

Développer des applications avec Appcelerator Titanium

Xavier Lacot - Septembre 2014 - Web@cadémie

< joliCode />



Xavier Lacot

- Expert Web et Mobile chez **<JoliCode/>** - <http://jolicode.com>
- 贡献者 à plusieurs projets Open Source
- Développeur Titanium depuis 2009
 - Conférencier à CodeStrong et TiConf
- Expert frameworks Web - notamment Symfony en PHP
 - Contributions Open Source
 - Plusieurs conférences - Symfony Live, Forum PHP, Symfony Day
- Ex-Président de l'Association Française des Utilisateurs de PHP (afup.org)
- <http://twitter.com/xavierlacot>





- **Introduction au développement mobile**
 - Solutions de développement
 - Architecture de Titanium
- **Initialisation et configuration d'un projet Titanium**
- **Éléments de l'interface utilisateur : fenêtres, vues, widgets, etc.**
 - Layout, positionnement et style des éléments graphiques
- **Détecter et utiliser le réseau, construire un client HTTP**
- **Accéder aux APIs matérielles :**
 - Accéléromètre / système de fichiers / caméra / GPS / Carnet d'adresses
- **Présentation d'Alloy, le framework MVC officiel pour Titanium**
 - Architecture et principes de Alloy
 - Gestion des vues (templates XML et styles .tss) et contrôleurs (application de comportements)
 - passage de variables entre contrôleurs
 - Création de widgets et réutilisabilité
- **Utiliser les APIs Appcelerator Cloud Services**
 - gestion des utilisateurs et d'objets métier
- **Liens et ressources intéressantes**



- Objectifs :

- Comprendre les concepts et les mécanismes de fonctionnement de Titanium
- Bien faire la différence entre Titanium et WebApps
- Savoir développer des applications mobiles iPhone et Android
- Maîtriser le framework Alloy

- Quelques conseils

- Posez des questions ;-)
- N'hésitez pas à tester





- Un cours unique : aujourd'hui
 - Apprendre les concepts
 - Installer Titanium et l'environnement
 - Faire un tout petit projet en démo
- Un projet individuel
 - contraintes imposées : page suivante
 - 5/09 → 19/09 → 17/10
- Un projet en groupe
 - 17/10 → 14/11



- Méthodologie / livrables attendus

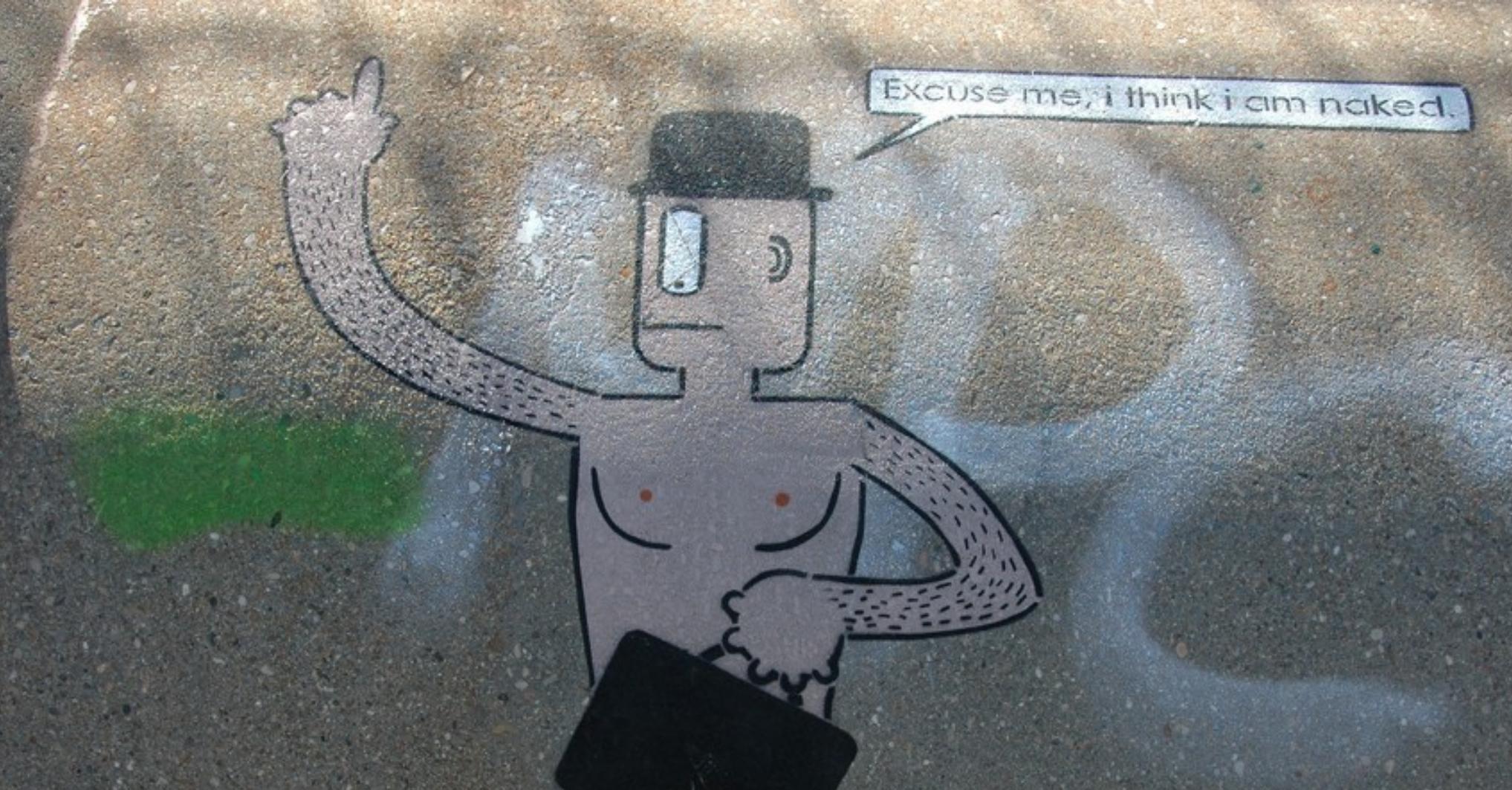
- Mockups de l'application (papier / powerpoint / paint / Balsamiq)
- Texte décrivant le rôle de l'application
- Code détaillé et commenté

