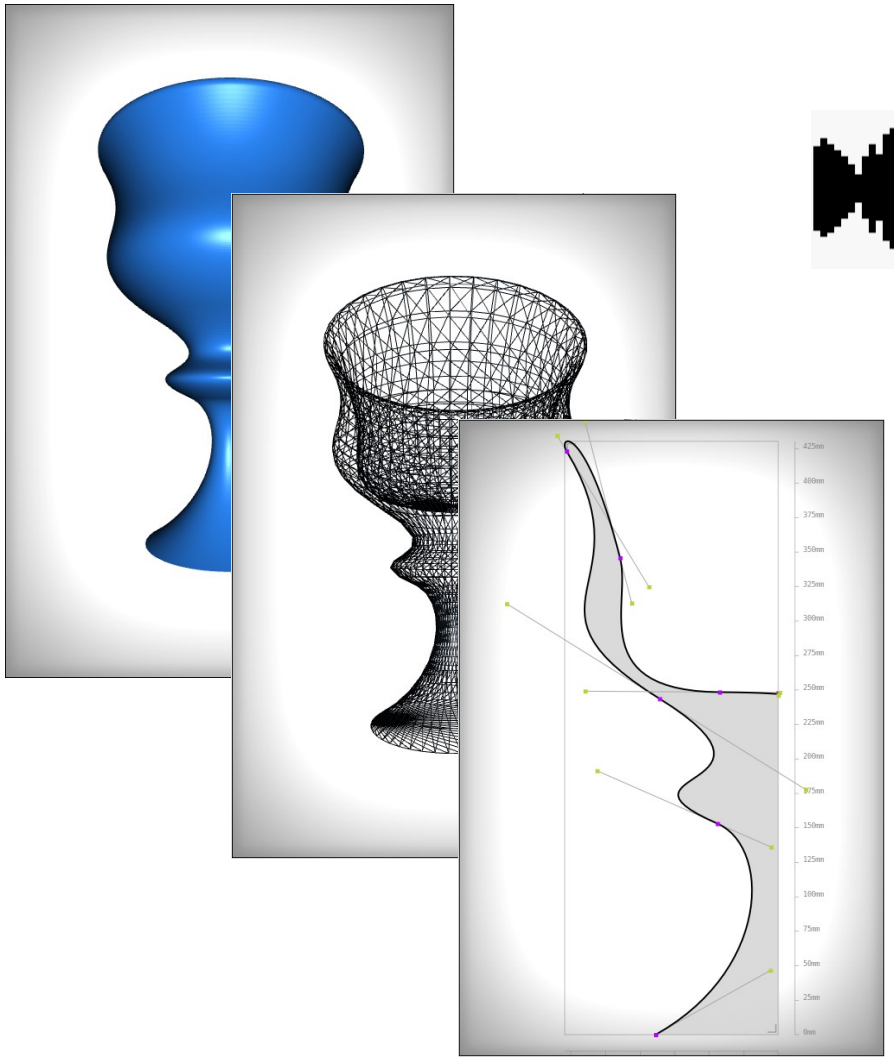
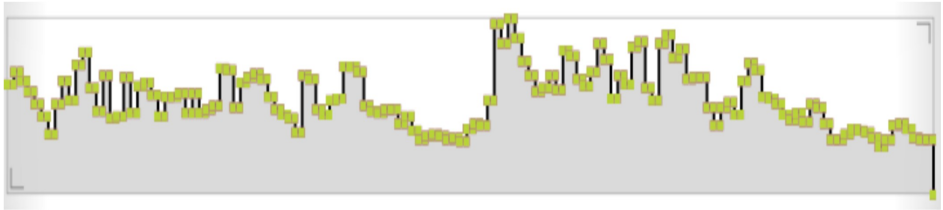
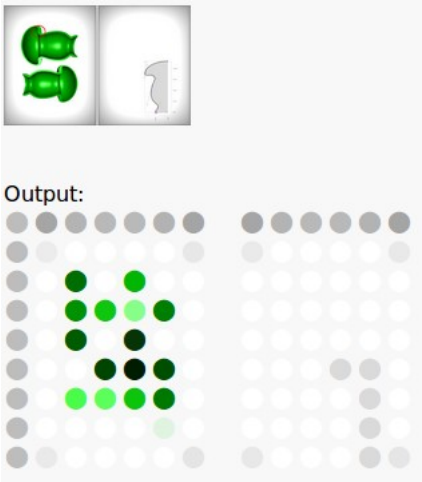
WebGL und WebAudio

