LUXOFT
Expertise in Software Services

WEB-007
JavaScript

Сборник упражнений
(версия 2.0 от 10.12.2015)

Содержание

Упражнение 1 (модуль 3).
Типы данных.

Цели упражнения.

Ознакомиться с имеющимися типами данных JavaScript, освоить принципы конвертации данных между различными типами данных, понять способы передачи ссылочных данных.

Описание.

При выполнении данного упражнения необходимо создать HTML-страницу, либо JS файл в обычном текстовом редакторе, и добавить блок JavaScript:
<script>
</script>.
Внутрь блока поместить код.

Задачи упражнения.

Часть 1. Конвертация число-строка.

Освоить следующие техники конвертации:

1. Конкатенация числа со строкой:

```
var n = 100;
var s = n + " bottles of beer on the wall.";
```

2. Конкатенация числа `n` с пустой строкой;
3. Вызов функции `String(number);`
4. Вызов метода `toString();`
5. Вызов метода `toString()` с указанием основания:

```
var n = 17;
binary_string = n.toString(2);
```

6. Дано число `n= 123456.789`, пользуясь функциями `toFixed(), toExponential(), toPrecision()` получить следующие строковые представления:
   a. `"123457";`
   b. `"123456.79";`
   c. `"1.2e+5";`
   d. `"1.235e+5";`
   e. `"1.235e+5";`
   f. `"123456.8".`

Часть 2. Конвертация строка-число.

1. В примере `var product = "21" * "2";` какого типа будет переменная `product,` и какое оно будет содержать значение;

2. Освоить технику конвертации
   `var number = string_value - 0;`
3. Освоить технику конвертации `Number(stringValue);`
4. Что возвратят следующие фрагменты кода:

```
a. parseInt("3 blind mice");
b. parseFloat("3.14 meters");
c. parseInt("12.34");
d. parseInt("0xFF");
e. parseInt("11", 2);
f. parseInt("ff", 16);
g. parseInt("zz", 36);
h. parseInt("077", 8);
i. parseInt("077", 10);
j. parseInt("eleven");
k. parseFloat("$72.47");
```

Часть 3. Boolean преобразования.

1. При условии, что `x` это:
   a. Значение `Undefined;`
   b. Значение `null`;
2. `x = "aaa";`
3. `x = 7;`
4. `x = 0;`
5. `x = Infinity;`
6. `x` – это объект;
7. Освоить технику конвертации `Boolean(x);`
8. Освоить технику конвертации `x!!`

Часть 4. Передача значения примитивного типа.

1. В примере, сказать чему будут равны n и m

```
var n = 1;
var m = n;
function add_to_total(total, x)
{
    total = total + x;
}
add_to_total(n, m);
```

Часть 5. Работа со ссылочными типами.

В примере:

```
1. var xmas = new Date(2007, 11, 25);
2. var solstice = xmas;
3. solstice.setDate(21);
4. xmas.getDate( );
5. xmas == solstice
6. var xmas = new Date(2007, 11, 25);
7. var solstice_plus_4 = new Date(2007, 11, 25);
8. xmas != solstice_plus_4
```

1. *Какое значение вернет вызов на строке 4?*

2. *Чему равно значение на строке 5?*
3. *Чему равно значение на строке 8?*
4. Сделать выводы

Упражнение 2 (модуль 4).
Переменные и области видимости.

Цели упражнения.

Отработать принципы глобальной и локальной видимости переменной. Закрепить декларацию переменной. Обратить внимание на отсутствие необходимости освобождать память.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

Часть 1. Локальная глобальная области видимости.

Указать, что выведется на экран при выполнении кода:

```
scope = "global";
function checkscope( ) {
   scope = "local";
   document.write(scope);
   myscope = "local";
   document.write(myscope);
}
checkscope( );
document.write(scope);
document.write(myscope);
```

Указать, что выведется на экран при выполнении кода, ознакомится с вложенными функциями:

```
var scope = "global scope";
function checkscope( ) {
   var scope = "local scope";
   function nested( ) {
      var scope = "nested scope";
      document.write(scope);
   }
   nested( );
}
checkscope( );
```

Объяснить результаты выполненных примеров с точки зрения цепочки вызова.

Упражнение 3 (модуль 5). Выражения и операторы.

Цели упражнения.

Усвоить правила неявной конвертации данных. Освоить операторы проверки идентичности и равенства. Логические операторы. Оператор modulo.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

Часть 1. Использование операторов equality(==) и identity(===).

