モバイルアプリケーション開発ツールの採用ポイント

はじめに

タブレットやスマホ上で動作するモバイルアプリケーションの種類には大きく 2 種類あります。1 つはブラウザで表示する Webアプリケーション で、もう 1 つはモバイルにインストールして利用するネイティブアプリケーションです。
また、ネイティブアプリケーションを開発する場合には、2 つの方法があります。1 つは Apple や Google などの OS メーカが提供している開発言語を利用した本来のネイティブアプリケーション開発で、もう 1 つは、単一の開発言語で実装するハイブリッドアプリケーション開発 です。ここではネイティブアプリケーション開発とハイブリッドアプリケーション開発についての選択ポイントを中心に、記載していきます。

ネイティブアプリケーション開発

ネイティブアプリケーション開発では、それぞれのプラットフォームによって開発言語や環境が異なります。例えばAndroid では、Eclipse*を使用して Java言語で開発を行います。また、iPhoneやiPadなどのiOS では Xcode †を使用してObjective-C言語で開発します。（ちなみに、iOSのアプリケーション開発には、Mac端末が必要となりますので、購入及びMac操作を習得するということも必要となります。）

どちらの言語もバリバリ使いこなせる開発者がいれば、特に問題はありませんが、こういった高度なエンジニアは多くありませんし、現状では一般的に単価が高いと言われております。

１つのOSサポートのみでOKであれば、あまり問題にはならないかもしれません。しかしながら、当初は Androidをターゲットに開発したが、将来的に iOSもサポートしたいとの要望がでてきたとき、開発者が Objective-C を知らなかったら、習得する必要があります。さらにiOS、Androidに加えて、Windows Phoneのサポートも必要となると、もう１つプラスとなります。将来的にどのOSが主流になっていくかも不透明の中、こういったリスクは非常に高いと思われます。加えて今後BYOD‡などを検討する可能性がある場合には、複数OSサポートは重要な要素となってきます。
ちなみに、１つの OS サポートだけだとしても、問題はあります。それは複数のバージョンのサポートが必要となるケースです。各プラットフォームの API は廃止、変更が繰り返されているため、すべてをサポートするためにはトリッキーなプログラミングを強いられることもあります。

これらの問題は、リソース（人、物、お金）に余裕があるのでしたら、大きなことではないかも知れませんが、基本的にはそうではないことが多いと思われますので、その解決策としてクロスプラットフォームを実現するハイブリッドアプリケーション開発ツールがあります。

---

* IBM によって開発された統合開発環境（IDE）の１つ。高機能ながらオープンソースであり、Java などのいくつかの言語に対応する。
† アップルの統合開発環境（IDE）。
‡‡ Bring Your Own Device の略。従業員が個人保有のモバイルを業務に使用することを示す。

ハイブリッドアプリケーション開発

上記の問題において、単一言語で Android と iOS の双方に対応するアプリケーションを開発できるのであれば、問題の解決または負担の軽減になるでしょう。

例えば、以下のような処理を実装することを考えてみます。

> 「何かの処理中を表すインジケータを表示する。」

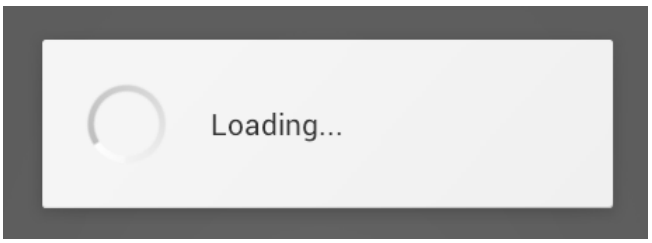
利用ケースとしては、通信中などで処理に時間が掛る場合、通信開始から通信終了までの間、処理を継続していることを表すためにアクティビティインジケータ *を使用します。

