le 1$
5.	$\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+...\right)$ $\lvert x\rvert<1$
6.	$\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-\left(x+\frac{x^2}{2}+\frac{x^3}{3}+...\right)$ $-1\le x\le 1$
7.	$arcctg\ x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}-\ ...$ $\lvert x\rvert\le 1$
8.	$arctg\ x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+...$ $x>1$
9.	$arctg\ x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ ...$ $\lvert x\rvert\le 1$
10.	$Arth\ x=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ ...$ $\lvert x\rvert<1$
11.	$Arth\ x=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+...$ $\lvert x\rvert>1$
12	$arctg\ x=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+...$ $x<$

Вариант	Разложение функции в ряд Тейлора
13.	$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-...$ $\lvert x\rvert<\infty$

14.	$\cos\ x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots \qquad \lvert x \rvert < \infty$
15.	$\frac{\sin\ x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} - \ldots \qquad \lvert x \rvert < \infty$

Требование по содержанию отчета

В отчете должны быть отображены следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задания.
4. Листинг кода сценария с комментариями.
5. Результат выполнения скрипта.
6. Выводы.

Контрольные вопросы для защиты:

1. Какие операторы управления вы знаете?
2. Для чего используется оператор **break** в операторе **switch**?
3. Как записать сложное условие?
4. В чем разница между ключевыми словами **else** и **else if**?
5. Что такое условная операция?
6. Какие операторы цикла вы знаете?
7. В чем отличие операторов цикла **while** и **do while**?
8. Что записывается в заголовке оператор цикла **for**?
9. Где используется цикл **for in**?
10. Как инициализируется счетчик цикла?
11. Как задается шаг изменения счетчика цикла?
12. Как принудительно выйти из цикла?

Лабораторная работа № 3
Использование функций в JavaScript

Цель работы: научиться использовать стандартные и создавать пользовательские функции в JavaScript.

Краткие теоретические сведения:

Функции представляют собой набор инструкций, выполняющих определенное действие или вычисляющих определенное значение.

Синтаксис определения функции:

```
function имя_функции([параметр [, ...]]){
    // Инструкции
}
```

Определение функции начинается с ключевого слова function, после которого следует имя функции. Наименование функции подчиняется тем же правилам, что и наименование переменной: оно может содержать только цифры, буквы, символы подчеркивания и доллара ($) и должно начинаться с буквы, символа подчеркивания или доллара.

После имени функции в скобках идет перечисление параметров. Даже если параметров у функции нет, то просто идут пустые скобки. Затем в фигурных скобках идет тело функции, содержащее набор инструкций.

Чтобы выполнить тело функции, необходимо вызвать функцию с передачей параметров или без параметров:

```
function sum(a, b, c){
    let d = a + b + c;
    console.log(d);
}

sum(1, 2, 3);
sum();
```

В отличии от компилируемых языков программирование, где количество фактических параметров, передаваемых при вызове функ-

ции, и количество формальных параметров, описанных при объявлении, должны совпадать, JavaScript позволяет не соблюдать это правило. Если для параметров не передается значение, то по умолчанию они имеют значение "undefined".

Для определения количества переданных параметров и доступа к их значениям при вызове функции создается массив arguments, доступный внутри тела функции.

Есть способ определения значения для параметров по умолчанию:

```
function sum(a = 1, b = 2, c = 3){
    let d = a + b + c;
    console.log(d);
}
```

С помощью spread-оператора мы можем указать, что с помощью параметра можно передать переменное количество значений:

```
function display(season, ...temps){
    console.log(season);
    for(index in temps){
        console.log(temps[index]);
    }
}
display("Весна", -2, -3, 4, 2, 5);
display("Лето", 20, 23, 31);
```

Функция может возвращать результат. Для этого используется оператор return:

```
function sum(a, b){
    return a + b;
}

console.log(sum(1, 2);
```

Функции могут выступать в качестве параметров других функций.

JavaScript позволяет определять функции внутри функций и возвращать результат в виде функции.

Среди функций отдельно можно выделить рекурсивные функции. Их суть состоит в том, что функция вызывает саму себя.

Рассмотрим, например, функцию, определяющую факториал числа:

```
function getFactorial(n){
   if (n === 1){
      return 1;
   }
   else{
      return n * getFactorial(n - 1);
   }
}
let result = getFactorial(4);
console.log(result); // 24
```

Практическая часть:

Использование встроенных функций

Задание 1

1. Создайте файлы z3_11.html и z3_12.html, демонстрирующие использование встроенных функций объекта Date:

1.1

```
<script>
     "use strict";
     let today = new Date();
     let hours = today.getHours();
     let minute = today.getMinutes();
     if (minute < 10) minute = "0" + minute;
     if (hours < 12) {
       var time12 = hours + ":" + minute + " am";
     } else {
       var time12 = hours - 12 + ":" + minute + " pm";
```

```
    }
    console.log("текущее время - ", time12);
</script>
```

1.2

```
<script>
    "use strict";
    let today = new Date();
    let newYearDay = new Date(today.getFullYear() + 1, 0, 1);
    const day = 86400000;
    const hour = 3600000;
    const minute = 60000;
    let time = newYearDay.getTime() - today.getTime();
    let days = Math.floor(time / day);
    let hours = Math.floor((time - days * day) / hour);
    let minutes = Math.floor((time - days * day - hours * hour) / minute);
    let str = days + " дней, " + hours + " часов, " + minutes + " минут.";
    console.log("До Нового года осталось ", str);
</script>
```

Создание пользовательских функций

Задание 2

Создайте файл z3_2.html, демонстрирующий создание пользовательской функции с аргументами и обращение к ней из основной части скрипта:

```
<script>
    "use strict";
    //Описание фvнкции вывода строки  с переходом на новую
    function PrintBR(txt) {
      document.write(txt, "<br />");
    }
```

```
        //Обращение к функции с передачей фактических парамет-
ров
        PrintBR("Это первая строка");
        PrintBR("Это вторая строка");
        PrintBR("Это еще одна строка");
    </script>
```

Задание 3

Создайте файл z2_31.html, демонстрирующий создание пользовательской функции, возвращающей значение:

```
    <script>
        "use strict";
        function AddNums(firstnum, secondnum) {
          return firstnum + secondnum;
        }
        document.write("3 + 5 = ", AddNums(3, 5));
        let a = 5;
        let b = 25;
        document.write("<br>", a, " + ", b, " = ", AddNums(a, b));
    </script>
```

Задание 4

Создайте файл z3_4.html, демонстрирующий использование функции с необязательными параметрами и параметрами по умолчанию:

```
    <script>
        "use strict";
        function printText(txt, size = 14) {
          document.write(`<div                                    style="font-
size:${size}pt">${txt}</div>`);
        }
        printText("<p>Крупный шрифт", 32);
        printText("<p>Шрифт по умолчанию - первая строка");
        printText("<p>Шрифт по умолчанию - вторая строка");
        printText("<p>Мелкий шрифт", 8);
    </script>
```

Задание 5

Создайте файл z3_5.html, демонстрирующий передачу функцию в качестве аргумента другой функции:

```
<script>
    "use strict";
    function kvadrat(a) {
      return a * a;
    }

    function polinom(a, k) {
      return k(a) + a + 5;
    }

    let result = polinom(3, kvadrat);
    document.write("<h2>result=", result);
</script>
```

Задание 6

Создайте файл z3_6.html, демонстрирующий использование функции как переменной:

```
<script>
    "use strict";
    let i = 5;
    function f(a, b, c) {
      if (a > b) return c;
    }
    document.write("Значение переменной i: " + i.valueOf());
    document.write("<p>Значение переменной f:<BR>" +
f.valueOf());
</script>
```

Задание 7

Создайте файл z3_7.html, демонстрирующий использование массива arguments:

```
<script>
    function mean(a, b) {
      if (arguments.length > mean.length) {
        alert("Аргументов больше, чем надо");
      }
      var result = 0;
      for (var i = 0; i < arguments.length; i++) {
        result += arguments[i];
      }
      return result / arguments.length;
    }
    document.write("Среднее - ", mean(2, 3), "<br>");
    document.write("Среднее - ", mean(2, 4, 6));
</script>
```

Индивидуальные задания

Во всех скриптах, в заголовке окна браузера должны быть ваши фамилия и имя!!!

Задание 1

В скрипте lab3-1.html, используя аргументы size, day и color функции weekDay(), отобразите названия дней недели уменьшающимся размером (начиная с 7) и разными цветами:

понедельник

вторник

среда

четверг

пятница

суббота

воскресенье

Задание 2

В соответствии со своим вариантом написать скрипт для вычисления значения функции b=f(x,y,z). Значения x, y и z должны вводиться пользователем, используя метод prompt. При выводе инфор-

мации предусмотреть форматирование документа, вывод текста задания, включая рисунок исходной функции, и вывод информации о разработчике скрипта.

Вариант	Вид функции	Вариант	Вид функции
1	$b=\frac{1+\cos^2(x+z)}{\lvert x^3-2y^2\rvert}$	16	$b=x+\frac{\sqrt[3]{zy}}{y+\cos x}$
2	$b=\frac{\ln^2\lvert z\rvert}{\sqrt[3]{\lvert x\rvert+\lvert y\rvert}}$	17	$b=\lg\left(\sqrt{e^{x-y}+x^{\lvert y\rvert}+z}\right)$
3	$b=\frac{y^3}{x+y^3\cos^2 z}$	18	$b=1+\frac{x^2+1}{3+y^2}+\sin 2z$
4	$b=\sqrt{x+\sqrt[4]{\lvert y\rvert}}+\cos^2 z$	19	$b=\lvert\cos x+\cos y\rvert+2\sin^2 z$
5	$b=\frac{\sqrt[3]{e^{\sin x}}\cdot\cos y}{z^2+1}$	20	$b=\frac{\ln(y^3)(z-x/2)}{2\cos^2 x}$
6	$b=z(tg\,y-e^{-(x+3)})$	21	$b=\sqrt{10\left(\sqrt[3]{z}+x^{(y+2)}\right)}$
7	$b=\lvert x-y\rvert(\sin^2 z+tgz)$	22	$b=(\sin z)^2+\lvert x+y\rvert$
8	$b=\sqrt{y+\sqrt[3]{x}}-1+2z$	23	$b=e^{2z}-\sqrt[3]{y\lvert x\rvert}$
9	$b=x(tg\,z+\cos^2 y)$	24	$b=e^{(x-1)}+\sin y$
10	$b=e^{\lvert x-y\rvert}(tg^2 z+1)^x$	25	$b=\sqrt{\lvert z\rvert e^{-(y+x/2)}}$
11	$b=\cos^2 z+tg\,2x+\lvert y\rvert$	26	$b=\frac{4y^2e^{2x}\sin^2 z}{3z^3+\ln x}$
12	$b=5tg\,z-4y^2+\lvert xy\rvert$	27	$b=\frac{\sqrt{y\ln x}-zx^2}{1+tg^2x^2}x$
13	$b=(z-x)\frac{y-\ln z}{1+(y-x)^2}$	28	$b=\frac{\lg\left(y+\sqrt{z+x^2}\right)}{y+x^2}$
14	$b=y^z+\sqrt{\lvert x\rvert+\lvert y\rvert}$	29	$b=\frac{x^2+4}{\sin^2 z^2+x/2}y$
15	$b=\frac{\lg\left(\sqrt{x}+\sqrt{y}+2\right)}{\lvert 2z\rvert}$	30	$b=\frac{\sin x+\sqrt{\lvert z-y\rvert}}{y(x-2)+x^2}$

Задание 3

В соответствии со своим вариантом вычислить значение функции с помощью разложения в ряд Тейлора. Задание реализовать с использованием реурсии.

Вариант	Разложение функции в ряд Тейлора
1.	$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right)$ $\lvert x\rvert>1$
2.	$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots$ $\lvert x\rvert<\infty$
3.	$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ \ \ldots$ $\lvert x\rvert<\infty$
4.	$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}\ -\frac{x^4}{4}-\ \ldots$ $-1<x\le 1$
5.	$\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right)$ $\lvert x\rvert<1$
6.	$\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-\left(x+\frac{x^2}{2}+\frac{x^3}{3}+\ldots\right)$ $-1\le x\le 1$
7.	$arcctg\ \ x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}-\ \ \ldots$ $\lvert x\rvert\le 1$
8.	$arctg\ x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots$ $x>1$
9.	$arctg\ \ x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ \ \ldots$ $\lvert x\rvert\le 1$
10.	$Arth\ \ x=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ \ \ldots$ $\lvert x\rvert<1$
11.	$Arth\ x=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots$ $\lvert x\rvert>1$
12	$arctg\ x=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots$ $x<$
13.	$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots$ $\lvert x\rvert<\infty$
14.	$\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$ $\lvert x\rvert<\infty$
15.	$\frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}-\ldots$ $\lvert x\rvert<\infty$

Требование по содержанию отчета

В отчете должны быть отображены следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задания.
4. Листинг кода сценария с комментариями.
5. Результат выполнения скрипта.
6. Выводы.

Контрольные вопросы для защиты:

1. Что представляет из себя функция?
2. Как создать пользовательскую функцию?
3. Что такое формальные и фактические параметры?
4. Чем отличается описание функций, не возвращающих и возвращающих значения?
5. В чем особенность использования функции как переменной?
6. В чем особенность использования функции как объект?
7. Для чего используется функция eval?
8. Чем отличаются глобальные и локальные переменные?
9. Как задать значение параметра функции по умолчанию?
10. Для чего используется массив arguments?

