Mobile Web Appplications Development with HTML5

Lecture 4: Canvas, File & Device Access

Claudio Riva

Aalto University March 2012

Lecture 4: Canvas, File & Device Access

Canvas API

Bytes, Blobs and File API

HTML Media Capture API

Motion Sensor API

GeoLocation API

Audio & Video

Canvas API WHATWG

One of the largest part of the HTML5 specification

It provides an API for 2D drawing

Some examples:

- Cut the Rope
- SketchPad
- Harmony

Bitmap vs Vector Graphics

A **bitmap** is just a list of color values for each pixels of the image (e.g. jpg, png, gif)

Fixed resolution

Large file

Faster to load and display

A **vector graphic** is a list of instructions how to draw an image (e.g. SVG, VML)

Resolution independent

Smaller file

Processor intensive

Canvas API

Javascript API providing low-level primitives for drawing on the Canvas

Interface of the Canvas element:

- **getContext('2d')**: returns the object for accessing the Javascript Canvas API
- **toDataURL([DOMString type])**: returns a data URL object for the image on the canvas. Type can be 'image/png', 'image/jpeg'
- **toBlob(callback, [DOMString type])**: creates a Blob representation of the image on the canvas and passes to the callback function.

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

```
//Get the 2D Canvas context object
var ctx = document.getElementById('canvas').getContext('2d');
```

Canvas API Cheat Sheet

Drawing tools

- Rectangles
- Arcs
- Paths and line drawing
- Bezier and quadratic curves

Transformations

- Scaling
- Rotation
- Translation
- Transformation matrix

Effects

- Fills and strokes
- Shadows
- Linear and radial gradients
- Alpha transparency
- Compositing

Importing/exporting images

- Importing images by URL, other canvases or capturing video
- Taking a snapshot of the canvas as a data URI

Drawing on the canvas

```
var ctx = document.getElementById('canvas_1').getContext('2d');
ctx.fillStyle = 'rgb(0, 255, 0)';
ctx.fillRect(10, 20, 50, 50); //Create a solid square
ctx.strokeStyle = 'rgb(0, 182, 0)';
ctx.lineWidth = 5;
ctx.strokeRect(9, 19, 52, 52); //Draws an outline
gradient = ctx.createLinearGradient(0, 0, 0 , 100);
gradient.addColorStop(0, '#fff');
gradient.addColorStop(1, '#f00');
ctx.fillStyle = gradient;
ctx.fillRect(100, 10, 100, 100);
```

Drawing paths

```
ctx.beginPath();
ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
ctx.closePath();
ctx.fillStyle = "rgb(255, 255, 0)";
ctx.fill();

ctx.fillStyle = "rgb(0, 0, 0)";
ctx.beginPath();
ctx.arc(120, 130, 20, 0, 2 * Math.PI, false);
ctx.closePath();
ctx.fill();

ctx.beginPath();
ctx.arc(180, 130, 20, 0, 2 * Math.PI, false);
ctx.closePath();
ctx.fill();

ctx.beginPath();
ctx.arc(150, 150, 70, Math.PI/10, Math.PI*
(1-1/10), false);
ctx.lineWidth = 5;
ctx.strokeStyle = "rgb(0,0,0)";
ctx.stroke();
```

Drawing an image

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

```
var img = new Image();

//wait until the image has loaded
img.onload = function () {
  ctx.drawImage(this, 0, 0);
};

img.src = "banksy.jpg"
```

Source can be an image, a canvas or a video.

