Wearables TechCon

WebKit

Development Platform for Wearable Devices

Overview

Why Web technologies and WebKit for Wearables?

Gear S2 – smartwatch powered by WebKit

Raspberry Pi as prototyping platform

Background

C++ developer, embedded systems and mobile OS development

Working on WebKit based IoT and webVR systems, VOD

Can we learn from mobiles?

How can we develop apps?

Can we reuse knowledge and tools?

Can we learn anything from mobile industry?

Web – Mobiles – Next Gen

We have made a long journey with mobile internet and web.

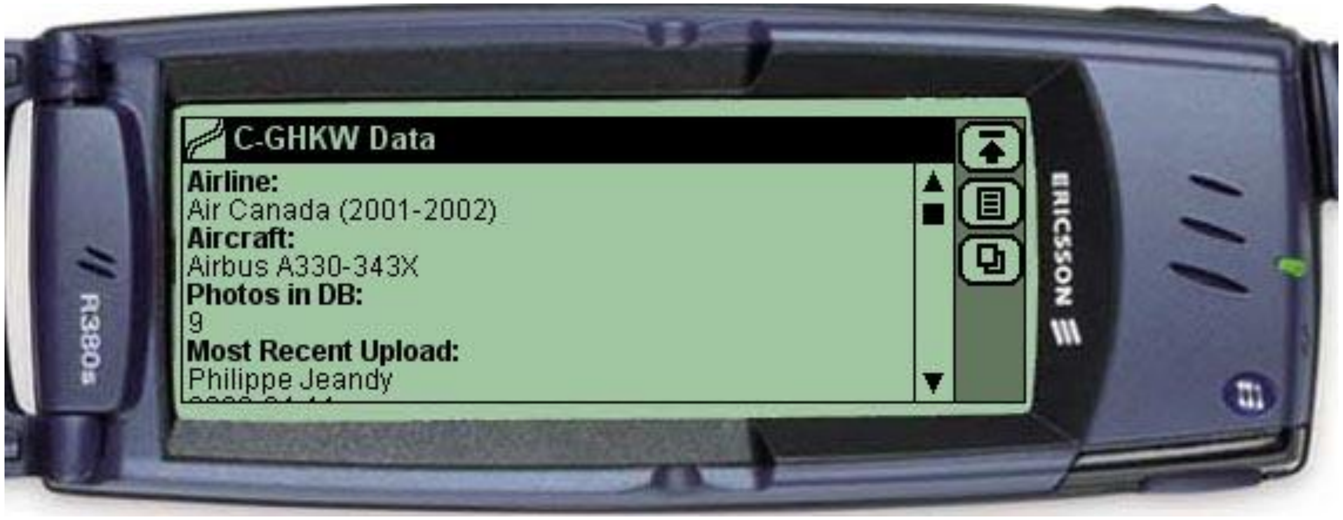
It took many steps to get where we are today – working mobile web

Smartphones

Smartphones are running version of desktop browsers.

Will we repeat the same mistakes with next generation devices?

This was going to be future of mobile web



C-GHKW Data

Airline:
Air Canada (2001-2002)
Aircraft:
Airbus A330-343X
Photos in DB:
9
Most Recent Upload:
Philippe Jeandy

History – Wireless Application Protocol

- Driven by mobile operators driven
- Mobile equivalent of HTML + HTTP
- Huge investment by entire industry for technology nobody really used

Mobile Browsers

Initially driven (and limited) by hardware and software (mobile phones).

This usually caused more issues than in solved.

Versions of desktop browsers are available on mobiles now.

NextGen Web Devices

Internet of Things is emerging technology – smart sensors, wearable tech, connected house, connected car, etc.

Reasonably established in some categories like wearable tech and smart watches.

Range of connected devices

- Many trivial devices – smart sensors
- More complex devices with dedicated OS – Google **Project Brillo**
- Devices capable of supporting web technologies – perhaps smallest group of IoT devices, but still significant (commercial or prototypes)

Wearables and Web

Why web technologies?