Лабораторная работа № 4
Работа с массивами в JavaScript

Цель работы: изучить возможности JavaScript для создания и обработки массивов.

Краткие теоретические сведения:

Для работы с наборами данных предназначены массивы. Для создания массива применяется литерал массива или конструкция new Array():

```
let array_name1 = [item1, item2, ...];
let array_name2 = new Array([item1, item2, ...]);
```

Для повышения производительности и читабельности программного кода рекомендуется использовать литерал массива.

Для получения доступа к элементам массива используется индекс. Индексация элементов начинается с нуля:

```
let cars = ["Saab", "Volvo", "BMW"];
console.log(cars[0]); // Saab
```

Индекс используется как для чтения, так и для записи элемента массива. Причем в отличие от других языков, таких как C# или Java, можно установить элемент, который изначально не установлен:

```
let cars = ["Saab", "Volvo", "BMW"];
cars[10] = "Toyota"
console.log(cars[10]); // Toyota
console.log(cars[3]); // undefined
```

Также стоит отметить, что в отличие от ряда языков программирования в JavaScript массивы не являются строго типизированными, один массив может хранить данные разных типов:

```
var objects = ["Tom", 12, true, 3.14, false];
console.log(objects.toString());
```

Массивы могут быть одномерными и многомерными. Каждый элемент в многомерном массиве может представлять собой отдельный массив.

```
const students = [
      ["Иванов", 20, 5.5],
      ["Петров", 18, 8.2],
      ["Сидоров", 21, 7.8]
    ];
students[0][1] = 19; // присваиваем отдельное значение
console.log(students[0][1]); // 56
```

В языке JavaScript все свойства и методы обработки массивов собраны в глобальном объекте Array.prototype, от которого автоматически наследуются все создаваемые массивы.

Все массивы обладают свойством length, которой устанавливает или возвращает количество элементов в массиве:

```
let cars = ["Saab", "Volvo", "BMW"];
console.log(cars.length); // 3
cars.length = 5;
console.log(cars[4]); // undefined
```

Методы массивов

Для добавления и удаления элементов массива используются следующие методы:

- **push(...items)** – добавляет элементы в конец,
- **pop()** – извлекает элемент из конца,
- **shift()** – извлекает элемент из начала,
- **unshift(...items)** – добавляет элементы в начало.

Для удаления эедемента массива можно использовать оператор **delete**. Однако этот оператор удаляет только значение элеменета с заданным ключом без переиндексации:

```
let cars = ["Saab", "Volvo", "BMW"];
delete cars[1];
console.log(cars.length); // 3
console.log(cars[1]); // undefined
```

Универсальный метод **splice()** используется для добавления, удаления и замены элементов массива:

```
splice(index[, deleteCount, elem1, ..., elemN])
```

Он начинает с позиции index удалять deleteCount элементов и вставлять elem1, ..., elemN на их место. Возвращает массив из удалённых элементов.

Метод **slice()** возвращает новый массив, в который копирует элементы, начиная с индекса start и до end (не включая end). Оба индекса start и end могут быть отрицательными. В таком случае отсчёт будет осуществляться с конца массива:

```
slice([start], [end])
```

Метод **forEach()** позволяет запускать функцию для каждого элемента массива. Его синтаксис:

```
forEach(function(item, index, array) {
  // ... делать что-то с item
});
```

Функция обратного вызова (callback) вызывается по очереди для каждого элемента массива и принимает следующие парметра:

- **item** – очередной элемент;
- **index** – его индекс;
- **array** – сам массив.

Для поиска элементов в массиве используются следующие методы:

- **indexOf(item, from)** ищет item, начиная с индекса from, и возвращает индекс, на котором был найден искомый элемент, в противном случае -1.
- **lastIndexOf(item, from)** – то же самое, но ищет справа налево.
- **includes(item, from)** – ищет item, начиная с индекса from, и возвращает true, если поиск успешен.

Методы **find**, **findIndex** и **filter** в качестве условия поиска используют функцию-предикат:

```
let result = arr.find(function(item, index, array) {
  // если true - возвращается текущий элемент и перебор прерывается
  // если все итерации оказались ложными, возвращается undefined
});

let result = arr.findIndex(function(item, index, array) {
  // если true - возвращается индекс, на котором был найден элемент, и перебор прерывается
  // если все итерации оказались ложными, возвращается -1
});


let results = arr.filter(function(item, index, array) {
  // если true - элемент добавляется к результату, и перебор продолжается
  // возвращается пустой массив в случае, если ничего не найдено
});
```

Метод **map()** является одним из наиболее полезных и часто используемых. Он вызывает функцию для каждого элемента массива и возвращает массив результатов выполнения этой функции:

```
let result = arr.map(function(item, index, array) {
  // возвращается новое значение вместо элемента
});
```

Метод **sort(fn)** сортирует массив на месте, меняя в нём порядок элементов. Он возвращает отсортированный массив, но обычно возвращаемое значение игнорируется, так как изменяется сам массив.

Полный список методов есть в справочнике MDN.

Задание к лабораторной работе

Задание 1

В файле со скриптом lab4-1.html:

1. Создайте массив treug «треугольных» чисел, т.е. чисел вида n(n+1)/2 (где n=1,2,… 10) и выведите значения этого массива на экран в строку (через 2 пробела).
2. Создайте массив kvd квадратов натуральных чисел от 1 до 10, выведите значения этого массива на экран в строку.
3. Объедините эти 2 массива в массив rez, выведите результат на экран.
4. Отсортируйте массив rez в обратном порядке, выведите результат на экран.

Задание 2

В файле со скриптом lab4-2.html:

1. Сформируйте одномерный массив (не менее 15 элементов, для генерации значений массива можно воспользоваться функцией random объекта Math), выведите значения этого массива на экран в строку.
2. Выполните задание в соответствии с вариантом, не используя встроенные функции.

Вариант	Задание
1	Найти максимальный элемент и поменять его местами с последним элементом массива
2	Найти минимальный элемент и поменять его местами с предыдущим элементом массива
3	Найти минимальный элемент и поменять его местами с последующим элементом массива
4	Найти максимальный элемент и поменять его местами с шестым элементом массива
5	Найти максимальный элемент, присвоить его значение последнему элементу массива, а вместо максимального числа записать - 1
6	Найти минимальный элемент, присвоить его значение первому элементу массива, а вместо минимального элемента записать число 9999

Вариант	Задание
7	Найти минимальный элемент и поменять его местами с третьим элементом массива
8	Найти минимальный элемент и заменить его на сумму первого и последнего элементов
9	Найти максимальный элемент и поменять его местами с предпоследним элементом массива
10	Найти минимальный элемент, присвоить его значение последнему элементу массива, а вместо минимального элемента записать значение 3N
11	Найти минимальный элемент и поменять его местами с элементом массива, номер которого задан
12	Найти максимальный элемент и поменять его местами со вторым элементом массива
13	Найти минимальный элемент и поменять его местами с последним элементом массива
14	Найти минимальный элемент и поменять его местами с предпоследним элементом массива
15	Найти минимальный элемент и поменять его местами с третьим элементом массива

Задание 3

В файле со скриптом lab4-3.html:

1. Сформируйте одномерный массив (не менее 15 элементов, для генерации значений массива можно воспользоваться функцией random объекта Math), выведите значения этого массива на экран в строку.
2. Выполните задание в соответствии с вариантом, используя встроенные методы массивов.

Вариант	Задание
1	Найти количество чисел, принадлежащих промежутку [a,b], и сумму чисел, стоящих на местах, кратных 3.
2	Найти сумму чисел, меньших заданного D, и количество чисел, стоящих на четных местах и больших заданного C.
3	Найти произведение всех чисел, стоящих на местах, кратных 4, и количество чисел, небольших заданного A.
4	Найти количество чисел, меньших заданного X, и произведение всех отрицательных чисел, стоящих на нечетных местах.

Вариант	Задание
5	Найти количество чисел, непринадлежащих промежутку (X,Y], и сумму отрицательных чисел, стоящих на четных местах.
6	Найти количество неотрицательных чисел и определить сумму чисел, стоящих на местах, кратных 3, и неравных заданному F.
7	Найти среднее арифметическое отрицательных чисел и определить количество чисел, по величине больших A и стоящих на четных местах.
8	Найти среднее арифметическое положительных чисел, стоящих на нечетных местах, и количество чисел, меньших заданного B.
9	Найти среднее арифметическое чисел, принадлежащих промежутку [A,B), и количество положительных чисел, стоящих на местах, кратных 4.
10	Найти среднее арифметической чисел, неравных заданному C, и произведение неположительных чисел, стоящих на четных местах.
11	Найти среднее арифметическое чисел, больших заданного D и стоящих на нечетных местах, и определить количество чисел, небольших заданного F.
12	Найти среднее арифметическое чисел, непопадающих в промежуток [A,B], и количество положительных чисел, стоящих на местах, кратных 3.
13	Найти среднее арифметическое ненулевых чисел и количество чисел, по величине небольших A и стоящих на четных местах.
14	Вычислить произведение чисел, принадлежащих промежутку (A,B], и количество отрицательных чисел, стоящих на местах, кратных 3.
15	Найти среднее арифметическое положительных чисел, стоящих на нечетных местах, и произведение чисел, меньших заданного C.

Задание 4

В файле со скриптом lab4-4.html:

1. Создайте массивы
   fruits=[‘apple’,’pineapple’,’mango’,’melon’,’grape’];
   citrus=[‘orange’,’lemon’,’lime’].
2. Последовательно, в каждой отдельной строчке, выведите:
   а) массив fruits;
   б) массив citrus;
   в) объединенный массив fruits;
   г) массив fruits, в котором после mango вставлены: pear, cherry, plum, raspberry, strawberry;
   д) массив fruits, в котором удалены последние 3 элемента;

е) выведите отсортированный в алфавитном порядке массив fruits;

ж) выведите массив fruits в обратном алфавитном порядке;

з) выведите массив fruits, отсортированный по длине слов.

Требование по содержанию отчета

В отчете должны быть отображены следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задания.
4. Листинг кода сценария с комментариями.
5. Результат выполнения скрипта.

68. Выводы.

Контрольные вопросы для зашиты:

1. Что такое массив?
2. Какие массивы используются в JavaScript?
3. Как индексируются элементы массив?
4. Как создается многомерный массив?
5. Как просмотреть структуру и значения элементов массива?
6. Как вывести значения элементов массива в окно браузера?
7. Как отсортировать массив по возрастанию и убыванию?
8. Какие методы объекта Array вы знаете?
9. Как добавить элементы в массив?
10. Как объединить несколько массивов?
11. Как удалить элементы массива?

Лабораторная работа № 5
Работа со строками в JavaScript

Цель работы: получить навыки обработки символьной информации в JavaScript.

Краткие теоретические сведения:

В JavaScript любые текстовые данные являются строками. Не существует отдельного типа «символ», который есть в ряде других языков.

Внутренний формат для строк — всегда UTF-16, вне зависимости от кодировки страницы.

Для задания строк в JavaScript есть разные типы кавычек. Строку можно создать с помощью одинарных, двойных либо обратных кавычек:

```
let single = 'single-quoted';
let double = "double-quoted";
let backticks = `backticks`;
```

Одинарные и двойные кавычки работают, по сути, одинаково, а если использовать обратные кавычки, то в такую строку мы сможем вставлять произвольные выражения, обернув их в ${…}:

```
function sum(a, b) {
  return a + b;
}

console.log(`1 + 2 = ${sum(1, 2)}.`); // 1 + 2 = 3.
```

Для определения длины строки используется свойство length.

Чтобы получить символ, который занимает позицию pos, можно использовать квадратные скобки: [pos], а также можно использовать метод **charAt(pos)**. Первый символ занимает нулевую позицию:

```
let str = `Hello`;
console.log(str.length); //5
// получаем первый символ
```

```
console.log(str[0]); // H
console.log(str.charAt(0)); // H
```

Следует обратить внимание на то, что содержимое строки в JavaScript нельзя изменить.

```
let str = `Hello`;
console.log(str); // Hello
str[0] = "J";
console.log(str); // Hello
```

Методы **toLowerCase()** и **toUpperCase()** меняют регистр символов:

```
alert( 'Interface'.toUpperCase() ); // INTERFACE
alert( 'Interface'.toLowerCase() ); // interface
```

Для поиска подстроки используются следующие методы:

- **indexOf(substr, pos)** ищет подстроку substr в строке str, начиная с позиции pos, и возвращает позицию, на которой располагается совпадение, либо -1 при отсутствии совпадений;
- **lastIndexOf(substr, position)** ищет подстроку с конца строки к её началу;
- **includes(substr, pos)** возвращает true, если в строке str есть подстрока substr, либо false, если нет;
- **startsWith(substr)**, **endsWith(substr)** проверяют, соответственно, начинается ли и заканчивается ли строка определённой строкой.

Для получения подстроки в JavaScript есть 3 метода:

- **slice(start [, end])** возвращает часть строки от start до (не включая) end, если end отсутствует, slice возвращает символы до конца строки;
- **substring(start [, end])** возвращает часть строки между start и end;
- **substr(start [, length])** возвращает часть строки от start длины length.

Для сравнение строк используются стандартные операторы сравнения, строки сравниваются посимвольно в алфавитном порядке.

Задание к лабораторной работе:

Задание 1. В соответствии со своим вариантом написать Java-скрипт в файле **lab5-1.html**, выполняющий следующие действия.