h







WebGL: 3D im Webbrowser

- 3D-Grafik mit Hardwareunterstützung
- Seit ~2011

- Unterstützende Browserversionen

Internet Explorer	>= 11
Firefox	>= 33
Chrome	>= 38
Safari	>= 7.1
Opera	>= 24
iOS Safari	>= 8
Opera Mini	X
Android Browser	X
Chrome for Android	>= 38

Quelle: http://caniuse.com/#feat=webgl
Stand: 2014-10-15

WebGL: getting started

HTML

```
<canvas id="my_canvas"
        width="512"
        height="1024">
Your browser doesn't appear to
support the HTML5
<code>&lt;canvas&gt;</code>
element.
</canvas>
```

Javascript

```
try {
  var canvas =
    document.createElement('my_canvas');
  var context =
canvas.getContext('webgl') ||
canvas.getContext('experimental-webgl');
  var exts =
context.getSupportedExtensions();
} catch( e ) {
  console.log( "WebGL not supported." );
}
```

Hinweis/Bug: Canvasgröße nicht mit CSS definieren

WebGL: Würfelbeispiel

- `glDegMatrix.js`

  kleine Math-Bibliothek für Matrix- und Vektortransformationen

- `gpu.js`

  Hilfsfunktionen zur Shaderberechnung

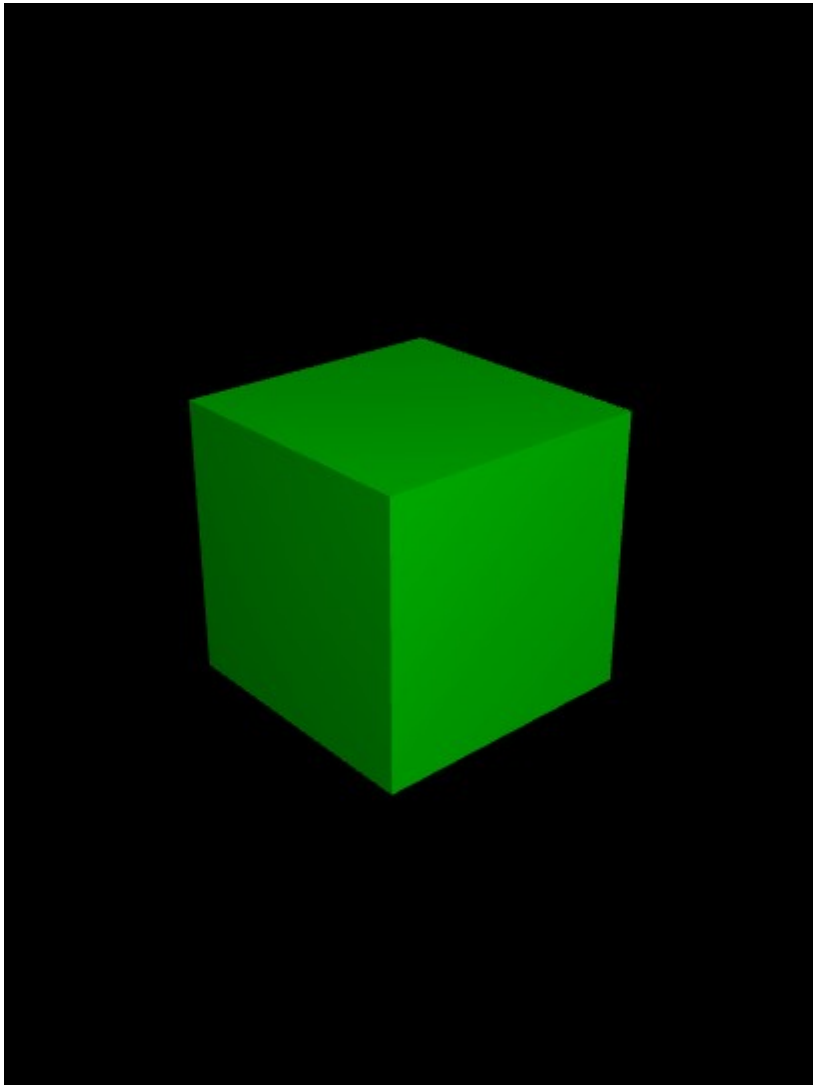
- Shaderdefinitionen:

```
<script id="shade_vert" type="x-shader/GLSL">
    #version 100
    precision mediump float;
[…]
</script>
```