Existing development tools and skilled developers

Samsung Gear S2

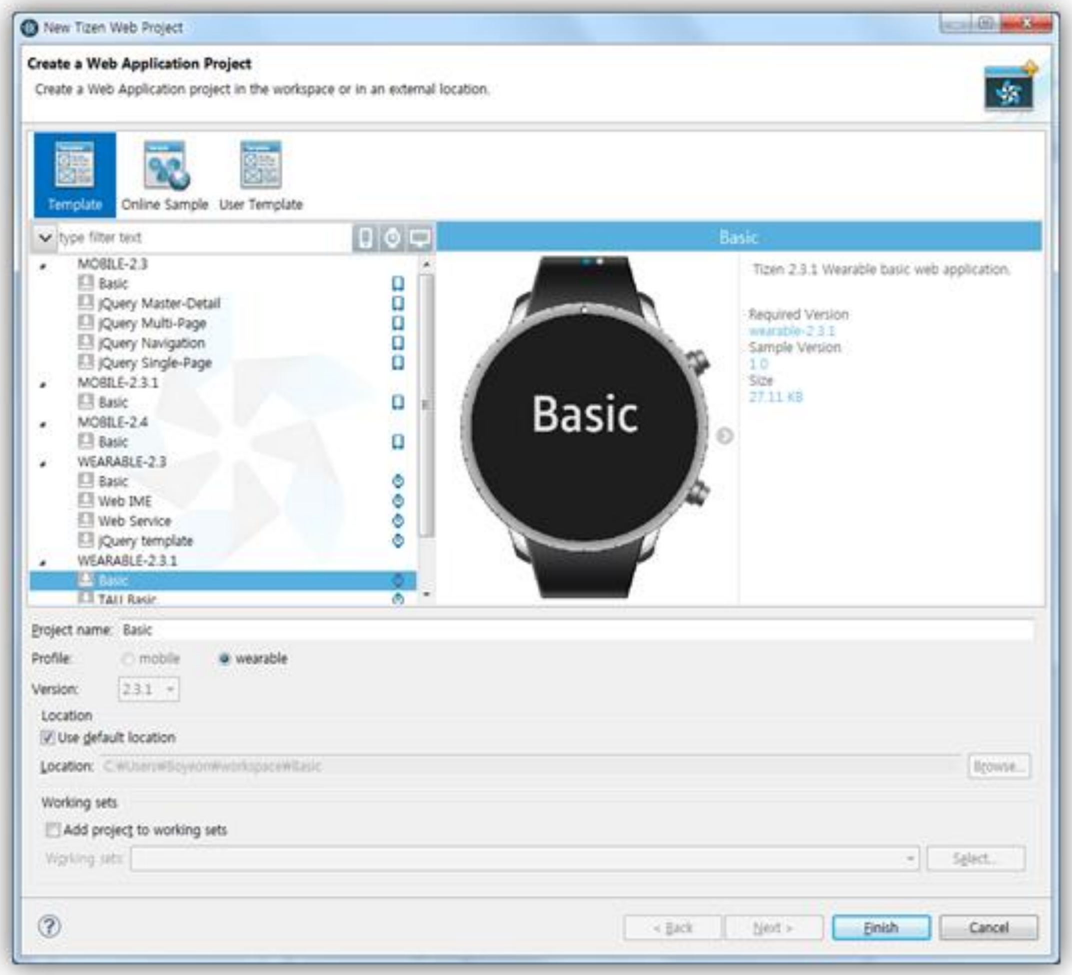
Runs WebKit

Gear applications can be built using a Native or a Web approach

Building a Tizen app using C or HTML5

Samsung developers Getting Started

http://developer.samsung.com/gear/develop/getting-started


New Tizen Web Project

Create a Web Application Project

Create a Web Application project in the workspace or in an external location.

Template | Online Sample | User Template

type filter text

- MOBILE-2.3
  - Basic
  - jQuery Master-Detail
  - jQuery Multi-Page
  - jQuery Navigation
  - jQuery Single-Page
- MOBILE-2.3.1
  - Basic
- MOBILE-2.4
  - Basic
- WEARABLE-2.3
  - Basic
  - Web IME
  - Web Service
  - jQuery template
- WEARABLE-2.3.1
  - Basic
  - TAU Basic

Basic

Tizen 2.3.1 Wearable basic web application.