Вариант	Задание
01	Подсчитать общее количество символов '+ ' и '-' и заменить каждый символ ';' на ',' и '.'
02	После каждого символа ',' вставить пробел и подсчитать количество букв 'A' и 'B' отдельно
03	Заменить символ '*' на '++' и подсчитать общее количество букв 'F' и 'D'
04	Подсчитать количество букв 'C' и 'D' отдельно и заменить каждую пару символов '**' на символ ' '
05	После каждого символа '!' вставить символ 'I' и подсчитать общее количество цифр в строке
06	Удалить каждую пару символов 'PQ' и подсчитать общее количество символов '.' и ',' в строке
07	Подсчитать количество пар символов '+ -' и заменить каждый символ '*' на '/-'
08	После каждой цифры вставить такую же цифру и подсчитать количество пар 'AC' в строке
09	Удалить каждый символ 'A', стоящий после ',' и подсчитать количество пар 'BC'
10	Подсчитать количество символов '.' , стоящих перед пробелом, и заменить каждую пару символов 'ST' на символ 'P'
11	После каждого символа 'A' вставить пробел и подсчитать количество символов 'B', стоящих между знаками '+' и '-'
12	Удалить каждый символ '?', стоящий после ';', и подсчитать общее количество символов 'o и 'O'
13	Подсчитать количество символов '+', стоящих между'A' и 'B', заменить каждый символ '0' на 'OO'
14	В каждую пару символов 'AB' вставить символ '*', подсчитать, сколько раз в строке символ 'I' стоит перед '2'
15	Вставить символ ';' после каждого символа 'A' и после каждого 'B', подсчитать, сколько раз символ'C' встречается между символами '*' и '/'

Задание 2. В соответствии со своим вариантом написать Java-скрипт в файле **lab5-2.html**, выполняющий следующие действия.

Вариант	Задание
01	После каждого слова поставить запятую. Подсчитать количество слов, в которых есть буква 'п'
02	Подсчитать количество букв в третьем слове. Поменять местами первое и последнее слова
03	Во втором слове после каждой буквы вставить пробел. Определить количество слов, которые заканчиваются на 'е'
04	Перед первой буквой каждого слова вставить символ '*'. Определить количество слов, в которых нет ни одной буквы 'г'
05	Для первого слова указать, сколько букв 'и' в нем содержится. Переставить первое слово в конец строки
06	Определить количество слов, начинающихся с буквы 'А'. После каждой буквы предпоследнего слова вставить символ '*'
07	Подсчитать количество букв во втором слове. Каждое слово заключить в кавычки.
08	Подсчитать количество слов, длина которых больше 5. Удалить пробелы, стоящие между первым и вторым словом.
09	Определить количество слов, в которых буква 'и' встречается хотя бы один раз. Поменять местами первое и второе слово
10	Третье слово строки поставить после первого. Определить количество слов, в которых первая и последняя буквы совпадают
11	Определить количество слов, вторая буква которых 'р'. Удалить последнюю букву в каждом слове
12	Подсчитать количество букв в предпоследнем слове. В каждом слове поменять местами первую и последнюю буквы
13	Перед каждой буквой третьего слова поставить '-'. Определить количество слов, после которых один пробел
14	После последней буквы каждого слова вставить точку. Для пятого слова указать, сколько букв 'И' в нем содержится.
15	Удалить все пробелы из строки, кроме тех, которые стоят между первым и вторым словом. Определить количество слов, которые по длине меньше 3

Задание 3. Написать простейшую программу шифрования. Программа каждую букву заменяет на следующую за ней в алфавите («я» переходит в «а»).

Задание 4. Написать программу, которая в каждом слове перемешивает буквы местами за исключением первой и последней буквы слова.

Требования по содержанию отчета

Отчет должен включать следующие пункты:
1. Титульный лист.
2. Цель работы.
3. Задания на лабораторную работу.
4. Листинги программного кода.
5. Результаты выполнения скриптов.
6. Выводы.

Контрольные вопросы для защиты:

1. Как выводится текстовая информация с использованием JavaScript?
2. Как осуществитьт конкатенация строк?
3. Какая функция осуществляет поиск заданного элемента в строке?
4. Как определить длину строки?
5. Как из исходной строки выделить подстроку?
6. Как выполнить замену подстроки в исходной строке?
7. Для чего используются регулярные выражения?
8. Какие функции используются в JavaScript для работы с регулярными выражениями?
9. Как вывести подстроки в соответствии с шаблоном?

Лабораторная работа № 6
Работа с объектами

Цель работы: получить навыки создания объектов, доступа к полям и методам, ознакомится с прототипным наследованием.

Краткие теоретические сведения

На сегодняшний день объектно-ориентированное программирование (ООП) на сегодняшний день является одной из господствующих парадигм в разработке приложений. JavaScript предоставляет возможности ООП, но имеет некоторые особенности.

Основным понятием ООП является объект – сложная комплексная структура, объединяющая в себе данные и методы их обработки.

Для работы с подобными структурами в JavaScript используются объекты. Каждый объект может хранить свойства, которые описывают его состояние, и методы, которые описывают его поведение.

Создание нового объекта

Есть несколько способов создания нового объекта.

Первый способ заключается в использовании конструктора Object:

```
const user = new Object();
```

В данном случае объект называется user. Он определяется также, как и любая обычная переменная. Выражение new Object() представляет вызов конструктора – функции, создающей новый объект. Для вызова конструктора применяется оператор new. Вызов конструктора фактически напоминает вызов обычной функции.

Второй способ создания объекта представляет использование фигурных скобок – объектного литерала:

```
const user = {};
```

На сегодняшний день более распространенным является второй способ.

Свойства и методы объекта

После создания объекта мы можем определить в нем свойства. Чтобы определить свойство, надо после названия объекта через точку указать имя свойства и присвоить ему значение, или использовать альтернативный способ с помощью синтаксиса массивов:

```
const user = {};
user.name = "Alex";
user.age = 21;
user["group"] = "IP-31"
```

В данном случае объявляются три свойства name, age и group, которым присваиваются соответствующие значения. После этого мы можем использовать эти свойства, например, вывести их значения в консоли:

Методы объекта определяют его поведение или действия, которые он производит. Методы представляют собой функции. Например, определим метод, который бы выводил имя и возраст человека:

```
user.info = function(){
   console.log(user.name);
   console.log(user.age);
};
```

Также свойства и методы могут определяться непосредственно при определении объекта:

```
var user = {

   name: "Alex",
   age: 21,
   group: "IP-31",
   info: function(){
      console.log(this.name);
      console.log(this.age);
   }
};
```

Чтобы обратиться к свойствам или методам объекта внутри этого объекта, используется ключевое слово this. Оно означает ссылку на текущий объект.

Конструктор объектов

Кроме создания новых объектов JavaScript предоставляет нам возможность создавать новые типы объектов с помощью конструкторов.

Конструктор позволяет определить новый тип объекта. Тип представляет собой абстрактное описание или шаблон объекта. Определение типа может состоять из функции конструктора, методов и свойств.

Для начала определим конструктор:

```
function User(pName, pAge) {
    this.name = pName;
    this.age = pAge;
    this.info = function(){
        document.write("Имя: " + this.name + "; возраст: " + this.age +
"<br/>");
    };
}
```

Конструктор – это обычная функция за тем исключением, что в ней мы можем установить свойства и методы. Для установки свойств и методов используется ключевое слово this. В данном случае устанавливаются два свойства name и age и один метод info. Как правило, названия конструкторы в отличие от названий обычных функций начинаются с большой буквы.

После этого в программе мы можем определить объект типа User и использовать его свойства и методы:

```
const alex = new User("Alex", 21);
console.log(alex.name); // Alex
alex.info();
```

Оператор **instanceof** позволяет проверить, с помощью какого конструктора создан объект. Если объект создан с помощью определенного конструктора, то оператор возвращает true:

```
const alex = new User("Alex", 21);
console.log(alex instanceof User); // true
```

Расширение объектов. Prototype

Кроме непосредственного определения свойств и методов в конструкторе мы также можем использовать свойство prototype. Каждая функция имеет свойство prototype, представляющее прототип функции. То есть свойство User.prototype представляет прототип объектов User. И любые свойства и методы, которые будут определены в User.prototype, будут общими для всех объектов User.

```
function User(pName, pAge) {
 this.name = pName;
 this.age = pAge;
 this.info = function() {
   document.write(
     "Имя: " + this.name + "; возраст: " + this.age + "<br/>"
   );
 };
}

User.prototype.hello = function() {
 document.write(this.name + " говорит: 'Привет!'<br/>");
};
User.prototype.group = "IP-31";

const alex = new User("Alex", 21);
alex.hello();
const max = new User("Max", 21);
max.hello();
console.log(max.group);
```

Классы

С внедреием стандарта ES2015 (ES6) в JavaScript появился новый способ определения объектов – с помощью классов. Класс представляет описание объекта, его состояния и поведения, а объект является конкретным воплощением или экземпляром класса.

Для определения класса используется ключевое слово class.

После этого мы можем создать объекты класса с помощью конструктора:

```
class Person{}

const alex = new Person();
const max = new Person();
```

По умолчанию классы имеют один конструктор без параметров. Поэтому в данном случае при вызове конструктора в него не передается никаких аргументов. Однако есть возможность определить в классе свои конструкторы. Также класс может содержать свойства и методы:

```
class Person{
    constructor(name, age){
        this.name = name;
        this.age = age;
    }
    info(){
        console.log(this.name, this.age);
    }
}

const tom = new Person("Alex", 21);
alex.info(); // Tom 34
console.log(alex.name); // Tom
```

Конструктор определяется с помощью метода с именем constructor. По сути это обычный метод, который может принимать параметры. Основная цель конструктора – инициализировать объект начальными данными. И в данном случае в конструктор передаются два значения – для имени и возраста пользователя.

Для хранения состояния в классе определяются свойства. Для их определения используется ключевое слово this. В данном случае в классе два свойства: name и age.

Поведение класса определяют методы. В данном случае определен метод info(), который выводит значения свойств на консоль.

Наследование

Одни классы могут наследоваться от других. Наследование позволяет сократить объем кода в классах-наследниках. Например, определим следеющие классы:

```
class Person{
    constructor(name, age){
        this.name = name;
        this.age = age;
    }
    info(){
        console.log(this.name, this.age);
    }
}

class Employee extends Person{
    constructor(name, age, company){
        super(name, age);
        this.company = company;
    }
    info(){
        super.info();
        console.log("Employee in", this.company);
    }
    work(){
        console.log(this.name, "is hard working");
    }
}

const bob = new Person("Bob", 34);
const bill = new Employee("Bill", 64, "Microsoft");
bob.info();
```

```
bill.info();
bill.work();
```

Для наследования одного класса от другого в определении класса применяется оператор extends, после которого идет название базового класса. То есть в данном случае класс Employee наследуется от класса Person. Класс Person еще называется базовым классом, классом-родителем, суперклассом, а класс Employee – классом-наследником, подклассом, производным классом.

Производный класс, как и базовый, может определять конструкторы, свойства, методы. Вместе с тем с помощью слова super производный класс может ссылаться на функционал, определенный в базовом. Например, в конструкторе Employee можно вручную не устанавливать свойства name и age, а с помощью выражения super(name, age); вызвать конструктор базового класса и тем самым передать работу по установке этих свойств базовому классу.

Статические методы

Статические методы вызываются для всего класса вцелом, а не для отдельного объекта. Для их определения применяется оператор static. Например:

```
class Person{
    constructor(name, age){
        this.name = name;
        this.age = age;
    }
    static nameToUpper(person){
        return person.name.toUpperCase();
    }
    info(){
        console.log(this.name, this.age);
    }
}
const tom = new Person("Tom Soyer", 34);
let personName = Person.nameToUpper(tom);
console.log(personName);       // TOM SOYER
```

В данном случае определен статический метод nameToUpper(). В качестве параметра он принимает объект Person и переводит его имя в верхний регистр. Поскольку статический метод относится классу вцелом, а не к объекту, то мы НЕ можем использовать в нем ключевое слово this и через него обращаться к свойствам объекта.

Практическая часть

Задание 1. В соответствии со своим вариантом определить объект, с заданными полями и методом вывода информации об объекте. Объект создается с помощью объектного литерала.