- Contraintes techniques imposées

- En Titanium avec le framework Alloy
- Au moins trois contrôleurs
- Des points en plus si vous faites un widget Alloy *utile*
- Code commenté, bien indenté, découpé en petites méthodes (pas un gros pâte illisible)
- Fonctionne sur iOS et Android sans bug majeur
- Utilise au moins une de ces trois features : réseau, cartographie et GPS, ou bases de données

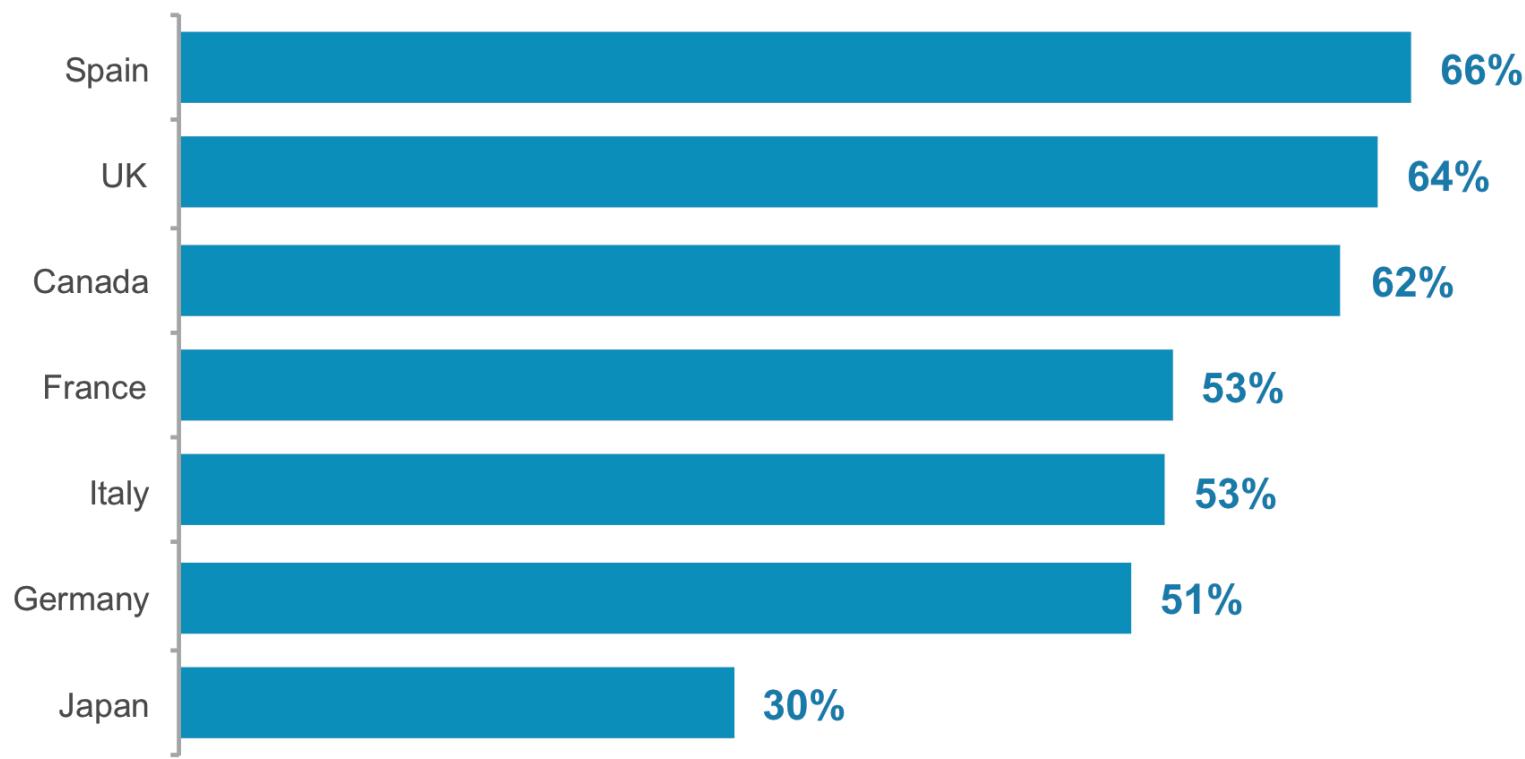


- Mercredi 10/09 au soir : par mail à xlacot@jolicode.com
 - Un texte décrivant votre application
 - Quelques mockups dessinés pour comprendre le fonctionnement
 - Des points accordés à cette étape et au respect du délai
 - Je vous ferai une réponse par mail, vous attaquez les développements
- 19/09 : journée d'atelier, je serai présent toute la journée
- 17/10 : journée de soutenance
 - Entre 5 et 10 slides :
 - Présenter votre projet (Quoi ? Pourquoi ?)
 - Expliquez comment ça marche
 - Parcours du code ensemble - vous devez savoir l'expliquer



Introduction au développement mobile

Marché des smartphones



Part de la population ayant un smartphone (dec. 2012, source : Comscore)



- De nombreuses plateformes techniques différentes :
 - iOS
 - Android
 - Windows Mobile
 - Blackberry
 - Firefox OS
 - Symbian
 - Bada
 - Linux
 - etc.