1. Указать, чему равен результат выполнения условного оператора в каждом случае и объяснить почему используя правила неявной конвертации данных:

```
a. alert(null == undefined);
b. alert(null === undefined);
c. alert(5 == "5");
d. alert(5 === "5");
e. alert("true" == true);
f. alert(1 == true);
g. alert("1" == true);
h. alert(1 === true);
i. alert("0" == false);
j. alert("0" === false);
k. var x = { id: 123, name: "Blah" };
l. var y = { id: 456, name: "Not Blah,
   definitely" };
m. alert(x == "[object Object]");
n. alert(y == "[object Object]");
o. alert(x == y);
p. alert(1 != true);
q. alert(0 != false);
r. alert(5 != "5");
s. var x = new Object();
t. alert(x != "[object Object]");
```

2. В примере, оператор equality более правилен, так как с формы приходит строка, которая сравнивается с числом 20: `if(document.getElementById("age").value== 20);`

3. Сравнить оператор сравнения и равенства в двух случаях:

```
var k;
if (k==null) {
   alert("k must be defined " + k);
}
```

и

```
var k = null;
if (k==null) {
   alert("k must be defined " + k);
}
```

Часть 2. Строковые и арифметические операторы.

1. Чему будет равна переменная x, какой тип сравнения используется, каков тип x:

```
a. x = "The answer is " + 42;
b. x = 42 + " is the answer";
c. x = "37" - 7;
d. x = "37" + 7;
e. x = 1 + 2;
f. x = "11" < "3";
g. x = "11" < 3;
h. x = "one" < 3;
i. x = 1 + 2 + " blind mice";
j. x = "blind mice: " + 1 + 2;
```

Часть 3. Оператор остатка от деления.

1. Чему равно:

```
k. 5%3
l. 5.7%3
m. -3.93%2
```

2. *Как используя, оператор % можно получать последовательность 0,1,0,1,0,1 и т.д.?* Эту возможность можно использовать для раскраски четных рядов таблицы в один цвет, нечетных – в другой. Реализуйте код, выводящий такую последовательность.

Упражнение 4 (модуль 6). Утверждения.

Цели упражнения.

Отработка навыков использования утверждений `switch`, `do-while`, `for-in`, работа с метками, возбуждение и перехват исключений.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

Часть 1. Утверждение `switch`.

1. Указать, корректен ли следующий пример (**`x`** – не константа) и выполнится ли строка 6. Если да, то почему, и как сделать так, чтобы сообщение "`Question`" не выводилось.

```
1. var x = 10;
2. switch (x) {
3.   case 10:
4.     x = 15;
5.   case 15:
6.     alert("Question");
7. }
```

Часть 2. Утверждение `do-while`.

1. Дан массив строк **`a`**. Написать функцию, которая выводит на экран сообщение «`Empty Array`», если массив пуст и элементы массива, если массив не пуст. Использовать утверждение `do-while`.

Часть 3. Утверждение for-in.

1. В client-side JavaScript (браузер) существует объект `document`. Вывести на экран все свойства и их значения объекта `document`.
2. Создайте массив и выведите все элементы этого массива с помощью метода for-in. Почему не рекомендуется использоваться for-in для этой цели?

Часть 4. Выход по меткам (опционально).

1. Рассмотреть пример, указать, что будет выведено на экран:

```
outerloop:
  for(var i = 0; i < 10; i++) {
    innerloop:
      for(var j = 0; j < 10; j++) {
          if (j > 3) break;              // Quit the innermost loop
          if (i == 2) break innerloop;  // Do the same thing
          if (i == 4) break outerloop;  // Quit the outer loop
          document.write("i = " + i + " j = " + j + "<br>");
```

```
        }
    }
    document.write("FINAL i = " + i + " j = " + j + "<br>");
```

Упражнение 5 (модуль 7). Объекты и массивы.

Цели упражнения.

Создание объектов, рассмотрение различных способов доступа к свойствам объекта. Создание массива, использование свойства `length` массива, итерирование массива, работа с массивом как с объектом.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

Часть 1. Создание объекта и работа со свойствами объекта.

1. Создать объект a без свойств;
2. Удостовериться, что `a.prop` возвращает значение `undefined;`
3. В произвольном месте программы обратиться к несуществующей переменной: `alert(nosuchprop).` Убедиться, что возбуждается runtime исключение;
4. Используя информацию с шагов 1 и 2, а также тот факт, что глобальный объект в client JavaScript это `Window`, выполнить проверку существования свойства.

Часть 2. Объект как ассоциативный массив.

1. Создать пустой объект `obj`;
2. Добавить свойство `prop` к объекту `obj`;
3. Удостоветиться, что свойство определено у объекта с помощью оператора `for/in`. Для этого вывести все свойства объекта `obj`;
4. Удостоветиться, что свойство определено у объекта с помощью метода `hasOwnProperty`;
5. Удалить свойство с помощью метода `delete`;
6. Проверить, что свойство `prop` удалено используя `for/in` и `hasOwnProperty`.

Часть 3. Создание массива.

1. Пояснить, что означает конструкция:

```
a = new Array({"attr1":"text1","attr2":"text2"},
{"attr1":"text3","attr2":"text4"});
```

Часть 4. Длина массива.

1. Пояснить, какова длина массива в примере ниже, сколько памяти этот массив занимает:

```
var a = []
a[1] = 1
```

```
a[999999] = 2
```

2. Убедиться в эквивалентности добавления нового элемента в конец массива:

```
a[a.length] = "new element"
a.push("new element")
```

3. Получить последний элемент массива с помощью метода `pop`. Посмотреть, как изменится длина массива;
4. Освоить очистку массива с помощью приема: `array.length = 0`.