Вариант	Описание	Свойство
01	Процессор	Тактовая частота Количество ядер Разрядность процессора Техпроцесс
02	Память	Тип памяти Объем Скорость чтения Скорость записи
03	Корпус	Мощность блока питания Тип корпуса Форм-фактор материнской платы Охлаждение
04	Жесткий диск	Тип накопителя Объем Интерфейс Скорость вращения шпинделя
05	SSD диск	Объем Интерфейс Тип микросхем Flash Контроллер
06	Оптический носитель	Тип установки Тип Интерфейс подключения Максимальная скорость чтения Максимальная скорость чтения

Вариант	Описание	Свойство
07	Клавиатура	Тип Технология переключения Назначение Интерфейс подключения
08	Мышь	Интерфейс подключения Назначение Тип сенсора Модель сенсора Максимальное разрешение сенсора
09	Звуковая карта	Аналоговые выходы Аналоговые входы Микрофонные входы Тип Количество каналов Интерфейс подключения
10	Видеокарта	Интерфес Производитель графического процессора Графический процессор Частота графического процессора Видеопамять (объем)
11	Блок бесперебойного	Тип AVR Мощность Время автономного питания
12	Планшет	Диагональ экрана Процессор Разрешение экрана Операционная система Оперативная память Внутренняя память
13	Электронная книга	Размер экрана Тип экрана Разрешение экрана Флэш-память
14	Акустика	Тип Усилитель Максимальная мощность Материал корпуса

Вариант	Описание	Свойство
15	Фитнес-браслет	Объем оперативной памяти Поддерживаемые платформы Датчики
16	Аккумулятор	Тип аккумулятора Емкость Напряжение Полярность
17	Мобильный телефон	Тип Операционная система Размер экрана Разрешение экрана Оперативная память
18	Радиоприемник	Питание Тип Выходная мощность звука Тип тюнера Модуляция
19	Телевизор	Тип Диагональ Частота Разрешение
20	Фотоаппарат	Тип камеры Тип матрицы Размер экрана Количество точекматрицы Максимальное разрешение видео
21	Видеокамера	Разрешение видеосъемки Формат сжатия видео Размер экрана Количество точек матрицы
22	Принтер	Тип Формат Количество цветов Технология печати Скорость печати
23	Игровая приставка	Тип Объем накопителя Максимальное разрешение в играх Оптический привод

Вариант	Описание	Свойство
24	IP камера	Тип матрицы Конструкция Стандарты беспроводной связи Сетевой интерфейс
25	Портативная	Стандарт Количество каналов Дальность радиосвязи Количество радиостанций в комплекте
26	Модем	Тип Интерфейс подключения Поддержка сотовой связи Поддержка карт памяти
27	Монитор	Диагональ Соотношение сторон Разрешение Частота обновления экрана
28	Велосипед	Класс Материал рамы Тип трансмиссии Диаметр колес
29	Микроволновая печь	Тип Объем Потребляемая мощность Выходная мощность микроволн
30	Стиральная машина	Загрузка белья Максимальная загрузка Класс энергопотребления Класс стирки Максимальная скорость отжима

Задание 2. Написать функцию-конструктор для создания объектов в соответствии с вариантом. Определить геттеры и сеттеры для получения доступа к свойствам. В прототип объектов добавить свойство – дата выхода на рынок, метод отображения информации об объекте.

Задание 3. Описать класс, описывающий объекты в соответствии с вариантом. Предусмотреть конструктор с параметрами, геттеры и сеттеры для получения доступа к свойствам, метод метод отображения информации об объекте.

Задание 4. Описать класс, являющийся наследником класса задаания 3. В производном классе добавить свойства: дата выхода на рынок, стоимость. Переопределить метод вывода информации об объекте.

Задание 5. На основе разработанного класса задания 4 создать массив объектов. Определить объекты с максимальной и минимальной стоимостью, суммарную стоимость всех объектов, вычислить среднюю стоимость объекта, подсчитать количество объектов со стоимостью выше средней. Для выполнения задания использовать встроенные методы массивов и объект Math.

Требования по содержанию отчета

Отчет должен включать следующие пункты:
1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг программного кода с комментариями.
5. Результаты выполнения скрипта.
6. Выводы.

Контрольные вопросы для защиты работ:

1. Как создать объект в JavaScript?
2. Что такое функция конструктор?
3. Что такое Prototype?
4. Как создать метод объекта?
5. Как перебрать свойства объекта?
6. Как определить собственные свойства объекта?
7. Как добавить, удалить свойство?
8. Как определить класс?
9. Как определить класс, наследуемый от заданного?
10. Как обратиться к функционалу базового класса из производного?

Лабораторная работа № 7
Объектная модель документа (DOM)

Цель работы: изучить возможности взаимодействия JavaScript с элементами страницы и объектами DOM.

Краткие теоретические сведения:

Одой из ключевых задач JavaScript является взаимодействие с пользователем и манипуляция элементами веб-страницы. Для JavaScript веб-страница доступна в виде объектной модели документа (document object model) или сокращенно DOM.

В соответствии с объектной моделью документа, каждый HTML-тег является объектом. Вложенные теги являются дочерними объектами родительского элемента. Текст, который находится внутри тега, также является объектом.

Все эти объекты представляют собой узлы в иерархической структуре DOM и доступны при помощи JavaScript для изменения страницы.

Существует 12 типов узлов. Но на практике в основном используются 4 из них:

- **Element**: html-элемент;
- **Document**: корневой узел html-документа;
- **Comment**: элемент комментария;
- **Text**: текст элемента.

У DOM-узлов есть свойства и методы, которые позволяют выбирать любой из элементов, изменять, перемещать их на странице и многое другое.

Объект document

Для работы со структурой DOM в JavaScript предназначен объект document, который определен в глобальном объекте window. Объект document предоставляет ряд свойств и методов для управления элементами страницы.

Для поиска элементов на странице применяются следующие методы:

- **getElementById(value)**: выбирает элемент, у которого атрибут id равен value;
- **getElementsByTagName(value)**: выбирает все элементы, у которых тег равен value;
- **getElementsByClassName(value)**: выбирает все элементы, которые имеют класс value;
- **querySelector(value)**: выбирает первый элемент, который соответствует css-селектору value;
- **querySelectorAll(value)**: выбирает все элементы, которые соответствуют css-селектору value.

Кроме методов объект document позволяет обратиться к определенным элементам веб-страницы через свойства:

documentElement: предоставляет доступ к корневому элементу <html>;

body: предоставляет доступ к элементу <body> на веб-странице;

images: содержит коллекцию всех объектов изображений (элементов img);

links: содержит коллекцию ссылок - элементов <a> и <area>, у которых определен атрибут href;

anchors: предоставляет доступ к коллекции элементов <a>, у которых определен атрибут name;

forms: содержит коллекцию всех форм на веб-странице.

Эти свойства не предоставляют доступ ко всем элементам, однако позволяют получить наиболее часто используемые элементы на веб-странице.

Объект Node. Навигация по DOM

Каждый отдельный узел, будь то html-элемент, его атрибут или текст, в структуре DOM представлен объектом Node. Этот объект предоставляет ряд свойств, с помощью которых мы можем получить информацию о данном узле:

- **childNodes**: содержит коллекцию дочерних узлов;
- **firstChild**: возвращает первый дочерний узел текущего узла;
- **lastChild**: возвращает последний дочерний узел текущего узла;

- **previousSibling**: возвращает предыдущий элемент, который находится на одном уровне с текущим;
- **nextSibling**: возвращает следующий элемент, который находится на одном уровне с текущим;
- **ownerDocument**: возвращает корневой узел документа;
- **parentNode**: возвращает элемент, который содержит текущий узел;
- **nodeName**: возвращает имя узла;
- **nodeType**: возвращает тип узла в виде числа;
- **nodeValue**: возвращает или устанавливает значение узла в виде простого текста;

Создание, добавление и удаление элементов веб-станицы

Для создания элементов объект document имеет следующие методы:

- **createElement(elementName)**: создает элемент html, тег которого передается в качестве параметра. Возвращает созданный элемент.
- **createTextNode(text)**: создает и возвращает текстовый узел. В качестве параметра передается текст узла.

Для добавления элементов можно использовать один из методов объекта Node:

- **appendChild(newNode)**: добавляет новый узел newNode в конец коллекции дочерних узлов;
- **insertBefore(newNode, referenceNode)**: добавляет новый узел newNode перед узлом referenceNode.

Иногда элементы бывают довольно сложными по составу, и гораздо проще их скопировать, чем с помощью отдельных вызовов создавать из содержимое. Для копирования уже имеющихся узлов у объекта Node можно использовать метод **cloneNode().** В метод cloneNode() в качестве параметра передается логическое значение: если передается true, то элемент будет копироваться со всеми дочерними узлами; если передается false - то копируется без дочерних узлов.

Для удаления элемента вызывается метод **removeChild()** объекта Node. Этот метод удаляет один из дочерних узлов.

Для замены элемента применяется метод **replaceChild(newNode, oldNode)** объекта Node. Этот метод в качестве первого параметра принимает новый элемент, который заменяет старый элемент oldNode, передаваемый в качестве второго параметра.

Объект Element. Управление элементами

Кроме методов и свойств объекта Node в JavaScript мы можем использовать свойства и методы объектов Element. Важно не путать эти два объекта: Node и Element. Node представляет все узлы веб-станицы, в то время как объект Element представляет непосредственно только html-элементы. То есть объекты Element - это фактически те же самые узлы – объекты Node, у которых тип узла (свойство nodeType) равно 1.

Основные свойства объекта Element:

- **nodeType** позволяет узнать тип DOM-узла. Его значение – числовое: 1 для элементов, 3 для текстовых узлов, и т.д.
- **tagName** возвращает название тега (записывается в верхнем регистре, за исключением XML-режима).
- **innerHTML** устанавливает или получает внутреннее HTML-содержимое элемента.
- **outerHTML** устанавливает или получает полный HTML-код элемента. Запись в elem.outerHTML не меняет elem. Вместо этого она заменяет его во внешнем контексте.
- **innerText, textContent** устанавливает или получает текст внутри элемента за вычетом всех <тегов>.
- **hidden** скрывает или отображает элемент. Когда значение установлено в true, делает то же самое, что и CSS display:none.
- **offsetWidth** определяет ширину элемента в пикселях. В ширину включается граница элемента.
- **offsetHeight** определяет высоту элемента в пикселях. В высоту включается граница элемента.
- **clientWidth** определяет ширину элемента в пикселях без учета границы.
- **clientHeight** определяют высоту элемента в пикселях без учета границы.

Среди методов объекта Element можно отметить методы управления атрибутами:

- **hasAttribute(name)** проверяет на наличие атрибута.
- **getAttribute(attr)** возвращает значение атрибута attr.
- **setAttribute(attr, value)** устанавливает для атрибута attr значение value. Если атрибута нет, то он добавляется.
- **removeAttribute(attr)** удаляет атрибут attr и его значение.

Изменение стиля элементов

Для работы со стилевыми свойствами элементов в JavaScript применяются, главным образом, два подхода:

- изменение свойства style;
- изменение значения атрибута class.

Свойство **style** представляет сложный объект для управления стилем и напрямую сопоставляется с атрибутом style html-элемента. Этот объект содержит набор свойств CSS.

Свойства объекта style совпадает со свойством css. Однако ряд свойств css в названиях имеют дефис, например, font-family. В JavaScript для этих свойств дефис не употребляется. Только первая буква, которая идет после дефиса, переводится в верхний регистр: fontFamily.

С помощью свойства **className** можно установить атрибут class элемента html. Благодаря использованию классов не придется настраивать каждое отдельное свойство css с помощью свойства style. Но при этом надо учитывать, что прежнее значение атрибута class удаляется. Поэтому, если нам надо добавить класс, надо объединить его название со старым классом в виде конкатенации строк.

Для управления множеством классов гораздо удобнее использовать свойство **classList**. Это свойство представляет объект, реализующий следующие методы:

- add(className): добавляет класс className.
- remove(className): удаляет класс className.
- toggle(className): переключает у элемента класс на className. Если класса нет, то он добавляется, если есть, то удаляется.

События

Для взаимодействия с пользователем в JavaScript определен механизм событий. Например, когда пользователь нажимает кнопку, то возникает событие нажатия кнопки.

В JavaScript есть следующие типы событий:

- события мыши (перемещение курсора, нажатие мыши и т.д.);
- события клавиатуры (нажатие или отпускание клавиши клавиатуры);
- события жизненного цикла элементов (например, событие загрузки веб-станицы);
- события элементов форм (нажатие кнопки на форме, выбор элемента в выпадающем списке и т.д.);
- события, возникающие при изменении элементов DOM;
- события, возникающие при касании на сенсорных экранах;
- события, возникающие при возникновении ошибок.

Событию можно назначить обработчик, то есть функцию, которая сработает, как только событие произошло.

Есть несколько способов назначить событию обработчик.

Обработчик может быть назначен прямо в разметке, в атрибуте, который называется on<событие>.

Например, чтобы назначить обработчик события click на элементе input, можно использовать атрибут onclick, вот так:

```
<input value="Нажми меня" onclick="alert('Клик!')" type="button">
```

При клике мышкой на кнопке выполнится код, указанный в атрибуте onclick.

Атрибут HTML-тега – не самое удобное место для написания большого количества кода, поэтому лучше создать отдельную JavaScript-функцию и вызвать в обработчике.

Можно назначать обработчик, используя свойство DOM-элемента on<событие>.

К примеру, elem.onclick:

```
<input id="elem" type="button" value="Нажми меня!">
<script>
  elem.onclick = function() {
```

```
  alert('Спасибо');
 };
</script>
```

Фундаментальный недостаток описанных выше способов назначения обработчика – невозможность повесить несколько обработчиков на одно событие.

Разработчики стандартов предложили альтернативный способ назначения обработчиков при помощи специальных методов addEventListener и removeEventListener.

Синтаксис добавления обработчика:

```
element.addEventListener(event, handler[, options]),
```

где event – имя события, например "click", handler – ссылка на функцию-обработчик, options – дополнительный объект со свойствами:

- once: если true, тогда обработчик будет автоматически удалён после выполнения.
- capture: фаза, на которой должен сработать обработчик, подробнее об этом будет рассказано в главе Всплытие и погружение. Так исторически сложилось, что options может быть false/true, это тоже самое, что {capture: false/true}.
- passive: если true, то указывает, что обработчик никогда не вызовет preventDefault().

Для удаления обработчика следует использовать removeEventListener с аналогичным синтаксисом.

