Draw/Error/Storage

JavaScript
Courses

vk.com/**js.courses**

js.courses.dp.ua/files

Локальное хранилище

```
1  <script>
2      function pay(){
3          var phone = localStorage.getItem("x_phone");
4
5          var phone = prompt("Введите номер для пополенения:", phone);
6          alert("Номер " + phone + " пополнен!");
7
8          localStorage.setItem("x_phone", phone);
9      };
10
11 </script>
12 <button onclick="pay()">Пополнить телефон</button>
```

localStorage *– объект который позволяет хранить данные на стороне клиента, т.е. в браузере, серверная часть (даже если она есть) не будет иметь доступ к этим данным. Данные которые хранит объект* ***localStorage*** *доступны всем закладкам которые содержат страницы с одного домена.*

Локальное хранилище

```
1  <script>
2      function pay(){
3          var phone = localStorage.getItem("x_phone");
4
5          var phone = prompt("Введите номер для пополенения:", phone);
6          alert("Номер " + phone + " пополнен!");
7
8          localStorage.setItem("x_phone", phone);
9      };
10
11 </script>
12 <button onclick="pay()">Пополнить телефон</button>
```

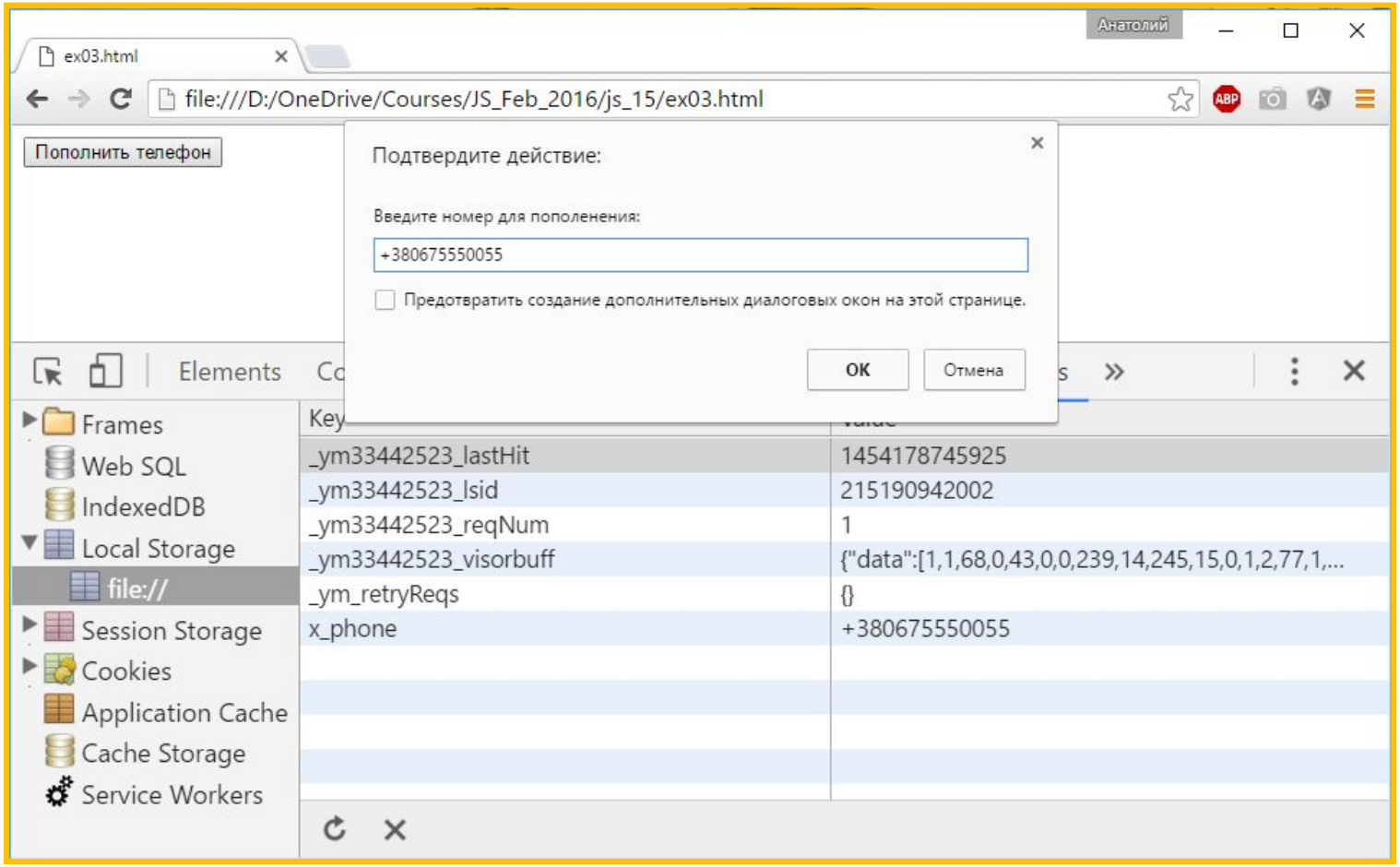
???

