Buzz it for real!

... the tortuous road to Mobile HTML5 Apps

Andrea Giammarchi @WebReflection

Senior SW Engineer @NOKIA

Many Choices

where would you like to create your app ?

- Symbian ?
-
-

Many Choices

where would you like to create your app ?

- Symbian ?
- Meego ?
-

Many Choices

where would you like to create your app ?

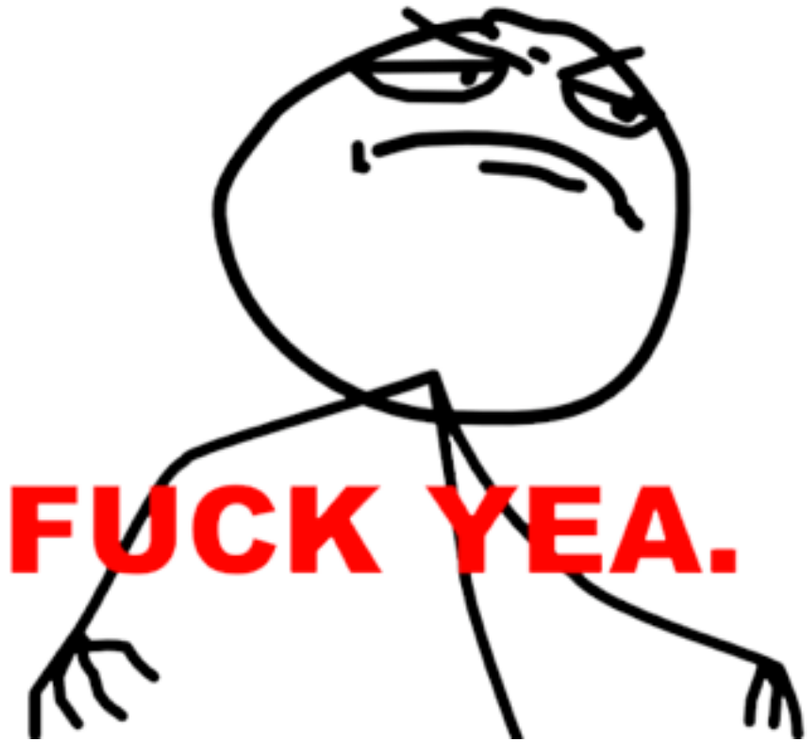
- Symbian ?
- Meego ?
- Windows Phone ?

My Choice

HTML5

My Choice

HTML5



HTML5

I am part of a team entirely focused on Mobile HTML5.
Our main goals are:

