

JAVASCRIPT В HTML5



Георги Пенчев
Съвременно уеб програмиране с PHP 2012

geolocation

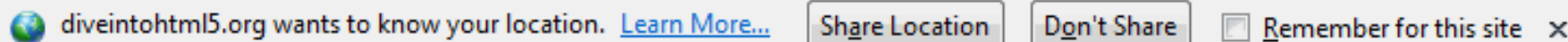
Защо ни е?

- ✓ Къде сме на картата
- ✓ Какво има наблизо
- ✓ Навигация
- ✓ Place-based алерти
- ✓ Location tagging
- ✓ Локализирано съдържание
- ✓ ... защото е 2012 и ще ни се смеят без него!



Какво трябва да знаем?

✓Permissions



✓Точност

- GPS – до 10м (харчи батерия и трафик)
- WiFi – до 20м (ако има!)
- GSM/CDMA мрежа – до 100м
- IP geolocation – city level (!)

Как да го ползваме?


✓ Проверяваме поддръжка

```
if (navigator.geolocation) if (Modernizr.geolocation)
```

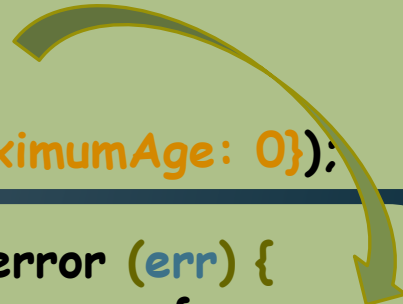
✓ Взимане на текуща локация

```
navigator.geolocation.getCurrentPosition(  
  successCallback, errorCallback,  
  {enableHighAccuracy: true, timeout: 10*1000, maximumAge: 0});
```

Position object



```
function success (pos) {  
  alert("You are at: " +  
    pos.latitude + ", " +  
    pos.longitude + ", accuracy: "  
    + pos.accuracy);  
}
```



```
function error (err) {  
  var errors = {  
    1: 'Permission denied',  
    2: 'Position unavailable',  
    3: 'Request timeout'  
  };  
  alert(errors[err.code]);  
}
```

Как да го ползваме?

✓ Ако трябва да ъпдейтваме постоянно

```
var watchID = navigator.geolocation.watchPosition(  
    successCallback, errorCallback,  
    {enableHighAccuracy: true, timeout: 10*1000, maximumAge: 0});
```

```
clearWatch(watchID);
```


Къде да го ползваме?



7.0	3.6	17.0			4.0-4.1		11.1	2.3
8.0	11.0	18.0			4.2-4.3		11.5	3.0
9.0	12.0	19.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0

- geolocation

use with fallback

[geolocation shim](#) uses Google's IP-GeoCoding service as a fallback. [geo-location-javascript](#) doesn't but has hooks into BlackBerry, WebOS, and Google Gears specific APIs. In most cases, you should just not expose Geo features in your app if the feature is not natively present.

Recommended polyfills: [geolocation shim](#), [geo-location-javascript](#)

[View browser share %](#)

[Edit this info](#)

✓ А ако няма поддръжка?

- Geo.js
- Gears

history

Защо ни е?

✓AJAX vs Shareable URLs

✓Hashbang hacks (#!)

- [https://github.com/balupton/history.js/wiki/Intelligent-State-Handl](https://github.com/balupton/history.js/wiki/Intelligent-State-Handling)
- <http://blog.benward.me/post/3231388630>



Как да го ползваме?

✓ Проверяваме поддръжка

```
if (window.history && history.pushstate)
```

```
if (Modernizr.history)
```

✓ Ajax call ➡ Променяме URL

```
history.pushState(data, null, link);
```

data - can be any object ; **title** - unused ; **link** - goes on location bar

✓ Когато натиснем “Back”

```
window.addEventListener('popstate', function(event) {  
    doAjaxWithData(event.state);  
});
```

//можем да ползваме и **location.href** и неговите компоненти

Как да го ползваме?

✓ А когато отворим „фалшив“ линк?