*Работа с объектом **localStorage:***

localStorage.setItem('ключ', значение) *– функция задающая ключ (имя) и значение которое нужно сохранить (тот же принцип, что и в ассоциативных массивах);*

localStorage.getItem('ключ') *– функция достающая из локального хранилища значение которое сохранено под именем заданным ключом.*

Локальное



Консоль разработчика покажет всё, что содержится в локальном хранилище.

Локальное

```
<script>
    window.onload = function(){

        var div          = document.querySelector("div");

        var counter      = Number(localStorage.getItem("counter"));
        var last_vizit   = localStorage.getItem("last_vizit");

        counter++;
        div.innerHTML = "Посещений: " + counter + "; Последний визит: " + last_vizit;

        localStorage.setItem("counter", counter);
        localStorage.setItem("last_vizit", (new Date).toLocaleString());
    }

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

???

Посещений: 49; Последний визит: 19.04.2016, 12:57:31

counter	49
last_vizit	19.04.2016, 12:58:04

Локальное хранилище сохраняет данные доступные, даже после закрытия браузера, этим можно пользоваться для сохранения промежуточных значений, или данные которые с целью безопасности лучше не передавать на сервер.

Локальное хранилище

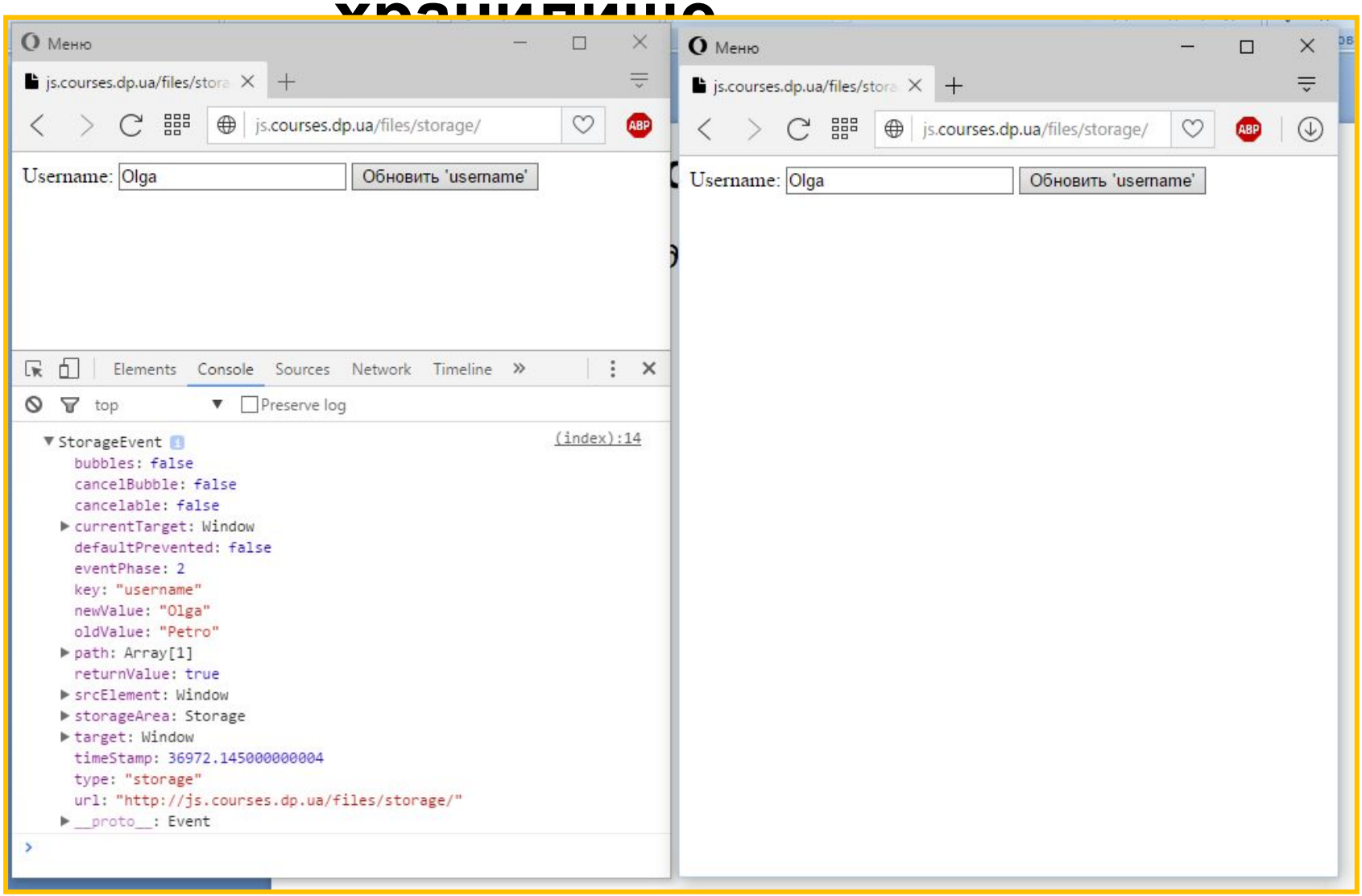
Данные из локального хранилища доступны для всех вкладок браузера в которых открыты страницы одного и того же домена. Но когда в одной из вкладок меняется какое-либо из значений хранимых в локальном хранилище, то хорошо бы уведомить осп