Pixel-level Processing

```
var pixels = getImageData(sx, sy, sw, sh)
putImageData(pixels, dx, dy)
pixels => [ r1, g1, b1, a1, r2, g2, ...]
```

```
var img = new Image();
img.onload = function () {
  ctx.drawImage(this, 0, 0);
  var a = ctx.getImageData(0, 0, this.width,
this.height);
  var b = $.map(a, function(v,i) {
    if (i % 4 < 3) return 255-v;
    else return v;
  });
  ctx.putImageData(b, 0, 0);
};
img.src = "imgs/lesson4/street-maid-graffiti-
banksy.jpg"
```

Animating the canvas

Animations can be achieved by re-drawing the canvas:

1. Initialize the canvas and the objects
2. Clear the canvas
3. Draw the canvas

Draw Smile

```
function drawSmile(ctx) {
  ctx.beginPath();
  ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
  ctx.closePath();
  ctx.fillStyle = "rgb(255, 255, 0)";
  ctx.fill();

  ctx.fillStyle = "rgb(0, 0, 0)";
  ctx.beginPath();
  ctx.arc(120, 130, 20, 0, 2 * Math.PI, false);
  ctx.closePath();
  ctx.fill();

  ctx.beginPath();
  ctx.arc(180, 130, 20, 0, 2 * Math.PI, false);
  ctx.closePath();
  ctx.fill();

  ctx.beginPath();
  ctx.arc(150, 150, 70, Math.PI/10, Math.PI*
(1-1/10), false);
  ctx.lineWidth = 5;
  ctx.strokeStyle = "rgb(0,0,0)";
  ctx.stroke();
}
```

Draw Smile that blinks

```
function drawSmileBlink(ctx) {
  ctx.beginPath();
  ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
  ctx.closePath();
  ctx.fillStyle = "rgb(255, 255, 0)";
  ctx.fill();

  ctx.fillStyle = "rgb(0, 0, 0)";
  ctx.lineWidth = 5;
  ctx.beginPath();
  ctx.moveTo(100,120);
  ctx.lineTo(140,140);
  ctx.closePath();
  ctx.stroke();

  ctx.beginPath();
  ctx.arc(180, 130, 20, 0, 2 * Math.PI, false);
  ctx.closePath();
  ctx.fill();

  ctx.beginPath();
  ctx.arc(150, 150, 70, Math.PI/10, Math.PI*
(1-1/10), false);
  ctx.lineWidth = 5;
  ctx.strokeStyle = "rgb(0,0,0)";
  ctx.stroke();
}
```

Blinking Smile

```
function clear(ctx) { ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height); }
var timer1 = setInterval(function() {
  clear(ctx);
  drawSmileBlink(ctx);

  var timer2 = setTimeout(function() {
    clear(ctx);
    drawSmile(ctx);
  }, 300);

}, 5000);
```

Rotating Blinking Smile canvasSmile

Rotating Blinking Smile - 1

```
function createCtx() {
  var canvas = document.createElement('canvas');
  canvas.width = 200;
  canvas.height = 200;
  return canvas.getContext('2d');
}
//Create all the contexts
var smileCtx = createCtx();
var smileBlinkCtx = createCtx();
var mainCtx = createCtx();

//Draw the smiles on their own canvas
drawSmile(smileCtx);
drawSmileBlink(smileBlinkCtx);

//Translate the center of the main context
mainCtx.translate(100,100);
mainCtx.drawImage(smileCtx.canvas, 0, 0, 200, 200, -100, -100, 200, 200);
ctx.drawImage(mainCtx.canvas, 0, 0);
```

Rotating Blinking Smile - 2

```
var current = smileCtx.canvas;
var timer1 = setInterval(function() {
  current = smileBlinkCtx.canvas;
  var timer2 = setTimeout(function() {
    current = smileCtx.canvas;
  }, 300);
}, 5000);

//Draw on the canvas every 50ms
var timer3 = setInterval(function() {
  clear(ctx);
  mainCtx.rotate(Math.PI/180*5);
  mainCtx.drawImage(current, 0, 0, 200, 200, -100, -100, 200, 200);
  ctx.drawImage(mainCtx.canvas, 0, 0);
}, 50);
```

Exporting the canvas as a Data URI canvasInstagram

- Take a snapshot of the canvas and store it into PNG/JPEG image
- Image is a Base64 encoded data URL
- Cross origin policy

```
image = ctx.canvas.toDataURL('image/png');
```

```
image => ...
```

Export Image as DataURL

Scalable Vector Graphics (SVG)