  (OpenGL Shading Language, GLSL)

- `main.js`

```
[…]
```



Quelle:
http://cs.anu.edu.au/~Hugh.Fisher/webgl/index.html

Hugh.Fisher

Reines WebGL: lieber nicht

```
/**
 * @date   2014-10
 * @source https://developer.mozilla.org/en-
US/docs/Web/WebGL/Adding_2D_content_to_a_WebGL_co
ntext
 **/


var gl;      // A global variable for the WebGL
context


// Until requestAnimationFrame works everywhere,
use this code from Google
var requestAnimFrame = (function() {
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        window.msRequestAnimationFrame ||
        function(callback, element)
{ window.setTimeout(callback, 1000 / 60); };
})();


// Cube geometry
var cubeVerts = new Float32Array([
    -0.5,  0.5,  0.5,   // 0
    -0.5, -0.5,  0.5,
     0.5,  0.5,  0.5,
     0.5, -0.5,  0.5,
    -0.5,  0.5, -0.5,   // 4
    -0.5, -0.5, -0.5,
    -0.5,  0.5,  0.5,
    -0.5, -0.5,  0.5,
     0.5,  0.5, -0.5,   // 8
    -0.5,  0.5, -0.5,
     0.5,  0.5,  0.5,
    -0.5,  0.5,  0.5,
     0.5,  0.5,  0.5,   // 12
     0.5, -0.5,  0.5,
     0.5,  0.5, -0.5,
     0.5, -0.5, -0.5,
     0.5, -0.5,  0.5,   // 16
    -0.5, -0.5,  0.5,
     0.5, -0.5, -0.5,
    -0.5, -0.5, -0.5,
     0.5,  0.5, -0.5,   // 20
     0.5, -0.5, -0.5,
    -0.5,  0.5, -0.5,
    -0.5, -0.5, -0.5,
]);
```

```
var cubeNorms = new Float32Array([
     0,  0,  1,
     0,  0,  1,
     0,  0,  1,
     0,  0,  1,
    -1,  0,  0,
    -1,  0,  0,
    -1,  0,  0,
    -1,  0,  0,
     0,  1,  0,
     0,  1,  0,
     0,  1,  0,
     0,  1,  0,
     1,  0,  0,
     1,  0,  0,
     1,  0,  0,
     1,  0,  0,
     0, -1,  0,
     0, -1,  0,
     0, -1,  0,
     0, -1,  0,
     0,  0, -1,
     0,  0, -1,
     0,  0, -1,
     0,  0, -1,
]);


var cubeIdx = new Uint16Array([
     0,  1,  2,  1,  3,  2,
     4,  5,  6,  5,  7,  6,
     8,  9, 10,  9, 11, 10,
    12, 13, 14, 13, 15, 14,
    16, 17, 18, 17, 19, 18,
    20, 21, 22, 21, 23, 22,
]);


// The scene. Really ought to be an object
var cubeVBuf   = -1;
var cubeNBuf   = -1;
var cubeIdxBuf = -1;


var pMatrix    = mat4.create();
var mvMatrix   = mat4.create();
var cubeColor  = [ 0, 1, 0, 1 ];
var cubeSpin   = 0;


// Shader program and handles to uniforms, attributes
// Also should be an object
var gpuShade          = null;
var hProjectionMatrix = -1;
var hModelViewMatrix  = -1;
var hNormalMatrix     = -1;
var hLightPos         = -1;
var hColor            = -1;
var vaPosition        = -1;
var vaNormal          = -1;



// Setting up WebGL
var initShaders = function() {
    var vShader, fShader;


    vShader = gpu.loadShader(gl.VERTEX_SHADER, "shade_vert");
    fShader = gpu.loadShader(gl.FRAGMENT_SHADER, "shade_frag");
    gpuShade = gpu.newProgram(vShader, fShader);
```

```


    hProjectionMatrix = gpu.getUniform(gpuShade, "gProjectionMatrix");
    hModelViewMatrix  = gpu.getUniform(gpuShade, "gModelViewMatrix");
    hNormalMatrix     = gpu.getUniform(gpuShade, "gNormalMatrix");
    hLightPos = gpu.getUniform(gpuShade, "gLightPos");
    hColor = gpu.getUniform(gpuShade, "gColor");


    vaPosition = gpu.getAttribute(gpuShade, "vPosition");