Android ネイティブ言語 Java の例

この処理を Android で実装するなら、Java では以下のようなコードになります。

```
ProgressDialog progress = new ProgressDialog(this);
progress.setMessage("Loading...");
progress.setProgressStyle(ProgressDialog.STYLE_SPINNER);
progress.setCancelable(false);
progress.show();
// 時間の掛かる処理
progress.dismiss();
```

実行イメージは以下のようになります。



* 歯車のようなものが「ぐるぐる」と回っているように見えるものです。

iOS ネイティブ言語 Objective-C の例

同様に、iOS なら Objective-C では以下のようなコードになります。

```
UIView *myView = self.view;
CGRect bounds = myView.bounds;
// indicator frame
UIView *frame = [[UIView alloc] init];
frame.bounds = CGRectMake(0, 0, 100, 100);
frame.backgroundColor = [UIColor viewFlipsideBackgroundColor];
frame.layer.cornerRadius = 10;
frame.center = CGPointMake(bounds.size.width / 2, bounds.size.height / 2);
// indicator
UIActivityIndicatorView *indicator = [[UIActivityIndicatorView alloc]
initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhiteLarge];
bounds = frame.bounds;
indicator.center = CGPointMake(bounds.size.width / 2, bounds.size.height / 2);
[frame addSubview:indicator];
[myView addSubview:frame];
[indicator startAnimating];
// 時間の掛かる処理
// …
```

実行イメージは以下のようになります。

（余談ですが、Android で "Loading..." のようなメッセージを表示しない場合、このまま使用することはできず、カスタムダイアログ等を作成して対応しなければなりません。コードを、setMessage("") のように変更しても、文字が消えるだけで右側に空白が残ってしまいます。）

このように、それぞれのプラットフォームで同じような機能を実現するには、(当たり前ですが)双方の環境でコツコツ調べて実装しなければなりません。もちろん、世の中には優れた多くのオープンソースライブラリ等がありますので、それらを見つけて利用することでも解決できます。

しかしこれが、初めからクロスプラットフォームに対応するものであれば、上記のようなことを意識しないで、通常、一つの書き方で Android でも iOS でも同じような機能を実現できます。

それでは、クロスプラットフォームに対応するツールを使用すると、どのように書けるかを、ハイブリッドアプリケーション開発ツールである「PhoneGap」と「Caede」を例に見てみます。

PhoneGap/Apache Cordova の例

ハイブリッドアプリケーションを作成するための有名なツール PhoneGap(または Apache Cordova) を使えば、JavaScript で以下のように書けます。(PhoneGap については http://phonegap.com/を参照)
※ここではバージョン 2.7.0 を使用。

```
navigator.notification.activityStart('', 'Loading...');
// 時間の掛かる処理
navigator.notification.activityStop();
```

実行イメージは、以下のようになります。(Android 上で確認。iOS では未確認。)

Caede （カエデ）の例

PhoneGap と同じく、ハイブリッドアプリケーションを作成する Caede を使用すると以下のようになります。なお、Caede では Curl 言語を使用します。(Caede および Curl 言語については http://www.curl.com 参照)

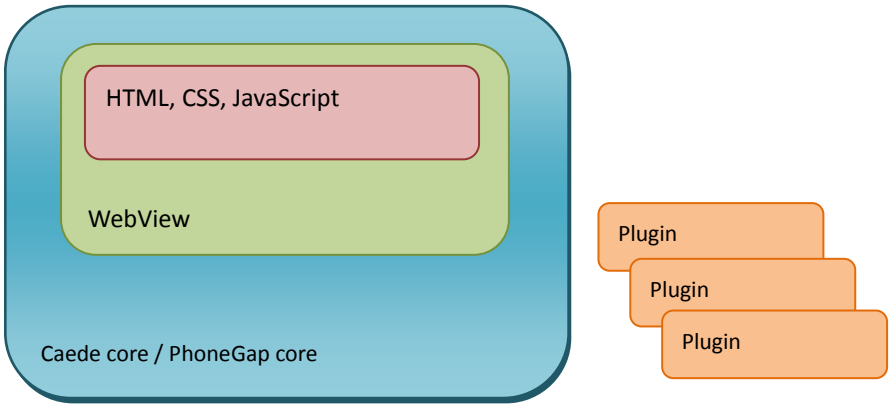
```
{with-busy-indicator
    message = "Loading..."
 do
    || 時間の掛かる処理
}
```