XML based vector graphics format

SVG objects are available in the DOM and can be created/traversed/manipulated like HTML elements

```
%svg(xmlns="http://www.w3.org/2000/svg" width="300" height="200")
  %text(style="font-size:30; stroke: blue" x="10"  y="100")
    Hello World
%rect(x="5" y="50" width="150" height="30" style="fill:yellow")
```

SVG vs Canvas

- Plenty of graphic tools export SVG
- SVG is retained in the DOM, the canvas not
- We can bind for events to SVG objects
- SVG handles scaling better than canvas that supports only zooming
- Canvas is more suitable for one-time drawing (no interaction and no events)

Javascript Libraries

- RaphaelJS
- JS Port of Processing language
- Pixastic
- Canvg
- SVG to Canvas

What can you do with a canvas ?

An App that

What can you do with a canvas ?

An App that

- is #1 in 79 countries

What can you do with a canvas ?

An App that

- is #1 in 79 countries
- has been downloaded 35+ million times just 6 weeks after launch

What can you do with a canvas ?

An App that

- is #1 in 79 countries
- has been downloaded 35+ million times just 6 weeks after launch
- generates $1 million+ revenues/day

What can you do with a canvas ?

An App that

- is #1 in 79 countries
- has been downloaded 35+ million times just 6 weeks after launch
- generates $1 million+ revenues/day
- has 15+ million users

What can you do with a canvas ?

An App that

- is #1 in 79 countries
- has been downloaded 35+ million times just 6 weeks after launch
- generates $1 million+ revenues/day
- has 15+ million users
- generates 3,000 drawings per second

What can you do with a canvas ?

An App that

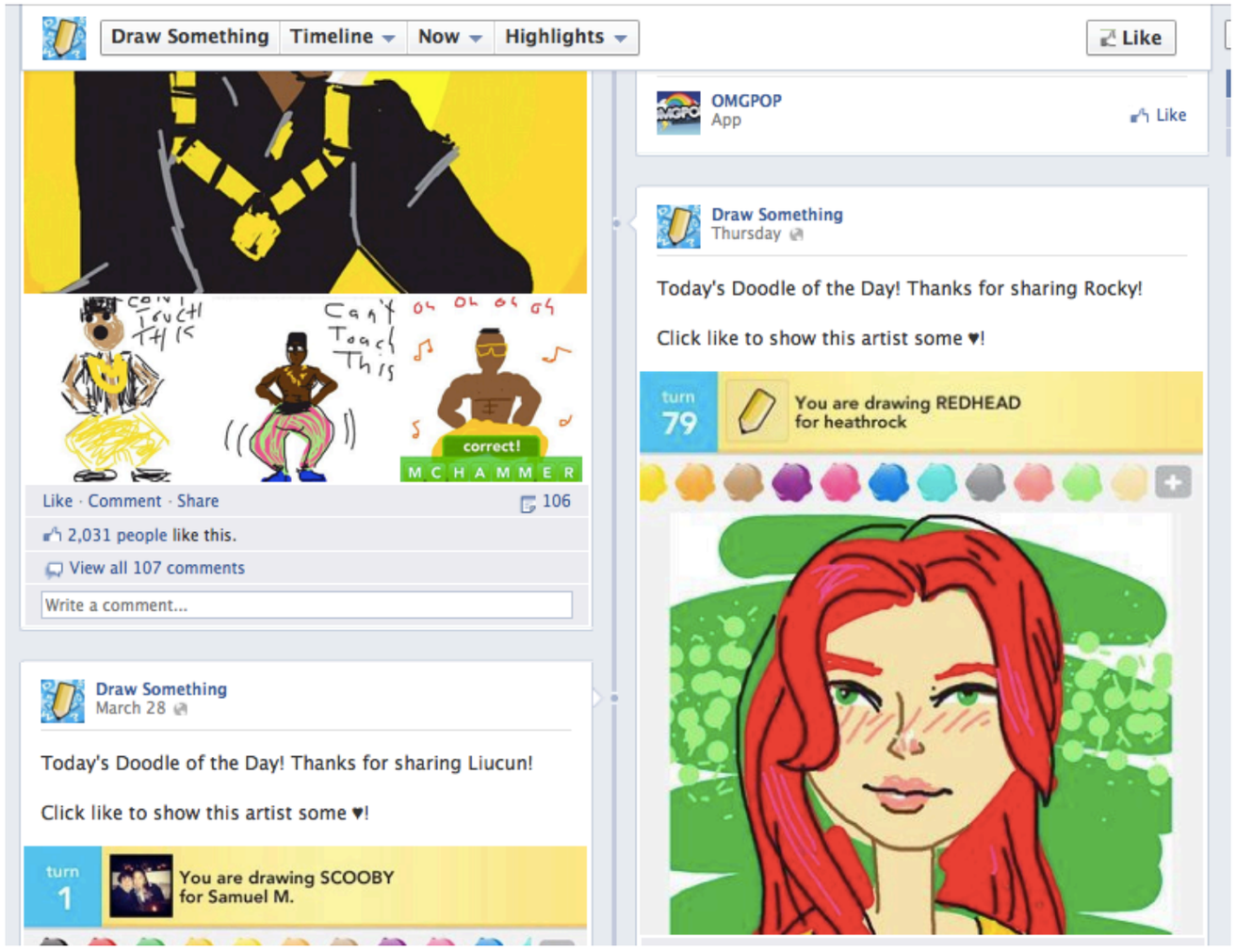
- is #1 in 79 countries
- has been downloaded 35+ million times just 6 weeks after launch
- generates $1 million+ revenues/day
- has 15+ million users
- generates 3,000 drawings per second
- generated 2 billion drawings since launch (15 Feb 2012)