Часть 5. Перебор элементов массива.

Дан массив с разрывом:

```
var arr = [];
arr[1] = 123;
arr[9999] = 456;
```

Проитерировать массив (вывести на экран `123` и `456`), используя оператор `for/in`. Воспользоваться методом `hasOwnProperty`.

Часть 6. Использование методов массива

Даны 2 массива:

```
var arr1 = [1,2,3];
var arr2 = [10,5,1];
```

Выполняйте следующие задачи и выводите результат на экран:

1. Воспользуйтесь методом join() для объединения двух массивов
2. Воспользуйтесь методом reverse() для вывода массива в обратном порядке
3. Воспользуйтесь методом sort() для сортировки массива
4. Воспользуйтесь методом slice() для удаления 2-х последних элементов массива
5. Воспользуйтесь методом splice() для замены 2-ого элемента массива на число 100
6. Воспользуйтесь методом unshift() для добавления элемента в начало массива

Часть 7. Массив – это объект.

Определить, правильна ли следующая конструкция. Если да, указать, что будет выведено на экран:

```
<script type="text/javascript">
  a = new Array();
  a[-2]=-2;
  a[-1]=-1;
  a[0]=0;
  a[1]=1;
```

```
  a[2]=2;
  alert("Length: " + a.length);
  for (var i=0; i<a.length; i++) {
     alert(a[i]);
  }
  alert(a[a[-2]+a[1]]);
  alert(a[3]);
</script>
```

Упражнение 6 (модуль 8)
Исключения

Часть 1. Возбуждение исключений.

В примере указать, что будет выведено на экран, объяснить:

```
<script type="text/javascript">
  try {
    alert(variable);
  }
  catch (ex) {
    alert(ex);
  }
  alert("Here");
</script>
```

Часть 2. Перехват исключений.

В примере указать, что будет выведено на экран, объяснить:

```
<script type="text/javascript">
  function f() {
       try {
              throw 1;
       } catch(ex) {
              return 2;
       } finally {
              alert("Here!");
       }
  }
  alert(f());
</script>
```

Часть 3. Использование исключений.

Реализуйте метод для ввода возраста пользователя readAge().

Этот метод должен читать возраст с помощью метода

```
var age = prompt("Введите ваш возраст");
```

В случае, если возраст выходит за рамки 18..90, должно возбуждаться исключение с информацией об ошибке.

В функции getAge() вызывайте функцию readAge() и обрабатывайте исключение, выводя информацию об ошибке на экран.

Часть 4. Использование исключений для предотвращения останова программы.

Создайте заведомо некорректный код, например:

```
document.write("hello");
someerrorcode;
document.write("ok");
```

Используйте исключения, не изменяя основного кода, чтобы на экране появилось и hello, и ok.

Упражнение 7 (модуль 9). Функции.

Цели упражнения.

Развитие навыков декларации функции, рассмотрение особенностей областей видимости функции, работа с аргументами функции, замыкания. Полезные практики работы с функциями.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

Часть 1. Декларация функции.

1. Сравнить два примера анонимной и именованой декларации функции. Указать, будет ли ошибка во втором примере, объяснить причину:

```
/* функция sum определена ниже */
var a = sum(2,2)

function sum(x,y) {
    return x+y
}
```

```
    /* будет ошибка, т.к sum еще не существует */
var a = sum(2,2)


var sum = function(x,y) {
    return x+y
}
```

Часть 2. Возвращаемое значение функции.

1. Что будет присвоено переменной x при различных значениях y?

```
function f(p) {
     if (p>0)  return p;
}

x = f(5);
alert(x);
x = f(-5);
alert(x);
```

Часть 3. Область видимости функции.

1. Следующий пример показывает, что в JS нет понятия «перегруженных» функций. Что выведет на экран данный фрагмент?:

```
function func(param) {
     alert("First declaration");
}
function func() {
     alert("Second declaration");
}
func(2);
```

Часть 4. Область видимости функции.

1. Что выведется на экран в первом и втором примере. Объяснить способ декларации переменной функции:

```
function f() {
     p = 5;
}

f();
alert(p);
```

```
function f() {
     var p = 5;
}

f();
alert(p);
```

Часть 5. Аргументы функции.

1. Дана функция:

```
var run = function(distance, speed) {
     speed = speed || 10;
     var time = distance / speed;
     return time;
}
```

2. Можно ли сделать вызовы run(), run(10), run(10,5), run(10, 5, 2, 7);
3. Какой будет результат в каждом случае (либо ошибка)?
4. Модифицировать код, выводить (alert) количество фактически переданных параметров в функцию.

Часть 6. Аргументы функции.

1. Написать функцию `f()`, которая принимает произвольное число параметров и вычисляет их среднее арифметическое значение (для простоты не проверять, что переданы числа);
2. Вызов `f(1,2,3)` должен возвращать 2;
3. Дан массив `m = [1,2,3]`, подсчитать]]среднее арифметическое его элементов используя функцию `f()`. Подсказка: воспользоваться `apply`.