⇒ **mod_rewrite** to the rescue!

```
<ifModule mod_rewrite.c>  
  RewriteEngine On  
  RewriteCond %{REQUEST_FILENAME} !-f  
  RewriteCond %{REQUEST_FILENAME} !-d  
  RewriteCond %{REQUEST_URI} !index  
  RewriteRule (.*) index.html [L]  
</ifModule>
```

- <http://www.josscrowcroft.com/2012/code/htaccess-for-html5-history-pushstate-url-routing/>

✓ Остана само началото:

```
history.replaceState(initialData, document.title, location.href);
```

Къде да го ползваме?

								
7.0	3.6	17.0			4.0-4.1		11.1	2.3
8.0	11.0	18.0			4.2-4.3		11.5	3.0
9.0	12.0	19.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
10.0	13.0	20.0	5.2	12.0				

- history

use with fallback

The `History` API provides a way for JavaScript to change the URL displayed in the browser without reloading the page. There are several approaches to providing a fallback. The simplest is to fall back to page refreshes. Alternatively, the [History.js](#) plugin smooths out some browser implementation differences and provides an optional hashchange fallback for HTML 4 browsers. GitHub uses [pjax](#) (pushState + ajax).

[View browser share %](#) [Edit this info](#)

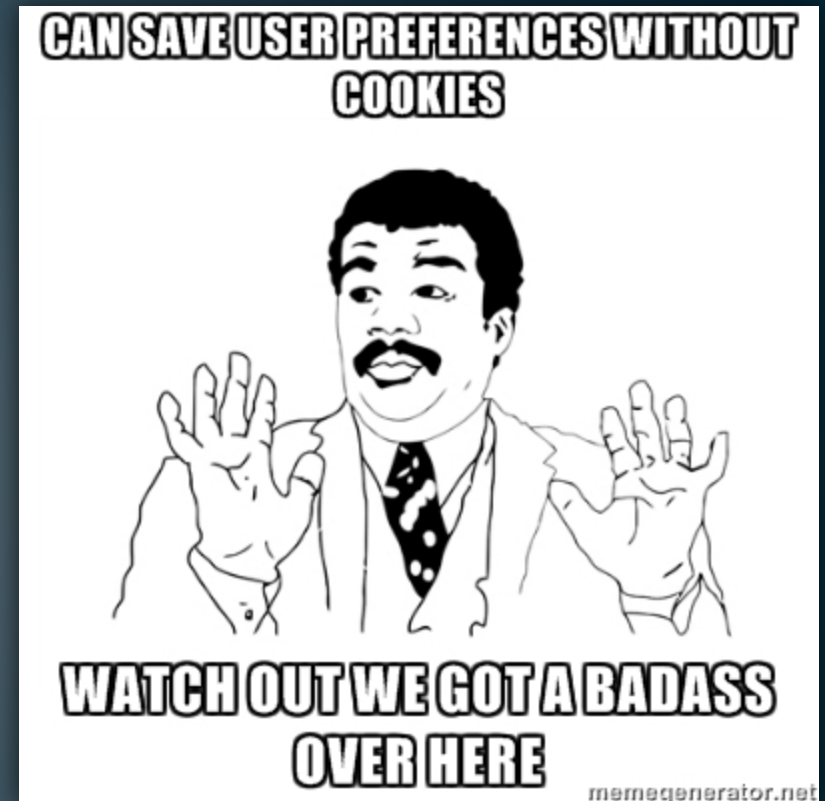
✓ А когато няма поддръжка?

- History.js

storage

Защо ни е?

- ✓ Cookies са лоши
 - малки са
 - хабят трафик
 - не са сигурни
- ✓ Зависим от сървъра
- ✓ Товарим сървъра



Какво трябва да знаем?

- ✓ Какво е „origin“?
- ✓ Имаме по 5MB място за всеки тип storage
- ✓ **localStorage**
 - представлява речник key-value двойки
 - много лесен за употреба
 - cookies имат смисъл, ако ни трябва в PHP
 - пази всичко в ТЕКСТ
 - не търси ефективно
 - пази данни вечно (докато ги изтрием)
 - Webkit има и по-интересни методи:
[http://playground.html5rocks.com/#show usage and quota request](http://playground.html5rocks.com/#show_usage_and_quota_request)
- ✓ **sessionStorage**
 - същото, но данните стоят до затваряне

Какво трябва да знаем?

