Лабораторная работа №5. URI. Позиционирование.

В данной работе нужно решить две практических задача:

1) Отображение графика указанной функции.
2) Визуализация раздачи карт.
3) При этом решение этих задач следует расположить на одной двухколоночной html-странице.

Теоретическая часть

Параметры URI

Для передач данных в скрипты размещаемые на html - страницах могут применяться так называемые параметры URI.
Параметры URI — это набор символов, следующих в конце адреса после символа `?` (вопросительный знак) и до `#` (хэш-символ) при наличии. Для их считывания в скрипте можно использовать следующий код:

```
function getUrlParameter(sParam) {
 var sPageURL = decodeURIComponent(window.location.search.substring(1)),
 sURLVariables = sPageURL.split('&'), sParameterName, i;

 for (i = 0; i < sURLVariables.length; i++) {
  sParameterName = sURLVariables[i].split('=');
  if (sParameterName[0] === sParam) {
   return sParameterName[1] === undefined ? false : sParameterName[1];
  }
 }
}
```

Например, если в URI указана следующая строка:

```
…../test1.html?x0=300&y0=300&k=1.03
```

то следующий код позволит получить строковые значения параметров `x0`, `y0` и `k`:

```
var x0 = getUrlParameter("x0");
var y0 = getUrlParameter("y0");
var k = getUrlParameter("k");
```

Построение графиков

Далее показано два искусственных метода, позволяющих выводить произвольные рисунки и графики функций.
Первый метод основывается на многократном клонировании тега `<img>` содержащего рисунок `2x2` пикселя (цвет и размер рисунка в пикселях можно задать). При этом в теге `<img>` указываются абсолютные отступы сверху (`margin-top`) и слева (`margin-left`), за счёт них находится нужная позиция для размещения рисунка.
Второй метод основывается на многократном клонировании тега `<div>` содержащего точку (цвет и размер точки также при желании можно задать). При этом в теге `<div>` также как и в предыдущем методе указываются абсолютные отступы сверху (`margin-top`) и слева (`margin-left`), за счёт них находится нужная позиция для размещения точки.
Для этих целей на практике возможно использование и других тегов (например, тег `<p>` с каким-то символом и/или стилем).

```
/*Пример html – страницы (test1.html), на которой отображается график.
Вариант параметров для uri: /test1.html?x0=300&y0=300&k=1.03*/

<!DOCTYPE html>
<html>
 <head><meta charset="UTF-8"><title></title></head>
 <body></body>

<script type="text/javascript">
/*  параметры из URI: ../test1.html?x0=300&y0=300 */
var x0 = Number(getUrlParameter("x0"));
var y0 = Number(getUrlParameter("y0"));
var k = Number(getUrlParameter("k"));
graph(x0,y0);
line(200,300,400,300);
linev(300,100,310);

function xmultx(x) {
 return x * x;
}

function graph(ml,mt) {
 var xstr = "";
 var xstr0 = "<img src='pixel.png' width=2 height=2 style='position:absolute;";
 var x = -3;
 while (x <= 3.1) {
  xstr += xstr0 + "margin-top:" + k*(mt - 20 * xmultx(x)) + "px;margin-left:" +
(20 * x + ml) + "px'>";
  x += 0.1;
 }
 document.write(xstr);
}

function linev(x,y1,y2) {
 var xstr = "";
 //var xstr0 = "<img src='pixel.png' width=1 height=1
style='position:absolute;";
 var xstr0 = "<div style='position:absolute;";
 var y = y1;
 while (y <= y2) {
  xstr += xstr0 + "margin-top:" + (y) + "px;margin-left:" + x + "px'>.</div>";
  y++;
 }
 document.write(xstr);
}

function line(x1,y1,x2,y2) {
 var xstr = "";
 var xstr0 = "<img src='pixel.png' width=1 height=1 style='position:absolute;";
 var k = (y2 - y1) / (x2 - x1);
 var x = x1;
 while (x <= x2) {
  xstr += xstr0 + "margin-top:" + (k * (x - x1) + y1) + "px;margin-left:" + x +
"px'>";
  x++;
 }
 document.write(xstr);
}

function getUrlParameter(sParam) {
 var sPageURL = decodeURIComponent(window.location.search.substring(1)),
 sURLVariables = sPageURL.split('&'), sParameterName, i;
```

```
 for (i = 0; i < sURLVariables.length; i++) {
  sParameterName = sURLVariables[i].split('=');
  if (sParameterName[0] === sParam) {
   return sParameterName[1] === undefined ? false : sParameterName[1];
  }
 }
}

</script>
</html>
```