What can you do with a canvas ?

An App that

- is #1 in 79 countries
- has been downloaded 35+ million times just 6 weeks after launch
- generates $1 million+ revenues/day
- has 15+ million users
- generates 3,000 drawings per second
- generated 2 billion drawings since launch (15 Feb 2012)
- reached 1 million users in 9 days

Draw Something



File API W3C

The File API provides the interfaces for accessing file objects in web applications:

- **`Blob`**: an interface for raw binary data
- **`File`**: readonly information about a file
- **`FileList`**: an array of individually selected files from the file system (e.g. through the file input tag)
- **`FileReader`**: asynchronous interface for reading the content of a File or Blob
- **`URI scheme`**: for referencing a blob (e.g. `blob:4342e43fa43-34ea53-4343dfa-37675`)

It works in conjunction witht the new XMLHttpRequest level 2

Bytes and Blobs (more info)

- `Blob` = Binary Large Object
- A binary data interchange mechanism (File API)
- Useful for moving data around
- Basic API for:
  - **size** : size in bytes of the blob
  - **type** : MIME type of the data contained in the blob
  - **slice(start, end, contentType)** : a new blob containing the data in the specified range of the source blob
- Read only
- May live on disk, so requires asynchronous access (e.g. FileReader)
- Created with the `BlobBuilder` interface or by slicing an exisitng blob

Why we need Blobs ?

- File uploading from the client file system to the server
- Downloading a file from web or filesystem and storing in localStorage
- Injecting a file from the file system into the DOM
- Decoding binary data (e.g. from websockets)
- Audio Synthesis
- Image processing

BlobBuilder

It provides the interface to construct a blob, append data and retrieve it:

- **append(ArrayBuffer | Blob | String)** : append chucks of data to the blob
- **getBlob(contentType)** : retrieves the blob with the desired MIME type.

```
var BlobBuilder = window.MozBlobBuilder || window.WebKitBlobBuilder || window.BlobBuilder;
b = new BlobBuilder();
b.append("Hello World");
b.getBlob('text/plain');
```

File and Blobs

`File` objects represent file data from the file system.