- performances
- cross platform compatibility
- user experience
- Kiss and YAGNI

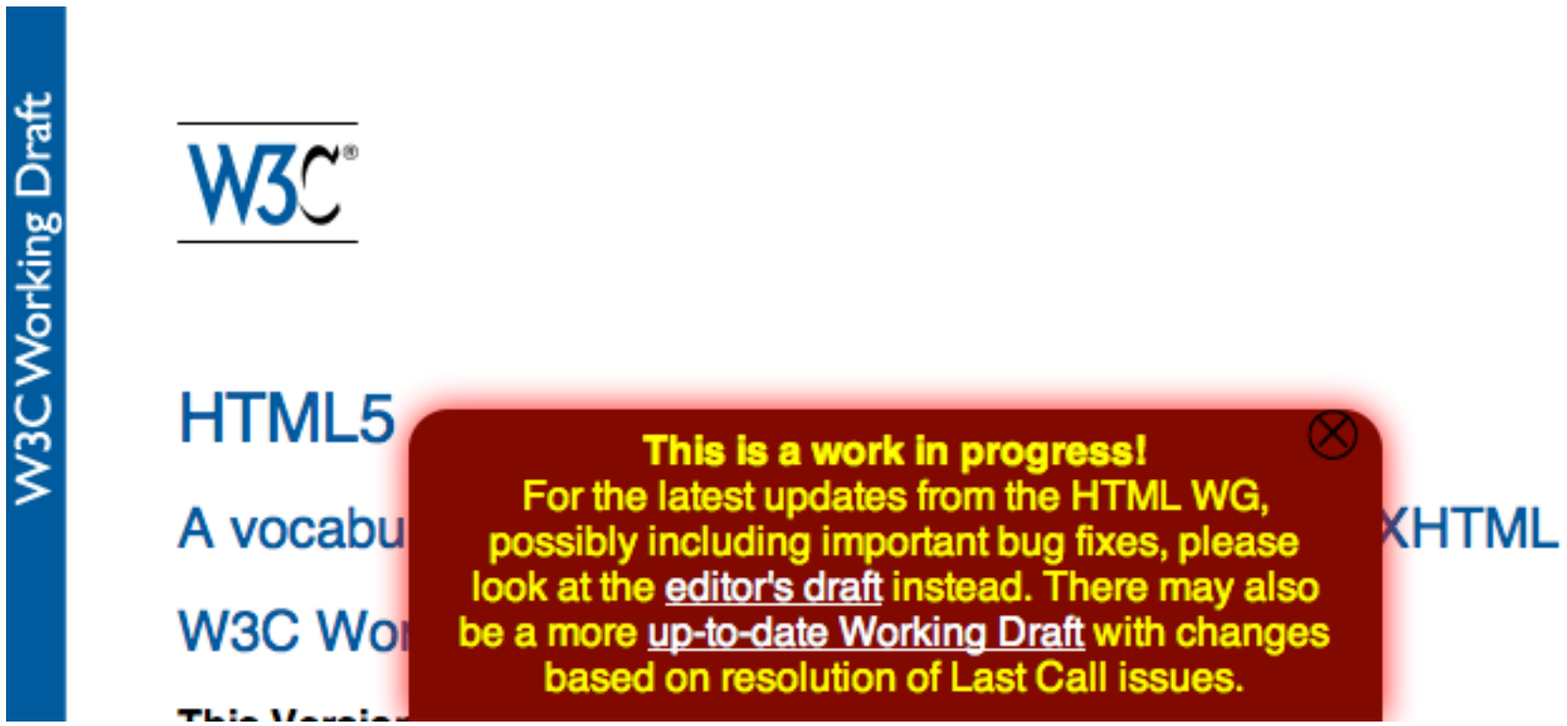
Mobile HTML5 Maps @Nokia

What Is HTML5 ?

One of the most overrated Buzz words in the WWW history ... well, it's something more concrete too:

- HTML5 in Wikipedia
- Dive Into HTML5
- HTML5 Rocks
- W3C HTML5

W3C HTML5


W3C Working Draft
W3C
HTML5
A vocabu
XHTML
W3C Wo
This is a work in progress!
For the latest updates from the HTML WG, possibly including important bug fixes, please look at the editor's draft instead. There may also be a more up-to-date Working Draft with changes based on resolution of Last Call issues.

HTML5

- Audio & Canvas (2D only, no WebGL yet)
- Application Cache
- Touches
- CSS3
- History
- GeoLocation
- Storage
- Web Workers & Sockets
- ... and much more buzzes ...

Coming Soon ...

- mic (voice recognition + calls)
- camera (face recognition + video calls)
- accelerometers (physics recognition)
- FileSystem
- device access (and its HardWare)
- link download=”fileName” (no server)
- text/event-stream
- StructType, ArrayType, ParallelsArray
- ... and much more buzzes ...

Nobody Scores !

- Html5Test
- MobileHtml5
- Modernizr

- fuckn.es
- ... and much more f#@$! ...

Polyfills

- SVG and Canvas + WebGL
- Web Storage
- Audio & Video
- Geo Location
- ... more ...

Features Detection

In order to provide fallbacks or inform the user about his browser lack of features.