```
<html>
<head>
<script>

    window.onload = function(){
        document.querySelector("#username").value = localStorage.getItem("username");
        document.querySelector("button").onclick = function(){
            localStorage.setItem("username", document.querySelector("#username").value);
        }
    }

    window.onstorage = function(e){
        document.querySelector("#username").value = e.newValue;
        console.dir(e);
    }

</script>
</head>
<body>
    Username: <input type="text" id="username">
    <button>Обновить 'username'</button>
</body>
</html>
```

Такая возможность есть – события ***onstorage*** *объекта* ***window***.

http://js.courses.dp.ua/files/storage/

Локальное хранилище



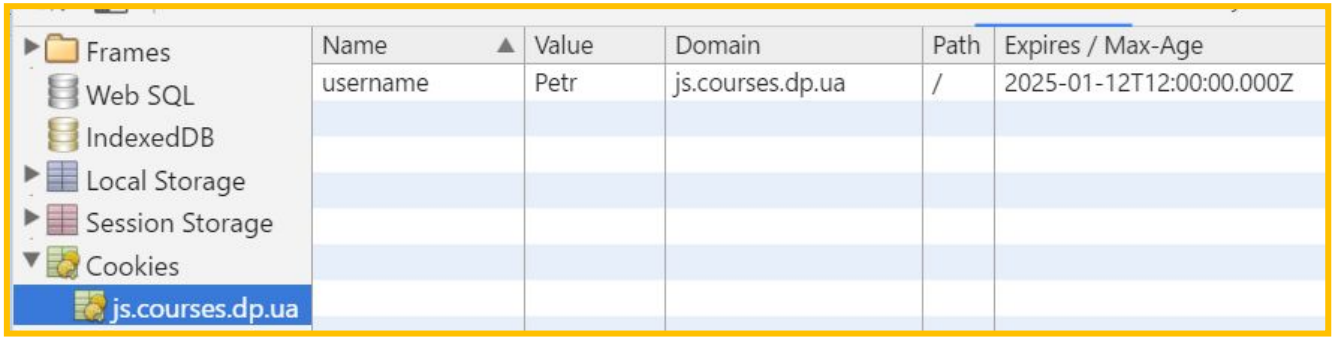
*Уведомление об изменении **localStorage**, событие **onstorage**.*

Cookie-файлы

Поскольку cookie-файлы хранятся в браузере, и передаются на сервер только в момент запроса страницы, то JavaScript позволяет их менять.

```
1  <script>
2
3      var cookie_data = document.cookie;
4      console.log("Полученные cookie: " + cookie_data);
5
6      var username = prompt("Введите имя пользователя: ");
7
8      document.cookie = "username=" + username + "; expires=Sun, 12 Jan 2025 12:00:00 UTC; path=/";
9      console.log("Установлен cookie: " + document.cookie);
10
11 </script>
```

```
Полученные cookie: username=Ivan                    cookie.html:4
Установлен cookie: username=Petr                    cookie.html:9
```



http://js.courses.dp.ua/files/storage/cookie.html

Обработка ошибок

```
1  <script>
2
3      var a = 35;
4
5      a();
6
7      console.log("after a();");
8
9  </script>
```

?!?

Обработка ошибок

```
1  <script>
2
3      var a = 35;
4
5      try{
6
7          a();
8
9      }catch(error_info){
10         console.dir(error_info);
11     }
12
13     console.log("after a();");
14
15 </script>
```

???