A `File` is a `Blob` (but a Blob is not a File) and in addition:

- **name**
- **lastModifiedDate**

`FileReader` object provides the asynchronous read methods to access the files

How to get a File

- With an input element of type file

```
<input type='file' multiple='true' accept='image/*'>
```

- With the drag-and-drop feature
- Read from a private filesystem sandbox (FileSystem API)

FileList

`FileList` objects are an array-like sequence of `File` objects

```
<input type='file' id="fileSelect" multiple='true'>
```

```
$("#fileSelect").live('change', function(e) {
  $.each(this.files, function (i,file){
    console.log(file.name, file.size, file.type, file.lastModifiedDate);
  }
}
```

Test here: Choose File no file selected

XMLHttpRequest Level 2

Old XMLHttpRequest was limited to send DomString or Document (XML) objects

New XMLHttpRequest level 2 allows to send the following types:

- DomString
- Document (XML)
- FormData
- Blob
- File
- ArrayBuffer

Sending a File fileAPITests

```
$.each(this.files, function (i,file){
  if (!file.type.match(/image.*/)) {
    return;
  }
  var xhr = new XMLHttpRequest();
  xhr.open('POST', '/uploadFile', true);
  xhr.onload = function(e) {console.log (this.responseText); };
  xhr.setRequestHeader("X-File-Name", file. name);
  xhr.setRequestHeader("Content-Type", "application/octet-stream");
  xhr.send(file); //XHR2
});
```

FormData

`FormData` objects provide a way to easily build a set of key/value pairs (like Forms).

The objects can be sent to the server with the XHR2 `send` method as a `multipart/form-data`

The interface allows to `append` a key/value pair where:

- **name** is the a string representing the key
- **value** is the value of the field: Blob, File or string

Sending files with FormData fileAPITests

```
var fd = new FormData(); // XHR2
$.each(this.files, function (i,file){
  fd.append(file.name, file);
});

var xhr = new XMLHttpRequest();
xhr.open('POST', '/uploadFormData', true);
xhr.onload = function(e) {console.log (this.responseText); };

xhr.send(fd); // as multipart/form-data
```

Downloading a Blob

XHR2 allows to specify the type of the response:

- **xhr.responseType** : `text` | `arraybuffer` | `blob` | `document`
- **xhr.response** : the requested data

What can we do with a Blob ?

Create an object and insert it in the DOM

- `URL.createObjectURL()`
- `URL.revokeObjectutL()`

Read the content with a FileReader:

- `FileReader.readAsText( File|Blob )`
- `FileReader.readAsBinaryString( File|Blob )`
- `FileReader.readAsArrayBuffer( File|Blob )`
- `FileReader.readAsDataURL( File|Blob )`

Messages to/form WebWorkers

Store in localStorage or IndexedDB

Downloading a Blob fileAPITests

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'logo.png', true);
xhr.responseType = 'blob';

xhr.onload = function(e) {
  if (this.status == 200) {
    var blob = this.response;

    var img = document.createElement('img');
    img.onload = function(e) {
      window.URL.revokeObjectURL(img.src); // Clean up after yourself.
    };
    img.src = window.URL.createObjectURL(blob);
    $('#blobOutput').append(img);
  }
};
xhr.send();
```

ArrayBuffer

`ArrayBuffer` is a generic container for binary data.

With typed arrays we can create different views from a single `ArrayBuffer` :

- DataView
- Float32Array
- Float64Array
- Int16Array
- Int32Array
- Int8Array
- Uint16Array
- Uint32Array
- Uint8Array

Downloading to an Array fileAPITests

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'logo.png', true);
xhr.responseType = 'arraybuffer';

xhr.onload = function(e) {
  if (this.status == 200) {
    var a = new Uint8Array(this.response);
    html = 'First 3 bytes: ' + a[0] + ',' + a[1] + ',' + a[2];
    $('#arrayOutput').html(html);
  }
};
xhr.send();
```

What can I do with an ArrayBuffer?

- Interface with Audio API
- WebGL
- WebSockets

`FileReader`

`FileReader` is an object that enables asynchronous reading of `File` and `Blob` objects. The following read methods are available:

- **readAsText** : the result is a text string with default UTF-8
- **readAsArrayBuffer** : the result is stored in a ArrayBuffer object
- **readAsBinaryString** : the result is a binary string where each byte is an integer in the range [0-255]
- **readAsDataURL** : the result is stored in a DataURL object using the data URI scheme (e.g. `...`)

For tracking the progress, the following callbacks are available: onloadstart, onprogress, onload, onabort, onerror, and onloadend.