В браузере часть данной страницы с графиком будет выглядеть примерно следующим образом, рисунок 5.1:

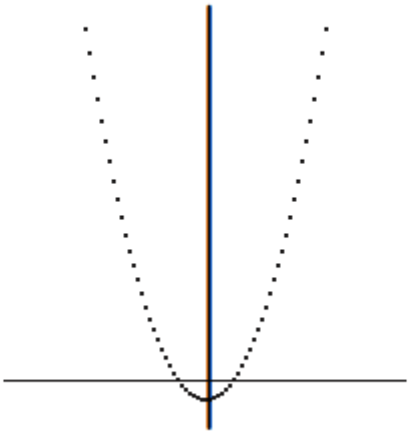

Рис. 5.1 Примерная страница с графиком

Полезно проверить корректность отображения страницы в разных браузерах.

Покер

Массивы: функция map

Функция `map` берет массив и делает из него новый, применяя указанную в её параметре функцию к каждому элементу. В следующем примере функция, которая возвращает `num * num`, применяется к каждому элементу для создания нового массива.

```
// создаем массив из чисел
var nums = [1, 2, 3, 4, 5];
// применим функцию map
// для создания нового массива
var squares = nums.map(function (num) {
 return num*num;
});
print(squares);

/*
1,4,9,16,25
*/
```

Ещё пример:

```
// создаем массив имен
var names = [ "эмили", "марк", "брюс", "андреа", "пабло" ];
// а сейчас мы создадим массив,где первая буква каждого имени прописная
var capitalizedNames = names.map(function (name) {
 // получаем первую букву
 var firstLetter = name[0];
 // возвращаем ее в виде прописной в строку, начинающуюся с индекса 1
 return firstLetter.toUpperCase() + name.substring(1);
});
print(capitalizedNames);

/*
Эмили,Марк,Брюс,Андреа,Пабло
*/
```

Как видите, мы создали массив из имён с первой прописной буквой, даже не перебирая элементы самого исходного массива!

Библиотеки функций

Обычно разработка проектов связана с написанием ряда библиотек функций, обеспечивающих функциональное ядро разрабатываемого приложения. Далее излагается пример, который может быть положен в основу такой библиотеки.

Определение покерных раздач

Все покерные раздачи базируются на комбинациях, которые могут появляться в раздаче пяти игральных карт. Комбинации могут быть такие:

- пара — две карты одного номинала;
- две пары — две карты одного номинала и ещё две карты другого;
- тройка — три карты одного номинала;
- стрит — пять карт, идущих по порядку, причём туз может считаться как тузом (следующим после короля), так и единицей;
- флеш — пять карт одной и той же масти;
- фулл-хауз — две карты одного номинала и три карты другого номинала;
- каре — четыре карты одного номинала;
- стрит-флеш — пять карт одной масти, номиналы которых идут по порядку;
- роял-флеш — стрит-флеш, в которой последовательность начинается с десятки и заканчивается тузом.

Любая другая раздача, где нет таких комбинаций, называется «нет игры». Обратите внимание на то, что комбинации не эксклюзивны — например, раздача, содержащая тройку, автоматически содержит и пару, а фулл-хауз содержит две пары.
Таким образом, задача, которой посвящён данный пример, состоит в определении типа покерной раздачи.
Посмотрим, как может выглядеть раздача:

```
var hand = [
 { "rank":"двойка", "suit":"пик" },
 { "rank":"четверка", "suit":"червей" },
 { "rank":"двойка", "suit":"треф" },
 { "rank":"король", "suit":"пик" },
 { "rank":"восьмерка", "suit":"бубен" }
];
```

В нашем примере раздача содержит пару двоек. Следует отметить, что на практике следует создать примеры для всех остальных типов комбинаций, так как они будут полезны при отладке разрабатываемого приложения.