The “in” operator is the safest one and should not produce false positives * :

API test example

```
“geolocation” in navigator
```

DOM Events (abusing Level 0 implementation)

```
“ontouchend” in document
```

* to know more about the safer “in”

Still Not Enough

Features detection may produce unexpected results such:

XMLHttpRequest Version 2

```
“addEventListener” in (new XMLHttpRequest)
```

failed!

The curious case of Android 1.6 via HTC Device Touch interface exposed only after the very first user interaction:

```
“ontouchend” in document; // false
“onmouseup” in document; // true

document.ontouchstart = function () {
  “ontouchend” in document; // true !!! WHAAAT ???!
};
```

You Stoned, We Have A Problem !

- how to understand if the browser exposes touches ?

- how to understand if the browser exposes multiple touches ?

Common In Today Web Apps

```
“ontouchend” in document; // partially reliable

“Touch” in (this || window || global); // same
```

The aim of features detection is to be present and as future proof as possible.

The past, present, and future of touch events is unpredictable!

Common In Today Web Apps

```
“ontouchend” in document; // partially reliable

“Touch” in (this || window || global); // same
```

The aim of features detection is to be present and as future proof as possible.

<u>The past, present, and future of touch events is unpredictable!</u>

Common In Today Web Apps

Some device may be driven via both fingers or external mouse/trackpad.

The user is King and our purpose is to make the King happy:

- the King uses the mouse, everything should work
- the King touches the screen, everything should work

As summary: everything should simply work !

Lazy features detection is not widely adopted yet.

Think about it and make the King, or the Queen, happy!

Lazy Features Detection

```
// a truly basic proposal for lazy/async detection
var features = (function (cache/*:object*/, undefined) {
  // reused in more than one place
  function has(key/*:string*/)/*:bool*/ {
    return cache.hasOwnProperty(key);
  }
  return {
    has: has,
    set: function (key/*:string*/, value/*:any*/)/*:void*/ {
      // one time only to avoid multiple notifications mess
      if (has(key)) return;
      // notify all listeners
      var event = document.createEvent("Event");
      event.initEvent("featuredetect", false, true);
      event.key = key;
      event.value = value;
      cache[key] = value;
      window.dispatchEvent(event);
    },
    get: function (key/*:string*/)/*:any*/ {
        // be sure if no key no Object.prototype is involved
        // e.g. cache["toString"] should still be undefined
        return has(key) ? cache[key] : undefined;
    }
  };
}({}));
```

Lazy Features Detection

Mouse or Touch Detection Example

```
for (var
  detect = function detect(e) {
    features.set("touch", !/mouse/i.test(e.type));
    for (i = events.length; i--;) {
        documentElement.removeEventListener(
          events[i], detect, true
        );
    }
  },
  documentElement = document.documentElement,
  events = [
    "mousemove", "mousedown", "mousewheel",
    "DOMMouseScroll", "touchstart"
  ],
  i = events.length; i--;
) {
  documentElement.addEventListener(events[i], detect, true);
}
```

Lazy Features Detection

Mouse or Touch Detection Example

```
window.addEventListener("featuredetect", function (e) {
  // this could be a long list of things to detect
  // let's use the handy switch ...
  switch (e.key) {
    case "touch":
      if (e.value) {
        // ... touch events
      } else {
        // ... mouse events
        document.addEventListener("mousedown", function () {
          document.documentElement.innerHTML = Math.random();
        }, false);
      }
      break;
    // anything else that may be notified
  }
}, true);
```

The Undetectable (at least inline)

```
“TouchList” in this;
// most likely not what we are looking for ...
```

TouchList may be present without indicating anything concrete.

Every touch event comes inside a TouchList even if only one is supported/exposed.

Accordingly, lazy/async feature detection is the best way to understand if the browser can exposes multiple touches.

