Agenda

- Introduction
- Platform
- TAU framework
- Tools
- Q & A
Platform overview
Platform overview – application types

- Linked (Master – Follower)
- Integrated
- Standalone
Platform overview

• Only web
  • WebKit based engine
  • HTML 5
  • CSS
  • JavaScript (ECMAScript 5)

• Keeping W3C standard
  When it’s possible ;)
Platform overview - physical

• **Data sources**
  • 2.0 Megapixel camera *(n/a in Neo version)*
  • Accelerometer
  • Gyroscope
  • Heart Rate Sensor

• **Data output**
  • 320x320px Super AMOLED display
  • IR transmitter
  • Speaker and vibration
Platform overview - software

Access privileges

Platform
Package management, All other

Partner
Application Management, All of public

Public
Alarm, Part of Application Management, Content, Filesystem, IrLed, Motion, Part of Package management, Power, SAPService, SystemInfo, Media, Storage, Camera API

- Device API
- W3C/HTML5
- Supplementary API
Platform overview – example

Camera use – <video> as preview

```html
<video id="camera-preview"></video>
```

```js
var video = document.getElementById('camera-preview');

function onSuccess(stream) {
    var streamUrl = window.webkitURL.createObjectURL(stream);
    video.src = streamUrl;
    video.play();
}

function onError() { console.log("Error ..."); }

navigator.webkitGetUserMedia( {
    video: true,
    audio: false
}, onSuccess, onError );
```
Platform overview – example

Camera use – taking photos 1/2

```javascript
function onCameraControlCreated(control) { cameraControl = control; }
function onCameraControlError() { ... }
function registerStream(mediaStream) {
    navigator.tizCamera.createCameraControl(
        mediaStream,
        onCameraControlCreated,
        onCameraControlError);
}
function onSuccess(stream) {
    var streamUrl = window.webkitURL.createObjectURL(stream);
    video.src = streamUrl;
    video.play();
    registerStream(stream);
}
navigator.webkitGetUserMedia( { 
    video: true,
    audio: false
 }, onSuccess, onError );
```
// Start focusing
if (cameraControl.autoFocus()) {
    setTimeout(afterFocus, AUTOFOCUS_DELAY);
} else {
    setTimeout(afterFocus, AUTOFOCUS_DELAY); // Take the picture anyway
}

function takePicture() {
    cameraControl.image.takePicture(onPictureDone, onPictureError);
}

function afterFocus() {
    var settings = { /* fileName, pictureFormat, pictureSize */ };
    cameraControl.image.applySettings( settings, takePicture, handleError );
}
Platform overview – phone communication

• SAP
  • Uses Bluetooth Low Energy
  • Data services:
    • Alarm Service
    • Calendar Service
    • Context Service
    • File Transfer Service
    • Music Service
    • Notification Service
SDK Tools for wearable

What’s inside of the SDK

• Tizen IDE
• Device emulator
  • Sensor event simulator
  • Shell access
• Samples
• Documentation and guides
Tizen Advanced UI Framework
TAU: the origin

• Why?
  • Application launched slowly
  • Platform profiles running different UI frameworks
    • TAU for wearable
    • Web UI Framework for mobile

• Why TAU?
  • Starting from smallest device and building up to bigger and more powerful
TAU: satisfying needs

- Application launched slowly
  - Two phases of application launch
  - Removing DOM intensive operations (+ general optimization)

- Many platform profiles running different UI frameworks
  - One ultrafast core ;)
  - Widget designed for different platforms sharing same API
  - Modular building and moving deprecated API into compatibility layer
TAU: evolution(1/2)

- **Optimization**
  - Code optimization using Vanilla JS
  - Using full WebKit potential
  - Launch time optimization

- **Structure**
  - Widget construction separated into phases
  - One core
  - Profile specific code as separate modules

- **Backward compatibility**
  - Compatibility layer for supporting deprecated API
TAU: evolution(2/2)

Framework workflow

Run engine

Widget processing

Application workflow
Application building
TAU build: how does it look like (1/7)?

- Load html file
- Build widgets
- Bind events
TAU build: how does it look like(2/7)?

SDK
- Build widgets
- Package

Runtime
- Load html file
- Build widgets
- Bind events
TAU build: how does it look like (3/7)?

```html
<body>
  <div class="ui-page" id="main">
    <header class="ui-header">
      <h2 class="ui-title">Wearable UI</h2>
    </header>
    <div class="ui-content">
      <ul class="ui-listview">
        <li><a href="#1" class="ui-btn">Button 1</a></li>
        <li><a href="#2" class="ui-btn">Button 2</a></li>
        <li><a href="#3" class="ui-btn">Button 3</a></li>
      </ul>
    </div>
  </div>
</body>
```
TAU build: how does it look like (4/7)?

Widgets built:
- PageContainer
- Page
- ListView
- Button x 3
TAU build: how does it look like (5/7)?
TAU build: how does it look like (6/7)?
TAU build: how does it look like(7/7)?

```html
<div class="ui-popup" data-header="Header" data-header="footer">
  </div>
</div>

<div class="ui-popup" data-header="Header" data-header="footer" data-tau-built="popup" data-tau-name="popup" id="ns-2-1401104644454">
  <div class="ui-popup-header">Header</div>
  <div class="ui-popup-content"></div>
  <div class="ui-popup-footer">footer</div>
</div>
```
TAU Guide

• Separate your application logic
  • Do as much as possible during build phase and then run your code through application pre-builder

• Use or extend existing widgets
  • Prepare your code based on available widgets
  • Extend existing widgets to make your application logic simpler

• Think further
  • Using TAU API will allow you to easily move towards bigger screens
Questions?