✓ **WebSQL**

- представлява локална SQLite база
- ползваме го чрез SQL команди
- добър за големи данни и сложно търсене
- има транзакции
- deprecated е – Mozilla никога няма да го поддържа

✓ **IndexedDB**

- локална база, но през ОО интерфейс
- вероятно ще е окончателният стандарт

Как да ги ползваме - localStorage?

- ✓ Проверяваме поддръжка

```
if (window['localStorage'] !== null)
```

```
if (Modernizr.localStorage)
```

- ✓ Съхраняваме нещо

```
localStorage.setItem('kokoshki', 6); key - string; value - stringified
```

- ✓ Ключовете са и сетъри

```
localStorage.kokoshki = 6;
```

- ✓ Взимам си кокошките да ги заколя

```
var myKokoshki = localStorage.kokoshki; //= localStorage.getItem('kokoshki');
```

- ✓ По-сложни обекти

```
localStorage.kokoshki = JSON.stringify(['Къцка', 'Типка', 'Бела']);
```

- ✓ Трия

```
delete localStorage.kokoshki; //= localStorage.removeItem('kokoshki');
```

Как да ги ползваме - localStorage?

✓ Събития

```
window.addEventListener('storage', storageEvent, false);
```

```
function storageEvent(e)
{
    console.log('Changing' + e.oldValue + 'kokoshki into ' + e.newValue);
}
```

✓ Търкане на целия store

```
localStorage.clear();
```

Как да ги ползваме - WebSQL?

- ✓ Отваряме база данни

```
var db = openDatabase('mydb', '1.0', 'kokoshki_db', 2 * 1024 * 1024);
```

- ✓ Създаваме таблица

```
db.transaction(function (tx) {  
    tx.executeSql('CREATE TABLE IF NOT EXISTS foo (id unique,  
text)');
```

- ✓ Пълним обекти

```
tx.executeSql('INSERT INTO foo (id, text) VALUES (?, ?)', [id,  
userValue]);
```

- ✓ Взимаме обекти

```
tx.executeSql('SELECT * FROM foo', [], function (tx, results) {  
    var len = results.rows.length, i;  
    for (i = 0; i < len; i++) {  
        alert(results.rows.item(i).text);  
    });
```

Как да ги ползваме - IndexedDB?

- ✓ Сравнение на една и съща задача с двата подхода можете да намерите тук:

<http://hacks.mozilla.org/2010/06/comparing-indexeddb-and-webdatabase/>

```
var kids = [
  { name: "Anna" },
  { name: "Betty" },
  { name: "Christine" }
];

var request = window.indexedDB.open("CandyDB",
                                     "My candy store database");
request.onsuccess = function(event) {
  var objectStore = event.result.objectStore("kids");
  for (var index = 0; index < kids.length; index++) {
    var kid = kids[index];
    objectStore.add(kid).onsuccess = function(event) {
      document.getElementById("display").textContent =
        "Saved record for " + kid.name + " with id " + event.result;
    };
  }
};
```


Къде да го ползваме - localStorage?



7.0	3.6	17.0			4.0-4.1		11.1	2.3
8.0	11.0	18.0			4.2-4.3		11.5	3.0
9.0	12.0	19.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0

- localStorage

caution with polyfill

Local and session storage are a great way to store data without resorting to cookies. IE8 supported `localStorage` and `sessionStorage` so you may not need a polyfill. If you do, Remy's is a piece of cake to implement and use.

This is a simple key/value store, so if you want to store complex data use `JSON.parse(str)` and `JSON.stringify(obj)` on your way in and out. There is also no way to know if you exceeded the storage cross-browser, so wrap your store commands in try/catch. Up to 2.5MB is safe to use.

As part of keeping things simple, `localStorage` has a synchronous API that runs on the main UI thread in browsers; as a consequence of that, a [race condition](#) can occur if a user has the same site open in multiple windows or tabs running as separate processes. For many applications, that's never really a problem in practice. But it can cause data corruption—so applications where it's important to try to ensure that absolutely no data corruption can occur should instead use a more robust storage mechanism such as `IndexedDB`.