Часть 7. Сворачивание параметров в объект. Освоить следующую практику.

1. Дана функция:

```
function resize(toWidth, toHeight,
saveProportion, animate)  {
    // значения по умолчанию
    saveProportions = saveProportions || true
    animate = animate || true
    toHeight = toHeight || ...
}
```

2. Вызов `resize(100, null, null, true);`
3. Можно сделать элегантнее:

```
function resize(setup)  {
    // значения по умолчанию
    var saveProportions = setup.saveProportions ||
true
    var animate = setup.animate || true
    var toHeight = setup.toHeight || ...
}
```

4. Теперь вызов функции выглядит:
   `resize({toWidth: 100, animate: true}).`

Часть 8. Замыкания (closures) функции.

1. Что выведет данный пример, объяснить результат:

```
function makeShout() {
    var phrase = "Привет!";
    var shout = function() {
        alert(phrase);
    }
    phrase = "Готово!"
    return shout;
}
shout = makeShout();
// что выдаст?
shout();
```

Часть 9. Замыкания (closures) функции.

1. Используя замыкания написать функцию суммирования sum:
   `sum(2)(2); //4`
   `var plus1 = sum(1);`
   `plus1(3); // 4`

Часть 10. Замыкания (closures) функции.

1. Дана функция:

```
function makeProperty(o, name, predicate) {
    var value;
    o["get" + name] = function() { return value; };

    o["set" + name] = function(v) {
        if (predicate && !predicate(v))
            throw "set" + name + ": invalid value " + v;
        else
            value = v;
    };
}
```

2. Данная функция может быть использована для создания инкапсулированных данных с проверкой их типа;
3. Разобрать и объяснить вызов:

```
var o = {};
makeProperty(o, "Name", function(x) {
   return typeof x ==    "string";
});
o.setName("Frank");
print(o.getName());
o.setName(0);
```

Упражнение 8 (модуль 10). Классы и прототипы.

Цели упражнения.

Усвоить основные принципы создания классов и модели наследования в JavaScript.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

1. Разработать класс `Complex`, представляющий комплексное число. Определить методы `magnitude (квадратный корень из x*x+y*y), negative, add, multiply(x1*x2-y1*y2+(x1*y2+x2*y1)*i), equals, toString, valueOf (вещественная часть числа)` в классе `Complex`, используя prototype. Методы `magnitude, add, multiply` должны возвращать полученный результат, не изменяя состояние объекта, на котором они вызываются, реализуя парадигму immutable. Определить `Complex.ZERO` как свойство класса, представляющее константу `0`. Законченный пример example01.html;
2. Дан класс `Rectangle`:

```
function Rectangle(w, h) {
     this.width = w;
     this.height = h;
}
Rectangle.area = function( ) {
   return this.width * this.height;
}
```

Реализовать класс `PositionedRectangle`, который позволяет задать левый нижний угол прямоугольника, а так же его ширину и высоту. Класс `PositionedRectangle` должен наследовать `Rectangle`, в частности метод `area()`. Класс `PositionedRectangle` должен дополнительно определять метод `contains()`, возвращающий `true`, если заданная точка принадлежит позиционированному прямоугольнику.

Подробное руководство.

Часть 1. Разработка класса `Complex`.

1. Рассмотреть example01.html. Класс `Complex` содержит методы `magnitude (квадратный корень из x*x+y*y), negative, add, multiply(x1*x2-y1*y2+(x1*y2+x2*y1)*i), equals, toString, valueOf;`

2. Определить класс `Complex`, определить конструктор, принимающий вещественную и мнимую части в качестве параметров;
3. С помощью прототипа определить указанные методы. Методы `magnitude, add, multiply` должны возвращать полученный результат, не изменяя состояние объекта, на котором они вызываются, реализуя парадигму `immutable`;
4. Определить `Complex.ZERO` как свойство класса, представляющее константу `0;`
5. Обратить внимание, что в строковой операции `«строка» + complexNumber` вызывается `valueOf()`, так как он определен на объекте. Удалите `valueOf()`, удостоверьтесь, что вызывается `toString()`.

Часть 2. Разработка класса `PositionedRectangle`.

Дан класс **`Rectangle`**:

```
function Rectangle(w, h) {
    this.width = w;
    this.height = h;
}
Rectangle.area = function() {
   return this.width * this.height;
}
```

1. Реализовать класс `PositionedRectangle`, который позволяет задать левый нижний угол прямоугольника, а так же его ширину и высоту;
2. Определить соответствующий конструктор;
3. Класс `PositionedRectangle` должен вызывать конструктор `Rectangle`, передавая ему высоту и ширину прямоугольника, а так же определять свойства `x, y` своей левой нижней координаты. Использовать принцип constructor chaining;
4. Заместить прототип `PositionedRectangle` прототипом `Rectangle`;
5. Заместить свойство `PositionedRectangle.prototype.constructor` правильным конструктором (по умолчанию это свойство ссылается на функцию `Rectangle`);
6. В классе `PositionedRectangle` определить метод `contains()`, возвращающий `true`, если заданная точка принадлежит позиционированному прямоугольнику;
7. Использовать тест Example02.html, а также готовый пример examples\Rect.js в случае необходимости.