Caede では、正常に Android と iOS 両方で同じようにインジケータが表示されます。

クロスプラットフォーム開発環境では、同じロジックのソースコードを使って、複数のプラットフォームに対応できるのが強みで、プラットフォームごとに開発する場合よりも必要なスキルが少なくなり、実装が容易になります。こういったことから、近い将来、ほとんどのモバイルアプリケーションが、ハイブリッドアプリケーションとなると一般的には言われています。

ハイブリッドアプリケーションのコア WebView / UIWebView

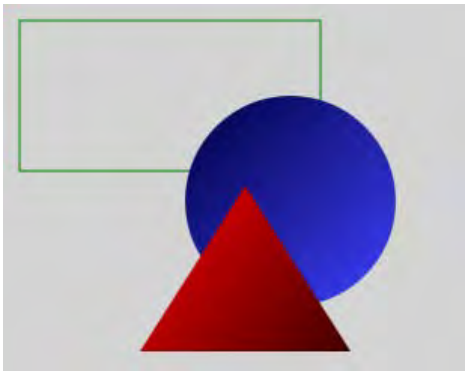
PhoneGap でも Caede でも、内部アーキテクチャとしては、インジケータ部分がネイティブ言語で実装されています。同様に HTML5 などのみでは実現できないようなデバイス固有の機能などは、ネイティブ言語での実装になります。また、作成されるアプリケーションはネイティブアプリケーションとしての殻を持ち、内側に WebView / UIWebView* を入れて、JavaScript や HTML、CSS の解釈と表示に使用しています。このように Web アプリケーションとネイティブアプリケーションを組み合わせたものを ハイブリッドアプリケーションと呼んでいます。（ちなみに、これは内部的な仕組みですので、もちろん利用者はネイティブ言語を覚える必要はありません。）



JavaScript + HTML5 の問題

Web アプリケーションを作成したことがあれば、HTML や CSS、JavaScript に慣れ親しんでいると思います。そのような開発者であれば、PhoneGap のような JavaScript、 HTML、CSS を利用することで、クロスプラットフォームアプリケーションを作成することが可能です。さらに、jQuery Mobile や Sencha Touch のようなフレームワークを組みわせることで、スマホやタブレットに適した ユーザインタフェースを簡単に作成できます。

また、HTML5 では Canvas や SVG という API を利用して、以下のような図形やグラフの描画も実現できるようになりました。



```
// Rectangle
context.strokeStyle = 'rgb(0, 128, 0)';
context.strokeRect(20, 30, 200, 100);
// Circle
var cx = 200, cy = 150, r = 70;
var grad = context.createLinearGradient(cx - r, cy - r, cx + r, cy + r);
grad.addColorStop(0, 'rgb(0, 0, 64)');
grad.addColorStop(1, 'rgb(64, 64, 255)');
context.fillStyle = grad; // gradation
context.beginPath();
context.arc(cx, cy, r, 0, Math.PI * 2, false);
context.fill();
// Triangular shape
var x1 = 100, y1 = 140, x2 = 240, y2 = 250;
grad = context.createLinearGradient(x1, y1, x2, y2);
grad.addColorStop(0, 'rgb(255, 0, 0)');
grad.addColorStop(1, 'rgb(64, 0, 0)');
context.fillStyle = grad; // gradation
context.beginPath();
context.moveTo(x1 + 70, y1);
context.lineTo(x1, y2);
context.lineTo(x2, y2);
context.closePath();
context.fill();
```

これらは、HTML5 をサポートしているブラウザおよび WebView / UIWebView 上で動作しますので、このようなことまで、ハイブリッドアプリケーション開発では実現できます。

