Державний університет
«Житомирська політехніка»

# Web APIs

*Морозов А.В., к.т.н., доц.,*
*morozov@ztu.edu.ua*

# Page Visibility API

Дозволяє відстежити, коли сторінка стає **видимою** або **невидимою**, а також отримати поточний стан видимості/невидимості сторінки.

При згортанні/розгортанні вікна або перемиканні на іншу вкладку ініціюється подія **visibilitychange**.

Можливі варіанти використання:
- зупинка програвання відео/аудіо, коли користувач не переглядає сторінку;
- на сторінці є карусель зображень, яка не повинна переходити до наступного зображення, якщо користувач не переглядає сторінку.

# Page Visibility API

```
var videoElement =
          document.querySelector("video");
if (document.visibilityState == "visible") {
    videoElement.play();
}
function handleVisibilityChange() {
    if (document.visibilityState == "hidden") {
        videoElement.pause();
    } else {
        videoElement.play();
    }
}
document.addEventListener('visibilitychange',
                          handleVisibilityChange);
```

# Notifications API

Відправка запиту на дозвіл відображення вспливаючих повідомлень:

```
Notification.requestPermission().then(function(result) {
    console.log(result);
});
```

Створення вспливаючого повідомлення:

```
var img = 'адреса зображення';
var text = 'Текст повідомлення';
var notification = new Notification('Заголовок',
                                   { body: text, icon: img });
```

Примусово закрити повідомлення:

```
setTimeout(notification.close.bind(notification), 4000);
```

# Notifications API

Можливі події:

- click
- close
- error
- show

# Geolocation API

Перевірка, чи підтримує браузер Geolocation API:

```
if (navigator.geolocation) {
    /* geolocation is available */
} else {
    /* geolocation IS NOT available */
}
```

Запит на отримання даних про геолокацію:

```
navigator.geolocation.getCurrentPosition(
   function(position) {
      console.log(position.coords.latitude,
                  position.coords.longitude);
   });
```

# Geolocation API

Автоматичний виклик функції при зміні положення:

```
var watchID =
      navigator.geolocation.watchPosition(
       function(position) {
          console.log(position.coords.latitude,
                      position.coords.longitude);
});
```

Зупинка слідкування за зміною положення:

```
navigator.geolocation.clearWatch(watchID);
```

# Geolocation API

Обробка помилок та передача додаткових опцій:

```
function geo_success(position) {
    console.log(position.coords.latitude,
                position.coords.longitude);
}
function geo_error() {
    console.log("Sorry, no position available.");
}
var geo_options = {
    enableHighAccuracy: true,
    maximumAge        : 30000,
    timeout           : 27000
};
var wpid =
navigator.geolocation.watchPosition(geo_success,
                                    geo_error, geo_options);
```

# Web Workers API

## Запуск воркера:

```
var myWorker = new Worker("worker.js");
```

## Примусова зупинка воркера:

```
myWorker.terminate();
myWorker = undefined;
```

## Отримування результатів від воркера:

```
myWorker.addEventListener('message', function(event)
{
    document.getElementById("result").innerHTML =
event.data;
});
```

# Web Workers API

**В js-файлі воркера:**

```
var i = 0;

function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}

timedCount();
```

# Fullscreen API

Розгорнути тег на весь екран у повноекранному режимі:

```
tag.requestFullscreen();
```

Вийти з повноекранного режиму:

```
document.exitFullscreen();
```

В CSS можна задавати стилі для тегу, який розгорнуто в повноекранному режимі:

```
div:fullscreen {
    width: 100% !important;
}
```

# Fullscreen API

Отримати тег, який розгорнуто в повноекранному режимі:

```
document.fullscreenElement
```

Подія переходу у повноекранний режим та виходу з нього:

```
document.addEventListener("fullscreenchange",
    function(event) { ... }
);
```

https://davidwalsh.name/demo/fullscreen.php

# Web Storage API

В браузері передбачено **2 механізми зберігання даних**:

- **sessionStorage** (сховище сесій)
- **localStorage** (локальне сховище)

**Розміри сховища:**

**- для десктопних браузерів:**

- Chrome, IE, Firefox: 10 МБ
- Safari: 5MB для localStorage, необмежений розмір для sessionStorage