The Undetectable (at least inline)

Only when and if the user puts more than a finger:

```
document.documentElement.addEventListener(
  "touchstart",
  function touchstart(e) {
    if (1 < e.touches.length) {
      features.set("multitouch", true);
      document.documentElement.removeListener(
        e.type, touchstart, true
      );
    }
  },
  true
);
```

The Partial HTML5 Canvas Case

Symbian browser exposes Canvas since long time ...

```
myCanvas.getContext(“2d”).drawImage(
  myPreviouslyDownloadedImage
);
```

The Partial HTML5 Canvas Case

... wait for it ...

The Partial HTML5 Canvas Case

The Partial HTML5 Canvas Case

All common features detection techniques fail.

```
“HTMLCanvasElement” in window; // fail

try {
  document.createElement(“canvas”).getContext(“2d”);
  // I can show my purrrrfect drawing skills !

} catch (e) {
  // OMG how do I fallback now ?

  // o_O no FlashPlayer in most mobile browsers
}
```

There Is Still A Room For UA Sniffing

The Webkit fragmentation is massive per each brand, Operating System, or specific device.

We cannot possibly test all features we need on the application startup would cost too much time, battery, and user patience.

Many old and modern applications relies into UserAgent sniffing, either on the client or the server side.

In some very specific case, UA sniffing is still the fastest, easier, best solution.

P.S. if still wondering about Symbian browser, don't worry:
we provide better native Map experience there!

About Storing Persistent Data

localStorage

localStorage is synchronous (kinda slow too) and limited in both size and features.

On average, localStorage has about 2.5Mb of size limit, but:

- there is no way to know the current size
- there is no way to increase the storage limit
- there are possibilities that things go wrong

localStorage

```
// better than localStorage[key] = value; *
// also shim friendly
localStorage.setItem(key, value);
```

However, if we reach the limit we have an Exception, so ...

```
try {
  localStorage.setItem(key, value);
} catch (e) {
  // maybe we should clean some stored content here
  // keys have not TIMESTAMP or usage history tho,
  // so it's not that trivial
  // also every script in this page may use or not
  // the same localStorage
}
```

* why direct key get/set is not a good idea

Appcache

Appcache is a must have in order to provide offline usage.

Appcache is also strongly coupled with the online server and it is not possible yet to add runtime new content.

Appcache has an average limit of 50 Mb per tab/application. This limit can be hacked but hacks are both ugly and not reliable for future versions of browsers that allow today this hack.

A bug can be fixed and if this hack is considered a bug it will be fixed and our app gonna have troubles.

It is also difficult for the user to understand which app is using such big amount of data: use Appcache with caution!

Web SQL Database

Ideal combination of performances and simplicity thanks to the widely known, powerful, and in-core optimized SQL syntax (in this case, a subset of SQL 92 provided by SQLite)

About 50 Mb of space in iOS, "unlimited" in Android. Incremental requests to enable more space when/if necessary so the King knows how much data is needed.

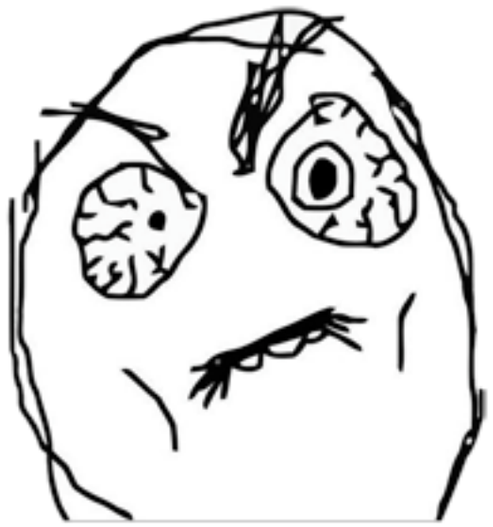
Easy to surf via debuggers, easy to store multiple columns including TIMESTAMP per each row.

Web SQL Database

too fast, too easy ... and **ABANDONED** by W3C !