FileReader.readAsText() fileAPITests

```
var reader = new FileReader();
reader.onload = function() {
  var text = reader.result;
  $("#fileTextOutput").html(text);
}
reader.readAsText(this.files[0]);
```

FileReader.readAsDataURL() fileAPITests

```
var reader = var reader = new FileReader();
    reader.onload = function() {
      var text = '<img src="'+reader.result+'"/>';
      $("#fileDataURLOutput").html(text);
    }
    reader.readAsDataURL(this.files[0]);
```

What works on mobile ?

	Firefox Android 11	Opera Mobile 11.50	Android 4	iOS 5	N9	Chrome 17	Firefox 11
XHR 2							
Upload Files	Yes	No	Yes	No	Yes	Yes	Yes
Text response type	Yes	No	Yes	Yes	No	Yes	Yes
Document response type	No	No	No	No	No	No	No
Array buffer response type	Yes	No	Yes	Yes	No	Yes	Yes
Blob response type	Yes	No	No	No	No	No	Yes
Files API							
FileReader API	Yes	Partial	Yes	No	No	Yes	Yes
FileSystem API	No	No	No	No	No	Yes	No
WebGL							
Native binary data	Partial (all but DataView)	No	Partial all but Float64Array)	Partial all but Float64Array)	No	Yes	Partial (all but DataView)

Opera doesn't support createObjectURL, revokeObjectURL, BlobBuilder, readAsArrayBuffer

- HTML5 Specs: File API , XHR2 , Native binary Array
- Opera: Opera Mobile File API
- Firefox: FireFox XHR2 , FireFox Typed Arrays , FileReader
- Html5Rocks Tricks: XHR2 , Local files

Good old `multipart/form-data`

Content type for submitting forms that contain binary data

Works with all browsers (except iOS)

```
%form(id="uploadFormPost" action="/uploadFormData" method="post" data-ajax='false'
enctype="multipart/form-data")
  %input(type="file" id="fileChoose0" name="file"  multiple="true" accept="image/*"
style="height:0;width:0;")
  %button(id="uploadFormPostButton") Upload via Form POST
```

```
$("#uploadFormPostButton").click(function(e) {
  e.preventDefault();
  $("#fileChoose0").click();
});
$("#fileChoose0").live('change', function() {
  $("#uploadFormPost").submit();
});
```

HTML Media Capture API W3C

Basic API for capturing media from the device (images, video and sound)

More powerful API will come with `getUserMedia`

Only implemented in Android Browser 3.0 onwards and Firefox for Android 11

Based on the extension of the `accept` attribute with the `capture` paramenter:

```
<input type="file" accept="image/*;capture=camera">
```

- **accept:** `"image/*", "sound/*", "video/*"`
- **capture:** `camera, camcoder, microphone, filesystem`

The MediaFile Interface

If the media capture succeds the `files` attribute contains `MediaFile` objects instead of `File` objects

`MediaFile` objects inherit from `File` and have the additional attribute `FormData`

`FormData` contains the attributes:

- codecs
- bitrate
- height
- width
- duration

Device Orientation API W3C

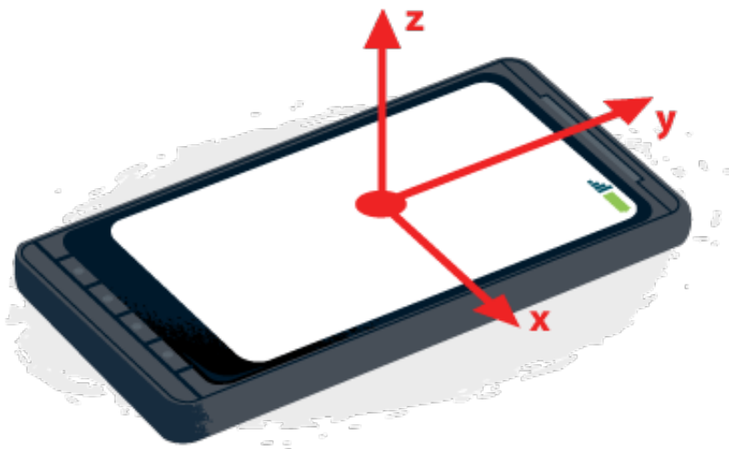
API to access the information of accelererometers, gyroscopes and compasses of mobile devices