Следующий важный шаг — это разработка набора вспомогательных функции, обеспечивающих определение типов комбинаций карт. Например, полезной будет такая функция, которая принимает три аргумента: массив, элемент для поиска и минимальное количество раз повторения этого элемента в массиве. Также возможно будет полезной функцию, которая принимает те же три аргумента и возвращает `true`, если массив содержит указанный элемент строго `n` раз.
А сейчас представьте, что мы отправляем первой из упомянутых функций массив карточных номиналов:

```
// тут у нас две двойки
containsNTimes(["двойка","четверка","двойка","король","восьмерка"], "двойка",
2);
//=> true
```

Это говорит нам, что в массиве с номиналами есть две двойки!
Мы можем использовать тот же принцип для определения наличия тройки или каре.

```
// тройки из двоек здесь нет
containsNTimes(["двойка","четверка","двойка","король","восьмерка"], "двойка",
3);
//=> false
```

Таким образом, мы можем частично свести поставленную задачу к представлению массива карточных объектов в виде массива карточных номиналов. Для этого очень легко и удобно использовать функцию `map`:

```
var hand = [
 { "rank":"двойка", "suit":"пик" },
 { "rank":"четверка", "suit":"червей" },
 { "rank":"двойка", "suit":"треф" },
 { "rank":"король", "suit":"пик" },
 { "rank":"восьмерка", "suit":"бубен" }
];
var r = hand.map(function (card) {
 return card.rank;
});
print(r);

/* двойка,четверка,двойка,король,восьмерка */
```

Результат работы вызова функции `map` мы можем обработать как переменную, а затем отправить его в другую функцию `containsNTimes`, чтобы определить, есть ли там пара двоек:

```
var hand = [
 { "rank":"двойка", "suit":"пик" },
 { "rank":"четверка", "suit":"червей" },
 { "rank":"двойка", "suit":"треф" },
 { "rank":"король", "suit":"пик" },
 { "rank":"восьмерка", "suit":"бубен" }
];
var handRanks, result;
handRanks = hand.map(function (card) {
 return card.rank;
});
result = containsNTimes(handRanks, "двойка", 2);
print(result);
```

```
/* true */
```

Целесообразным является создание массива для всех возможных номиналов, а затем использование цикла для определения, например, того, содержит ли раздача пару одинаковых значений из этого массива. Далее показано, как можно задать и использовать этот массив:

```
// это все возможные номиналы
var ranks = ["двойка","тройка","четверка","пятерка","шестерка",
"семерка","восьмерка","девятка","десятка","валет","дама","король","туз"];

var hand = [
 { "rank":"семерка", "suit":"пик" },
 { "rank":"четверка", "suit":"червей" },
 { "rank":"двойка", "suit":"треф" },
 { "rank":"семерка", "suit":"пик" },
 { "rank":"восьмерка", "suit":"бубен" }
];

function containsNTimes(handRanks, rank, n) {
 var count = 0;
 for (var i = 0; i < handRanks.length; i++)
  if (handRanks[i] == rank) count++;
 return n <= count ? true : false;
}

function containsPair(hand) {
 // предположим, что пар нет, пока не доказано обратного
 var result = false,
     handRanks;
 // создаем массив карточных номиналов
  handRanks = hand.map(function(cards) { return cards.rank; });
 // поиск пары любого номинала
 for (var i = 0; i < ranks.length; i++)
  if (containsNTimes(handRanks, ranks[i], 2)) {
   result = true; // мы нашли пару
  }
 // в результате получаем true, если пара найдена,
 // и false, если нет
 return result;
}

print(containsPair(hand));

/*
true
*/
```

После рассмотрения этого примера вам будет значительно проще написать функции для всех остальных комбинаций (например, `containsThreeOfAKind` для проверки наличия тройки). Для двух пар и фулл-хауза понадобится отслеживать значения номиналов элементов, которые вы обнаружили. А для флеша нужно будет извлекать из массива объектов масти, а не номиналы.