    vaNormal   = gpu.getAttribute(gpuShade, "vNormal");
};


var createCube = function() {
// Transfer data to GPU
cubeVBuf = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, cubeVBuf);
gl.bufferData(gl.ARRAY_BUFFER, cubeVerts, gl.STATIC_DRAW);
gl.bindBuffer(gl.ARRAY_BUFFER, null);


cubeNBuf = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, cubeNBuf);
gl.bufferData(gl.ARRAY_BUFFER, cubeNorms, gl.STATIC_DRAW);
gl.bindBuffer(gl.ARRAY_BUFFER, null);


cubeIdxBuf = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeIdxBuf);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, cubeIdx, gl.STATIC_DRAW);
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);
};


var initGL = function(canvas) {
    // Do we have a context?
    try {
        gl = canvas.getContext("webgl") ||
canvas.getContext("experimental-webgl");
        gl.viewportWidth = canvas.width;
        gl.viewportHeight = canvas.height;
    } catch(e) {
        gl = null;
    }
    if (! gl) {
        alert("Could not get WebGL context: does your browser support
WebGL?");
    }
    // Regular OpenGL setup
    gl.clearColor(0, 0, 0, 1);
    gl.enable(gl.DEPTH_TEST);
    //gl.enable(gl.CULL_FACE);
    mat4.identity(pMatrix);
    mat4.identity(mvMatrix);
    initShaders();
    createCube();
};
```

```
// Rendering the scene.
// I always separate into projection - viewpoint - world

var setProjection = function() {
    mat4.perspective(60, gl.viewportWidth / gl.viewportHeight, 0.1, 10.0, pMatrix);
}

var setViewpoint = function() {
    mat4.lookAt([0, 2, 4], [0, 0, 0], [0, 1, 0], mvMatrix);
}

var drawWorld = function() {
    mat4.rotate(mvMatrix, cubeSpin, [0, 1, 0], mvMatrix);

    var nv3 = mat4.toInverseMat3(mvMatrix);
    mat3.transpose(nv3, nv3);
    var nvMatrix = mat3.toMat4(nv3);

    gl.useProgram(gpuShade);
    gl.uniformMatrix4fv(hProjectionMatrix, false, pMatrix);
    gl.uniformMatrix4fv(hModelViewMatrix, false, mvMatrix);
    gl.uniformMatrix4fv(hNormalMatrix, false, nvMatrix);
    gl.uniform4f(hLightPos, 0.5, 1.0, 1.0, 0.0);
    gl.uniform4f(hColor, cubeColor[0], cubeColor[1], cubeColor[2], cubeColor[3]);

    gl.bindBuffer(gl.ARRAY_BUFFER, cubeVBuf);
gl.enableVertexAttribArray(vaPosition);
gl.vertexAttribPointer(vaPosition, 3, gl.FLOAT, false, 0, 0);

    gl.bindBuffer(gl.ARRAY_BUFFER, cubeNBuf);
gl.enableVertexAttribArray(vaNormal);
gl.vertexAttribPointer(vaNormal, 3, gl.FLOAT, false, 0, 0);

gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeIdxBuf);
    gl.drawElements(gl.TRIANGLES, cubeIdx.length, gl.UNSIGNED_SHORT, 0);
}

var draw = function() {
    gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    try {
        setProjection();
        setViewpoint();
        drawWorld();
    } catch(e) {
        alert("draw: " + e.message);
    }

    cubeSpin += 0.5;
    requestAnimFrame(draw);
};

function cubeMain() {

    //window.alert( "jo" );

    var canvas = document.getElementById("my_canvas");
    try {
        initGL(canvas);
        draw();
    } catch(e) {
        alert("initGL: " + e.message);
    }
}


// Add the event listener
window.addEventListener( "load", cubeMain, false );
```