```
▼ TypeError: a is not a function at file:///D:/OneDrive/Courses/JS_Feb_2016/js_15/ex06.html:7:3   ex06.html:10
    message: "a is not a function"
    stack: "TypeError: a is not a function↵    at file:///D:/OneDrive/Courses/JS_Feb_2016/js_15/ex06.html:7:3"
  ▶ get stack: function get stack()
  ▶ set stack: function set stack()
  ▶ __proto__: Error
after a(); 35                                                                          ex06.html:13
```

Обработка ошибок

*Блок **try{ … } catch(){}** предназначен для перехвата и обработки ошибок (**исключительных ситуаций, exeptions**) времени выполнения (т.е. те ошибки которые появились во время работы скрипта).*

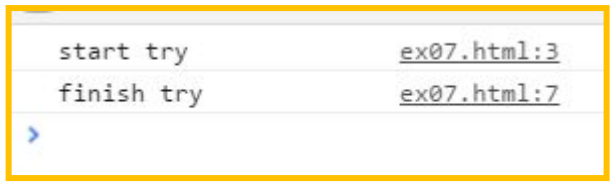
```
<script>

    var a = 35;

    try{

        document.querySelector(a);

    }catch(error_info){
        console.dir(error_info);
    }

    console.log("after a();");

</script>
```

*Если в блоке **try** произойдёт ошибка, выполнение блока прекратится и перейдёт к блоку **catch**, в котором могут быть выполнены какие-либо действия направленные на нивелирования влияния ошибки на работу сприпта. Если в блоке **try** ошибка не произошла, то блок **catch** не выполняется. Независимо от того произошла ошибка или нет, после try-catch скрипт пойдёт выполняться дальше, как ни в чём не бывало.*

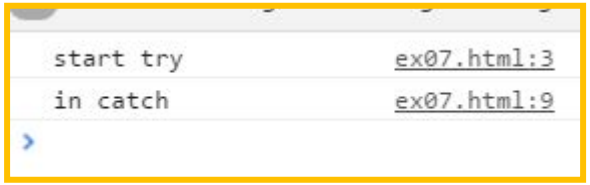
Обработка ошибок

*При ошибке скрипт не перестаёт выполнятся, а перескочит в блок catch, если ошибка не случилась, то блок **catch** не будет выполняться вообще*

```
<script>
    try{
        console.log("start try");

        //..code..

        console.log("finish try");
    }catch(e){
        console.log("in catch");
    }
</script>
```



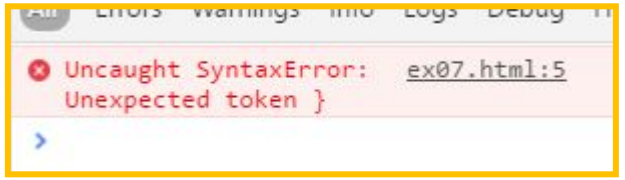

```
<script>
    try{
        console.log("start try");

        blabla();

        console.log("finish try");
    }catch(e){
        console.log("in catch");
    }
</script>
```



Обработка ошибок

Перехват ошибок возможен только для синтаксически верного кода

```
1  <script>
2      try{
3          var a = 35;
4
5          a( };
6
7      }catch(e){
8
9      }
10 </script>
```



В приведенном примере перехват ошибок, не поможет

Обработка ошибок

*Информация об ошибке содержится в параметре который получает блок **catch**.*

```
1  <script>
2      try{
3          var a = 35;
4
5          document.querySelector(a);
6
7      }catch(e){
8          console.dir(e);
9      }
10 </script>
```