Due to the shortcomings of `localStorage`, there are calls [to stop advocating for and building examples that use it](#).

Recommended polyfills: [Remy's storage polyfill](#), [sessionstorage](#)

[View browser share %](#) [Edit this info](#)

Къде да го ползваме - WebSQL?



7.0	3.6	17.0			4.0-4.1		11.1	2.3
8.0	11.0	18.0			4.2-4.3		11.5	3.0
9.0	12.0	19.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
10.0	13.0	20.0	5.2	12.0				

WebSQL DB

avoid

Although it initially found favor as a clientside database API, it has now [been abandoned](#) in favor of IndexedDB. We therefore recommend that you do not use WebSQL.

[View browser share %](#) [Edit this info](#)

Къде да го ползваме - IndexedDB?



8.0	11.0	moz	18.0	webkit			4.2-4.3		11.5	3.0
9.0	12.0	moz	19.0	webkit	5.1	11.6	5.0	5.0-6.0	12.0	4.0
10.0	ms	moz	20.0	webkit	5.2	12.0				

- IndexedDB

caution with fallback

IndexedDB was a volatile spec for a year, but has settled down. In addition to Chrome and Firefox, IE10 will have it; Opera has not yet implemented it and Safari has not yet committed to it.

[IDBWrapper](#) helps smooth out the cross-browser differences. You may consider falling back to WebSQL when IndexedDB isn't available, but do keep in mind that WebSQL has been abandoned.

Recommended polyfills: [IDBWrapper](#)

[View browser share %](#) [Edit this info](#)

drag & drop

Защо ни е?



Какво трябва да знаем?

- ✓ И преди HTML5 елементи се влачат
 - ✓ Например можем да завлачим линк
- ✓ Ние трябва да стилизираме различно елементите при влачене – не става автоматично
- ✓ Влаченето не е само за местене – можем да подаваме каквато искаме информация от завлечения елемент към елемента, върху който пускаме
 - ✓ Използваме специалния **dataTransfer** обект, който представлява dictionary с ключ тип на данните

Какво трябва да знаем?

- ✓ Възможно е да се драгват файлове от и към вашия десктоп (като в Gmail)
 - ✓ <http://www.sitepoint.com/html5-file-drag-and-drop/>
 - ✓ <http://www.thecssninja.com/javascript/gmail-dragout>
 - ✓ **dataTransfer.files** ни дава файловете, завлечени от десктопа
- ✓ Критики
 - ✓ Моделът на събитията е доста сложен
 - ✓ Имплементациите са разнородни
 - ✓ Влаченето не започва при натискане на мишката, а чак при движение
 - ✓ Не е добре обмислено при вложени контейнери
- ✓ Има много библиотеки, които досега и все още можем да използваме за D&D

Какво трябва да знаем?

- ✓ Това са събитията, които се ползват
 - ✓ **dragstart** – започва влачене, this = dragged element
 - ✓ **drag** – мишката се мести влачейки, this = dragged element
 - ✓ **dragenter** – минава над елемент, this = drop element
 - ✓ **dragover** – мишката се мести над елемент, this = drop element
 - ✓ **dragend** – мишката се пуска някъде, this = dragged element
 - ✓ **dragleave** – мишката напуска елемент, this = drop element
 - ✓ **drop** – мишката се пуска над елемент, this = drop element
 - ✓ **Важно:** При **dragenter** & **dragover** трябва да спрем поведението по подразбиране на събитието, за да продължим влаченето (по подразбиране спира или редиректва)

Как да го ползваме?

- ✓ Проверяваме поддръжка

```
if (Modernizr.draganddrop)
```

- ✓ Декларираме елемент за влачене

```
<div id="myDiv" draggable="true">
```

- ✓ Внимание: може да ни трябва това за Safari

```
[draggable="true"] { -webkit-user-drag: element; }
```

- ✓ В **dragstart** обработчика описваме визуализацията:

```
function onDragStart(e) {  
  this.style.opacity = '0.4';
```

- ✓ ... както и какви данни носи елементът:

```
e.dataTransfer.setData('text/html', this.innerHTML);
```

- ✓ можем да описваме каквито си искаме стойности, тук ползваме съдържанието на елемента
 - ✓ ако искаме по-сложен обект за стойност => JSON

Как да го ползваме?