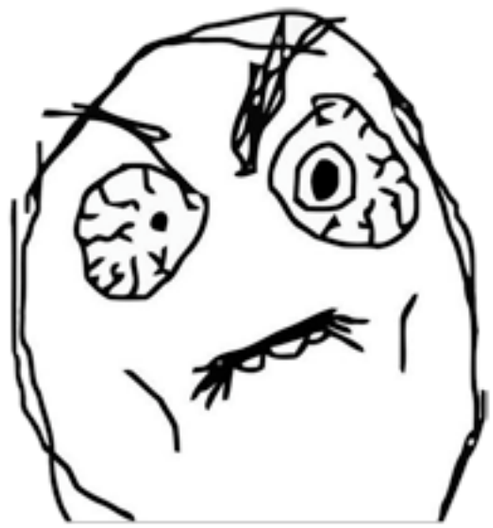
Web SQL Database

too fast, too easy ... and **ABANDONED** by W3C !



Web SQL Database

too fast, too easy ... and **ABANDONED** by W3C !



This almost perfect solution has been dropped by Mozilla browsers while Microsoft has never implemented it.

At least Mozilla lets developers use SQLite via XUL: db.js

IndexDB

A NoSQL DB solution, most likely based on SQLite behind the scene (as many NoSQL solutions are based over a DB ...)

IndexDB aim is to put an abstract layer on top of a database but many things that are simple via SQLite become slower and more complicated via IndexDB.

IndexDB is still not available in Webkit Nightly but is available in Mozilla browsers and Internet Explorer 10

Internet Explorer Mobile 9 has no Web SQL, neither IndexDB, while Google gears project is death and there is no easy fallback.

Which HTML5 Is 100% Cross Platform

Which HTML5 Is 100% Cross Platform

NONE OF IT

Which HTML5 Is 100% Cross Platform

NONE OF IT

However, there are few well known facts we can remember

Touch Events Capable

Unfortunately touch events are not everywhere yet.

Who we assume is supporting them properly:

- Opera Mobile 11 (touch and multi touches)
- Webkit (mainly touch only and not all of them)
- Safari Mobile (touch, multi touch, and gestures)
- Blackberry 7 (and some 6)

Who is not (yet) supporting them:

- IE9 Mobile (partially hackable gestures)
- webOS, at least all Pre (dunno about tablet)

HW Accelerated Canvas

Unfortunately HW Acceleration is not everywhere yet.

Which one we assume is supporting it properly:

- Some Android phone, tablet, or browser
- iOS5 with newer or older phones/pods/pads
- Blackberry 7 (they claim 40% performances boost)
- IE9 Mobile

Hell yeah, the good news is that IE9 has quite fast canvas inside.
The bad news is that without proper touch and multi touches support the canvas is not suitable, as example, for games.

What Did We Learn

In Mobile HTML5 Team we learned that going for HTML5 was much harder than expected: great ideal world not mature yet!

When you develop following partially implemented drafts over fragmented browsers there may be many hacks, considerations, limits, and inevitable compromises you gonna face for sure.

We all believe that HTML5 is “the way to go” or, at least, the best way to go cross platform.

There are still many missing parts or things truly hard to implement efficiently compared with native applications but these are also the reason we are constantly excited about our present and future HTML5 Application features: give as a try!

http://m.maps.nokia.com/

My Choice

HTML5

My Choice

HTML5

Thank You !

Question ?

Credits

NOKIA Mobile HTML5 Team !

Rage faces borrowed in ragefac.es

iPad on keyboard and mouse via redmondpie.com

Peter Paul Koch for his constant effort !

Thank You !

Question ?

NEXT ONE ? ... just kidding :{D

Credits

NOKIA Mobile HTML5 Team !

Rage faces borrowed in ragefac.es

iPad on keyboard and mouse via redmondpie.com

Peter Paul Koch for his constant effort !

Thank You !

if you think you learned something: SpeakerRate

if you think it wasn't worth it: SpeakerRate