**- для мобільних браузерів:**

- Chrome, Firefox: 10 МБ
- iOS Safari and WebView: 5 МБ для localStorage, sessionStorage необмежений в iOS6, iOS7 – 5 МБ
- Android Browser: 2 МБ для localStorage, необмежений - sessionStorage

# Web Storage API

**1) Додавання елемента:**

```
localStorage.setItem("lastname", "Smith");
localStorage.lastname = "Smith";
localStorage[‘lastname’] = "Smith";
```

**2) Отримання елемента:**

```
var elem = localStorage.getItem("lastname");
var elem = localStorage.lastname;
var elem = localStorage[‘lastname’];
```

**3) Видалення елемента**

```
localStorage.removeItem("lastname");
```

**4) Отримання кількості елементів:**

```
localStorage.length
```

# Web Storage API

**5) Очищення сховища:**

localStorage.clear()

```
if (localStorage.clickcount) {
    localStorage.clickcount = +(localStorage.clickcount) + 1;
} else {
    localStorage.clickcount = 1;
}
document.getElementById("result").innerHTML =
    "Кількість завантажень сторінки " +
    localStorage.clickcount;
```

# Web Storage API

**JavaScript-об’єкт:**

```
var myCar = {
    wheels: 4,
    doors: 4,
    engine: 1,
    name: "Jaguar"
};
```

**JSON:**

```
var json = {
    "firstName": "Іван",
    "lastName": "Іванов",
    "address": {
        "streetAddress": "Київська, 10, кв.101",
        "city": "Житомир",
        "postalCode": 10010
    },
    "phoneNumbers": [
        "098 123-1234",
        "093 123-4567"
    ]
};
```

# Web Storage API

**Для збереження об’єкта в localStorage або sessionStorage:**

```
localStorage.setItem('ключ', JSON.stringify(об’єкт));
```

**Читання об’єкта з localStorage або sessionStorage:**

```
var retrievedObject = JSON.parse(
                        localStorage.getItem('ключ'));
```

# Web Workers API

**Запуск воркера:**

```
var myWorker = new Worker("worker.js");
```

**Примусова зупинка воркера:**

```
myWorker.terminate();
```

**Передача даних у воркер:**

```
myWorker.postMessage(значення1, значенняN);
```

**Отримування даних від воркера:**

```
myWorker.addEventListener('message', function(event) {
    document.getElementById("result").innerHTML = event.data;
});
```

# Web Workers API

## В js-файлі воркера:

```
var i = 0;
function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}
timedCount();
```

## Для отримання переданих даних у воркері:

```
onmessage = function(e) {
    var workerResult = 'Result: ' +
                       (e.data[0] * e.data[1]);
    postMessage(workerResult);
};
```

# XMLHttpRequest

**Дозволяє підвантажувати контент з інших файлів або сайтів.**

**Відправка синхронного запиту (блокуючого):**

```
var xhr = new XMLHttpRequest();

// Конфігурування: GET-запит на URL 'phones.json'
xhr.open('GET', 'phones.json', false);

// Відправляємо запит
xhr.send();

// Якщо код статусу не 200, то це помилка
if (xhr.status != 200) {
   alert( xhr.status + ': ' + xhr.statusText );
          // наприклад: 404: Not Found
} else {
    // вивести результат
    alert( xhr.responseText );
                // responseText – відповідь з сервера.
}
```

# XMLHttpRequest

## Синтаксис:

```
xhr.open(method, URL, async, user, password)
```

## Коли запит виконується:

- **status** - HTTP-код відповіді: 200, 404, 403. Може бути також рівний 0, якщо сервер не надав відповіді.
- **statusText** - текстовий опис статусу від сервера: OK, Not Found, Forbidden і т. д.
- **responseText** - текст відповіді сервера.
- **responseXML** – якщо сервер повернув XML, то тут зберігатиметься DOM-дерево цього XML-документа. У ньому можно виконувати запити вигляду: `xhr.responseXml.querySelector("...")` та ін.