Случай стрита несколько сложнее, но я дам вам подсказку: стрит не содержит пар (что вы можете определить в результате работы функции `containsPair`, использовав оператор !), а разница между низшей и высшей картами составляет 4.

Лучше всего будет написать набор вспомогательных функций для нахождения высшего и низшего номинала карт в виде чисел (валет в данном случае можно считать за 11, даму — за 12, короля — за 13, а туз — за 14). После того как у вас будут эти функции, вы сможете

определить, является ли раздача стритом, подтвердив, что отсутствуют пары, а разница между низшей и высшей картами составляет 4.
Затем можно будет создать и остальные функции, используя те, что у вас уже есть. Например, стрит-флеш автоматически содержит стрит, а роял-флеш является стрит-флешем, где высшая карта — туз (или низшая — 10).

Практическая часть

График

1. Ознакомиться с теоретическим материалом и практическими примерами лабораторной работы.
2. Обеспечить отображение графика определённой функции без использования специализированных библиотек, обеспечивающих построение диаграмм и сводных таблиц.
3. График должен располагаться в выделенной области на странице (например 300px на 300px или, например, 400px на 500px).
4. Следует обеспечить защиту от неверного ввода через параметры URI.
5. Обеспечить масштабирование графика функции через:
   1 — коэффициент пропорциональности указываемый в параметре URI[7],
   2 — кнопки или другие удобные интерактивные, кликабельные элемент поочерёдное нажатие на которые приводит к увеличению либо уменьшению масштаба графика.
6. Обеспечить сдвиг графика функции влево и вправо через:
   1 — параметр указываемый в параметре URI (как смещение влево — отрицательное число пикселей, смещение вправо — положительное число пикселей),
   2 — кнопки или другие удобные интерактивные, кликабельные элементы, которые работают в паре: нажатие на одну приводит к смещению влево, нажатие на другую приводит к смещению — вправо.
7. Обеспечить сдвиг графика функции вверх и вниз через:
   1 — параметр указываемый в параметре URI (как смещение вверх — отрицательное число пикселей, смещение вниз — положительное число пикселей),
   2 — кнопки или другие удобные интерактивные, кликабельные элементы, которые работают в паре: нажатие на одну приводит к смещению вверх, нажатие на другую приводит к смещению — вниз.
8. На графике должна присутствовать координатная сетка (по умолчанию — это шесть горизонтальных линий и шесть вертикальных линий, оси координат с указанием стрелок и надписей осей и цифр), для улучшения восприятия графика допускается увеличивать либо уменьшать число ячеек сетки:
   1 — сплошная сетка,
   2 — штрих пунктирная.
9. Обеспечить масштабирование координатной сетки при изменении масштаба графика.
10. Математическая формула для отображения графика функции:
    1 — x^2+x^3
    2 — $\sin^2(x)$
    3 — $\cos^2(1-x)$
    4 — $2-x^3$
    5 — $\ln^2(1-x)$
    6 — $4-\ln(3\cdot x)$
    7 — $e^{2\cdot x-6}$
    8 — $x-2\cdot e^x$
    9 — $x^4+8\cdot x^2$
    10 — $23\cdot x^{2-}x^3$
    11 — $tg^2(x)$
    12 — $ctg(1-x)$
    13 — $x\cdot\cos(1-x)$

7 Рекомендуется использовать пару: «наименование параметра» и «значение параметра». Обеспечить защиту от неверного ввода.