Объект Event

При обработке события браузер автоматически передает в функцию обработчика в качестве параметра объект Event, который инкапсулирует всю информацию о событии. И с помощью его свойств мы можем получить эту информацию:

- **bubbles**: возвращает true, если событие является восходящим;
- **cancelable**: возвращает true, если можно отменить стандартную обработку события;
- **currentTarget**: определяет элемент, к которому прикреплен обработчик события;

- **defaultPrevented**: возвращает true, если был вызван у объекта Event метод preventDefault();
- **eventPhase**: определяет стадию обработки события;
- **target**: указывает на элемент, на котором было вызвано событие;
- **timeStamp**: хранит время возникновения события;
- **type**: указывает на имя события.

С помощью метода **preventDefault()** объекта Event мы можем остановить дальнейшее выполнение события.

События мыши

Одну из наиболее часто используемых событий составляют события мыши:

- **click**: возникает при нажатии указателем мыши на элемент;
- **mousedown**: возникает при нахождении указателя мыши на элементе, когда кнопка мыши находится в нажатом состоянии;
- **mouseup**: возникает при нахождении указателя мыши на элементе во время отпускания кнопки мыши;
- **mouseover**: возникает при вхождении указателя мыши в границы элемента;
- **mousemove**: возникает при прохождении указателя мыши над элементом;
- **mouseout**: возникает, когда указатель мыши выходит за пределы элемента.

Объект Event является общим для всех событий. Однако для разных типов событий существуют также свои объекты событий, которые добавляют ряд своих свойств. Так, для работы с событиями указателя мыши определен объект MouseEvent, который добавляет следующие свойства:

- **altKey**: возвращает true, если была нажата клавиша Alt во время генерации события;
- **button**: указывает, какая кнопка мыши была нажата;
- **clientX**: определяет координату X окна браузера, на которой находился указатель мыши во время генерации события;
- **clientY**: определяет координату Y окна браузера, на которой находился указатель мыши во время генерации события;

- **ctrlKey**: возвращает true, если была нажата клавиша Ctrl во время генерации события;
- **metaKey**: возвращает true, если была нажата во время генерации события метаклавиша клавиатуры;
- **relatedTarget**: определяет вторичный источник возникновения события;
- **screenX**: определяет координату X относительно верхнего левого угла экрана монитора, на которой находился указатель мыши во время генерации события;
- **screenY**: определяет координату Y относительно верхнего левого угла экрана монитора, на которой находился указатель мыши во время генерации события;
- **shiftKey**: возвращает true, если была нажата клавиша Shift во время генерации события

События клавиатуры

Другим распространенным типом событий являются события клавиатуры:

- **keydown**: возникает при нажатии клавиши клавиатуры и длится, пока нажата клавиша;
- **keyup**: возникает при отпускании клавиши клавиатуры;
- **keypress**: возникает при нажатии клавиши клавиатуры, но после события keydown и до события keyup. Надо учитывать, что данное событие генерируется только для тех клавиш, которые формируют вывод в виде символов, например, при печати символов. Нажатия на остальные клавиши, например, на Alt, не учитываются.

Для работы с событиями клавиатуры определен объект KeyboardEvent, который добавляет к свойствам объекта Event ряд специфичных для клавиатуры свойств:

altKey: возвращает true, если была нажата клавиша Alt во время генерации события;

charCode: возвращает символ Unicode для нажатой клавиши (используется для события keypress);

keyCode: возвращает числовое представление нажатой клавиши клавиатуры;

ctrlKey: возвращает true, если была нажата клавиша Ctrl во время генерации события;

metaKey: возвращает true, если была нажата во время генерации события метаклавиша клавиатуры;

shiftKey: возвращает true, если была нажата клавиша Shift во время генерации события.

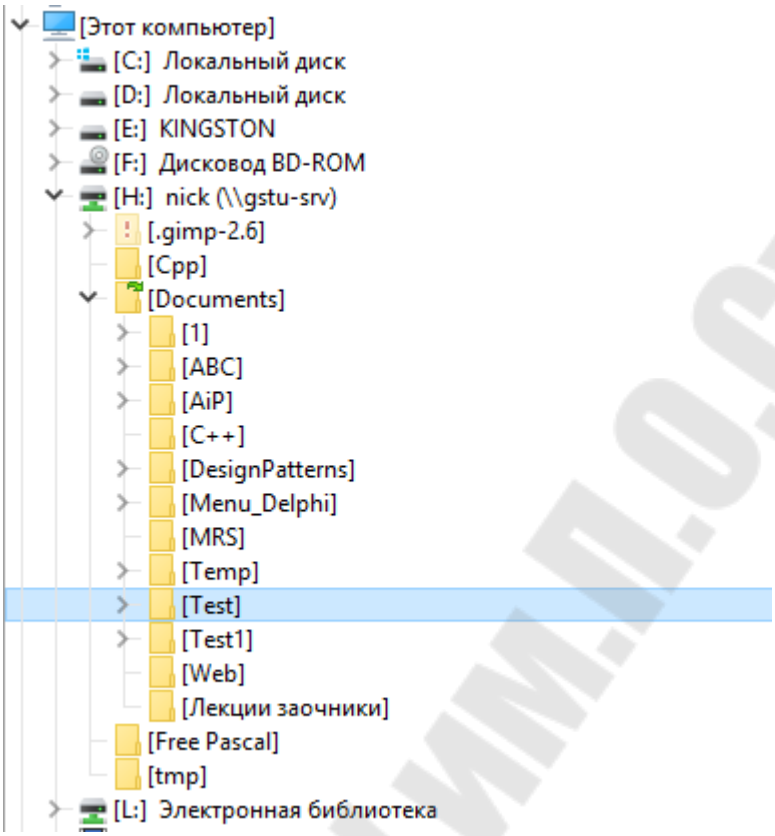
Практическая часть

Задание 1. На основании задания 4 предыдущей лабораторной работы сформировать таблицу для отображения элементов массива. Каждое свойство объекта выводится в отдельную ячейку. Таблица должна формироваться динамически с использованием JavaScript. Предусмотреть форматирование таблицы.

Задание 2. Создать HTML-документ со списком ссылок. Ссылки на внешние источники (которые начинаются с http:// или https://) необходимо подчеркнуть пунктиром. Искать такие ссылки в списке и устанавливать им дополнительные стили необходимо с помощью JS.

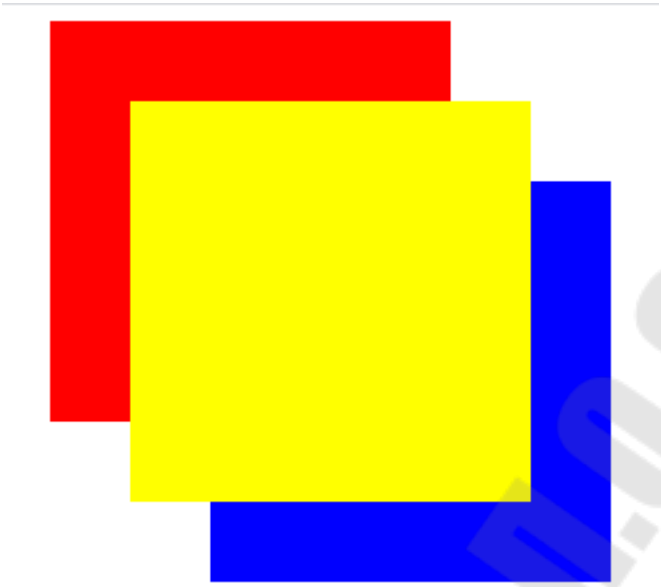
- index-html
- https://google.com
- https://www.gstu.by/
- file:///G:/users/RPI/test.html
- /local/path
- http://www.php.su/
- /images/cat.jpg
- https://www.tut.by/

Задание 3. Создать HTML-документ с деревом вложенных директорий (см. рисунок ниже). Размер дерева и содержимое – произвольные. Каждый узел является элементом списка. При клике на элемент списка, он должен сворачиваться или разворачиваться. При наведении на элемент, шрифт должен становится жирным (с помощью свойства classList).



Задание 4. Написать сценарий, который позволяет менять фоновое изображение документа выбором из таблицы цветов.

Задание 5. Создать HTML-документ, в котором присутствуют три перекрывающихся (но не полностью) блока с различным цветом фона и разными значениями z-индекса. Написать сценарий, в котором при клике мыши на блоке, соответствующий блок будет отображаться поверх остальных.



Задание 6. Создать HTML-документ со списком книг (автор и название). При щелчке на элементе спика, цвет текста должен меняться на оранжевый. При повторном щелчке на элементе необходимо возвращать прежний цвет. Если при клике мышкой была нажата клавиша Ctrl, то элемент добавляется/удаляется из выделенных. Если при клике мышкой была нажата клавиша Shift, то к выделению добавляются все элементы в промежутке от предыдущего кликнутого до текущего.

Требования по содержанию отчета

Отчет должен включать следующие пункты:
1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг прогрммного кода.
5. Результаты выполнения сценария.
6. Выводы.

Контрольные вопросы для защиты:

1. Как изменить правила стилей для заданного элемента?
2. Как программно получить доступ к элементу?
3. Какие методы и свойства объекта document Вы знаете?
4. Что такое DOM?

5. Что такое коллекция?
6. Как добавить атрибут к элементу?
7. Как создать (удалить) узел в DOM?
8. Какие события Вы знаете?
9. Опишите события клавиатуры.
10. Опишите события мыши.

Лабораторная работа № 8
Обработка и валидация форм с использованием JavaScript

Цель работы: научиться обрабатывать значения, введенные пользователем в элементы форм, применять валидацию формы на стороне клиента.

Краткие теоретические сведения:

Один из способов взаимодействия с пользователями представляют html-формы. Например, если нам надо получить от пользователя некоторую информацию, мы можем определить на веб-странице форму, которая будет содержать текстовые поля для ввода информации и кнопку для отправки. После ввода данных пользователем мы можем обработать введенную информацию.

В JavaScript форма представлена объектом **HtmlFormElement**. И после создания формы мы можем к ней обратиться различными способами.

Первый способ заключается в прямом обращении по имени формы:

```
<form name="search">
</form>
<script>
    let searchForm = document.search;
</script>
```

Второй способ состоит в обращении к коллекции форм документа и поиске в ней нужной формы:

```
let searchForm;
for (var i = 0; i < document.forms.length; i++) {
    if(document.forms[i].name==="search")
        searchForm = document.forms[i];
}
```

Форма имеет ряд свойств, из которых наиболее важными являются свойство name, а также свойство **elements**, которое содержит коллекцию элементов формы.

Среди методов формы надо отметить метод **submit()**, который отправляет данные формы на сервер, и метод **reset()**, который очищает поля формы:

Элементы форм

Форма может содержать различные элементы ввода html: input, textarea, button, select и т.д. Но все они имеют ряд общих свойств и методов.

Также, как и форма, элементы форм имеют свойство name, с помощью которого можно получить значение атрибута name.

Для получения и установки значения элементов **input** и **textarea** используется свойство input.value (строка) или input.checked (булево значение) для чекбоксов, textarea.value:

```
input.value = "Новое значение";
textarea.value = "Новый текст";
input.checked = true; // для чекбоксов и переключателей
```

Элемент **select** имеет 3 важных свойства:

- select.options – коллекция из подэлементов option;
- select.value – значение выбранного в данный момент option;
- select.selectedIndex – номер выбранного option.

Они дают три разных способа установить значение в select:

- найти соответствующий элемент option и установить в option.selected значение true;
- установить в select.value значение нужного option;
- установить в select.selectedIndex номер нужного option.

В отличие от большинства других элементов управления, select позволяет нам выбрать несколько вариантов одновременно, если у него стоит атрибут multiple. Эту возможность используют редко, но в этом случае для работы со значениями необходимо использовать первый способ, то есть ставить или удалять свойство selected у подэлементов option.

Валидация формы

HTML5 представил новую концепцию валидации HTML, названную проверкой ограничений, которая основана на:

- атрибутах элементов ввода HTML;
- псевдо-селекторах CSS;
- свойствах и методах DOM.

Для валидации формы используются следующие атрибуты элементов ввода HTML:

- **disabled**: указывает, что элемент ввода должен быть отключен;
- **max**: определяет максимальное значение элемента ввода;
- **min**: определяет минимальное значение элемента ввода;
- **pattern**: определяет шаблон значения элемента ввода;
- **required**: указывает, что для поля ввода требуется элемент;
- **type**: определяет тип элемента ввода.

Проверка ограничений CSS использует следующие псевдо-селекторы:

- **:disabled** – выбирает элементы ввода с указанным атрибутом «disabled»;
- **:invalid** – выбирает элементы ввода с недопустимыми значениями;
- **:optional** – выбирает элементы ввода без указания атрибута «required»;
- **:required** – выбирает элементы ввода с указанным атрибутом «required»
- **:valid** – выбирает элементы ввода с допустимыми значениями

Методы DOM включают функции:

- **checkValidity ()**: возвращает true, если входной элемент содержит допустимые данные;
- **setCustomValidity ()**: устанавливает свойство **validationMessage** элемента ввода.

Свойства DOM:

- **validity**: содержит логические свойства, связанные с достоверностью входного элемента;
- **validationMessage**: содержит сообщение, которое браузер отобразит, когда допустимость ложна;
- **willValidate**: указывает, будет ли проверен входной элемент.