では、JavaScript と HTML5 に各種フレームワークさえあれば、何の問題もなくクロスプラットフォーム開発ができるでしょうか。

例えば、上記の言語やフレームワークの扱いには、独特のクセがあったりします。また、HTML5 をサポートしているといっても、プラットフォーム及びバージョンによっては実装状況が違っていることがあります。
同様に、一概にJavaScript といっても標準である ECMAScript* のどのバージョンを実装のベースとしているかによっても違いますし、同じベースでも実装エンジンによっては「方言」と呼ばれるような、微妙な違いがあります。

ECMAScript 実装状況

アプリケーション（JavaScript engine）	呼称	基本 ECMAScript リビジョン
Mozilla Firefox *(SpiderMonkey)*	JavaScript	ECMA-262 5th Edition
Microsoft Internet Explorer 8	JScript	ECMA-262 3rd Edition
Microsoft Internet Explorer 9 ~ *(Chakra)*	JScript	ECMA-262 5th Edition
Safari 4 *(Nitro)*	JavaScript	ECMA-262 5.1th Edition
Google Chrome *(V8 JavaScript Engine)*	JavaScript	ECMA-262 5th Edition
Adobe Flash	ActionScript	ECMA-262 4th (draft)

- ✓ ECMAScript 3rd Edition: 1999/12 公開。正規表現、try/catch 構文の追加など。
- ✓ ECMAScript 4th Edition: 破棄。
- ✓ ECMAScript 5th Edition: "strict mode"の追加。getter, setter の追加、JSON ライブラリサポートなど。
- ✓ ECMAScript 5.1th Edition: ISO/IEC 16262:2011 3rd Edition に完全準拠。

* http://www.ecma-international.org/publications/standards/Ecma-262.htm

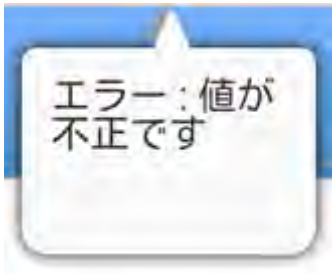
モバイルブラウザでの HTML5, CSS3 などの対応差異

機能	Android (標準ブラウザ)					iOS (Safari)					IE (参考)	
	~2.3	3.0	4.0	4.1	4.2	3.2	4.0,4.1	4.2,4.3	5.0,5.1	6.0,6.1	9	10
CSS position:fixed (絶対位置+固定)	△					×	×	×	△	△		
getComputedStyle	△	△				△	△	△				
JSON パース						×						
型付き配列	×	×				×	×				×	
ECMAScript5 “strict mode”	×					×	×	×			×	
SVG (基本機能)	×											
SVG filter	×	×	×	×	×	×	×	×	×		×	

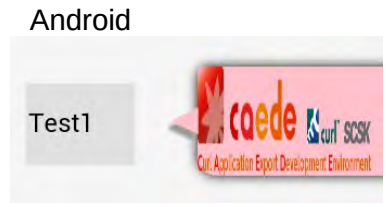
※凡例：×… 未サポート、△…部分サポート、なし…サポート

Windows Phone 8 の IE は Desktop 版 IE10 と同じエンジンを使用。

例えば CSS ですと、以下のようなバルーンと呼ばれる吹き出しのようなものを表示させたい場合、iOS と Android で同じように見せるために、CSS の Style 属性をそれぞれ個別に指定します。



これを例えば、Android に合うように CSS を設定すると iOS では以下のよう△の噴き出しの部分が離れてしまう問題が発生します。



このように、Android と iOS の WebView / UIWebView では CSS の適用方法が異なることがあります。このような仕様の違いは１つの言語で実装可能と言え動作が異なってくるため、プラットフォームごとにテスト、調査、実装を行う必要があります。これは非常に負荷の高い作業となり、工数が膨れ上がってきます。