14 — $(1-x)\cdot\cos(x+8)$
15 — $x^2\cdot\cos(2-x)$
16 — $2\cdot x\cdot\sin(1-x^2)$

Покер

1. Ознакомиться с теоретическим материалом и практическими примерами лабораторной работы.
2. Обеспечить генерацию раздачи:
   1 — вручную (либо через текстовые поля, либо через выпадающие списки, либо через параметры URI, либо одной строкой, в которой как-то закодирована раздача, в общем — как вам удобно, но ручками);
   2 — случайным образом (генерируется по нажатию на кнопку либо какого-то другого кликабельный элемент и отображается на странице). При этом возможно два случая отображения раздачи: в не редактируемых текстовых полях либо в виде изображений картинок карт (рекомендуется).[8]
3. Выявить раздачи двух указанных типов:
   1 — две пары — две карты одного номинала и ещё две карты другого;
   2 — тройка — три карты одного номинала;
   3 — стрит — пять карт, идущих по порядку, причём туз может считаться как тузом (следующим после короля), так и единицей;
   4 — флеш — пять карт одной и той же масти;
   5 — фулл-хауз — две карты одного номинала и три карты другого номинала;
   6 — каре — четыре карты одного номинала;
   7 — стрит-флеш — пять карт одной масти, номиналы которых идут по порядку (туз может считаться как тузом (следующим после короля), так и единицей);
   8 — роял-флеш — стрит-флеш, в которой последовательность начинается с десятки и заканчивается тузом.
4. Использовать для перебора элементов массивов функцию `map`:
   1 — нет;
   2 — да.
5. Использовать для демонстрации случая «нет игры» механизм обработки исключений:
   1 — без явного использования инструкции `try/catch/finally` (только `throw`);
   2 — с явным использованием инструкции `try/catch/finally`.
6. Информацию о том, что раздача соответствует случаю «нет игры» выводить в:
   1 — консоль;
   2 — на страницу либо в абзац либо в не редактируемое текстовое поле.
7. Продемонстрировать в вашем коде использование одного из следующих методов (что не запрещает использование любого из остальных):
   1 — `join()`;
   2 — `sort()`;
   3 — `slice()`;
   4 — `splice()`.
   Рекомендуется, чтобы это использование было вполне осмысленным.

8 Архив изображений карт размещён в ББ. Допускается использовать свои изображения карт.

Задания на лабораторную работу

Табл. №5.1

	График					**Покер**					
№	(П.5)	(П.6)	(П.7)	(П.8)	(П.10)	(П.2)	(П.3)	(П.4)	(П.5)	(П.6)	(П.7)
1	2	2	1	2	4	2	5	1	1	2	2
2	1	2	1	1	4	2	5	2	1	2	3
3	2	2	2	1	2	1	6	1	2	1	3
4	2	2	2	2	8	1	7	2	1	2	1
5	1	2	1	1	12	2	6	1	2	2	4
6	1	1	1	1	2	1	1	2	1	2	2
7	2	1	2	2	15	2	2	1	2	2	2
8	2	2	2	1	9	2	6	1	2	2	3
9	2	1	1	1	8	1	3	1	2	1	2
10	2	1	1	1	16	1	5	1	1	2	3
11	1	2	2	1	10	1	6	1	2	2	2
12	1	2	2	2	4	2	4	2	2	2	4
13	1	1	2	2	7	1	6	2	1	2	2
14	1	1	2	2	4	1	1	1	1	2	1
15	2	1	2	2	16	2	7	1	1	1	2
16	1	1	1	2	13	2	8	1	1	2	3
17	1	1	2	2	15	2	8	2	2	1	2
18	2	2	2	2	11	1	8	2	2	2	1
19	2	1	1	2	2	2	3	1	1	2	2
20	2	1	1	1	2	2	3	2	1	2	1
21	1	1	1	1	8	2	5	1	1	2	4
22	2	1	2	2	8	2	4	1	1	2	3
23	1	2	2	1	16	1	3	1	1	2	1
24	1	1	1	1	9	2	2	2	1	2	3
25	2	2	1	1	14	1	5	2	1	1	3
26	2	1	1	2	14	2	2	1	1	2	2
27	1	1	2	1	12	1	2	1	1	1	4
28	2	2	1	1	4	1	2	1	1	1	2
29	1	1	2	2	3	2	2	2	1	1	1
30	1	1	1	1	16	2	3	1	1	2	3
31	2	2	2	2	12	2	4	2	2	2	3
32	1	2	2	1	14	2	1	2	1	1	3
33	2	1	1	2	8	1	3	2	1	2	3
34	1	1	2	1	7	1	2	1	2	1	4
35	1	2	1	2	5	2	8	2	1	1	2
36	2	2	2	1	10	2	3	1	2	2	3
37	2	2	1	1	15	2	5	2	1	1	2
38	2	1	1	1	6	1	5	1	1	2	4
39	2	2	2	1	11	2	3	2	1	1	3
40	1	1	2	2	9	1	4	2	1	2	3
41	1	2	2	1	13	2	4	1	1	1	2
42	2	2	2	2	8	1	8	1	2	1	2
43	2	2	2	1	3	1	6	2	1	2	2
44	1	2	2	1	10	1	5	1	1	2	2
45	1	1	1	1	1	2	7	2	2	2	3
46	1	1	2	2	14	2	7	2	2	2	3
47	1	1	1	2	10	2	5	1	1	2	3