В свою очередь свойство validity является объектом со следующими свойствами:

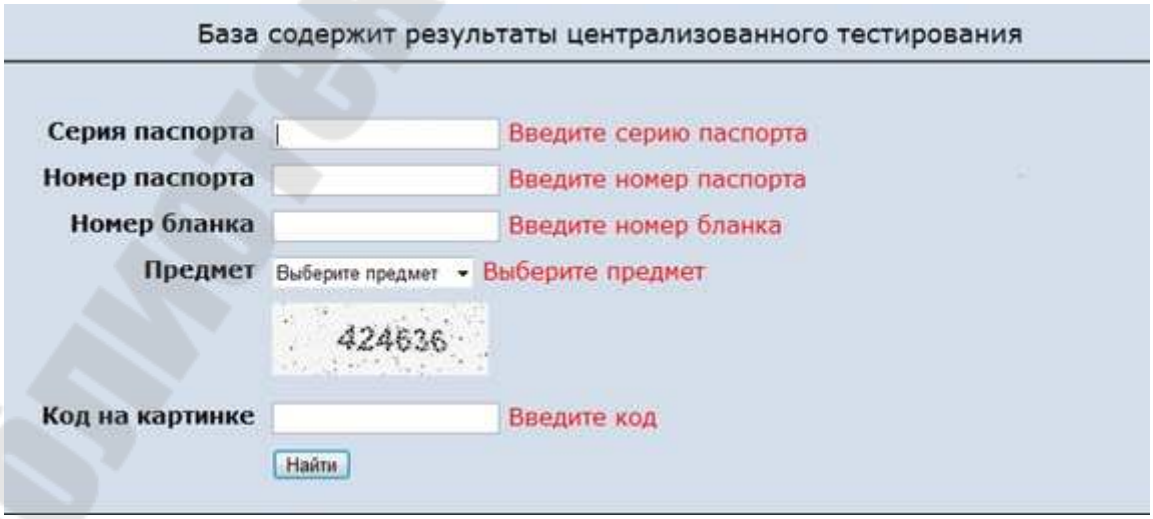
- **customError**: устанавливается в true, если настроено пользовательское сообщение о достоверности;
- **patternMismatch**: устанавливается в true, если значение элемента не соответствует его атрибуту pattern;
- **rangeOverflow**: устанавливается в true, если значение элемента больше, чем его атрибут max;
- **rangeUnderflow**: устанавливается в true, если значение элемента меньше его атрибута min;
- **stepMismatch**: устанавливается в true, если значение элемента недопустимо для его атрибута шага;
- **tooLong**: устанавливается в true, если значение элемента превышает его атрибут maxLength;
- **typeMismatch**: устанавливается в true, если значение элемента недопустимо для его атрибута типа;
- **valueMissing**: устанавливается в true, если элемент (с обязательным атрибутом) не имеет значения;
- **valid**: устанавливается в true, если значение элемента допустимо.

Практическая часть

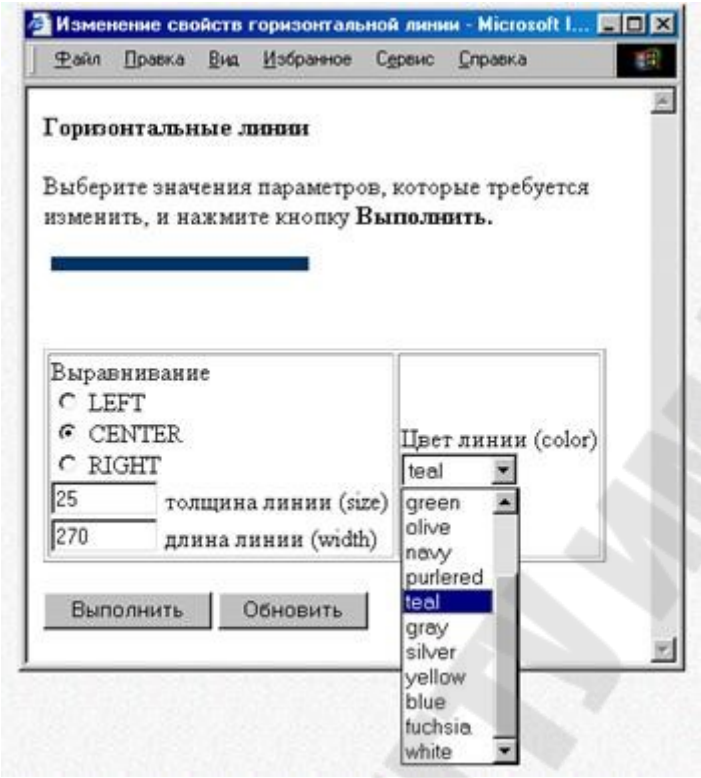
Задание 1. Необходимо создать веб-страницу с формой.

Написать скрипт, который находит среднее арифметическое введенных значений и выводит результат в поле формы при нажатии на кнопку «Вычислить». В программе предусмотреть обработку исключительной ситуации, когда пользователь не ввел значений (хотя бы одного) или введенные значения не являются числами, с выводом сообщения в модальном окне. Модальное окно реализовать самостоятельно, не использую стандартные диалоговые окна объекта window.

Задание 2. Необходимо реализовать форму с валидацией. Валидация осуществляется без использования атрибутов элементов ввода HTML. В поле паспорт могут вводится только две латинские буквы, в поле номер паспорта – только 7 цифр, номер бланка ответа – 7 цифр, предмет предполагает выбор языка (белорусский или русский), математика, физика. Код на картинке – 6 цифр. При не заполнении какого-либо поля должна выводится соответствующая подсказка.



Задание 3. Создайте файл с HTML-формой. Разработайте код JavaScript, в котором анализируются введенные пользователем данные, позволяющие изменить размер, положение на странице и цвет горизонтальной линии:



Вместо линии можно использовать блок и свойства CSS – width, height, backgroundColor. Для выравнивания блока используйте свойство margin.

Задание 4. В соответсвии со своим вариантом (задание 1 лабораторной работы №6), создать форму для ввода свойств объекта. Сформировать массив объектов, организовав цикл, в котором вызывается форма для ввода как модальное окно. Введенный макссив должен быть отображен на странице в виде таблицы.

Требования по оформлению отчета

Отчет должен включать следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задание.

4. Листинг программного кода.
5. Результаты выполнения сценариев.
6. Выводы.

Контрольны вопросы для защиты:

1. Какие механизмы используются при валидации форм?
2. Какие атрибуты элементов ввода используются при валидации форм?
3. Какие псевдоселекторы CSS используются при проверке ограничений?
4. Как получить доступ к конкретной форме на веб-странице?
5. Как получить доступ к элементам формы?
6. Как получить значение, введенное пользователем в элемент ввода?
7. Как проверить состояние переключателей (radio и checkbox)?
8. Как получить значение элемента select?
9. Какие методы DOM используются при валидации форм?
10. Какие свойства DOM используются при валидации форм?

Лабораторная работа № 9
Объектная модель браузера (BOM). Таймеры

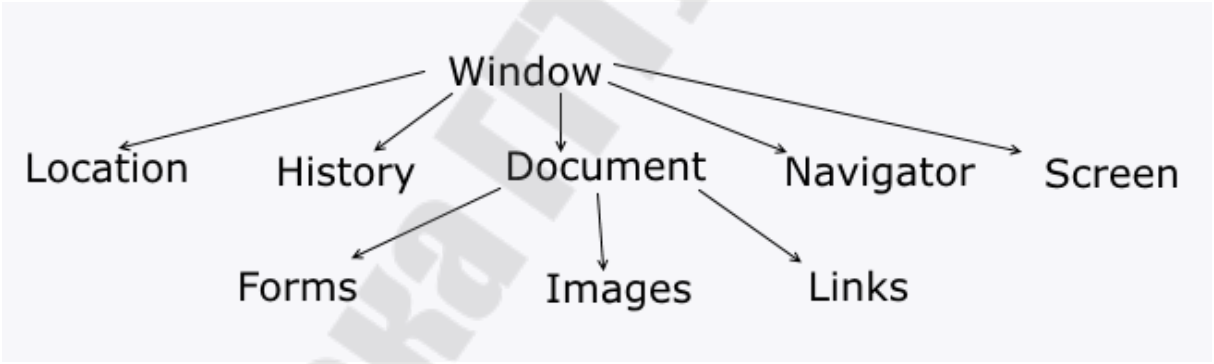
Цель работы: изучить объекты BOM, научиться использовать таймеры в JavaScript.

Краткие теоретические сведения:

Большое значение в JavaScript имеет работа с веб-браузером и теми объектами, которые он предоставляет. Например, использование объектов браузера позволяет манипулировать элементами html, которые имеются на странице, или взаимодействовать с пользователем.

Все объекты, через которые JavaScript взаимодействует с браузером, описываются таким понятием как **Browser Object Model** (Объектная Модель Браузера).

Browser Object Model можно представить в виде следующей схемы:



В вершине находится главный объект – объект window, который представляет собой браузер. Этот объект в свою очередь включает ряд других объектов, в частности, объект document, который представляет отдельную веб-страницу, отображаемую в браузере.

Объект window

Объект window представляет собой окно веб-браузера, в котором размещаются веб-страницы. Window является глобальным объектом, поэтому при доступе к его свойствам и методам необязательно использовать его имя.

Все объявляемые в программе глобальные переменные или функции автоматически добавляются к объекту window.

Ранее были рассмотрены 3 методы объекта window для взаимодействия с пользователем:

- **alert(str)**: показывает сообщение в диалоговом окне.
- **prompt(str, default)**: показывает сообщение и запрашивает ввод текста от пользователя. Возвращает напечатанный текст в поле ввода или null, если была нажата кнопка «Отмена» или Esc с клавиатуры.
- **confirm(str)**: показывает сообщение и ждёт, пока пользователь нажмёт OK или Отмена. Возвращает true, если нажата OK, и false, если нажата кнопка «Отмена» или Esc с клавиатуры.

Все эти методы являются модальными: останавливают выполнение скриптов и не позволяют пользователю взаимодействовать с остальной частью страницы до тех пор, пока окно не будет закрыто.

Объект window также предоставляет методы для работы с окном браузера:

- **window.open ()**: открыть новое окно;
- **window.close ()**: закрыть текущее окно;
- **window.moveTo ()**: переместить текущее окно;
- **window.resizeTo ()**: изменить размер текущего окна.

Объект history

Объект history предназначен для хранения истории посещений веб-страниц в браузере. Нам этот объект доступен через объект window.

Все сведения о посещении пользователя хранятся в специальном стеке (history stack). С помощью свойства **length** можно узнать, как много веб-станиц хранится в стеке.

Для перемещения по страницам в истории в объекте history определены методы **back()** (перемещение к прошлой посмотренной странице) и **forward()** (перемещение к следующей просмотренной странице)

Также в объекте history определен специальный метод **go()**, который позволяет перемещаться вперед и назад по истории на определенное число страниц. Положительное число определет перемещение вперед, а отрицательное – назад.

Объект location

Объект location содержит информацию о расположении текущей веб-страницы: URL, информацию о сервере, номер порта, протокол. С помощью свойств объекта мы можем получить эту информацию:

- **href**: полная строка запроса к ресурсу;
- **pathname**: путь к ресурсу;
- **origin**: общая схема запроса;
- **protocol**: протокол;
- **port**: порт, используемый ресурсом;
- **host**: хост;
- **hostname**: название хоста;
- **hash**: если строка запроса содержит символ решетки (#), то данное свойство возвращает ту часть строки, которая идет после этого символа;
- **search**: если строка запроса содержит знак вопроса (?), например, то данное свойство возвращает ту часть строки, которая идет после знака вопроса.

Также объект location предоставляет ряд методов, которые можно использовать для управления путем запроса:

- **assign(url)**: загружает ресурс, который находится по пути url.
- **reload(forcedReload)**: перезагружает текущую веб-страницу. Параметр forcedReload указывает, надо ли использовать кэш браузера. Если параметр равен true, то кэш не используется.
- **replace(url)**: заменяет текущую веб-станицу другим ресурсом, который находится по пути url. В отличие от метода assign, который также загружает веб-станицу с другого ресурса, метод replace не сохраняет предыдущую веб-страницу в стеке истории переходов history, поэтому мы не сможем вызвать метод history.back() для перехода к ней.

Объект navigator

Объект navigator содержит информацию о браузере и операционной системе, в которой браузер запущен. Он определяет ряд свойств и методов, основным из которых является свойство **userAgent**, представляющее браузер пользователя.

Объект navigator хранит свойство **geolocation**, с помощью которого можно получить географическое положение пользователя. Для получения положения используется метод **getCurrentPosition()**. Этот

метод принимает два параметра: функцию, которая срабатывает при удачном запуске, и функцию, которая срабатывает при ошибке запроса геоданных.

```
function success(position) {
    let latitude = position.coords.latitude;
    let longitude = position.coords.longitude;
    let altitude = position.coords.altitude;
    let speed = position.coords.speed;

    document.write("Широта: " + latitude + "<br/>");
    document.write("Долгота: " + longitude + "<br/>");
    document.write("Высота: " + altitude + "<br/>");
    document.write("Скорость перемещения: " + speed + "<br/>");
};

function error(obj) {
    document.write("Ошибка при определении положения");
};
navigator.geolocation.getCurrentPosition(success, error);
```

Таймеры

Для выполнения действий через определенные промежутки времени в объекте window предусмотрены функции таймеров. Есть два типа таймеров: одни выполняются только один раз, а другие постоянно через промежуток времени.