DeviceOrientation Event (degrees)

- alpha: around the Z-axis
- beta: around the X-axis
- gamma: around the Y-axis

MotionOrientation Event (m/s2)

- acceleration
- accelerationIncludingGravity

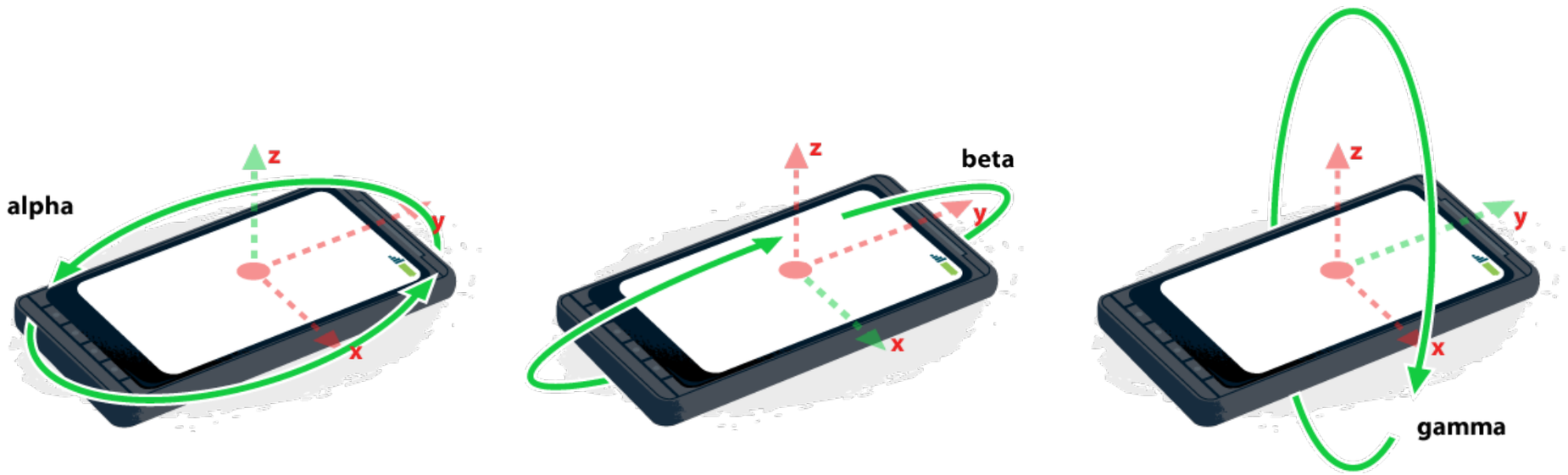


Device Orientation API

Heading (Yaw)
[0, 360)

Elevation (Pitch)
[-180, 180)

Bank (Roll)
[-90, 90)



Orientation Events deviceOrientation

```
$(function() {
if (window.DeviceOrientationEvent) {
  window.addEventListener("deviceorientation", function( event ) {
    }
    //alpha: rotation around z-axis
    $(".alpha").html(event.alpha);

    //gamma: left to right
    $(".beta").html(event.beta);

    //beta: front back motion
    $(".gamma").html(event.gamma);

    var rotation = "rotateX("+(event.beta*-1)+"deg)"
    rotation += " rotateY("+(event.gamma)+"deg)"
    rotation += "rotateZ("+(event.alpha*-1)+"deg)";

    $(".transform").html(rotation);
    $("#imgLogo").css('-moz-transform',rotation);
  }, false);
  }
}
```