クロスプラットフォーム開発製品「Caede / Curl」のご紹介

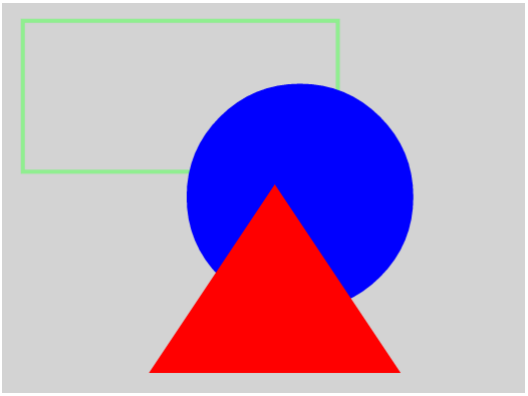
各言語の特性や様々なフレームワーク独自の扱い方に惑わされない開発環境が欲しいと思います。それを解決できるのが Caede です。

Caede では、Curl という言語のみで Android と iOS をサポートするアプリケーションを開発できます。これまでに述べたような、プラットフォームごとに必要なスキルを習得する期間も、複数の言語を習得することもなく、単一の言語でモバイルアプリケーションを開発できます。

ネイティブ		ハイブリッド	
Android	iOS	PhoneGap/ Cordova	Curl
Java	Objective-C	HTML5 CSS3 JavaScript SVG ＋ jQueryMobile Sencha Touch …	Curl

Caede は、Curl 言語で書かれたソースコードを 内部的には JavaScript、HTML、CSS、SVG に変換して、WebView / UIWebView 上で、JavaScript を実行したり HTML を表示したりしています。(WebView / UIWebView の利用については、PhoneGap でも同様です。)
この変換時に、プラットフォームごとに異なる部分を吸収しますので、開発者がプラットフォーム間の差異を意識しなくて済むため、余計な負担を強いられることなく、開発に専念できることになります。

前述の HTML5 の例では Canvas を使用して図形を書きましたが、Caede でも Shape クラスを使用して、以下のように図形の描画ができます。



図形の描画部分のコードは、以下のようになります。

```
def canvas = {self.find-graphic-by-name "canvas"} asa Canvas
|| Rectangle
def rectangle = {RectangleShape
                  {GRect 0cm, 2.5cm, 0cm, 1.2cm},
                  color = "transparent",
                  border-color = "lightgreen",
                  border-width = 2px,
                  translation = {Distance2d 0.5cm, 0.5cm}
                }
|| Circle
def ellipse = {EllipseShape
                {GRect 0cm, 1.8cm, 0cm, 1.8cm},
                color = "blue",
                translation = {Distance2d 1.8cm, 1.0cm}
              }
|| Triangular shape
def triangle = {value
                 def p = {Path}
                 {p.move-to {Distance2d 2cm, 0cm}}
                 {p.line-to {Distance2d 3.0cm, 1.5cm}}
                 {p.line-to {Distance2d 1.0cm, 1.5cm}}
                 {RegionShape
                     {Region.from-path p},
                     color = "red",
                     translation = {Distance2d 0.5cm, 1.8cm}
                 }
               }
{canvas.add rectangle}
{canvas.add ellipse}
{canvas.add triangle}
```

また、JavaScript* で記述する他の開発環境とは異なり、Curl ではクラスベースのオブジェクト指向プログラミングを行えるので、Java や C++、C# などと同じように開発できます。
さらに、Curl は静的型付き言語 †であるため、コンパイル時の型チェック等により型安全な機能を実装できます。

さらなる特徴としてオプティマイザという機能があります。Caede は Curl 言語を JavaScript や HTML に変換しますが、このことを翻訳(トランスレート)と呼んでいます。トランスレートされた JavaScript コードは、各プラットフォームの WebView / UIWebView 上で、JavaScript エンジンにより解釈され、実行されます。その Caede の翻訳機(トランスレータ)には、「定数伝播」「定数の畳込」「共通部分式の削除」「条件分岐の確定」「デッドコードの削除」「不要コードの削除」の最適化機能を実装しています。これがオプティマイザです。