Для одноразового выполнения действий через промежуток времени предназначена функция **setTimeout()**. Она может принимать два параметра:

```
let timerId = setTimeout(someFunction, period)
```

Параметр period указывает на промежуток, через который будет выполняться функция из параметра someFunction. А в качестве результата функция возвращает id таймера.

Для остановки таймера применяется функция **clearTimeout(id)**.

Функции **setInterval()** и **clearInterval()** работают аналогично функциям setTimeout() и clearTimeout() с той лишь разницей, что

setInterval() постоянно выполняет определенную функцию через промежуток времени.

Функция **requestAnimationFrame()** действует аналогично setInterval() за тем исключением, что он больше заточен под анимации, работу с графикой и имеет ряд оптимизаций, которые улучшают его производительность. В метод window.requestAnimationFrame() передается функция, которая будет вызываться определенное количество раз (обычно 60) в секунду.

Практическая часть:

Задание 1. «Написать свой браузер»! На странице в верхней части размещается форма, в нижней части – плавающий фрейм. Форма обеспечивает следующие возможности:

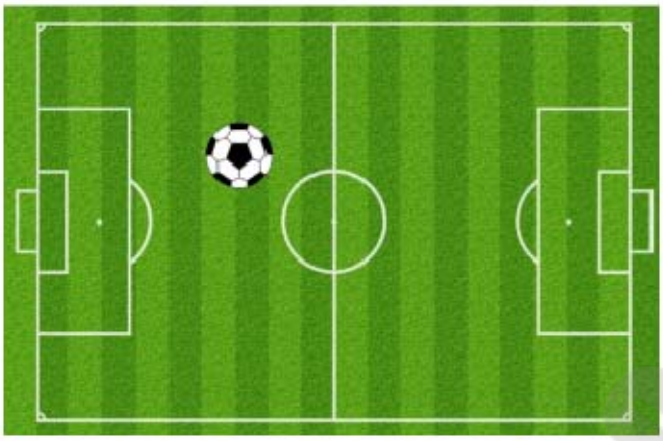
- ввод адреса;
- загрузка страницы в нижний фрейм или новое окно;
- переход назад и вперед по истории страниц нижнего фрейма;
- повторная загрузка страницы в нижнем фрейм.

Задание 2. Создать HTML-документ с кнопкой. При подведении курсора мыши к кнопке с любой стороны кнопка должна исчезать и через 3 секунды появляться в случайном месте окна.

Задание 3. Создать HTML-документ с футбольным полем, которое занимает всю ширину и высоту экрана, и мячом размером 100 на 100 пикселей.

Сделать так, чтобы при клике мышкой по полю, мяч плавно перемещался на место клика. Учтите: необходимо, чтобы центр мяча останавливался именно там, где был совершен клик мышкой.

Также предусмотрите, чтобы мяч не выходил за границы поля.



Задание 4. Создать HTML-документ с кнопкой или блоком div небольшого размера. Написать сценарий, вращающий элемент вокруг текущего положения курсора мыши. При изменении положения курсора, центр вращения так же должен смещаться следом за ним.

Требования по оформлению отчета

Отчет должен включать следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг программного кода.
5. Результаты выполнения сценариев.
6. Выводы.

Контрольные вопросы для защиты:

1. Как определить размеры окна?
2. Как создать новое окно?
3. Как определить адрес загруженного документа?
4. Как с помощью сценария перегрузить документ в текущем окне?
5. Как определить размеры экрана?
6. Какие свойства объекта **location** Вы знаете?
7. Какие свойства объекта **history** Вы знаете?
8. Какие методы объекта **history** Вы знаете?
9. В чем отличие функций **setInterval** и **setTimeout**?
10. Как сбросить таймер?
11. Какие свойства объекта **navigator** Вы знаете?

Лабораторная работа № 10
Сериализация объектов. Библиотека jQuery

Цель работы: ознакомиться с нотацией JSON, научиться использовать библиотеку jQuery.

Краткие теоретические сведения

Сериализация объектов

Для хранения и обмена данными между клиентом и сервером чаще всего используется формат JSON. JSON (JavaScript Object Notation) – способ записи данных в виде строки. Поскольку формат JSON является только текстовым, его можно легко отправлять на сервер и с сервера и использовать в качестве формата данных на любом языке программирования. Например, клиент использует JavaScript, а сервер написан на Ruby/PHP/Java или любом другом языке.

JavaScript предоставляет следующие методы для сериализации и десериализации объектов:

- **JSON.stringify(obj)** используется для преобразования объектов в строку JSON;
- **JSON.parse(str)** используется для преобразования строки в формате JSON обратно в объект.

JSON поддерживает три типа данных: примитивные значения, объекты и массивы. Примитивные значения представляют стандартные строки, числа, значение null, логические значения true и false.

```
<script>
    const student = {
      name: "Alex",
      age: 21,
      marks: {
        OAIP: 9,
        "System programming": 8,
        RPI: 9,
        ACS: 8
      }
    };
```

```
        let str = JSON.stringify(student);
        console.log(str);                                                    //
{"name":"Alex","age":21,"marks":{"OAIP":9,"System
programming":8,"RPI":9,"ACS":8}}
        let alex = JSON.parse(str);
        console.log(alex); // {name: "Alex", age: 21, marks: {…}}
    </script>
```

Web storage

Для хранения данных на стороне клиента в HTML5 была внедрена новая концепция – web-storage. Web storage состоит из двух компонентов: **session storage** и **local storage**.

Session storage представляет временное хранилище информации, которая удаляется после закрытия браузера.

Local storage представляет хранилище для данных на постоянной основе. Данные из local storage автоматически не удаляются и не имеют срока действия. Объем local storage составляет в Chrome и Firefox 5 Мб для домена.

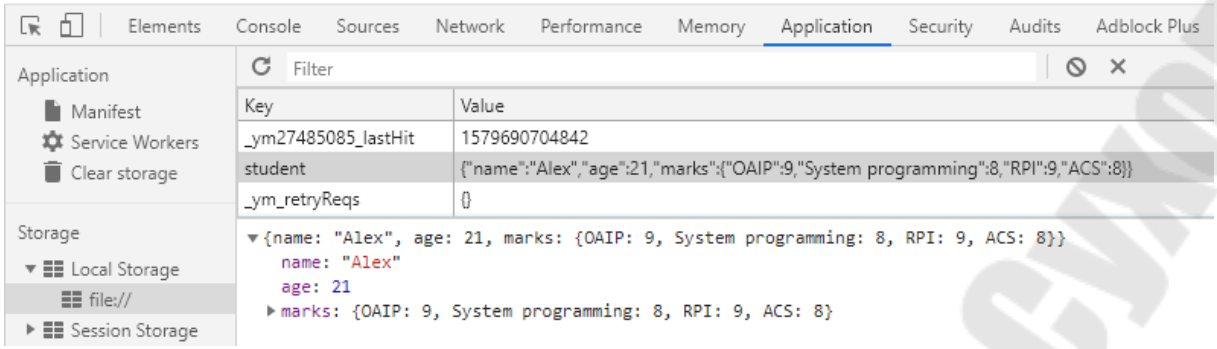
Все данные в web storage представляют набор пар ключ-значение. То есть каждый объект имеет уникальное имя-ключ и определенное значение.

Для работы с local storage в javascript используется объект **localStorage**, а для работы с session storage – объект **sessionStorage**. Эти объекты предомтавляют следующие методы для работы с данными:

- **setItem()** – для сохранения данных;
- **getItem()** – для получения сохраненных данных;
- **removeItem()** – для удаления объект из хранилища;
- **clear()** – для для полного удаления всех объектов из хранилища.

Например:

```
localStorage.setItem("student", JSON.stringify(student));
```



Библиотека jQuery

Современное веб-программирование и создание веб-приложений уже невозможно представить без использования языка JavaScript. Однако в настоящее время, все чаще используется не чистый код JavaScript, а JavaScript-фреймворки и библиотеки. Одной из таких библиотек является jQuery.

Подключив библиотеку jQuery вместо десятков команд на JavaScript можно написать несколько команд. Команды основаны на селекторах и классах CSS.

Библиотеку jQuery можно скачать с сайта http://code.jquery.com/. Если нет желания скачивать библиотеку её можно подключить с CDN (Content Delivery Network) компаний Google или Microsoft.

Вся работа с библиотекой сводится к выбору (query) HTML элемента и выполнения над ним различных действий

Основной синтаксис:

```
$(selector).action(),
```

где $() – функция доступа (определения) jQuery. (jQuery() – полное название функции $), selector – "query(запрос)" для определения HTML элементов, action() – действия над элементом (элементами).

Примеры:

```
$(this).hide() // скрывает текущий элемент.
$("p").hide() // скрывает все абзацы.
$(".test").hide() // скрывает все элементы класса "test".
$("#test").hide() // скрывает элемент с id="test".
```

Создание элементов DOM

```
$("<div>Hi</div>");
```

```
$("<div>")
```

Можно использовать кавычки, можно апострофы. Но наличие тегов обязательно. Задавать просто текст нельзя. Сама по себе эта команда ничего не выведет, надо этот узел привязать к родительскому элементу. Делается это так:

```
$("<div>Hi</div>").insertAfter("#a1")
```

Здесь a1 – это идентификатор объекта, после которого необходимо вставить элемент.

Работа с полученным набором значений

В терминах jQuery эти наборы называют обернутыми.

Для определения количество выбранных элементов используется свойство length или метода size, который также возвращает число выбранных элементов:

```
let num1 = $("tr:nth-child(odd)").length;
let num2 = $("tr:nth-child(odd)").size();
```

Для получения доступа к элементу выборки можно использовать индекс или использовать функцию eq(index). Второй способ более удобен так как функция возвращает объект:

```
let firstElement = $("tr:nth-child(odd)")[0];.
let elem = $("tr:even").eq(0);.
```

Для перебора элементов выборки можно использовать стандартные методы JavaScript, а также использовать специальный метод each:

```
$(function(){
    $("tr:even").each(function(index, elem){
        console.log(index + ". " + elem.innerHTML);
    });
});
```

В метод each в качестве параметра передается безымянная функция, которая принимает два параметра: index - индекс элемента в наборе и elem - сам элемент.

Манипулирование объектами на странице

Библиотека jQuery предлагает нам инструментарий для манипуляции свойствами и атрибутами элементов:

- **prop(property, [value])** получает или изменяет значение свойства элемента;
- **removeProp(property)** удаляет свойство;
- **attr(attributeб [value])** получить или изменяет значение атрибута элемента;
- **css(property, [value]**) применяется для работы со стилями элемента;
- **html()** используется для получения или установки разметки;
- **text()** используется для получения или установки текста;
- **val()** используется для получения значений элементов форм.

Обработка событий

Различные браузеры по-своему могут обрабатывать события, jQuery пытается сгладить эти неприятности.

Модель событий jQuery обладает следующими свойствами:

- поддерживает единый метод установки событий;
- позволяет устанавливать несколько обработчиков для события;
- использует стандартные названия типов событий;
- предоставляет единые методы отмены события и блокирования действий по умолчанию.

jQuery предоставляет специальный метод **on**, который позволяет создать и зарегистрировать обработчики как для существующих, так и для будущих элементов, которых еще не в структуре DOM. Этот метод имеет следующие варианты использования:

- on('событие', 'селектор', обработчик_события);
- on('событие', 'селектор', данные_события, обработчик_события) то же самое, что и в предыдущем случае, плюс параметр для передачи данных события в объект Event (e.data).

Для удаления обработчика используется команда unbind()

Метод **trigger()** используется для вызова обработчиков событий вручную. Например, вызвав метод trigger для нужного элемента мы можем также и вызвать обработчик события. Он имеет следующие варианты использования:

- trigger('событие'): вызов обработчика для данного события;
- trigger(объект_Event): вызов обработчика с использованием объекта Event, который содержит данные о том, какой именно обработчик надо вызвать.

Кроме специализированных методов jQuery предоставляет прямые методы для обработки событий. Эти методы, как правило, носят наименование обрабатываемого события, а в качестве параметра принимают функцию обработчика данного события:

```
$('button').first().click(function(){
      $(this).css('background-color', 'silver');
   });
```

Эффекты анимации

К **базовым эффектам** в jQuery относятся эффекты скрытия и отображения элементов, которые достигаются с помощью методов **show()**, **hide()** и **toggle()**.

Все эти методы могут использоваться в трех вариантах (рассмотрим на примере метод hide):

- hide(): метод без параметров
- hide([duration][, complete]): принимает два необязательных параметра. Параметр duration указывает как долго анимация элемента будет длиться. По умолчанию его значение равно 400 миллисекунд. Параметр complete представляет функцию, вызываемую методом по завершению анимации
- hide([duration] [, easing][, complete]): то же самое, только добавляется параметр easing, который принимает название функции плавности анимации в виде строки. По умолчанию его значение равно "swing". Также можно использовать значения 'slow' и 'fast', которые соответствуют длительности эффекта в 600 и 200 миллисекунд.

Эффекты скольжения позволяют нам плавно скрыть или раскрыть элемент. Эффекты скольжения реализованы в виде методов **slideUp()**, **slideDown()** и **slideToggle()**.

Эффекты прозрачности позволяют нам, плавно изменяя прозрачность элемента, скрыть его или отобразить. Эффекты прозрачности реализованы с помощью методов **fadeOut()**, **fadeIn()**, **fadeTo()** и **fadeToggle()**.