Required Version
wearable-2.3.1
Sample Version
1.0
Size
27.11 KB

Project name: Basic

Profile: mobile [✓ wearable]

Version: 2.3.1

Location

[✓] Use default location

Location: C:₩Users₩Soyeon₩workspace₩Basic

Working sets

Add project to working sets

Working sets:

Samsung developers Create Project

http://developer.samsung.com/gear/develop/getting-started/web/create-project

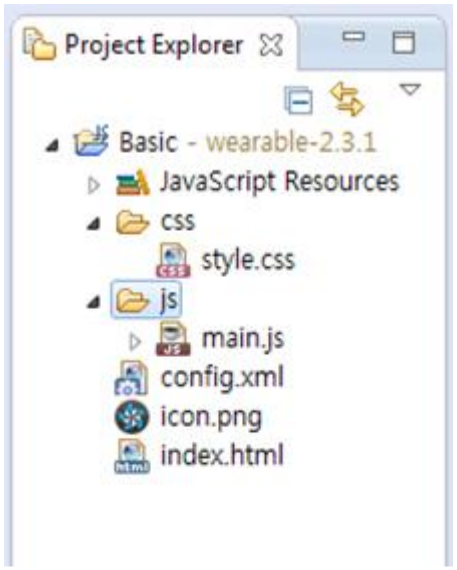
Project structure

css folder: contains .css files for styling the contents of the application.

js folder: contains .js files for handling the functionalities of the application.

config.xml file: contains configuration information for the platform to install and launch the application.

index.html file: contains the layout of the application screen.



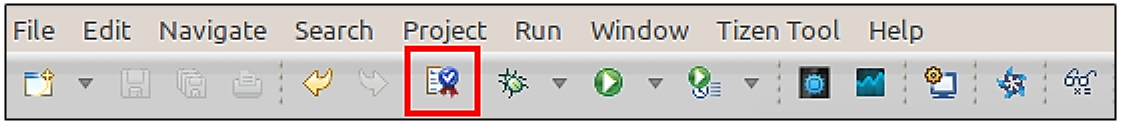
Samsung developers Create Project

http://developer.samsung.com/gear/develop/getting-started/web/create-project

Running Applications in Commercial Devices

Run the Update Manager to install Certificate Extension package

Request certificates by clicking icon on the IDE toolbar



Register the author and distributor certificates

Permit to install application on the device

All details in URL below

Issuing a Tizen Certificate

https://developer.tizen.org/community/tip-tech/issuing-tizen-certificate-certificate-extension-ver-1.2?langredirect=1

Hello Wearable index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic
Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div class="contents">
            <span id="content-text">Basic</span>
        </div>
    </div>
</body>
</html>
```

Hello Wearable main.js

```
window.onload = function() {
    // TODO:: Do your initialization job

    // add eventListener for tizenhwkey
    document.addEventListener('tizenhwkey', function(e) {
        if (e.keyName === "back") {
            try {
                tizen.application.getCurrentApplication().exit();
            } catch (ignore) {}
        }
    });

    // Sample code
    var mainPage = document.querySelector('#main');

    mainPage.addEventListener("click", function() {
        var contentText = document.querySelector('#content-text');

        contentText.innerHTML = (contentText.innerHTML === "Basic") ?
"Wearable" : "Basic";
    });
};
```

Hello Wearable

Clock App index.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable Web Basic
Template" />

    <title>Tizen Wearable Web Basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script src="js/main.js"></script>
</head>

<body>
    <div id="main" class="page">
        <div class="contents">
        <p style="font-size:60px">
            <span id="txt"></span>
        </div>
    </div>
</body>
</html>
```

Clock App main.js

```
window.onload = function startTime() {
var today = new Date();
    var h = today.getHours();
    var m = today.getMinutes();
    var s = today.getSeconds();
    m = checkTime(m);
    s = checkTime(s);
    document.getElementById('txt').innerHTML =
    h + ":" + m + ":" + s;
    var t = setTimeout(startTime, 500);
};

function checkTime(i) {
    if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
    return i;
}
```