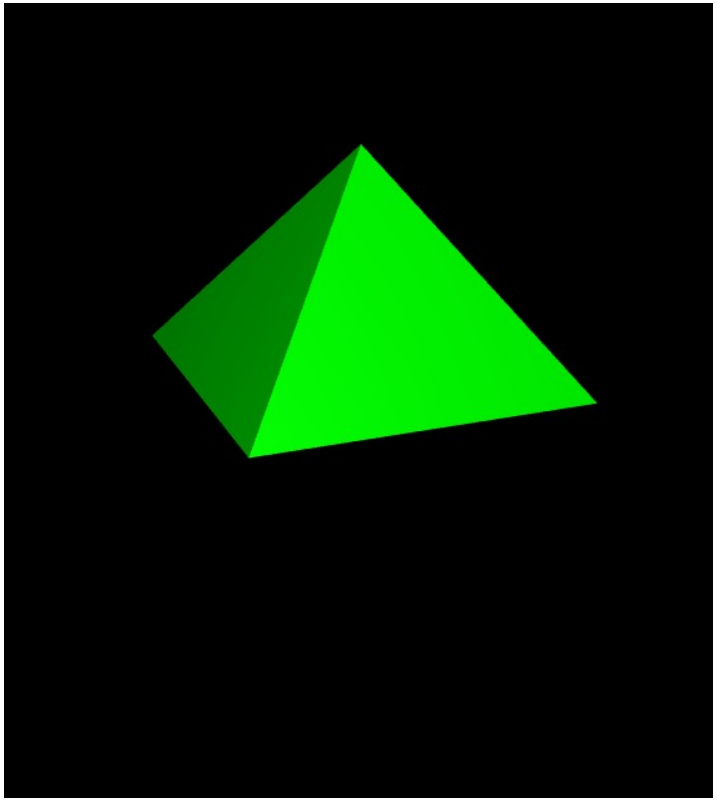
WebGL: Use libraries

THREE.js

```
// revolutions per second
var angularSpeed = 0.2;
var lastTime = 0;

// this function is executed on each animation frame
function animate(){
  // update
  var time = (new Date()).getTime();
  var timeDiff = time - lastTime;
  var angleChange = angularSpeed * timeDiff * 2 * Math.PI /
1000;
  cube.rotation.y += angleChange;
  lastTime = time;

  // render
  renderer.render(scene, camera);

  // request new frame
  requestAnimationFrame(function(){
      animate();
  });
}

// renderer
var renderer = new THREE.WebGLRenderer( { canvas:
document.getElementById("my_canvas") } );
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

// camera
var camera = new THREE.PerspectiveCamera(45, window.innerWidth /
window.innerHeight, 1, 1000);
camera.position.z = 800;
camera.position.y = 300;
camera.lookAt( new THREE.Vector3(0,0,0) );

// scene
var scene = new THREE.Scene();

// cube
var cube = new THREE.Mesh(new THREE.CubeGeometry(200, 200, 200), new
THREE.MeshPhongMaterial( { color: 0x00ff00 } ) );
cube.overdraw = true;
scene.add(cube);

// Add light
var light = new THREE.PointLight(0xFFFFFF);
light.position.z = 800;
light.position.y = 500;
scene.add( light );

// start animation
animate();
```