- ✓ Накрая махаме

```
var dragIcon = document.createElement('img');  
dragIcon.src = 'logo.png'; dragIcon.width = 100;  
e.dataTransfer.setDragImage(dragIcon, -10, -10);
```

- ✓ В обработката на **dragover** & **dragenter** спираме дефолтното поведение, за да позволим дроп

```
function onDragEnter(e) {  
    e.preventDefault();
```

- ✓ Можем и да стилизираме елемента, над който сме, за да покажем, че там можем да пуснем

```
... this.style.borderStyle = 'dashed';
```

- ✓ Това не трябва да правим в обработката на **dragOver**, защото то се вика много пъти, докато сме над обекта!
- ✓ Не забравяме да махнем тази стилизация в **onDragLeave** & **onDragEnd**!

Как да го ползваме?

- ✓ Накрая в обработката на **drop** събитието:
 - ✓ Премахваме дефолтното поведение (напр. пренасочване, ако сме влачили линк)

```
function onDrop(e) {  
    e.preventDefault();  
    e.stopPropagation();  
}
```

- ✓ Взимаме данните, които записахме, и правим нещо с тях:

```
this.innerHTML += e.dataTransfer.getData('text/html');
```


Къде да го ползваме?



7.0	11.0				4.0-4.1		11.1	2.3
8.0	12.0	18.0		11.6	4.2-4.3		11.5	3.0
9.0	13.0	19.0	5.1	12.0	5.0	5.0-6.0	12.0	4.0
10.0	14.0	20.0	5.2					

- drag n drop

caution with polyfill

Drag and Drop has been standardized in HTML5 based on Internet Explorer's original implementation. Therefore, it already has wide support, but many feel frustrated when using the API. You may want to use jQuery UI Draggable (or another JavaScript library) to handle this for you. Meanwhile the proposed [\[dropzone\]](#) attribute will improve the situation as it gains browser support.

Recommended polyfills: [dropfile](#), [fileSaver](#), [jDataView](#)

[View browser share %](#) [Edit this info](#)

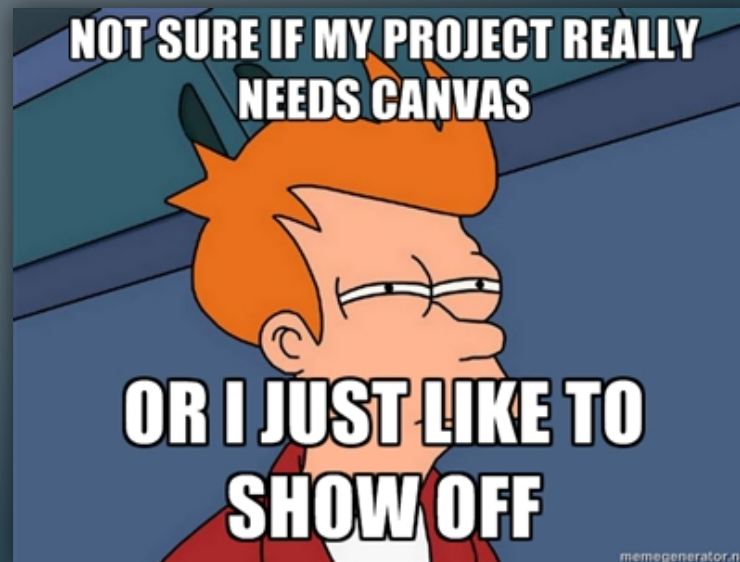
✓ А ако няма поддръжка?

- jQuery

canvas

Защо ни е?

- ✓ За да правим игри, DUN!
- ✓ Или редактори на изображения
- ✓ Спестява нуждата от Photoshop
- ✓ Отваря много врати пред уебдизайна



Какво трябва да знаем?