Clock App

Running Gear S2 examples

Raspberry Pi

Epiphany for the Raspberry Pi

Epiphany - fast, modern browser for the Raspberry Pi

WebKit based

Much-improved HTML5

JavaScript JIT

Hardware-accelerated video decoding

Epiphany for the Raspberry Pi

If Raspbian and NOOBS doesn't include Epiphany as the default browser, you can install:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install epiphany-browser
```

Epiphany for the Raspberry Pi

Epiphany is a full browser

Good enough for testing, but in reality only WebKit part is needed for simple devices to run Web Apps.

Clock App index.html

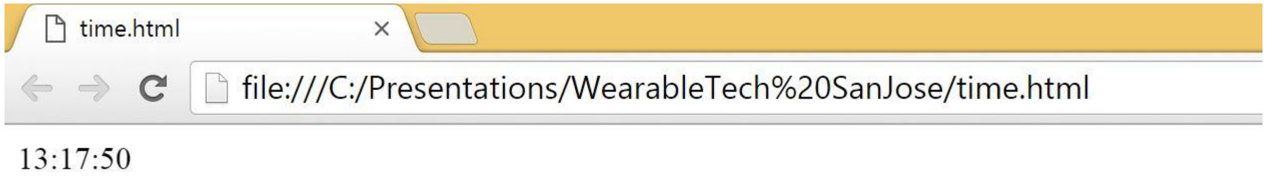
```
<!DOCTYPE html>
<html>
<head>
<script>
function startTime() {
    var today = new Date();
    var h = today.getHours();
    var m = today.getMinutes();
    var s = today.getSeconds();
    m = checkTime(m);
    s = checkTime(s);
    document.getElementById('txt').innerHTML =
    h + ":" + m + ":" + s;
    var t = setTimeout(startTime, 500);
}
function checkTime(i) {
    if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
    return i;
}
</script>
</head>

<body onload="startTime()">

<div id="txt"></div>

</body>
</html>
```

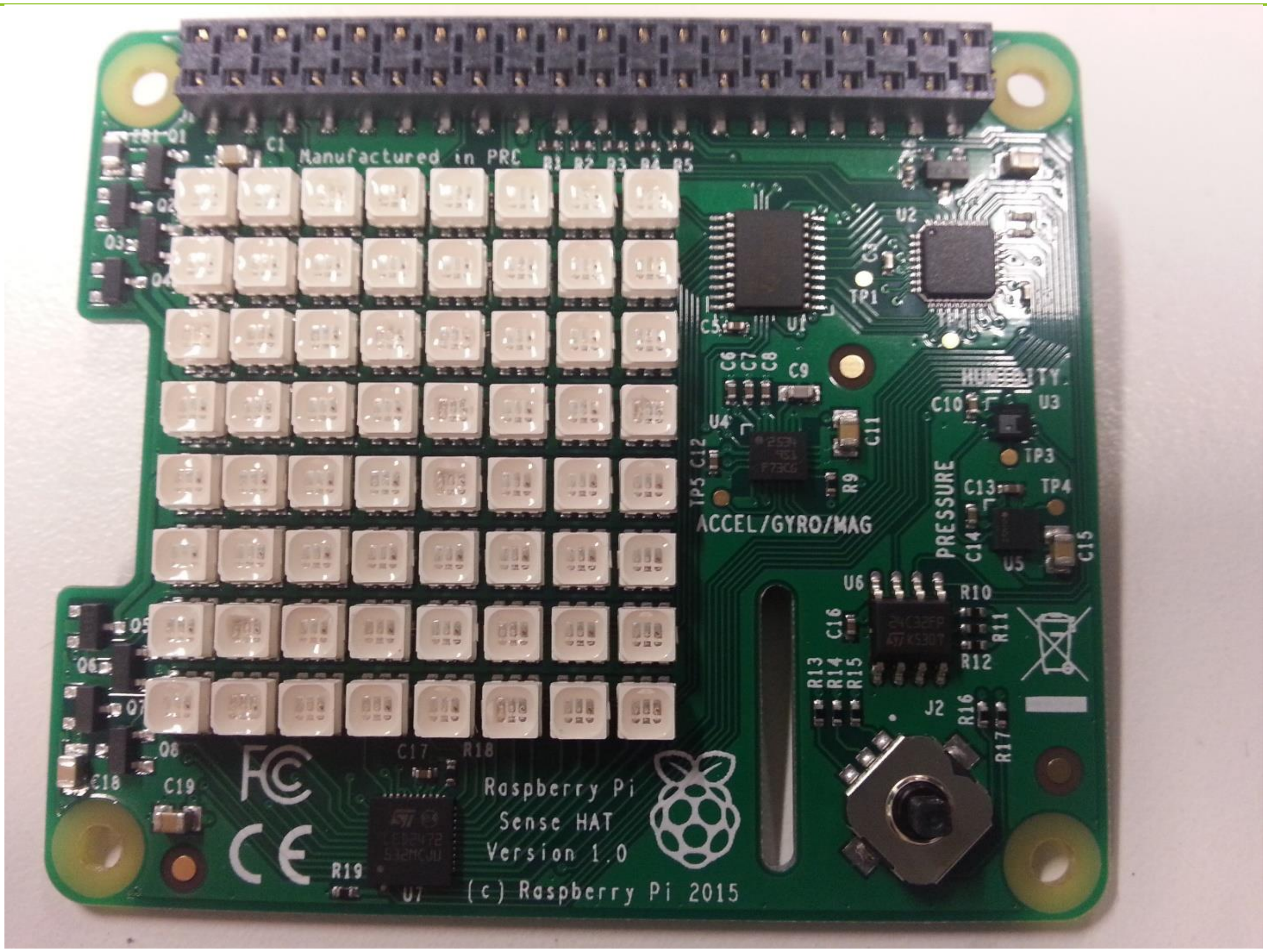
Clock App in WebKit browser



file:///C:/Presentations/WearableTech%20SanJose/time.html