---

* JavaScript は、「プロトタイプベース」のオブジェクト指向プログラミング言語。
† ちなみに JavaScript は、動的型付き言語です。

下記の Curl ソースを例に最適化の事例を説明します。
この例は、引数を 3 つ取り、第 2、第 3 引数である配列の内容を書き換える関数(プロシージャ)となっています。

```
{define-proc  {sample index:int,arg1:{Array-of int}, arg2:{Array-of int}}:void
    let i:int = 1
    let j:int = 2

    {if j > 0 then
        set arg1[index + j] = i + j
        set arg2[index + j] = i + j
     else
        set arg1[index + i] = -(i + j)
        set arg2[index + i] = -(i + j)
    }
}
```

この最適化前のソースに対して、定数伝播の最適化をすることにより、下記のようなコードになります。

```
{define-proc  {sample index:int,arg1:{Array-of int}, arg2:{Array-of int}}:void
    let i:int = 1
    let j:int = 2

    {if 2 > 0 then
        set arg1[index + 2] = 1 + 2
        set arg2[index + 2] = 1 + 2
     else
        set arg1[index + 1] = -(1 + 2)
        set arg2[index + 1] = -(1 + 2)
    }
}
```

次に、定数の畳込の最適化により下記のようになります。

```
{define-proc  {sample index:int,arg1:{Array-of int}, arg2:{Array-of int}}:void
    let i:int = 1
    let j:int = 2

    {if 2 > 0 then
        set arg1[index + 2] = 3
        set arg2[index + 2] = 3
     else
        set arg1[index + 1] = -3
        set arg2[index + 1] = -3
    }
}
```

さらに、共通部分式の削除を行うと下記のようになり、

```
{define-proc  {sample index:int,arg1:{Array-of int}, arg2:{Array-of int}}:void
    let i:int = 1
    let j:int = 2

    {if 2 > 0 then
        def opt1 = index + 2
        set arg1[opt1] = 3
        set arg2[opt1] = 3
     else
        def opt2 = index + 1
        set arg1[opt2] = -3
        set arg2[opt2] = -3
    }
}
```

そして、条件分岐の確定とデッドコードの削除の最適化で、下記になります。

```
{define-proc  {sample index:int,arg1:{Array-of int}, arg2:{Array-of int}}:void
    let i:int = 1
    let j:int = 2

    def opt1 = index + 2
    set arg1[opt1] = 3
    set arg2[opt1] = 3
}
```

最後に、不要コードの削除を行うと、下記のようになります。

```
{define-proc  {sample index:int,arg1:{Array-of int}, arg2:{Array-of int}}:void
    def opt1 = index + 2
    set arg1[opt1] = 3
    set arg2[opt1] = 3
}
```

ソースコードが徐々にシンプルになっていくのが分かるかと思います。このような最適化を行うことにより、最適化された Curl コードから生成される JavaScript コードもシンプルになり、JavaScript エンジンの解析速度が速くなるとともに、実行速度も速くなります。

また、よく問題となる点として、デバイスの多様化があげられます。（特に Android。）デバイスの種類が変われば、画面サイズや解像度にも違いが出てきます。そのような差異に対応するような画面デザインをするのは大変な労力です。Caede / Curl にはエラスティックと呼ばれるものがあり、画面レイアウトによってグラフィカルオブジェクトを自動的に整列、伸長、圧縮することができます。これにより、デバイスごとの画面サイズの差異を吸収できます。

最後に

実行速度やレスポンスの良さを求められるようなアプリケーション開発では、ネイティブ言語による開発にまだまだ分があると思います。
一方、そこまでのレスポンスに拘らないのであれば、ハイブリッドアプリケーションの方がクロスプラットフォーム対応や開発期間の短縮など生産性の面については圧倒的な利点があります。

それを踏まえて、モバイルのみならずデスクトップアプリケーションとしても開発できる Caede / Curl を利用してのクロスプラットフォーム開発は、いかがでしょうか。