```
                                                                    ex07.html:8
▼ DOMException: Failed to execute 'querySelector' on 'Document': '35' is not a valid selector. at Er
    code: 12
    message: "Failed to execute 'querySelector' on 'Document': '35' is not a valid selector."
    name: "SyntaxError"
    stack: "Error: Failed to execute 'querySelector' on 'Document': '35' is not a valid selector.↵
  ▶ __proto__: Error
>
```

Обработка ошибок

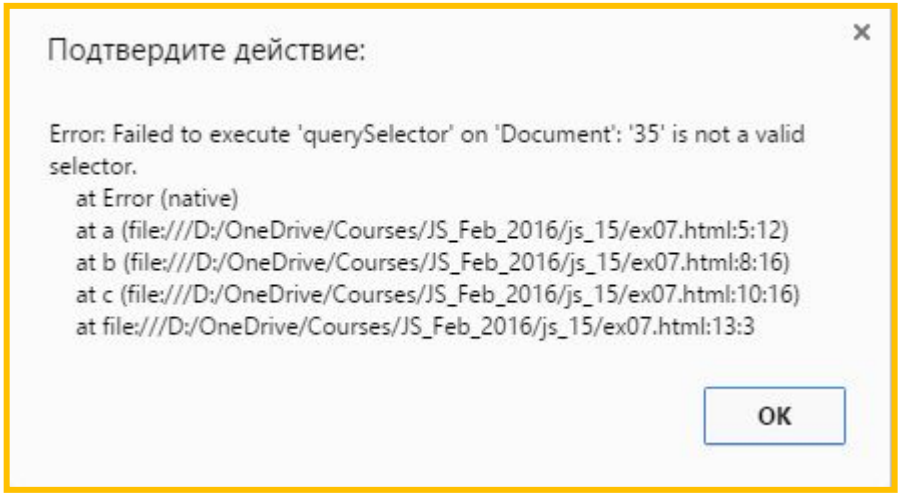
Часто ошибка случается где-то «в глубине»:

```
<script>

    function a(){
        var a = 35;
        document.querySelector(a);
    }

    function b(){ a(); };

    function c(){ b(); };

    try{
        c();
    }catch(e){
        console.dir(e);
        alert(e.stack);
    }
</script>
```

*Свойство **stack** объекта который содержит информацию об ошибке содержит описание цепочки вызовов которая привела к ошибке.*

Обработка ошибок

stack *– цепочка вызовов приведшая к ошибке*



Обработка ошибок

Не поможет обрабатывать код вызываемый асинхронно.

?!?

```
1  <script>
2
3      try{
4          setTimeout(function(){
5              aaa();
6          }, 3000);
7      }catch(e){
8          console.log("Error catch!");
9      }
10
11 </script>
```

```
Uncaught ReferenceError: aaa is not defined                ex08.html:5
```

Обработка ошибок

Не поможет если код который содержится в try-catch вызывается из вне.

?!?

```
1  <script>
2      try{
3          function test(){
4              console.log("In test();");
5              aaa();
6          };
7      }catch(e){
8          console.log("Error catch!");
9      }
10
11     test();
12 </script>
```

```
In test();                                              ex08.html:5
▶ Uncaught ReferenceError: aaa is not defined           ex08.html:6
```

Обработка ошибок

*Секция **finally** предназначен для размещения в нём кода который выполнится не зависимо от того, возникла ошибка в блоке **try** и выполнялся ли блок **catch** или нет.*

```
1  <script>
2      try{
3          console.log("try start;");
4          //...code...
5          console.log("try finish;");
6      }catch(e){
7          console.log("in catch;");
8      }finally{
9          console.log("in finally;");
10     }
11 </script>
```

```
try start;          ex08.html:3
try finish;         ex08.html:5
in finally;         ex08.html:9
>
```

```
1  <script>
2      try{
3          console.log("try start;");
4          aaa();
5          console.log("try finish;");
6      }catch(e){
7          console.log("in catch;");
8      }finally{
9          console.log("in finally;");
10     }
11 </script>
```

```
try start;          ex08.html:3
in catch;           ex08.html:7
in finally;         ex08.html:9
>
```

*Секция **finally** используется для того, чтобы завершить начатые операции при любом развитии событий*

Обработка ошибок

Ну вот ничего не помогло…

```
1  <script>
2      window.onerror = function(message, url, lineNumber) {
3          console.log("Глобальная ошибка!\n" +
4              "Сообщение: " + message + "\n(" + url + ":" + lineNumber + ")");
5      };
6
7    aaa();
8
9    console.log("После aaa();");
10 </script>
```

?!?

```
Глобальная ошибка!                                          ex09_winerror.html:3
Сообщение: Uncaught ReferenceError: aaa is not defined
(file:///D:/OneDrive/Courses/JS_Feb_2016/js_15/ex09_winerror.html:7)
Uncaught ReferenceError: aaa is not defined                 ex09_winerror.html:7
>
```

*Обработка события **window.onerror** – не предотвратит прекращение работы скрипта, но позволит выбросить информацию об ошибке которая привела к его «падению».*

Рисование, Графика,

```
<!DOCTYPE html>
<html>
<body>
    <canvas id="myCanvas" width="600px" height="450px" style="border:1px solid gray;">
        Your browser does not support the canvas element.
    </canvas>
    <script>
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        console.dir(context);
        //После этой строки будем писать код.