- ✓ **canvas** е празен HTML5 елемент и рисуваме в него чрез JS функции на обекта му **context**
- ✓ Технологията е растерна – позволява чертаене, стандартно за повечето библиотеки за рисуване (например Java2D)
- ✓ Стандартът дава само 2D API, но Mozilla & Opera имат експериментално 3D API (потърсете за WebGL)
- ✓ В HTML5 може да се рисува и векторно, чрез декларативни описания със стандарта SVG

Какво трябва да знаем?

- ✓ **canvas** се ресетва и изпразва, ако зададем ширина и височина, след като сме рисували, дори те да са същите стойности!
- ✓ Можем да правим пикселни манипулации (и следователно всякакви фотоефекти) чрез **getImageData**
- ✓ Началото на координатната система е горе ляво, координатите са релативни спрямо позицията на **canvas** елемента
- ✓ Има методи за геометрични трансформации, които влияят върху всяко последващо рисуване (като в OpenGL)

Какво трябва да знаем?

- ✓ **canvas** може да се конвертира до DataURL!
 - ✓ така можем да запазваме нарисуваното
 - ✓ после го зареждаме с обекта **Image** и **drawImage**
- ✓ Поддръжката на Canvas API не навсякъде включва поддръжка на Canvas Text API за чертаене на текст в **canvas** (но в повечето случаи е така)
- ✓ Може да се комбинира с други HTML5 технологии:
 - ✓ Video: <http://html5doctor.com/video-canvas-magic/>
 - ✓ Drag&Drop: <http://www.html5canvastutorials.com/labs/html5-canvas-drag-and-drop-resize-and-invert-images/>
- ✓ Можем да пробваме тук:
<http://jsbin.com/abagi3/edit#source>

Как да го ползваме?

- ✓ Проверяваме поддръжка

```
if (Modernizr.canvas)
```

```
if (Modernizr.canvastext)
```

- ✓ Създаваме си елемента

```
<canvas id="myDiv" width="200" height="300"/>
```

- ✓ Взимаме му контекста:

```
var canvas = $('myDiv'); //jQuery style  
var context = canvas.getContext('2d');
```

- ✓ Задаваме стил – цвят, градиент или pattern

```
context.setFillStyle = "#ee00ff";
```

- ✓ Рисуваме правоъгълник

```
context.fillRect(0,20,100,200); //w = 100, h = 200
```

Как да го ползваме?

- ✓ Само рамка на правоъгълник

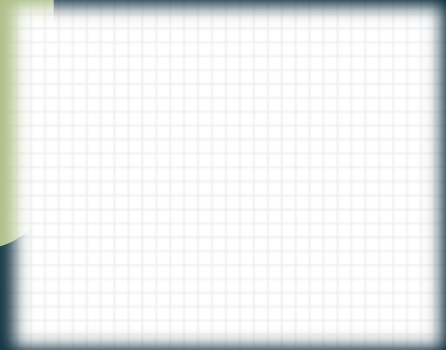
```
context.strokeStyle = "#000";  
context.strokeRect(0, 20, 100, 200);
```

- ✓ Изтриване на пикселите от регион

```
context.clearRect(0, 20, 100, 200);
```

- ✓ Paths:

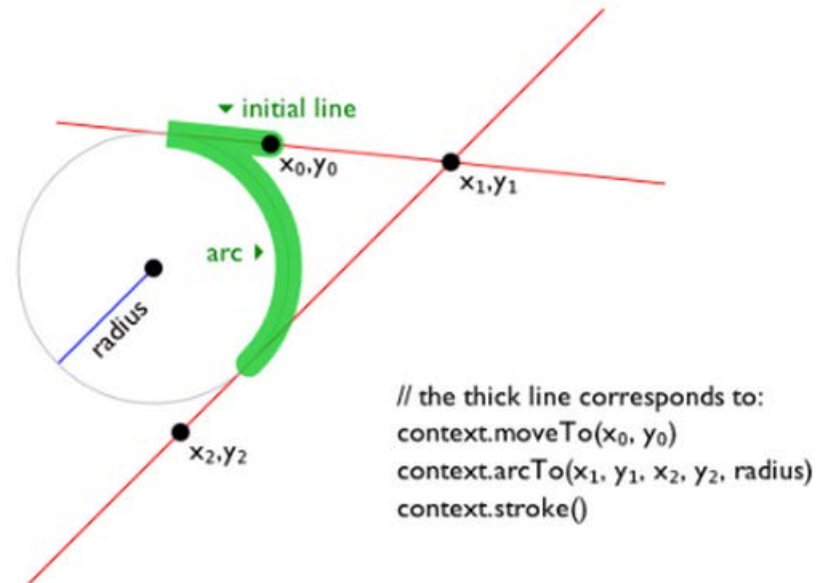
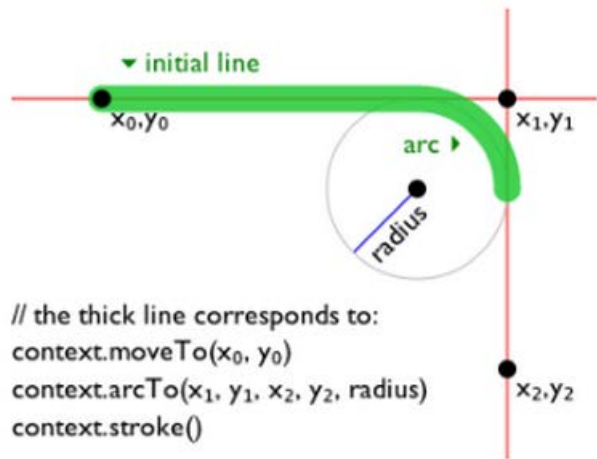
```
context.beginPath();  
for (var x = 0.5; x < 500; x += 10) {  
    context.moveTo(x, 0); context.lineTo(x, 375); }  
for (var y = 0.5; y < 375; y += 10) {  
    context.moveTo(0, y); context.lineTo(500, y); }  
context.strokeStyle = "#eee";  
context.stroke(); //вика closePath() само
```



Как да го ползваме?

✓ Имаме и по-сложни неща от прави:

```
context.quadraticCurveTo(cpx, cpy, x, y);  
context.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y);  
context.arcTo(x1,y1,x2,y2,radiusX [, radiusY, rotation ]);
```



Как да го ползваме?

✓ Рисуване на текст

```
context.font = "bold 12px sans-serif";  
context.textAlign = "right";  
context.textBaseline = "bottom";  
context.fillText("Mecho", 492, 370);
```

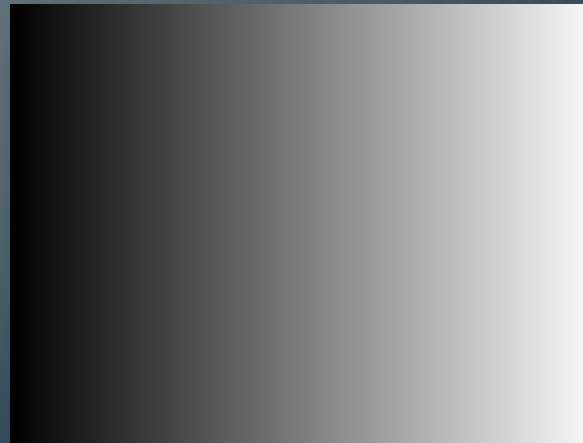
✓ Text baselines



Как да го ползваме?

✓ Градиенти

```
var gradient = context.createLinearGradient(0, 0, 200, 0);  
//нашият канвас е широк 200  
//параметрите са две точки на отсечка, по която се мени цветът  
gradient.addColorStop(0, "black"); //0 е единият край на отсечката  
gradient.addColorStop(1, "white"); //1 е другият  
context.fillStyle = gradient;  
context.fillRect(0, 0, 200, 125);
```