```
13:17:50
```

Raspberry Pi accessories



Raspberry Pi Sense HAT

Made especially for the Astro Pi mission – ISS last year

The Sense HAT has an 8×8 RGB LED matrix, a five-button joystick and includes a few sensors

Raspberry Pi Sense HAT

Sensors:

- Gyroscope
- Accelerometer
- Magnetometer
- Temperature
- Barometric pressure
- Humidity

Python library

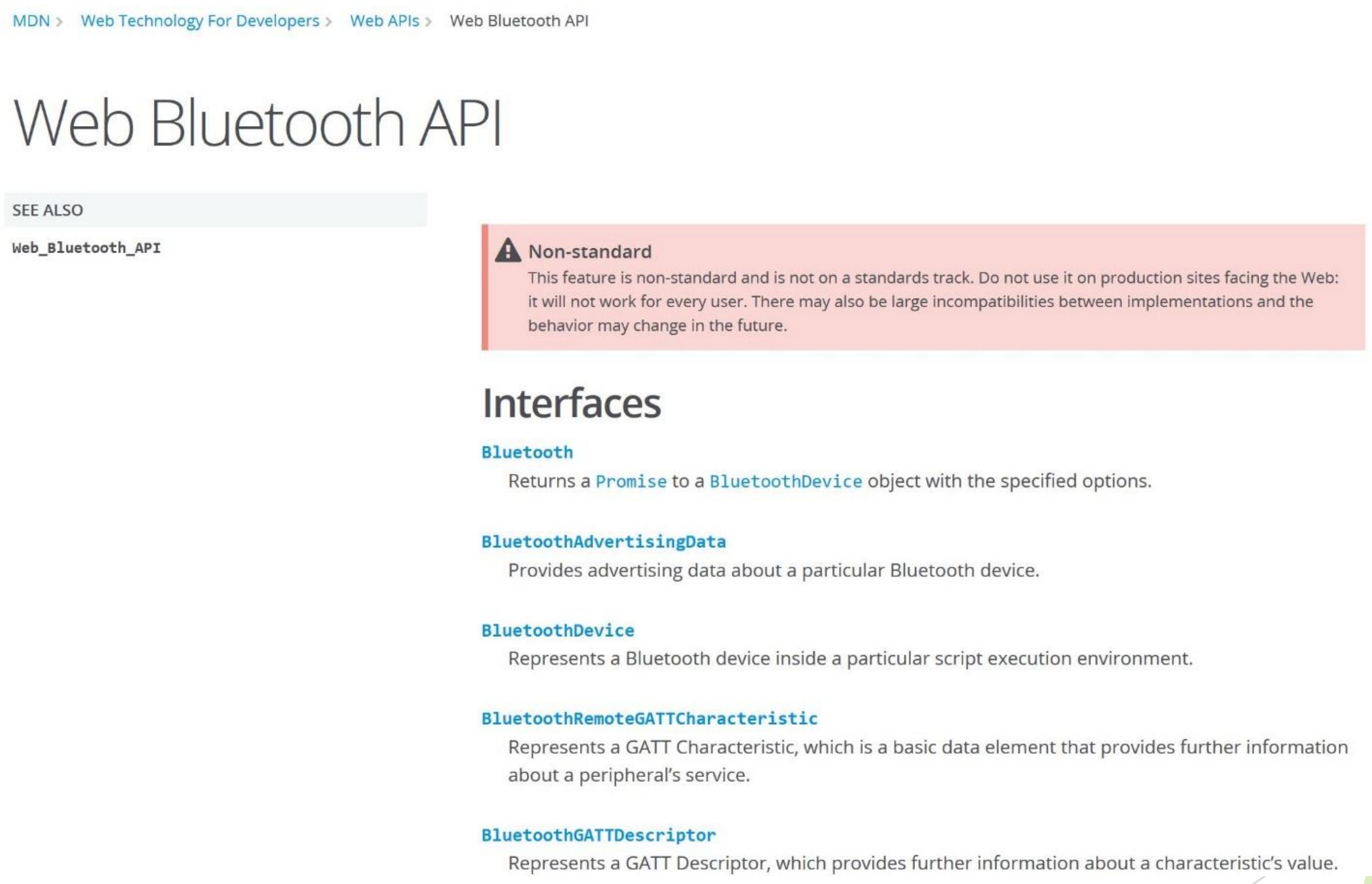
http://pythonhosted.org/sense-hat/

Accessing native functionality

Implementing new JavaScript extensions to enable and expose native functionality

Using JavaScript prototype extension methods

Web Bluetooth API



MDN > Web Technology For Developers > Web APIs > Web Bluetooth API

Web Bluetooth API

SEE ALSO

`Web_Bluetooth_API`

Non-standard
This feature is non-standard and is not on a standards track. Do not use it on production sites facing the Web: it will not work for every user. There may also be large incompatibilities between implementations and the behavior may change in the future.

Interfaces

`Bluetooth`
Returns a `Promise` to a `BluetoothDevice` object with the specified options.

`BluetoothAdvertisingData`
Provides advertising data about a particular Bluetooth device.

`BluetoothDevice`
Represents a Bluetooth device inside a particular script execution environment.

`BluetoothRemoteGATTCharacteristic`
Represents a GATT Characteristic, which is a basic data element that provides further information about a peripheral's service.