        //И вот до этой строки.
    </script>
</body>
</html>
```

*Тег **canvas** – представляет собой «**холст**», прямоугольную область в которой можно рисовать. **Контекст canvas'а** – объект который содержит множество методов для рисования на «**холсте**».*

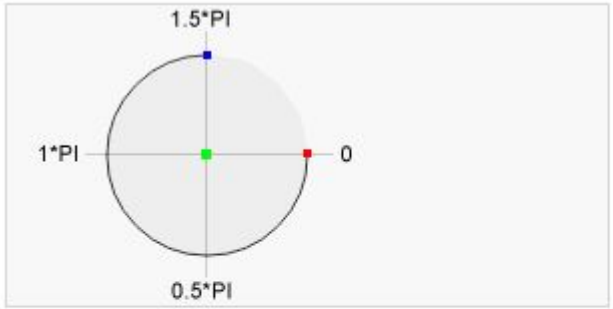
http://js.courses.dp.ua/files/canvas/example.html

Рисование примитивов

```
context.strokeStyle = "green";
context.rect(300, 200, 50, 80);
//rect(x,y, width, height)
context.stroke();
```

???

```
context.arc(200,300,45, 1.8*Math.PI, 2*Math.PI, false);
//arc(x, y, radius, startAngle, finishAngle, direction);
context.stroke();
```



- Center arc(**100,75**,50,0*Math.PI,1.5*Math.PI)
- Start angle arc(100,75,50,**0**,1.5*Math.PI)
- End angle arc(100,75,50,0*Math.PI,**1.5*Math.PI**)

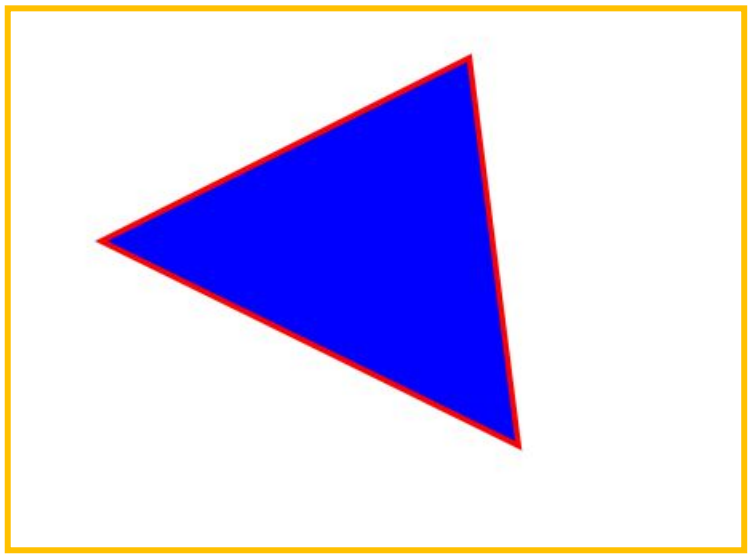
Рисование тивов

```
context.beginPath();
context.moveTo(175, 225);
context.lineTo(400,113);
context.lineTo(430, 350);
//context.closePath();
context.stroke();
```

```
context.beginPath();
context.moveTo(175, 225);
context.lineTo(400,113);
context.lineTo(430, 350);
//context.closePath();
context.fill();
```

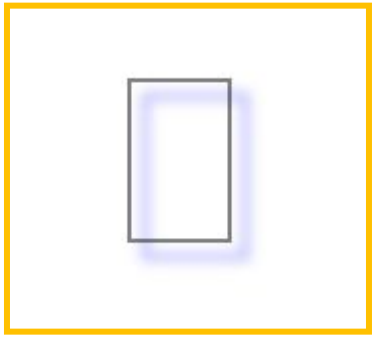
???

```
context.beginPath();
context.moveTo(175, 225);
context.lineTo(400,113);
context.lineTo(430, 350);
context.closePath();
context.lineWidth = 7;
context.strokeStyle = "red";
context.fillStyle = "blue";
context.stroke();
context.fill();
```



???