THREE.js: Datenstrukturen definieren

```
var IKRS = IKRS || {};

IKRS.PyramidGeometry = function( size ) {

    // Call super 'constructor'
    THREE.Geometry.call( this );

    var halfSize = size/2.0;

    // Add base points
    this.vertices.push( new THREE.Vector3(halfSize,0,halfSize) );    // at index 0
    this.vertices.push( new THREE.Vector3(halfSize,0,-halfSize) );   // at index 1
    this.vertices.push( new THREE.Vector3(-halfSize,0,-halfSize) );  // at index 2
    this.vertices.push( new THREE.Vector3(-halfSize,0,halfSize) );   // at index 3

    // Add top point
    this.vertices.push( new THREE.Vector3(0,size*0.66,0) );          // at index 4

    // Add faces
    this.faces.push( new THREE.Face3(4,0,1) );
    this.faces.push( new THREE.Face3(4,1,2) );
    this.faces.push( new THREE.Face3(4,2,3) );
    this.faces.push( new THREE.Face3(4,3,0) );

    // Add base face
    this.faces.push( new THREE.Face4(0,1,2,3) );

    this.computeCentroids();
    this.computeFaceNormals();
};

IKRS.PyramidGeometry.prototype = new THREE.Geometry();
IKRS.PyramidGeometry.prototype.constructor = IKRS.PyramidGeometry;
```

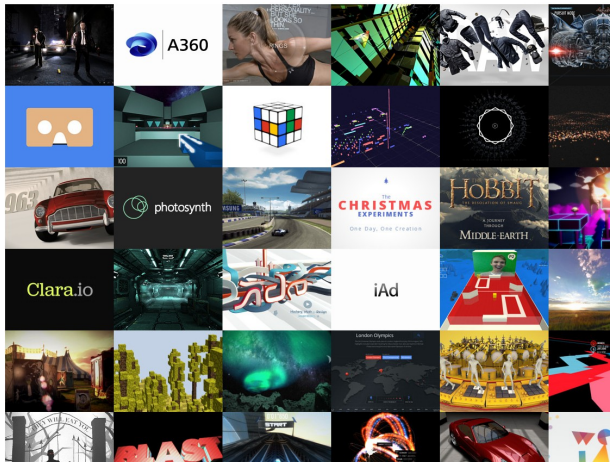


THREE.js: Docs

- http://threejs.org

  http://threejs.org/docs

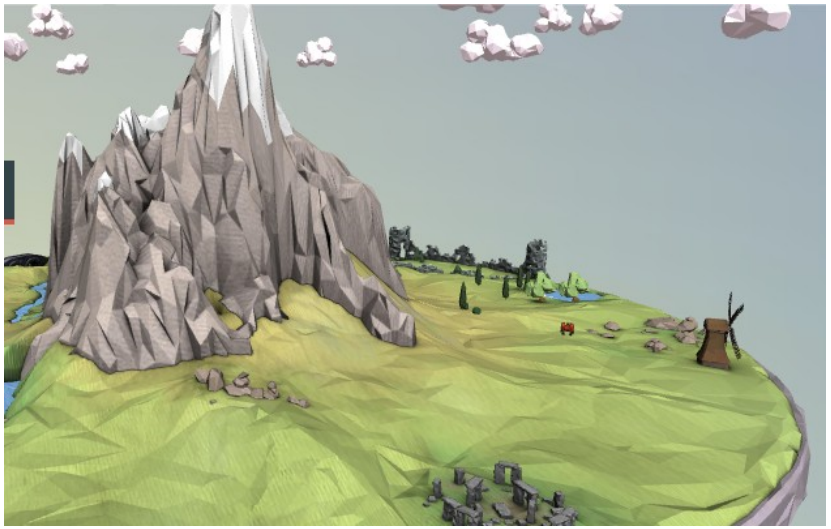
  http://github.com/mrdoob



- Andere Plattform:

  PlayCanvas

  http://playcanvas.com/



HTML5: FileReader class

https://developer.mozilla.org/en-US/docs/Web/API/FileReader

```
/** Pass an array of files **/
function readFile( file ) {
        if( file.files && file.files[0] ) {
            var reader = new FileReader();
            reader.onload = function(event) {
                //var dataURI = event.target.result;
                var arrayBuffer = event.target.result;

                Return arrayBuffer;
            };

            reader.onerror = function(event) {
                console.error( "File could not be read! Code " +
                                event.target.error.code );
            };
            //reader.readAsDataURL(file.files[0]);
            reader.readAsArrayBuffer( file.files[0] ); // Read as binary data!
        } else {
            window.alert( "Error: no files found to be read." );
        }
}
var arrayBuffer =
     readFile( document.forms["my_form"].elements["my_file"] );
```

Dateien lokal vom Massenspeicher beim Besucher lesen :)

Mögliche Formate:

- Binary (ArrayBuffer<byte>)
- Binary (String)
- Data-URI (String base64)
- Text (String)

Vorteil:

- keine Serverkommunikation
- Vermeidet

  Urheberrechtsprobleme :)

WebAudio

WebAudio initialisieren:

...

```
// Thanks to
//  http://www.html5rocks.com/en/tutorials/webaudio/intro/?redirect_from_locale=de
var context       = null;
Var audioAnalyzer = null;
// Fix up for prefixing
try {
    window.AudioContext =
         window.AudioContext || window.webkitAudioContext;
    context = new AudioContext();
    document.getElementById( "status_div" ).innerHTML =
        "AudioContext successfully created.";
    audioAnalyzer = new IKRS.AudioAnalyzer( context );
} catch( e ) {
    console.log( "Web Audio API is not supported in this browser." );
    throw "Web Audio API is not supported in this browser.";
}
```

HTML5: Read Audio File

```
var AudioFileReader = {
    readAudioFile : function() {
        var audioFile = document.getElementById( "input_audio_file" );
        AudioFileReader._readAudioFile( audioFile );
    },
    _readAudioFile : function( audioFile ) {
        if( audioFile.files && audioFile.files[0] ) {
            var reader = new FileReader();
            reader.onload = function(event) {
                //var dataURI = event.target.result;
                var arrayBuffer = event.target.result;

                // Global instance audioAnalyzer
                audioAnalyzer.setAudioByArrayBuffer( arrayBuffer );
            };

            reader.onerror = function(event) {
                console.error( "File could not be read! Code " +
                               event.target.error.code);
            };
            //reader.readAsDataURL(audioFile.files[0]);
            reader.readAsArrayBuffer( audioFile.files[0] ); // Read as binary data!
        } else {
            window.alert( "Error: no audio files found to be read." );
        }
    }
};
```

...

WebAudio: Audiodaten abspielen

Fetch the form data:

```
[...]

reader.onload = function(event) {

    var arrayBuffer = event.target.result;

    // Global instance audioAnalyze

    audioAnalyzer.playAudioByArrayBuffer(

         arrayBuffer

    );

};

reader.readAsArrayBuffer( audioFile.files[0] );
```

Play from audio buffer:

```
IKRS.AudioAnalyzer.prototype.playAudioByArrayBuffer = function( arrayBuffer ) {

    // Create a callback function
    var audio_decoded = function( audioBuffer ) {

       // Play data
       var src    = audioAnalyzer.context.createBufferSource();
       src.buffer = audioBuffer;
       src.connect( audioAnalyzer.context.destination );

       // Play immediately
       if( src.noteOn ) // Mozilla
             src.noteOn(0);
       else
             src.start(0);

    }
    // Start the decoder
    this.context.decodeAudioData(
                           arrayBuffer,
                                 audio_decoded,
                                 function( errmsg ) {
                             window.alert( "error: " + errmsg );
                          }
                         );
}; // END function
```

WebAudio: Audiodaten analysieren

Amplitude über die Zeit (Kanal 0):