`BluetoothGATTDescriptor`
Represents a GATT Descriptor, which provides further information about a characteristic's value.

Using WebKit in Wearables

As with any new technology, there are issues to be sorted out:

- Ease of development
- Security
- Privacy
- Less powerful hardware

Reusing proved technology

Issues with using WebKit as development environment for Wearables are balanced out by advantages:

- Existing development tools
- Developers
- Well understood Security implementation

Security

Reusing WebKit technologies at level provided on web today

https:// - based on TLS and SSL using OpenSSL libraries

Going forward big potential for blockchain

Privacy

Privacy in WebKit – for embedded devices, we can disable or configure some features

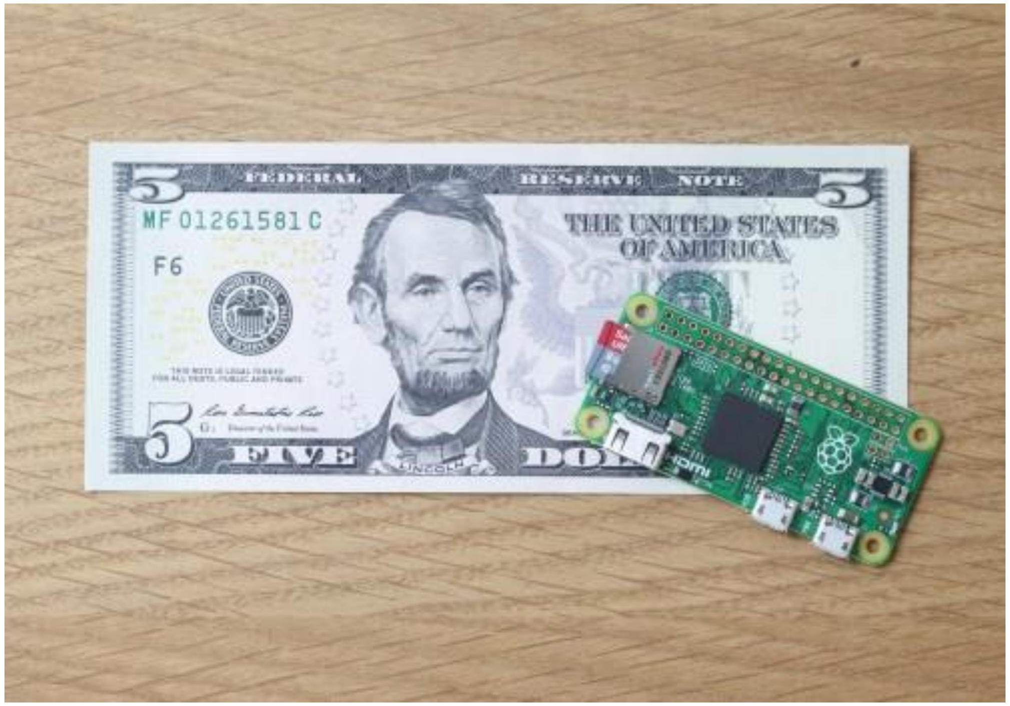
- Cookies
- Web storage
- CORS

Managing memory

If you run with memory budget of 200MB you are likely to hit OOM

At that level not much to do apart from redesigning original Web app

In comparison even entry level Raspberry Pi Zero comes with 512MB RAM



The Raspberry Pi Zero is half the size of a Model A+

https://www.raspberrypi.org/blog/raspberry-pi-zero/

Advanced memory management

Optimise the Web App

Improve WebKit OOM handling and Garbage Collection algorithm.

Specialised WebKit port may be needed to achieve more suitable memory management on low memory devices

Generic optimisation