symbian
OS

RIM
BlackBerry

 Windows phone

meeGo



palm webOS™

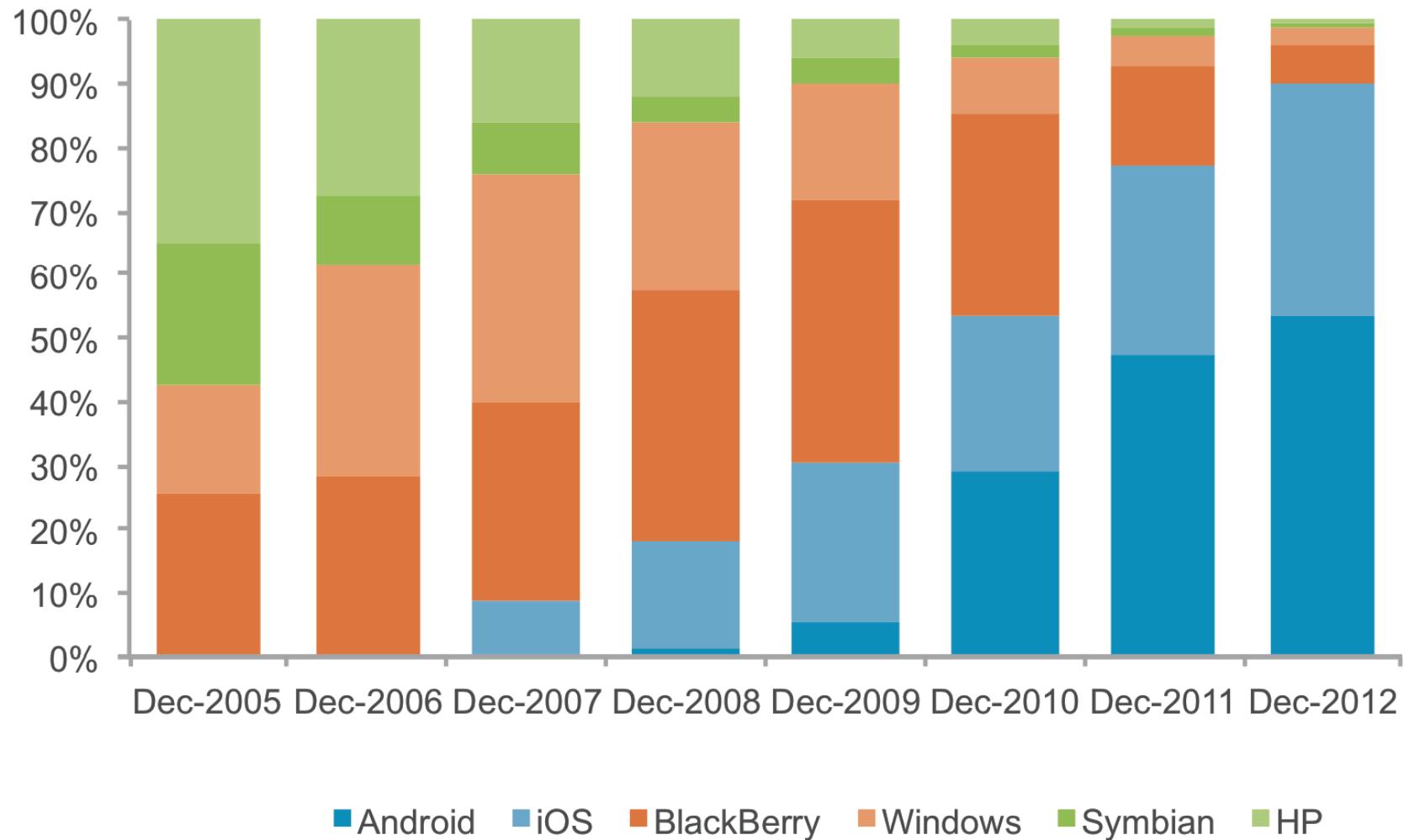


bada





Marché qui évolue





- Difficile de faire un choix de plateforme

- Faut-il privilégier certaines plateformes ?
- Comment anticiper les changements de leadership ?

- Des technologies différentes :

- Java
- Objective-C
- Visual C#
- C++
- Python
- etc.



- Société :
Google
- Open Source :
Oui
- Langage :
Java
- SDK :
Android SDK
- Catalogue en ligne :
Android Play! Store
- Parts de marché (ventes Q4 2013)
65%





- Société :
Apple
- Open Source :
Non
- Langage :
Objective-C / Cocoa
- SDK :
Iphone SDK
- Catalogue en ligne :
Apple AppStore
- Parts de marché (ventes Q4 2013)
environ 20%





Panel des solutions de développement mobile

- Plusieurs solutions :

- Applications natives, avec le SDK natif
 - iOS
 - Android
 - BlackBerry
 - Windows Phone 8
 - Tizen
- WebApp (site HTML adapté au mobile)
 - Toutes plateformes
- Application native sur la base d'un framework multiplateformes
 - Plusieurs plateformes



Panel des solutions de développement mobile

- Plusieurs solutions :

- Applications natives, avec le SDK natif
 - iOS
 - Android
 - BlackBerry
 - Windows Phone 8
 - Tizen
- WebApp (site HTML adapté au mobile)
 - Toutes plateformes
- Application native sur la base d'un framework multiplateformes
 - Plusieurs plateformes