Упражнение 9 (модуль 11).
DOM модель.

Цели упражнения.

Освоить упарвление DOM моделью: поиском, созданием, модификацией элементов.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

1. Создать HTML элемент (ссылка), осуществить его поиск по id
2. Изменить значение найденного элемента на другой
3. Изменить значение текстового элемента (текст ссылки) на данный
4. Выполнить поиск элемента, переместить после него другой элемент
5. Создать новый элемент ссылку, поместить его перед данным элементом

Подробное руководство.

Часть 1. Поиск элемента дерева.

1. Дан фрагмент HTML кода:

```
<div id="codesection">
<ul>
   <li>
     <a href="http://www.sitepoint.com/" id="splink">
       SitePoint
     </a>
   </li>
   <li>
     <a href="http://www.yahoo.com/" id="yalink">
       Yahoo!
     </a>
   </li>
</ul>
</div>
```

2. Создать HTML документ, в него вставить фрагмент, форму и кнопку;
3. Определить функцию - обработчик нажатия кнопки;
4. В функции вывести `(alert)` значение ссылки `http://www.sitepoint.com/` по id соответствующего элемента.

Часть 2. Изменение атрибута элемента.

1. Расширить предыдущий пример, добавить новую кнопку и обработчик;

2. В найденном элементе, заменить значение ссылки (атрибут `href`) с значения `http://www.sitepoint.com/` на `http://www.google.com/`.

Часть 3. Изменение текстового узла.

1. В предыдущем примере с помощью `sitepoint_link.childNodes[0]` вывести имя ссылки (`SitePoint`);
2. Понять, что `sitepoint_link.childNodes[0]` возвращает текстовый узел (проверить тип элемента);
3. Заменить значение этого текстового элемента (атрибут `nodeValue`) на `"Google"`.

Часть 4. Перемещение элементов .

1. Дан фрагмент:

```
<div id="codesection">
  <p id="codepara">
    <a href="http://www.google.com/" id="splink">Google</a>
  </p>
 <ul>
   <li></li>
   <li>
     <a href="http://www.yahoo.com/" id="yalink">Yahoo!</a>
   </li>
 </ul>
</div>
```

2. Осуществить поиск элемента с id `codepara`; выведите его содержимое
3. Осуществить поиск элемента с id `yalink`; выведите его содержимое
4. С помощью `para.appendChild` переместить ссылку `yahoo.com` из списка в параграф.

Часть 5. Создание элементов.

1. Расширить предыдущий пример, создать фрагмент `<a href="http://www.linux.org/">The Linux operating system</a>`;
2. Поместить эту ссылку перед ссылкой yalink используя метод `para.insertBefore(что,перед_чем)`.

Часть 6. Построение динамического содержания.

1. Рассмотреть пример Examples\Module11\example04.html;
2. Код JS извлекает все заголовки (h1) из тела документа, создает содержание и восстанавливает ссылки на соответствующие разделы из построенного содержания.

Упражнение 10 (модуль 12). Каскадные страницы стилей и динамический HTML.

Цели упражнения.

Выработка навыков управления стилями, спецификация атрибутов: position, z-index, построение меню.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

1. Запустить пример Example12.html. Рассмотреть, выполнить код самостоятельно. Использовать атрибут `position:relative`;
2. Запустить пример Example13.html. Рассмотреть, выполнить код самостоятельно. Использовать атрибут `z-index`;
3. Запустить пример Example14.html. Рассмотреть, выполнить код самостоятельно. Использовать обработчики события `onmouseover` и `onmouseout`, а также изменяя стиль фона (цвет);
4. Запустить пример Example15.html. Рассмотреть, выполнить код самостоятельно. Динамически изменять размер изображения, используя стиль. Изменять значение свойства `innerHtml` текстового поля;
5. Запустить пример Example16.html. Рассмотреть, выполнить код самостоятельно. Меню верхнего уровня определить в виде таблицы. Подменю данного элемента меню верхнего уровня определить также в виде таблицы, которая вложена в ячейку данного элемента. Использовать абсолютное позиционирование и скрыть подменю. JS код должен менять видимость подменю в зависимости от факта наведения мыши на пункт меню верхнего уровня;
6. Рассмотреть пример Example17.html как пример несколько более хитрого управления показом меню. Выполнить самостоятельно в случае необходимости.

Подробное руководство.

Часть 1. Атрибуты позиционирования. `position:relative`.

1. Дана заготовка Exercise01.html, все заголовки расположены по умолчанию (free floating) относительно документа;
2. Используя inline стиль, сдвинуть Header1 вправо на 20 px, а Header2 влево на 20 px. Применить атрибут `position:revative`;
3. Заменить атрибут позиционирования на `position:absolute`, осуществить наложение одного заголовка на другой;
4. Готовый пример в Example12.html.

Часть 2. Атрибут `z-index`.