```
for(var i=0; i<10; i++){
   var obj = {key:'val'};
   console.log(obj);
}
```

OOM stress test

Initialise variables

```
var numberOfImages = 0;
var memoryDiv = document.getElementById('memory');


//get references to the 6 static DOM elements up
front.
var imageDivs = [];
for (var i = 0; i < 6; i++) {
    imageDivs.push(document.getElementById('image' +
i));
}
```

WebKit OOM & GC

```
function interval() {
    var currentImageDiv =
        imageDivs[numberOfImages % 6];
    currentImageDiv.innerHTML =
        '<img src="testimage.png?cachebuster='
        + numberOfImages + '" />';

    numberOfImages++;
}
```

Issues around GC in WebKit

In low mem cases clearing the MemoryCache doesn't releases all CachedResource to system.

https://bugs.webkit.org/show_bug.cgi?id=111094

Garbage Collect to release the references of CachedResource

Responsibility of ‘browser’

Summary of memory handling

Existing WebKit ports proven for desktop and mobile.

Managing memory budget involves cache, font storage, Garbage Collection

Full Memory Pressure Handler may be needed in WebKit for low mem devices.

What next?

Web developers and designers need to consider memory optimisation if they wish to target wearables

Optimised memory management in WebKit is needed.

Specific embedded WebKit port for wearable and IoT devices would encourage wider use.

Conclusion

Using WebKit for wearables is feasible

Performance as such is not generally an issue

Graphics Hardware acceleration is not an issue

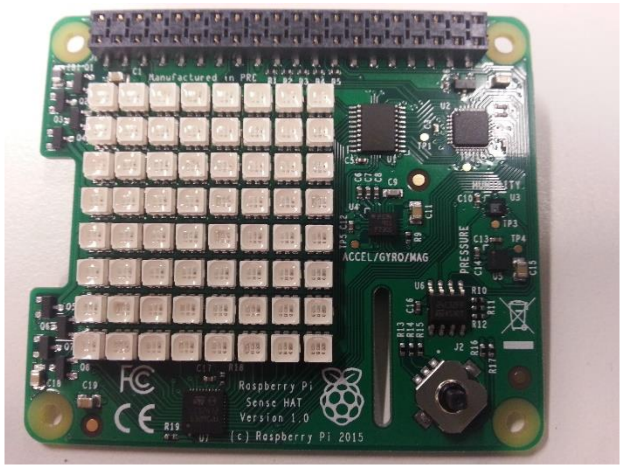
Memory usage is the issue

Alternatives

Node.js and pure JavaScript development for IoT and Wearables

Questions?

00:04
Brighton
15°C
21° / 13°


ACCEL/GYRO/MAG
PRESSURE
HUMIDITY
Raspberry Pi
Sense HAT
Version 1.0
(c) Raspberry Pi 2015