# XMLHttpRequest

```
// Синхронний запит
xhr.open('GET', 'phones.json', false);

// Відправляємо його
xhr.send();
// вкладка "підвисає", поки виконується запит
```

# XMLHttpRequest

## Щоб виконати асинхронний запит:

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'phones.json', true);
xhr.send();
xhr.addEventListener('readystatechange', function() {
    if (xhr.readyState != 4)
        return;
    button.innerHTML = 'Готово!';
    if (xhr.status != 200) {
        console.log(xhr.status + ': ' + xhr.statusText);
    } else {
        console.log(xhr.responseText);
    }
});
button.innerHTML = 'Завантажую...';
button.disabled = true;
```

# Fetch API

**Нова сучасна форма виконання асинхронних запитів:**

```
'use strict';

fetch('/article/fetch/user.json')
    .then(function(response) {
        alert(response.headers.get('Content-Type'));
        alert(response.status); // 200
        return response.json();
    })
    .then(function(user) {
        alert(user.name); // iliakan
    })
    .catch( (err) => { … } );
```

# Fetch API

**Результат з сервара може прийти у різних форматах:**

- `response.arrayBuffer()`
- `response.blob()`
- `response.formData()`
- `response.json()`
- `response.text()`

**Повна документація:**

https://developer.mozilla.org/ru/docs/Web/API/Fetch_API/Using_Fetch

# Audio/Video Capture API

```
var constraints =
    { audio: true,
        video: {
        width: 1280,
            height: 720
    }
    };

navigator.mediaDevices.getUserMedia(constraints)
    .then(function(mediaStream) {
        var video = document.querySelector('video');
        video.srcObject = mediaStream;
        video.onloadedmetadata = function(e) {
            video.play();
        };
    })
    .catch(function(err) { console.log(err.name + ": " +

err.message); }); // always check for errors at the end.
```

# Video/Audio API

**HTML:**

```
<video width="400" height="300"
  controls="controls">
  <source src="video/duel.mp4"
  type="video/mp4"/>
</video>
<audio controls ="controls">
  <source src="audio/music.mp3"
     type="audio/mpeg"/>
</audio>
```

var vid = document.getElementById("myVideo");

**1) Старт відео/аудіо:**

vid.start();

**2) Пауза:**

vid.pause();

**3) Скільки секунд закешовано (тільки читання):**

vid.buffered.length – скільки фрагментів закешовано;

vid.buffered.start(i) – початок і-ого фрагмента;

vid.buffered.end(i) – кінець і-ого фрагмента;

**4) Поточна позиція (читання + запис):**
vid.currentTime = 5; // 5 секунд
**5) Швидкість (читання + запис). Спрацьовує тільки після перезапуску:**
vid.defaultPlaybackRate = 0.5;
**6) Статус (читання):**
vid.paused
**7) Фрагменти, які були переглянуті /прослухані користувачем (читання):**
vid.played.length – количество отрывков
vid.played.start(i)
vid.played.end(i)
**8) рівень звуку (читання + запис)**
vid.volume = число від 0 до 1;

**Повна документація:**

https://www.w3schools.com/tags/ref_av_dom.asp

# Canvas API

Тег **<canvas></canvas>** призначений для створення графіки засобами JavaScript.

По замовчуванню розміри тега **<canvas></canvas>** **300 x 150**.

Фізичний розмір тега **<canvas></canvas>** задається через атрибути тега **width** та **height**.

Зміна розмірів за допомогою CSS виконує тільки зміну масштабу відображення тегу.

```
<canvas id="draw">Текст, який відобразиться, якщо браузер не підтримує Canvas API</canvas>
```

# Canvas API

***Приклад***.

**HTML:**

```
<body>
    <canvas id="canvas"></canvas>
</body>
```

**JavaScript:**

```
var canvas = document.getElementById('canvas');
var con = canvas.getContext('2d');
con.fillStyle = 'green';
con.fillRect(10, 10, 100, 100);
```

**Результат:**

https://jsfiddle.net/morozovandriy/d5f4jyx1/

# Canvas API

## Малювання прямокутників

```
con.fillRect(x, y, ширина, висота)
```

*Малює зафарбований прямокутник*

```
con.strokeRect(x, y, ширина, висота)
```

*Малює лише контур прямокутника (не зафарбований прямокутник)*