Titanium



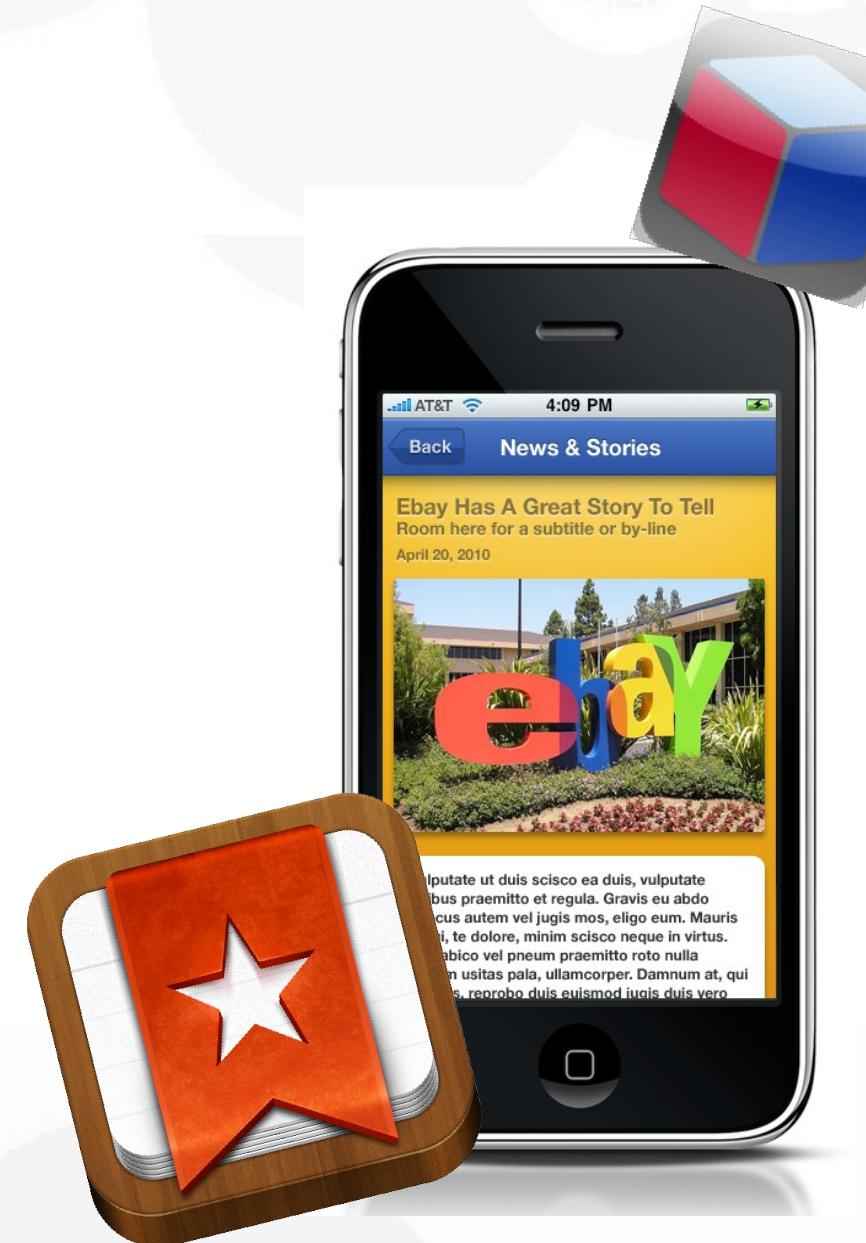
- HTML(5) / CSS / Javascript
- Frameworks de présentation « uniformisants »
 - Jquery Mobile - <http://jquerymobile.com/>
 - Sencha touch - <http://www.sencha.com/products/touch>
 - JQTouch - <http://jqtouch.com/>
- Pratique mais :
 - Lent (interfaces riches en js dans le navigateur)
 - Pas de widgets au look natif
 - Pas de mode offline
 - Pas dans les stores

Présentation de Titanium mobile



Historique de Titanium

- Publié en aout 2009 (version 0.4)
 - 1.0 en août 2010
 - 2.0 début 2012
 - 3.0 début 2013
 - 4.0 courant 2014
- Société sponsor : Appcelerator
- Mouvance x.commerce
- Sponsorisé par eBay
- De nombreuses applications à succès
 - Wunderlist
 - NBC Ipad
 - Ebay corporate
 - Sugar CRM Mobile
 - cf. « Application showcase »





- Open Source

- Oui

- Entreprise

- Appcelerator

- Langage

- javascript

- Plateformes

- iOS, Android, Tizen, (BlackBerry)

- Bilan : Applications natives avec un look natif

- APIs identiques suivant les cibles

- Look natif → convenable pour la plupart des applications !





« Titanium is an open source framework for building native desktop and mobile applications using open web technologies (HTML, CSS, and JavaScript) »

- Développement en javascript
- Accès aux fonctionnalités matérielles par le biais d'APIs javascript
- Widgets natifs : impossible de distinguer d'une app 100% native
- Application compilée : performances identiques aux applications traditionnelles
- Pas besoin d'apprendre Objective-C ou java
- Support d'iOS, d'Android, de Tizen et dans une moindre mesure de BlackBerry



Pourquoi Titanium ?