1. Дана заготовка Exercise02.html;

2. Используя встроенный стиль, определить свойство `z-index` и атрибут позиционирования для изображения так, чтобы оно было помещено на задний план. Т.о. заголовок должен быть написать поверх изображения;
3. Готовый пример в Example13.html;
4. Используя положительное значение `z-index`, поместить изображение на передний план, перекрыв заголовок.

Часть 3. Динамическое изменение стиля.

1. Дана заготовка Exercise03.html, определяющая 3 квадрата и обработчик события `bgChange()`, в который передается значение цвета. Обработчик вызывается при наведении и уводе мыши. События изучаются подробно в последующем модуле;
2. Реализовать функцию `bgChange()`, которая динамически меняет цвет фона документа на цвет, переданный в качестве параметра функции;
3. Готовый пример в Example14.html.

Часть 4. Динамическое изменение стиля. Изменение размера изображения.

1. Дана заготовка Exercise03.html;
2. Реализовать функции `moveover()` и `moveback()`;
3. `moveover()` выполняет поиск изображения и устанавливает его размер в 200x360. Осуществляет поиск элемента с `id="text1"` и устанавливает его свойство `innerHtml` в значение `"Enlarge image!"`;
4. `moveback()` выполняет поиск изображения и устанавливает его размер обратно в 100x180. Осуществляет поиск элемента с `id="text1"` и устанавливает его свойство `innerHtml` в значение `"Original size"`;
5. Готовый пример в Example15.html.

Часть 5. Построение меню.

1. Рассмотреть пример Example16.html
2. Воспользоваться заготовкой Exercise05.html, которая определяет меню верхнего уровня в виде таблицы. Подменю данного элемента меню верхнего уровня определяется также в виде таблицы, которая вложена в ячейку данного элемента;
3. Реализовать функции `showMenu()` и `hideMenu()`, которые осуществляют поиск по `id` таблицы, соответствующей подменю, и устанавливают свойство `visibility` в значения «`visible`» и «`hidden`» соответственно;
4. Установить следующие значения стиля для класса `menu` элемента `table`:
    a. абсолютное позиционирование, для того, чтобы отключить нормальное положение подменю (free floating) относительно ячейки, соответствующей меню верхнего уровня;
    b. скрыть элемент (атрибут `visibility`).
5. Механика меню работает. С помощью CSS можно установить атрибуты, влияющие на внешний вид меня. Рассмотреть готовый пример.

Упражнение 11 (модуль 13). События.

Цели упражнения.

Усвоить модели обработки событий, фазы capturing и bubbling.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

1. Создать простой обработчик, срабатывающий при нажатии ни кнопку и выводящий "Привет". Использовать оригинальную модель.
2. Используя модели DOM Level 2/IE добавить 2 обработчик на нажатие кнопки. Первый выводит "Спасибо", второй "Большое спасибо".
3. Рассмотреть Exercise02.html, освоить схему bubbling, выполнить остановку "всплытия" события.
4. Определить обработчик, срабатывающий при клике на ссылку. Использовать оригинальную модель. Вывести нажатую ссылку `alert()`, но отменить переход по ссылке.

Подробное руководство.

Часть 1. Оригинальная модель.

1. Создать кнопку, повесить обработчик как свойство объекта JavaScript. Функция-обработчик должна выводить `alert("1");`
2. Определить повторно обработчик, выводящий `alert("2");`
3. Наблюдать, что второй обработчик заместил первый.

Часть 2. Модель DOM 2. Установка двух обработчиков.

1. Создать элемент «кнопка» с `id=myElement;`
2. С помощью методов модели DOM Level 2/IE создать 2 обработчика, срабатывающих на нажатие кнопки. Первый выводит `alert('Спасибо!');`, второй `alert('Большое Спасибо!');`
3. Реализовать кросс-браузерный код;
4. Ответ для IE приведен в Exercise01.html.

Часть 3. Модель DOM 2. Порядок срабатывания событий. Схема bubbling, остановка bubbling.

1. Рассмотреть пример Exercise02.html, демонстрирующий фазу bubbling события;
2. Данный пример не выполняется в IE, внести соответствующие изменения;
3. Осуществить остановку bubbling на втором элементе, используя кросс-браузерный код. (Пример взять из презентации)

4. Модифицировать пример так, чтобы на всех элементах при возникновении события срабатывала схема capturing (не будет работать на IE)

Часть 4. Модель DOM 2. Остановка действия по-умолчанию.

1. Дана ссылка:

```
<a href="http://www.google.com" id="myElement2">
Press me<a/>
```

2. Зарегистрировать обработчик, срабатывающий на щелчок по ссылке, используя оригинальную модель:
   `myElement2.onclick=handler`;
3. Получить доступ к элементу, на котором произошел клик и вывести `alert(ссылка)`. Кросс-браузерно;
4. Отменить переход по ссылке, используя модель остановки действия DOM Level 2 (preventDefault(), либо модель IE);
5. Файл Exercise03.html содержит заготовку примера

Упражнение 12 (модуль 14). Формы.

Цели упражнения.