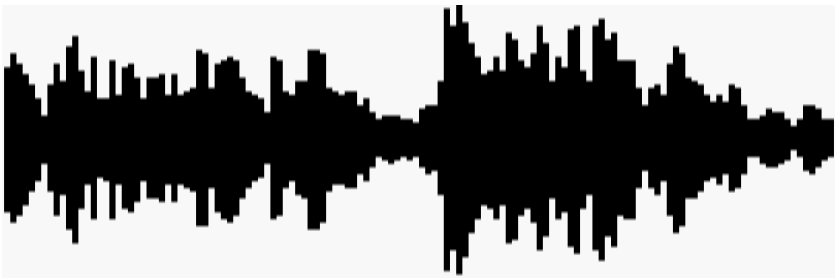
```
/**
 * @returns {Float32Array} Array of peaks.
 */
IKRS.AudioAnalyzer.prototype.getPeaks = function( buffer, length ) {
    var sampleSize = buffer.length / length;
    var sampleStep = ~~(sampleSize / 10) || 1;
    var channels   = buffer.numberOfChannels;
    var peaks      = new Float32Array(length);
    for (var c = 0; c < channels; c++) {
        var chan = buffer.getChannelData(c);
        for (var i = 0; i < length; i++) {
            var start = ~~(i * sampleSize);
            var end   = ~~(start + sampleSize);
            var max   = 0;
            for (var j = start; j < end; j += sampleStep) {
                var value = chan[j];
                if (value > max) {
                   max = value;
                // faster than Math.abs
                } else if (-value > max) {
                   max = -value;
                }
            }
            if (c == 0 || max > peaks[i]) {
                peaks[i] = max;
            }
        } // END for
    } // END for
    return peaks;
};
```

Quelle:

wavesurfer.js

https://github.com/katspaugh/wavesurfer.js

http://www.wavesurfer.fm/



Canvas und Bilddaten

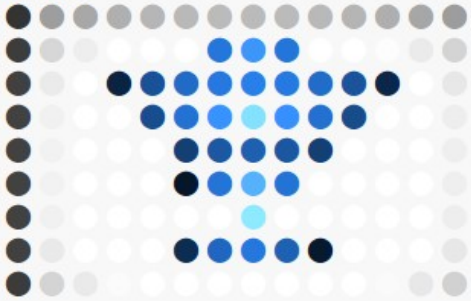
```
var context = null;
function processImageDataURI( dataURI ) {

    // A data URI:
    //   [...]

    // Display the image in <img id="preview_image" />
    var imgElement     =
           document.getElementById( "preview_image" );
    imgElement.src     = dataURI;

    // Create in-memory image object
    var image          = new Image();
    image.src          = dataURI;

    // Create an invisible in-memory canvas
    var tmpCanvas      = document.createElement( "my_canvas" );
    context     = tmpCanvas.getContext( "2d" );
    context.drawImage( image, 0, 0,
               tmpCanvas.width, tmpCanvas.height );

}
```

Bilddaten per `reader.readAsDataURL(...)` gelesen.

```
var x = …;
var y = …;
var imageData      =
      context.getImageData( x, y, 1, 1 );
var red            = imageData.data[0];
var green          = imageData.data[1];
var blue           = imageData.data[2];
var alpha          = imageData.data[3];
```

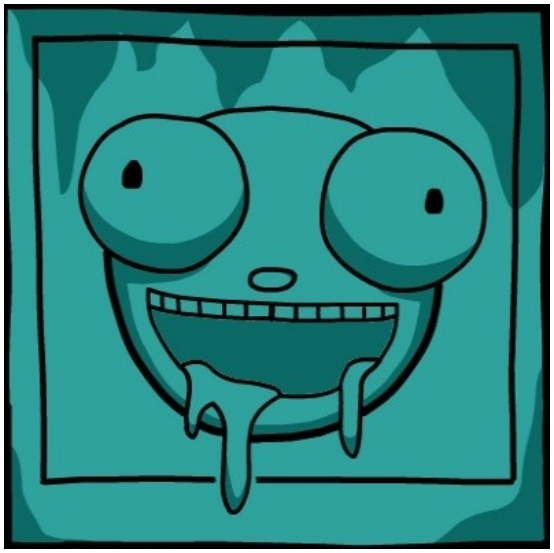
```
var colorHTML =
     "rgb( " + red +
     ","     + green +
     ","     + blue +
     ")";
```

Output:



Benutze Gummihuhn mit Karabinerhaken

…



https://github.com/IkarosKappler

2014-10-16