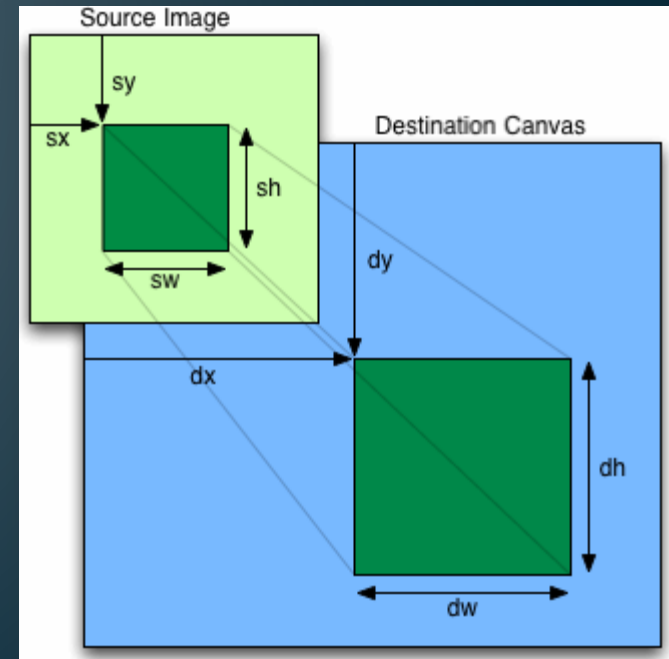
Как да го ползваме?

✓ Картинки

```
context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh);  
context.drawImage(image, dx, dy);  
context.drawImage(image, dx, dy, dw, dh);
```

✓ Можем да зареждаме от

```
  
<script>  
window.onload = function() {  
  .. // get canvas and context  
  var cat = $('cat');  
  context.drawImage(cat, 0, 0);  
};  
</script>
```



Как да го ползваме?

✓ Или с класа **Image**

```
var cat = new Image();  
cat.src = "images/cat.png";  
cat.onload = function() {  
    context.drawImage(cat, 0, 0);  
}; //картинката трябва да се е заредила!
```

✓ Запазване на **canvas** като картинка

```
myImgElement.src = canvas.toDataURL();
```

✓ Image обектите могат да се зареждат и от DataURL (save<->load)

✓ Как да намерим върху какво кликаме с мишката върху **canvas**? => вижте примера :)

✓ Още много функции в APIто:

<http://html5tutorial.net/downloads/canvas-cheat-sheet/HTML-5-Canvas-Cheat-Sheet.png>

Къде да го ползваме?



7.0	11.0				4.0-4.1		11.1	2.3
8.0	12.0	18.0		11.6	4.2-4.3		11.5	3.0
9.0	13.0	19.0	5.1	12.0	5.0	5.0-6.0	12.0	4.0
10.0	14.0	20.0	5.2					

<canvas>

use with polyfill

`canvas` is definitely good to go for modern browsers. If you want to support Internet Explorer 8 and below, ExplorerCanvas and FlashCanvas can be helpful in providing support for most `canvas` features. However, due to the complex nature of native `canvas` implementations, developers should be aware that the polyfills for `canvas` are not simple drop-in solutions in some cases.

For example, both [ExplorerCanvas](#) and [FlashCanvas](#) may have difficulties handling the commonly used `drawImage` method. FlashCanvas cannot be passed the bitmap data from a DOM-based `Image` object, and therefore has to re-request the asset in the Flash Player causing undesired latency and flickering. Developers should be careful when handling image data and ensure thorough testing due to the unreliability and technical limitations of these features in the polyfills.

That isn't to say `canvas` shouldn't be used if cross-browser compatibility is a concern. Existing polyfills are more than capable of rendering simpler bitmaps such as [charts/graphs](#), [visualizations](#), and even [starfields](#)! For these uses and many more, `canvas` is highly encouraged.

Recommended polyfills: [FlashCanvas](#), [ExplorerCanvas](#)

[View browser share %](#) [Edit this info](#)

others

За какво не остана време:

- ✓ File API
- ✓ Offline cache manifests
- ✓ Server-sent notifications
- ✓ Streaming API
- ✓ Web notifications
- ✓ Device orientation
- ✓ **БОНУС: Как да ползваме WebSockets с PHP**
Server => <http://net.tutsplus.com/tutorials/javascript-ajax/start-using-html5-websockets-today/>