Отработка навыков создания, валидации и механики форм.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

1. Создать форму, определить для нее всевозможные элементы. Зарегистрировать всевозможные обработчики (`onclick, onchange, onfocus, onblur, onselect`) для всех элементов. Выводить произошедшее событие, имя элемента, значение элемента.
2. Рассмотреть часть 2 из подробного руководства
3. Рассмотреть пример Example03.html, реализовать скрипт, переходящий на соседний текстовый элемент при заполнении данного.
4. Реализовать валидацию текстовых полей формы с помощью регулярного выражения, задаваемого с помощью HTML атрибута.

Подробное руководство.

Часть 1. Создание формы, управление событиями.

1. Рассмотреть заготовку Exercise01.html, усвоить правила создания различных элементов формы;
2. Создать функцию `addhandlers(form)`, которая итерируется по всем элементам формы и ставит функцию `report()` как обработчик событий `onclick, onchange, onfocus, onblur, onselect;`
3. Создать функцию `report()`, выводящую произошедшее событие, имя элемента, значение элемента;
4. Готовый пример в Example01.html.

Часть 2. Механика элементов.

1. Дана заготовка Exercise02.html;
2. Задать обработчик на элемент select так, чтобы при выборе браузера в combo box, выбранный элемент отображался в текстовом поле;
3. Готовый пример в Example02.html.

Часть 3. Перемещение фокуса между элементами.

1. Запустить пример Example03.html;
2. Рассмотреть заготовку Exercise03.html, обратить внимание на то, как задаются HTML атрибуты size, tabindex, maxlength на элеменете text формы. При работе с объектом, создадутся соответствующие свойства;
3. Реализовать функцию checkLen(), которая:

a. проверяет, что значение текущего поля равно максимально допустимому для этого поля: `y.length==x.maxLength`
b. определяет индекс следующего текстового поля: `var next=x.tabIndex;`
c. проверяет, если ли такое поле в документе: `next<document.getElementById("myForm").length`
d. переводит фокус на него: `element.focus().`

Часть 4. Полезные скрипты.

1. Рассмотреть примеры Example04.html, Example05.html, пополнить запас знаний web программиста.

Часть 5. Валидация формы.

1. Запустить пример Example06.html. Пример осуществляет валидацию полей формы. Первое поле – любые непробельные символы, второе поле – правильный адрес электроннной почты, третье поле – 5 чисел, последнее поле – произвольная строка;
2. Exercise04.html содержит заготовку упражнения;
3. Реализовать функцию init(), которая:
   a. интегрирует все формы и элементы в них, отбирая только текстовые поля;
   b. получает HTML атрибуты pattern и required и, в случае, если валидация текстового поля необходима, определяет обработчик `e.onchange = validateOnChange;`
   c. устанавливает обработчик формы `f.onsubmit = validateOnSubmit.`
4. Функцию validateOnChange(), которая:
   a. проверяет значение элемента на соответствие regexp шаблону `value.search(pattern) == -1;`
   b. в зависимости от результата проверки установить `textfield.className = invalid|valid.`
5. Функцию validateOnSubmit(), которая:
   a. пробегает все текстовые элементы формы, вызывая для каждого `e.onchange( );`
   b. проверяет результат `e.className == "invalid"`. Если хотя бы один элемент не прошел валидацию – вывести `alert("Ошибка валидации");`
   c. отменить сабмит формы.

Упражнение 13 (модуль 15). Cookie.

Цели упражнения.

Освоить способы установки, чтения и удаления cookie информации.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

Часть 1. Работа с cookie.

1. Создать документ с 3-мя кнопками: «Check cookie», «Set cookie» и «Delete cookie», определить для них 3 обработчика события `onclick`;
2. Нажатие на кнопку «Check cookie». Должно проверять установлен ли элемент cookie username, если да, выводить это значение, если нет – выводить «Имя пользователя не установлено»;
3. Нажатие на кнопку «Set cookie». Должно запрашивать у пользователя его имя (диалог `prompt()`) и устанавливать полученное значение как значение cookie username;
4. Нажатие на кнопку «Delete cookie». Должно удалять значение cookie username;
5. При выполнении упражнения использовать скрипт Cookie.js;
6. Готовый пример расположен в Example01.html.

Упражнение 14 (модуль 16). Управление окном браузера.

Цели упражнения.

Усвоение принципов использования функций `setTimeout()` / `clearTimeout()`, открытия нового окна, переход по-новому URL, свойств объекта `Screen`.

Описание.

Каждая часть этого упражнения представляет отдельную задачу, которая может расширяться в последующих частях.

Задачи упражнения.

1. Определить две кнопки и текстовое поле. При нажатии на первую кнопку инициализируется таймер, вызывающий функцию. Функция записывает число в текстовое поле, и инкрементирует свое значение, затем устанавливает себя на выполнение через 1 секунду. Вторая кнопка останавливает таймер
   Готовый пример в example01.html;
2. Открыть новое окно для http://www.google.com, установить декорацию окна: с тулбаром, меню, скролбаром, высота 400, ширина 400. Рассмотреть прочие доступные значения декорации окна;
3. Определить 2 кнопки. Первая кнопка отображает текущий URL. Вторая кнопка осуществляет переход на http://www.google.com. Готовый пример в example03.html.