Для создания более сложных по характеру эффектов используется метод **animate()**:

```
animate(properties [,duration] [,easing] [,complete])
```

Обязательный параметр properties содержит набор css-свойств, у которых указываются финальные значения. Параметр duration указывает, как долго будет длиться изменение прозрачности элемента. Параметр easing принимает название функции плавности анимации в виде строки. Параметр complete представляет функцию обратного вызова, вызываемую методом по завершении анимации

Задание к лабораторной работе:

Задание 1. В соответсвии с вариантом задания 1 лабораторной работы №6 сформировать массив объектов. Сериализовать массив в формат JSON с последующим сохранением его в объекте localStorage. Используя библиотеку jQuery, написать сценарий, считывающий информацию из локального хранилища и формирующий таблицу из полученных данных. Предусмотреть сортировку отображаемых данных при щелчке по заголовкам таблицы.

Задание 2. Создать HTML-документ с реализацией стандартногокалькулятора, выполняющего арифметические операции над введенными данными и имеющего возможность сохранения результатов в памяти. При написании сценария использовать. Предусмотреть ввод данных и выбор операции как с помощью мыши, так и с помощью клавиатуры.

Задание 3. Создать HTML-документ с полем ввода даты. Используя библиотеку jQuery, сформировать календарь для текущего

месяца введенной даты. Календарь оформить в виде таблицы, неделя начинается с понедельника, в ячейки таблицы выводится день месяца.

Задание 4. Воспользоваться созданным в предыдущем задании календарем. Сформировать строку JSON с информацией о мероприятиях на текущий месяц. Строка хранится в Local Storage. Предусмотреть выделения цветом дней месяца, для которых запланировано мероприятие. При щелчке на такой день, должна выдаваться полная информация о мероприятии. В скрипте предусмотреть возможность добавления мероприятия на выбранный день с сохранением информации в Local Storage. Для добавления мероприятия и просмотра информации использовать модальное окно.

Контрольные вопросы для защиты:

1. Какие механизмы предусмотены для хранения информации на стороне клиента?
2. Что такое JSON?
3. Как сериализуется объекты?
4. Какие средства используются для восстановления данных из JSON строки?
5. Что такое jQuery?
6. Как получить доступ к элементу, используя jQuery?
7. Как изменить свойства элемента, используя jQuery?
8. Как получить доступ к атрибутам, используя jQuery?
9. Как изменить содержимое разметки элемента, используя jQuery?
10. Как получить контекст элемента, используя jQuery?
11. Как задать стили элемента, используя jQuery?
12. Как назначить обработчик события, используя jQuery?

Лабораторная работа № 11
Взаимодействие с сервером. Технология AJAX

Цель работы: изучить технологию AJAX, научиться получать данные, используя REST API.

Краткие теоретические сведения:

Ajax представляет технологию для отправки запросов к серверу из клиентского кода JavaScript без перезагрузки страницы. Сам термин расшифровывается как Asynchronous JavaScript And XML. То есть изначально AJAX предполагал асинхронное взаимодействие клиента и сервера посредством данных в формате XML. В настоящее время основным форматом данных для обмена данных между клиентом и сервером является формат JSON.

Объект XMLHttpRequest

Для создания приложений, использующих Ajax, применяются различные способы. Но самым распространенным способом является использование объекта XMLHttpRequest:

```
const request = new XMLHttpRequest();
```

После создания объекта XMLHttpRequest можно отправлять запросы к серверу. Но для начала надо вызвать метод open() для инициализации:

```
request.open("GET", "http://localhost/test.txt", false);
```

Метод open() принимает три параметра: тип запроса (GET, POST, HEAD, PUT), адрес запроса и третий необязательный параметр – логическое значение true или false, указывающее, будет ли запрос осуществляться в асинхронном режиме. По умолчанию, если третий параметр не используется, то запрос отправляется в асинхронном режиме, что позволяет параллельно с выполнением запроса выполнять также и другой код JavaScript.

Кроме того, метод open() может принимать еще два параметра: логин и пароль пользователя, если для выполнения запроса нужна аутентификация:

```
request.open("GET", "http://localhost/home.php", true, "login", "password");
```

После инициализации запроса методом open() необходимо отправить запрос с помощью метода send():

```
request.send();
```

Объект XMLHttpRequest имеет ряд свойств, которые позволяют проконтролировать выполнение запроса:

- **status**: содержит статусный код ответа HTTP, который пришел от сервера. С помощью статусного кода можно судить об успешности запроса или об ошибках, которые могли бы возникнуть при его выполнении. Например, статусный код 200 указывает на то, что запрос прошел успешно. Код 403 говорит о необходимости авторизации для выполнения запроса, а код 404 сообщает, что ресурс не найден и так далее.
- **statusText**: возвращает текст статуса ответа, например, "200 OK"
- **responseType**: возвращает тип ответа.
- **response**: возвращает ответ сервера.
- **responseText**: возвращает текст ответа сервера.
- **responseXML**: возвращает xml, если ответ от сервера в формате xml.

При асинхронном запросе объект XMLHttpRequest использует свойство **readyState** для хранения состояния запроса. Состояние запроса представляет собой число:

0: объект XMLHttpRequest создан, но метод open() еще не был вызван для инициализации объекта

- 1: метод open() был вызван, но запрос еще не был отправлен методом send()
- 2: запрос был отправлен, заголовки и статус ответа получены и готовы к использованию
- 3: ответ получен от сервера

– 4: выполнение запроса полностью завершено (даже если получен код ошибки, например, 404)

Событие readystatechange возникает каждый раз, когда изменяется значение свойства readyState.

Отправка запросов

GET-запрос характеризуется тем, что данные могут отправляться в строке запроса:

```
<script>
// объект для отправки
var user = {
   name: "Alex",
   age: 21
};

const request = new XMLHttpRequest();
function reqReadyStateChange() {
   if (request.readyState == 4) {
      var status = request.status;
      if (status == 200) {
       document.getElementById("output").innerHTML=request.resp
onseText;
      }
   }
}
// строка с параметрами для отправки
var body = "name=" + user.name + "&age="+user.age;
request.open("GET", "http://localhost:8080/postdata.php?"+body);
request.onreadystatechange = reqReadyStateChange;
request.send();
</script>
```

Для отправки берем свойства объекта user и формируем из их значений сроку с параметрами: "name=" + user.name + "&age=" + user.age. Затем эта строка добавляется к строке запроса в методе open("GET", "http://localhost:8080/postdata.php?"+body)

Для отправки данных методом POST надо установить заголовок Content-Type с помощью метода **setRequestHeader()**. В данном случае заголовок имеет значение application/x-www-form-urlencoded. Данные встраиваются в тело запроса при посылке запроса.

```
var user = {
   name: "Tom",
   age: 23
};

const request = new XMLHttpRequest();
function reqReadyStateChange() {
   if (request.readyState == 4 && request.status == 200)
       docu-
ment.getElementById("output").innerHTML=request.responseText;
}
let body = "name=" + user.name + "&age="+user.age;
request.open("POST", "http://localhost:8080/postdata.php");
request.setRequestHeader('Content-Type',  'application/x-www-form-
urlencoded');
request.onreadystatechange = reqReadyStateChange;
request.send(body);
```

Promise в Ajax-запросах

Для более удобного способа организации асинхронного кода в современном JavaScript используется объект Promise, который оборачивает асинхронную операцию в один объект и позволяет определить действия, выполняющиеся при успешном или неудачном выполнении этой операции.

Promise – это специальный объект, который содержит своё состояние. Вначале pending («ожидание»), затем – одно из: fulfilled («выполнено успешно») или rejected («выполнено с ошибкой»).

Синтаксис создания Promise:

```
const promise = new Promise(function(resolve, reject) {
  // Эта функция будет вызвана автоматически
  // В ней можно делать любые асинхронные операции,
  // А когда они завершатся — нужно вызвать одно из:
  // resolve(результат) при успешном выполнении
```

```
  // reject(ошибка) при ошибке
})
```

Для обработки результата объекта Promise вызывается метод then(), который принимает два параметра: функцию, вызываемую при успешном выполнении запроса, и функцию, которая вызывается при неудачном выполнении запроса. Метод then() также возвращает объект Promise. Поэтому при необходимости мы можем применить к его результату цепочки вызовов метода then: promise.then().then().

Рассмотрим пример использования промиса с использованием промисификация – это когда берут асинхронную функциональность и делают для неё обёртку, возвращающую промис.

После промисификации использование функциональности зачастую становится гораздо удобнее. Например:

```
function httpGet(url) {
  return new Promise(function(resolve, reject) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.onload = function() {
      if (this.status == 200) {
        resolve(this.response);
      } else {
        var error = new Error(this.statusText);
        error.code = this.status;
        reject(error);
      }
    };
    xhr.onerror = function() {
      reject(new Error("Network Error"));
    };
    xhr.send();
  });
}

httpGet("/article/promise/user.json")
  .then(
    response => alert(`Fulfilled: ${response}`),
    error => alert(`Rejected: ${error}`)
  );
```

Метод Fetch

Современные браузеры поддерживает метод **fetch** – новый встроенный метод для AJAX-запросов, призванный заменить XMLHttpRequest. Он гораздо мощнее, чем приведенная выше функция httpGet.

Типичный запрос с помощью fetch состоит из двух операторов await:

```
let response = await fetch(url, options); // завершается с заголовками ответа
let result = await response.json(); // читать тело ответа в формате JSON
```

Или, без await:

```
fetch(url, options)
  .then(response => response.json())
  .then(result => /* обрабатываем результат */)
```

Метод fetch следующие опции для формирования запроса:

- **method** – HTTP-метод;
- **headers** – объект с запрашиваемыми заголовками;
- **body** – данные для отправки (тело запроса) в виде текста, FormData, BufferSource, Blob или UrlSearchParams.

Для получения тела ответа используются следующие методы:

- **response.text()** – возвращает ответ как обычный текст;
- **response.json()** – преобразовывает ответ в JSON-объект;
- **response.formData()** – возвращает ответ как объект FormData (кодировка form/multipart, см. следующую главу);
- **response.blob()** – возвращает объект как Blob (бинарные данные с типом);
- **response.arrayBuffer()** – возвращает ответ как ArrayBuffer (низкоуровневые бинарные данные).

Практическая часть:

Задание 1. Написать скрипт, в котором предусмотрена возможность получения данных с сайта Национального банка Республики Беларусь**,** используя службу **Web API** (документация **-** http://www.nbrb.by/APIHelp/ExRates).

На странице предусмотреть получение валюты страны со своим вариантом. Информацию отобразить в виде таблицы, указав флаг страны, название страны, название валюты по-английски и по-белорусски, курс валюты к белорусскому рублю на текущую дату. Для получения данных использовать объект **XMLHttpRequest.**

Номер варианта	Страна	Номер варианта	Страна	Номер варианта	Страна
1	Аргентина	11	Ирландия	21	Турция
2	Алжир	12	Израиль	22	Великобритания
3	Канада	13	Южная Корея	23	Россия
4	Чили	14	Мексика	24	Таджикистан
5	Китай	15	Монголия	25	Армения
6	Дания	16	Польша	26	Казахстан
7	Финляндия	17	Афганистан	27	Азербайджан
8	Грузия	18	Швеция	28	Малайзия
9	Гонконг	19	Украина	29	Индия
10	Исландия	20	Сингапур	30	Норвегия

Задание 2. Получить данные для отображения динамики курса валюты за указанный период времени. Период времени задается в полях ввода. Информацию отобразить в виде таблицы. Для получения данных использовать **промисификацию.**

Задание 3. Отобразить в виде таблицы курс конвертации валюты страны в соответствии с вариантом к валютам стран предыдущего и последующего вариантов. Информацию отобразить в виде таблицы. Для получения данных использовать метод **fetch** и специальный синтаксис для работы с промисами **«async/await».**

Контрольные вопросы для защиты:

1. Как осуществляется обмен данными между клиентом и сервером?
2. Что такое технология AJAX?
3. Что такое REST API?
4. Опишите свойства объекта XMLHttpRequest.
5. Как получить данные с сервера, используя GET запрос?
6. Как получить данные с сервера, используя POST запрос?
7. Что такое промис?
8. Основные действия при работе с промисами.
9. Как обрабатываются ошибки при использовании промисов?
10. Для чего предназначен метод fetch?

Литература

1. Флэнаган, Д. JavaScript. Подробное руководство / Дэвид Флэнаган ; пер. с англ. А. Киселева. - 6-е изд. – Санкт-Петербург : Москва : Символ-Плюс, 2017. - 1080 с.

2. Васильев, А. Н. Программирование на Java Script в примерах и задачах / Васильев А. Н.. - Москва : Издательство "Э", 2017. - 718 с.

3. Ajax = Айякс : JavaScript, JQuery, Prototype, Dojo, DWR, Google Web ToolKit, Yahoo UI Library, Maps, ASP.NET Ajax, JSF, SOAP, WSDL, RSS and Atom, Comet, Ruby on Rails, XML, XSLT. – India : Dreamlech Press, 2010. – 756 p.

41. Современный учебник JavaScript. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 27.03.2020.

5. JavaScript Tutorial. – Режим доступа: https://www.w3schools.com/js/default.asp. – Дата доступа: 27.03.2020.

Самовендюк Николай Владимирович

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ ИНТЕРНЕТ. FRONT-END

Практикум по выполнению лабораторных работ по одноименной дисциплине для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 26.02.21.

Рег. № 15Е.
http://www.gstu.by