```
con.clearRect(x, y, ширина, висота)
```

*Очищає прямокутну область, роблячи її прозорою*

# Canvas API

## Малювання ліній

| | |
|---|---|
| `con.moveTo(x, y)` | Встановлює координати точки, з якої починається малювання наступного об’єкта |
| `con.lineTo(x, y)` | Малює лінію до вказаної точки |
| `con.arc(x, y, радіус, почКут, кінКут)` | Малює круг або сектор круга. Кут потрібно задавати у радіанах, а не в градусах (радіани=(Math.PI/180)*градуси). |
| `con.rect(x, y, ширина, висота)` | Малює прямокутник |

# Canvas API

## Малювання ліній

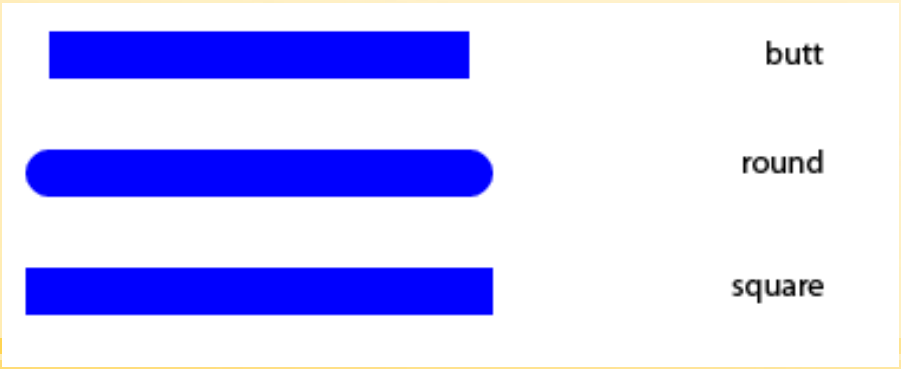
| | |
|---|---|
| `con.moveTo(x, y)` | Встановлює координати точки, з якої починається малювання наступного об’єкта |
| `con.lineTo(x, y)` | Малює лінію до вказаної точки |
| `con.arc(x, y, радіус, почКут, кінКут)` | Малює круг або сектор круга. Кут потрібно задавати у радіанах, а не в градусах (радіани=(Math.PI/180)*градуси). |
| `con.rect(x, y, ширина, висота)` | Малює прямокутник |

# Canvas API

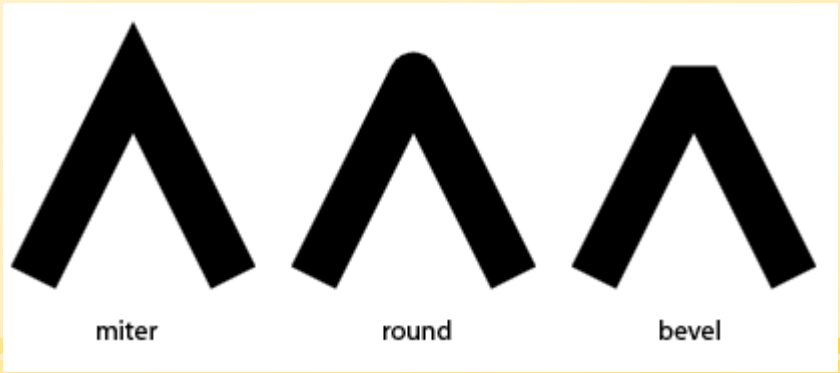
## Стилі ліній