```
context.rect(300, 200, 50, 80);
context.shadowBlur = 10;
context.shadowOffsetY = 8;
context.shadowOffsetX = 8;
context.shadowColor = "blue";
context.stroke();
```



Свойства (графические атрибуты «холста»)

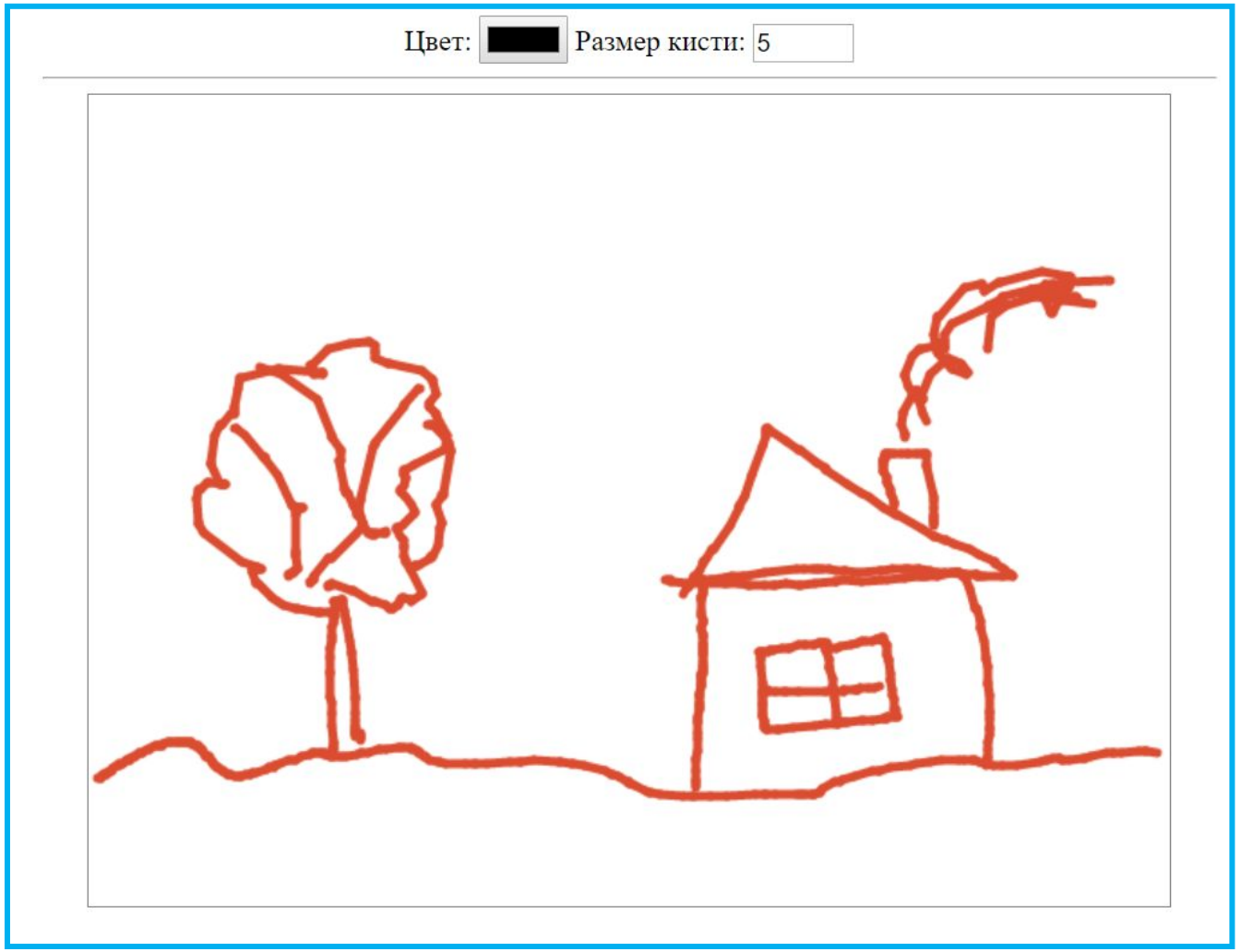
Свойство	Описание
fillStyle	цвет, градиент или шаблон, используемый для заливки
font	определение шрифта в формате CSS для команд рисования текста
globalAlpha	уровень прозрачности, назначаемый для всех пикселов при рисовании
globalCompositeOperation	способ объединения новых пикселов с существующими
lineCap	форма концов линий
lineJoin	форма вершин
lineWidth	толщина рисуемых линий
miterLimit	максимальная длина острых вершин
textAlign	выравнивание текста по горизонтали
textBaseline	выравнивание текста по вертикали
shadowBlur	четкость теней
shadowColor	цвет теней
shadowOffsetX	горизонтальное смещение теней
shadowOffsetY	вертикальное смещение теней
strokeStyle	цвет, градиент или шаблон, используемый для рисования линий

http://www.w3schools.com/tags/ref_canvas.asp

Загрузка изображений на холст