Geolocation API W3C

- API for locating the user exact position that contains three methods:
  - **`getCurrentPosition`**: asynchronous request for current location of the user. The handler is invoked with a `Position` object
  - **`watchPosition`**: polling at regular intervals and if the location has changed invoke the handler with a `Position` object
  - **`clearWatch`**: cancel the watching request
- Browser displays a dialogue to request for user's permission
- Positioning happens in two ways:
  - IP, cell id, WiFi access point + some black magic from a geocoding provider (less accurate, fast, works indoor)
  - GPS for enabled devices (accurate, slower, works outdoor)

Geolocation API W3C

```
//Get location ASAP, regardless of accuracy
navigator.geolocation.getCurrentPosition(function(position) {
  do_something(position.coords.latitude, position.coords.longitude);
});

//Watch for location changes, request high accuracy
var wpid = navigator.geolocation.watchPosition(
  function(position) { do_something(position.coords.latitude, position.coords.longitude); },
  function(error) { show_error(error.code, error.message); },
  {enableHighAccuracy:true, maximumAge:30000, timeout:27000}
);
```

Position

- coords
- timestamp

Coordinates

- latitude (decimal degrees)
- longitude (decimal degrees)
- accuracy (radius in m.)
- altitude (in m.)
- altitudeAccuracy (in m.)
- heading (degrees)
- speed (in m/s)

Error

- `code`
  - UNKNOWN_ERROR (0)
  - PERMISSION_DENIED (1)
  - POSITION_UNAVAILABLE (2)
  - TIMEOUT (3)
- `message`

Maps and Location deviceGeolocation

```
var myLatLng;
var map;
var marker;

$(function() {
  //Get position fast and initialized google map
  navigator.geolocation.getCurrentPosition(function(geodata) {
    myLatLng = new google.maps.LatLng(geodata.coords.latitude, geodata.coords.longitude);

    var myOptions = { center: myLatLng, zoom: 8, mapTypeId: google.maps.MapTypeId.ROADMAP };
    map = new google.maps.Map(document.getElementById("map"), myOptions);
    marker = new google.maps.Marker( { position: myLatLng, map: map, title:"My position" });
  });
  //Update the position at least every 5 seconds and use GPS if available
  navigator.geolocation.watchPosition(function(geodata) {
    $(".latitude").html(geodata.coords.latitude);
    $(".longitude").html(geodata.coords.longitude);
    $(".accuracy").html(geodata.coords.accuracy);
    $(".altitude").html(geodata.coords.altitude);
    $(".heading").html(geodata.coords.heading);
    $(".speed").html(geodata.coords.speed);

    myLatLng = new google.maps.LatLng(geodata.coords.latitude, geodata.coords.longitude);
    marker.setPosition(myLatLng);
    map.setCenter(myLatLng);
  },function() {},{enableHighAccuracy:true, maximumAge:30000, timeout:5000} );
})
```

Audio & Video

- Open standard for delivery of multimedia without plugins
- HTML native video and audio elements
- No support for DRM
- No access to webcam or microphone
- Mix with HTML, JS and CSS

Video Formats

- **.mp4 = H.264 + AAC**
  - Not an open codec
  - Not royalty free
  - Only codec supported on iOS
  - Not supported by Firefox and Opera
- **.ogg / .ogv = Theora + Vorbis**
  - Supported by Firefox, Chrome and Opera
- **.webm = VP8 + Vorbis**
  - High-quality, royalty free and open video format
  - Supported in Firefox, Chrome, Opera and Andorid

Video videoTest

```
%video(controls)
  %source(src="gaga.webm" type='video/webm; codecs="vp8, vorbis"')
  %source(src="gaga.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"')
  Video tag not supported. Download the video
  %a(href="gaga.webm") here
```

Audio and Video Element Attributes

- autoplay: automatically start playing the audio/video
- controls: show the controls (implementation varies among browsers)
- poster: image to show while the video is downloading
- muted: mute the audio by default
- height, width
- loop: repeat playing the audio/video
- preload: start buffering the video