```
// задає товщину лінії
con.lineWidth = число;
// задає стиль для кінців лінії
con.lineCap = 'стиль';
// задає стиль для стиків двох ліній
con.lineJoin = 'стиль';
```

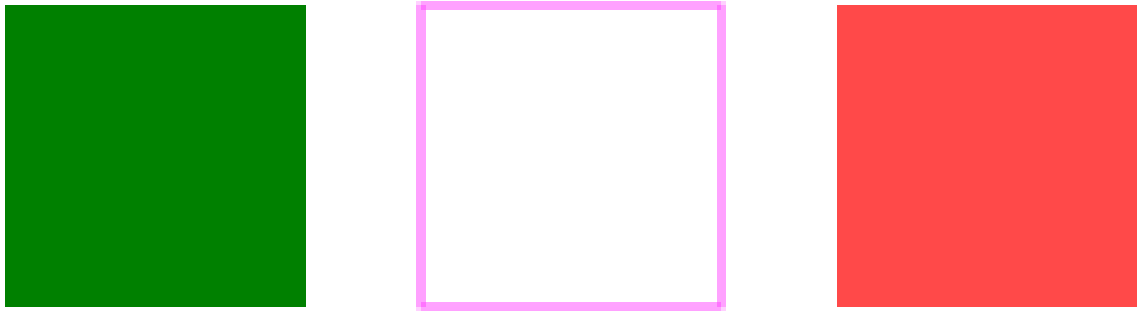
### Стилі для lineCap:




### Стилі для lineJoin:



# Canvas API



```
var canvas = document.getElementById("draw");
var con = canvas.getContext("2d");
con.fillStyle = "green";
con.strokeStyle = "#FF45FF";
con.fillRect(10, 40, 65, 65);
con.strokeRect(100, 40, 65, 65);
con.fillStyle = "rgb(255, 73, 73)";
con.fillRect(190, 40, 65, 65);
```

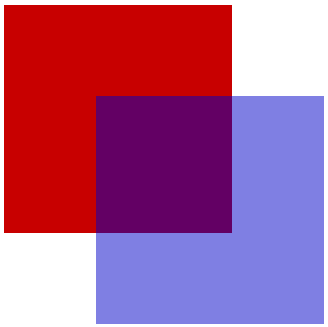
# Canvas API

## Приклад

```
var canvas = document.getElementById('canvas');
if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.fillStyle = 'rgb(200, 0, 0)';
    ctx.fillRect(10, 10, 50, 50);

    ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
    ctx.fillRect(30, 30, 50, 50);
}
```



# Canvas API

## Малювання фігур

```
con.beginPath();
/* виконання малювання
примітивів та простих фігур */

con.closePath();
        /* якщо треба замкнути контур */

con.stroke();   або    con.fill();
```

# Canvas API

## Малювання фігур

```
var con = canvas.getContext('2d');

con.beginPath();
con.moveTo(75, 50);
con.lineTo(100, 75);
con.lineTo(100, 25);
con.fill();
```

```
var con = canvas.getContext('2d');

con.beginPath();
con.arc(75, 75, 50, 0, Math.PI * 2, true);
con.moveTo(110, 75);
con.arc(75, 75, 35, 0, Math.PI, false);
con.moveTo(65, 65);
con.arc(60, 65, 5, 0, Math.PI * 2, true);
con.moveTo(95, 65);
con.arc(90, 65, 5, 0, Math.PI * 2, true);
con.stroke();
```

# Canvas API

## Малювання фігур

```
con.beginPath();
con.moveTo(25, 25);
con.lineTo(105, 25);
con.lineTo(25, 105);
con.fill();

// Stroked triangle
con.beginPath();
con.moveTo(125, 125);
con.lineTo(125, 45);
con.lineTo(45, 125);
con.closePath();
con.stroke();
```

# Canvas API

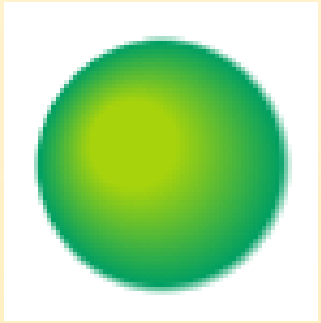
## Лінійний градієнт

```
// встановлення напряму градієнту
var grad = x.createLinearGradient(50,50,150,150);
// додавання кольорів
grad.addColorStop(0.0,'#d2006b');
grad.addColorStop(0.5,'#00a779');
grad.addColorStop(1.0,'#ffe800');
// встановлення градієнта замість кольору
con.fillStyle = grad;
// малювання фігури
con.fillRect(20,20,190,150);
```

# Canvas API

## Радіальний градієнт