```
var img1 = new Image();
img1.src = "http://js.courses.dp.ua/files/canvas/images/1.jpg";
img1.onload = function(){
    context.drawImage(img1, 100, 100);
}

var img2 = new Image();
img2.src = "http://js.courses.dp.ua/files/canvas/images/2.jpg";
img2.onload = function(){
    context.drawImage(img2, 50, 50, 250, 185);
}
```

Paint на



http://js.courses.dp.ua/files/canvas/paint.html

Paint на

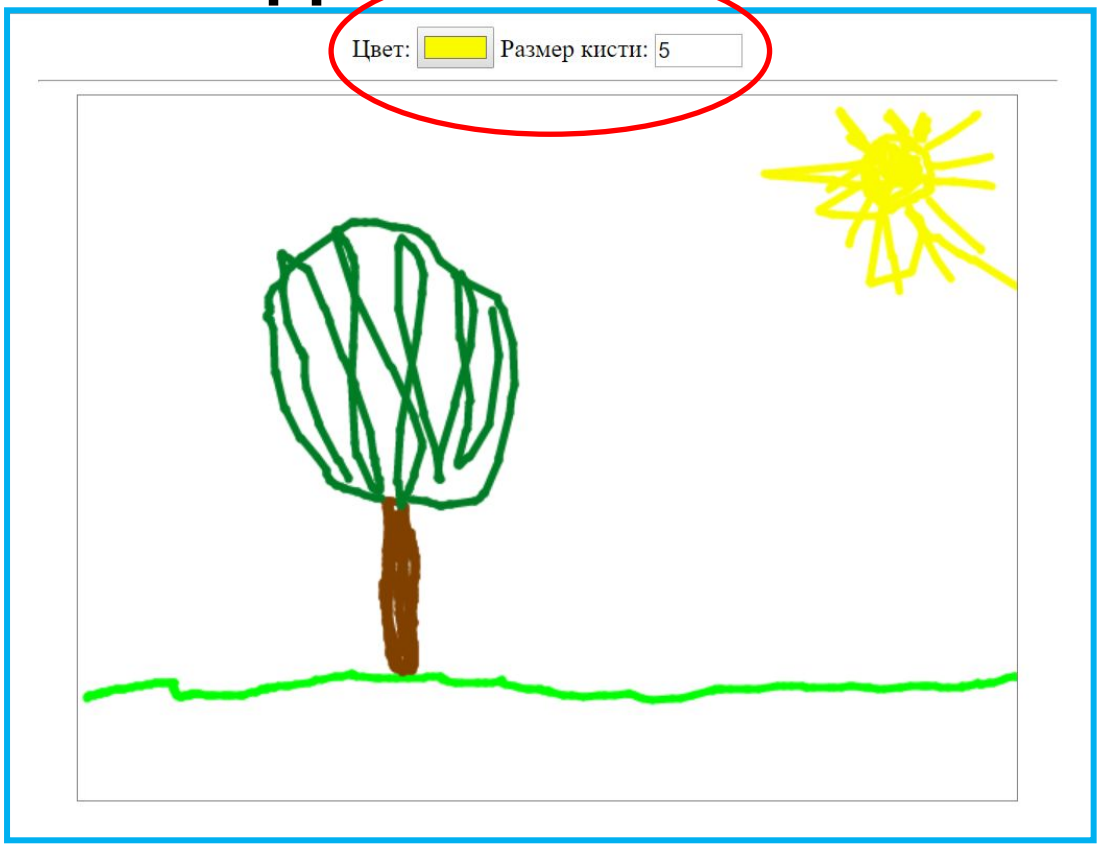
```
<!DOCTYPE html>
<html>
<body>
<center>
    Цвет: <input type="color" id="color_selector">
    Размер кисти: <input type="number" min="1" max="32" value="5">
    <hr width="650px">
    <canvas id="xxx" width="600px" height="450px" style="border:1px solid gray; cursor:pointer;">
    </canvas>
</center>
    <script>
        var canvas = document.getElementById("xxx");
        var context = canvas.getContext("2d");

        context.strokeStyle = "red";
        context.lineWidth = 5;

        var paint = false;

        canvas.onmousedown  = function(e){  paint = true; };
        canvas.onmouseup    = function(e){  paint = false; };
        canvas.onmouseleave = function(e){  paint = false; };

        canvas.onmousemove = function(e){
            var x = e.pageX - this.offsetLeft;
            var y = e.pageY - this.offsetTop;
            if(paint){
                context.beginPath();
                context.moveTo(x + e.movementX, y + e.movementY);
                context.lineTo(x, y);
                context.closePath();
                context.stroke();
            }
        }
    </script>
</body>
</html>
```

http://js.courses.dp.ua/files/canvas/paint.html

Домашнее задание

*Узнать зачем нужен оператор **throw**, как и для чего он используется.*

Домашнее задание



Доработать paint, добавить возможность выбора цвета и размера кисти.