48	1	1	2	2	2	2	8	2	2	2	2
49	1	1	1	1	1	1	2	2	2	1	3
50	1	1	2	2	10	2	1	1	1	2	4
51	2	1	2	2	7	2	6	1	1	1	2
52	2	1	1	2	6	1	2	1	1	1	4
53	2	1	1	1	15	1	4	1	2	2	2
54	1	1	1	2	10	2	5	1	2	1	3
55	1	1	1	1	5	1	4	1	1	1	2
56	1	2	1	2	2	1	1	1	1	2	2
57	2	2	2	2	10	2	7	1	1	1	2
58	1	1	1	2	3	2	4	2	2	1	2
59	2	2	1	2	3	2	4	1	2	1	3
60	2	1	2	1	10	1	3	1	1	1	1
61	2	2	2	1	7	1	4	2	1	1	1
62	1	1	1	1	6	2	4	2	1	2	2
63	2	1	2	1	13	1	1	2	2	2	4
64	1	2	2	2	15	1	2	2	1	1	4
65	2	1	2	1	2	1	2	1	2	1	3
66	2	1	1	1	11	1	5	1	1	2	1
67	2	2	1	1	12	1	5	2	2	2	3
68	2	2	1	2	7	1	4	1	2	1	3
69	2	2	1	2	14	2	5	1	2	1	2
70	2	2	1	1	12	1	1	1	1	1	3
71	1	1	2	1	7	1	5	1	2	2	4
72	2	2	1	2	6	1	1	1	1	1	4
73	2	2	2	1	2	1	8	1	1	2	3
74	2	1	2	2	1	1	1	2	2	1	1
75	1	1	2	1	9	2	2	1	1	1	1
76	1	1	1	2	13	2	2	1	1	2	3
77	2	2	2	2	2	2	7	1	1	1	3
78	1	2	2	2	5	2	7	1	1	1	4
79	2	1	2	2	15	1	3	2	2	1	1
80	1	1	2	2	10	1	5	1	2	2	2
81	1	1	2	1	8	1	3	1	1	2	3
82	2	2	2	2	6	1	5	1	1	2	3
83	1	1	1	2	3	2	4	2	2	1	2
84	2	1	2	1	13	1	4	2	2	1	1
85	2	1	2	1	12	2	3	1	1	1	3
86	2	1	2	1	4	2	7	2	2	1	2
87	1	1	2	1	16	1	8	2	1	2	2
88	2	2	2	2	10	1	5	1	2	2	2
89	2	2	2	2	6	2	4	2	1	2	2
90	2	1	2	2	5	2	6	1	2	1	2
91	1	2	2	1	2	1	8	1	1	1	2
92	1	2	2	2	11	1	6	2	1	1	2
93	2	2	1	1	5	2	8	2	2	2	3
94	2	2	2	1	13	2	3	1	1	1	2
95	2	1	2	2	13	1	3	1	1	1	2
96	2	1	1	2	2	1	6	2	2	2	1
97	2	1	1	1	4	1	6	1	1	1	3
98	2	2	1	1	2	2	8	2	2	1	3
99	2	2	2	2	9	1	6	1	2	2	3

100	1	1	1	2	14	1	7	1	1	2	3