```
var radgrad = con.createRadialGradient(
                              45, 45, 10, 52, 50, 30);
radgrad.addColorStop(0, '#A7D30C');
radgrad.addColorStop(0.9, '#019F62');
radgrad.addColorStop(1, 'rgba(1, 159, 98, 0)');
con.fillStyle = radgrad;
con.fillRect(0, 0, 150, 150);
```



# Canvas API

## Тінь

| | |
|---|---|
| shadowOffsetX | Зміщення тіні по горизонталі |
| shadowOffsetY | Зміщення тіні по вертикалі |
| shadowBlur | Величина розмиття тіні |
| shadowColor | Колір тіні (по замовчуванню – чорний) |

```
var canvas=document.getElementById("draw")
var con=canvas.getContext("2d");
con.shadowOffsetX = -3;
con.shadowOffsetY = 3;
con.shadowBlur = 8;
con.shadowColor = 'black';
con.fillStyle = '#ffaa00';
con.fillRect(50, 40, 55, 55);
```

# Canvas API

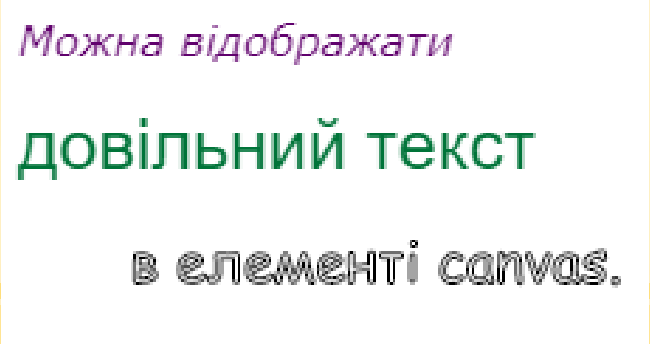
## Тінь

```
var canvas=document.getElementById("draw")
var con=canvas.getContext("2d");
con.shadowOffsetX = -3;
con.shadowOffsetY = 3;
con.shadowBlur = 8;
con.shadowColor = 'black';
con.fillStyle = '#ffaa00';
con.fillRect(50, 40, 55, 55);
```

# Canvas API

## Текст

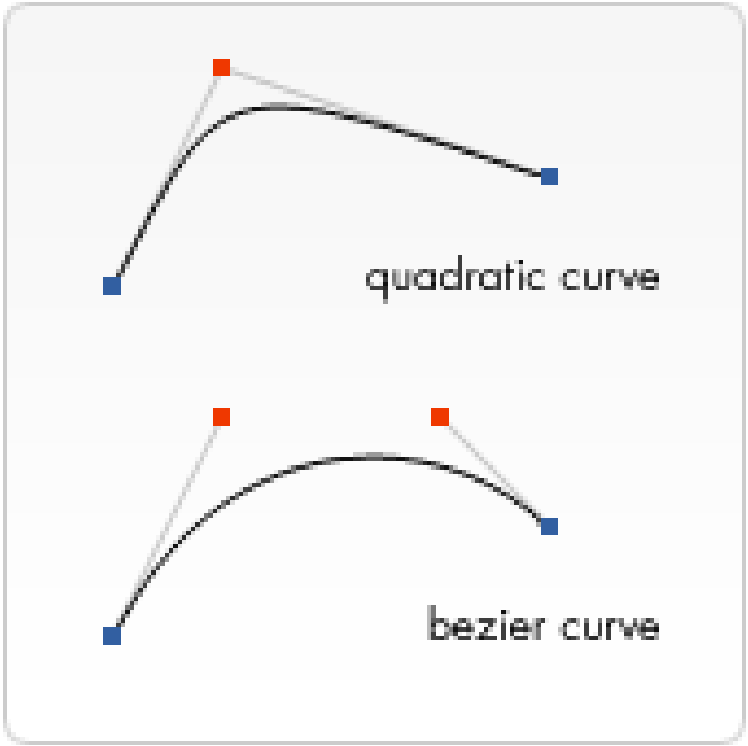
```
con.font = 'italic 15px Verdana';
con.fillStyle = '#60016d';
con.fillText("Можна відображати", 10, 40);
con.font = '25px Arial';
con.fillStyle = '#007439';
con.fillText("довільний текст", 10, 80);
con.fillStyle = '#a67800';
con.font = '20px Comic Sans MS';
con.strokeText("в елементі canvas.", 50, 120);
```



# Canvas API

## Криві