Подробное руководство.

Часть 1. Использование `setTimeout()`.

1. Определить две кнопки и текстовое поле;
2. При нажатии на первую кнопку инициализируется таймер, вызывающий функцию `timedCount()`. Функция записывает число в текстовое поле, и инкрементирует свое значение, затем устанавливает себя на выполнение через 1 секунду. Вторая кнопка останавливает таймер;
3. Таймер включается: `t=setTimeout("timedCount()",1000);`
4. Переменную t передать в качестве аргумента в `window.clearTimeout();`
5. Реализовать альтернативный вариант, используя `setInterval();`
6. Готовый пример в example01.html.

Часть 2. Открытие нового окна.

1. Определить кнопку, обработчик – функцию `open_win();`
2. Функция `open_win()` открывает новое окно для `http://www.google.com`, устанавливает декорацию окна: тулбар, меню, скролбар, высота 400, ширина 400;
3. Рассмотреть прочие доступные значения декорации окна;
4. Готовый пример в example02.html.

Часть 3. Свойство `window.location`.

1. Определить 2 кнопки;
2. Первая кнопка отображает текущий URL;
3. Вторая кнопка осуществляет переход на `http://www.google.com`, использовать `window.location`;
4. Готовый пример в example03.html.

Часть 4. Объект `Screen`.

1. Рассмотреть пример example03.html для усвоения доступных свойств объекта Screen.

Упражнение 15.
Финальное упражнение: разработка компонента ввода критерия выбора автомобилей

Цели упражнения.

Закрепить следующие практики в едином упражнении:

1. Получение динамического содержимого с помощью фреймворка DWR;
2. Управление элементами формы;
3. Валидация формы;
4. Управление документом, свойство `innerHTML;`
5. Установка таймера;
6. Создание стиля, используя внешний CSS файл;
7. Управление cookie, сохранение объекта с помощью JSON;
8. Отработка разнообразных конструкций языка JavaScript, включая работу с массивами, объектами, операторами цикла и ветвления.

Описание.

Упражнение представляет собой компонент по вводу критерия для поиска автомобилей.

При выборе автопроизводителя, обновляется список моделей данного автопроизводителя, используется Ajax. При выборе конкретной модели, обновляется список опций – годы выпуска автомобиля данной модели, а также тип коробки передач. При нажатии на кнопку “Submit”, при условии, что критерий поиска введен, происходит считывание введенных параметров поиска, отображение их в виде таблицы. Таблица исчезает через 3 секунды.

Введенные параметры записываются в cookie. При следующем заходе на данный сайт, происходит считывание информации из cookie и установка критерия, для удобства пользователя.

Полученный HTML документ оформляется с использованием внешнего CSS стиля.

Готовое упражнение можно найти в Examples/Module19

Спецификация приложения.

Упражнение состоит из 2 частей – серверная и клиентская. Серверная часть содержит данные об автопроизводителях и моделях автомобилей, также предоставляет сервисы получения этих данных через фреймворк DWR. Серверное приложение написано на Java и выходит за рамки тренинга.

Для запуска сервера необходимо выполнить скрипт Exercises/Module19/run.bat. В браузере следует перейти по URL: http://localhost:8080/ajaxdemo/

Клиентскому скрипту JavaScript доступны следующие методы:

1) `DwrService.getAutomakers()` – возвращает массив объектов `Automaker`. `Automaker` – объект, представляющий автопроизводителя, содержит свойства:

```
Automaker {
     name /*имя автопроизводителя*/,
     models[] /*массив объектов CarModel */
}
```

2) `DwrService.getAutomaker(automakerName)` – возвращает объект `Automaker` по имени
3) `DwrService.getModel(automakerName, modelName)` – возвращает объект `CarModel`, представляющий модель автомобиля.

```
CarModel {
     name /*название модели*/,
     yearFrom /*год, с которого начался выпуск
     модели*/,
     yearTo /*год, до которого продолжался выпуск
     модели*/,
     gearBox /* коробка передач, принимает
     строковые значения MANUAL, AUTOMATIC и BOTH*/
}
```

Задачи упражнения.

Выполнить упражнение самостоятельно в соответствии с описанием, рассмотрением готового примера, а также используя спецификацию удаленных Ajax сервисов, приведенную выше.

Подробное руководство.

1. Перейти к файлу Exercises\Module19\src\main\webapp\index.jsp;
2. Создать элемент форма с именем `carfilter`, в ней элемент select с именем `automaker`, элемент select с именем `model`, элемент select с именем `years`, 2 чекбокса с именем `gearbox`, 2 кнопки «Reset» с id `resetCriteriaBtn` и «Submit to sever» с id submitBtn;
3. Подключить скрипт cars.js, который должен содержать основную логику задачи
4. Файл Exercises\Module19\src\main\webapp\scripts\cars.js содержит спецификацию основных методов, необходимых для выполнения задачи, выполнить их реализацию;
5. Задать стили элементов формы с помощью внешнего css файла в соответствии с примером Examples/Module19