```
con.quadraticCurveTo(cp1x, cp1y, x, y)
con.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
```



# Canvas API

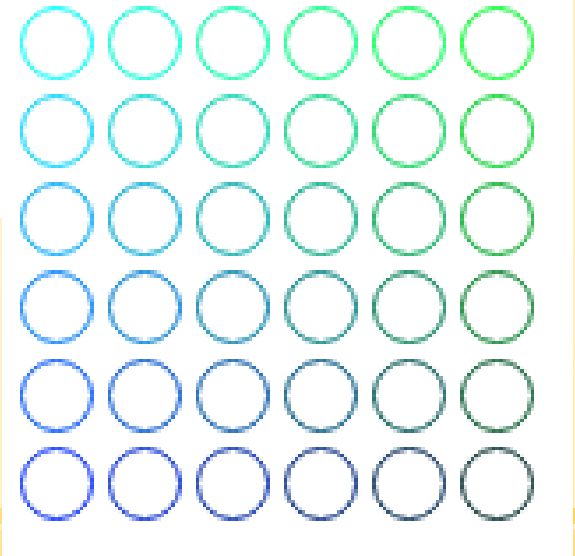
## Криві

```
con.beginPath();
con.moveTo(75, 25);
con.quadraticCurveTo(25, 25, 25, 62.5);
con.quadraticCurveTo(25, 100, 50, 100);
con.quadraticCurveTo(50, 120, 30, 125);
con.quadraticCurveTo(60, 120, 65, 100);
con.quadraticCurveTo(125, 100, 125, 62.5);
con.quadraticCurveTo(125, 25, 75, 25);
con.stroke();
```

# Canvas API

## Приклад

```
for (var i = 0; i < 6; i++) {
    for (var j = 0; j < 6; j++) {
        con.strokeStyle = 'rgb(0, ' +
            Math.floor(255 - 42.5 * i) + ', ' +
            Math.floor(255 - 42.5 * j) + ')';
        con.beginPath();
        con.arc(12.5 + j * 25, 12.5 + i * 25, 10, 0,
                Math.PI * 2, true);
        con.stroke();
    }
}
```



# Canvas API

## Графічні зображення

```
// знаходимо тег img (краще створити)
var img = document.getElementById('image1');
var canvas = document.getElementById("draw")
var con = canvas.getContext("2d");
con.drawImage(img,10,10);
```

# Canvas API

## Графічні зображення

```
var con = document.getElementById('canvas').
                                        getContext('2d');
var img = new Image();
img.addEventListener('load', function() {
    for (var i = 0; i < 4; i++) {
        for (var j = 0; j < 3; j++) {
            con.drawImage(img, j * 50, i * 38, 50, 38);
        }
    }
});
img.setAttribute('src',
    'https://mdn.mozillademos.org/files/5397/rhino.jpg');
```

# Canvas API

## Графічні зображення

**Параметри методу drawImage:**

| | |
|---|---|
| *img* | Specifies the image, canvas, or video element to use |
| *sx* | Optional. The x coordinate where to start clipping |
| *sy* | Optional. The y coordinate where to start clipping |
| *swidth* | Optional. The width of the clipped image |
| *sheight* | Optional. The height of the clipped image |
| *x* | The x coordinate where to place the image on the canvas |
| *y* | The y coordinate where to place the image on the canvas |
| *width* | Optional. The width of the image to use (stretch or reduce the image) |
| *height* | Optional. The height of the image to use (stretch or reduce the image) |

# Canvas API

## Анімації

```
var cnvs = document.getElementById("canvas");
var ctxt = cnvs.getContext("2d");
var x = 0;
function animation() {
    ctxt.clearRect(0,0,150,150);
    ctxt.fillRect(x,50,50,50);
    x = x + 0.2;
    if ( x > 100) { x = 0; }
    setTimeout(animation, 10)
}
animation();
```

https://jsfiddle.net/morozovandriy/uLe9zsnv/

# Canvas API

## Приклад реалізації арканоїда

https://developer.mozilla.org/en-US/docs/Games/Tutorials/2D_Breakout_game_pure_JavaScript