- Langages Web très populaires
 - Faciles à apprendre
 - Très répandus
- HTML / javascript / CSS
 - Standards ouverts
 - Cross platform : codez une fois, déployez sur plusieurs cibles !
- Applications natives
 - Rapides
 - Mode offline
 - Accès aux fonctionnalités matérielles
 - Actives en tache de fond





Installation de Titanium
et création d'un projet



Titanium Studio

The screenshot shows the Titanium Studio interface with the following components:

- Project Explorer:** Displays the project structure for "joli.api.js-app-demo [master]". It includes folders for bin, Resources (android, config, images, iphone), lib (controller, model, vendor, view), and files like add.js, app.js, display.js, and list.js.
- Code Editor:** Shows the content of the "list.js" file. The code initializes a window, sets its background color and title, loads library files, and creates a search bar.
- Outline View:** Shows a hierarchical list of variables and objects defined in the current file, such as win, backgroundColor, peoples, search, and tableview.
- Console:** An Aptana Scripting Console window is visible at the bottom.
- Status Bar:** Shows status indicators like "Writable" and "Smart" along with the user's name, "Xavier Lacot".



- Titanium Studio

- Aptana, éditeur basé sur Eclipse
- Racheté début 2011
- Forte intégration avec les SDK

- Permet :

- De créer un projet
- Debugger intégré
- Auto-complétion / aide au code
- De lancer les simulateurs / émulateurs iOS et Android



- Inclus au moment de télécharger le framework

- Nodejs

- Un environnement d'exécution javascript côté serveur
- Un « concurrent de PHP » mais en javascript
- Des tonnes de librairies
- Un outil d'installation : « npm »

- Installer Titanium :

- Installer nodejs : <http://nodejs.org/>

- Installer Titanium cli :

```
$ npm install -g titanium
```

- Installer le sdk Titanium :

```
$ titanium sdk install 3.2.0.GA
```



Créer un projet avec Titanium...

- Créez le projet Titanium :

```
$ titanium create
```

- Répondez aux différentes questions :

- Target platforms : plateformes ciblées. Par ex. « android,ios »
- App Id : identifiant d'application, eg. « com.company.appname »
- Project name : le nom de l'application

```
Target platforms (android\blackberry\ios\ipad\iphone\mobileweb\tizen) [android,blackberry,ios,ipad,iphone,mobileweb,tizen]: android,ios
App ID: com.jolicode.test
Project name: webacademie
[INFO] Creating Titanium Mobile application project
[INFO] Copying "android" platform resources
[INFO] Copying "iphone" platform resources
[INFO] Project 'webacademie' created successfully in 117ms
```

- Vous pouvez commencer à employer Titanium...

```
$ titanium build --platform ios
```



Layout d'un projet Titanium

	build
	Resources
	CHANGELOG.txt
	LICENSE
	LICENSE.txt
	manifest
	README
	tiapp.xml

- **build** : dossier de compilation
- **Resources** :
 - le code de l'application
 - tous les assets (images, etc.)
 - éventuellement la base SQLite
 - éventuellement les modules natifs
- **tiapp.xml** : les directives de configuration et de compilation
 - Noms, icône et copyright
 - Full screen ?
 - Analytics ?
 - Différents paramètres



Que se passe-t-il réellement ?

- **Pré-compilation**
 - Analyse des dépendances (API requises, etc.)
 - Pré-validation du code javascript
- **Compilation front-end**
 - Compilation des librairies de Titanium
 - Préparation à la compilation sur la plateforme cible
- **Compilation sur la plateforme cible**
 - Appel des outils de compilation fournis par les SDKs
 - IOS
 - transformation du js sous la forme de variables Objective-C
 - Interprétation à l'exécution par l'interpréteur javascript de l'iPhone
 - Android
 - Précompilation en bytecode et interprétation par l'interpréteur V8 (ou Rhino)



Pas de variables globales

```
// The bad
var key = 'value',
    foo = 'bar',
    human = 'john';

function hello(name) {
    alert('Hello ' + world);
}

function transform(value) {
    return 'Logging ' + value;
}

function log(text) {
    Titanium.API.log(
        transform(text)
    );
}

// call the function
log('Coucou poney!');
```

```
// The good
var application = {
    key: 'value',
    foo: 'bar',
    human: 'john'
};

// add a function to this namespace
application.hello = function(name) {
    alert('Hello ' + world);
};

(function() {
    // locally defined function
    function transform(value) {
        return 'Logging ' + value;
    }

    application.log = function(text) {
        Titanium.API.log(
            transform(text)
        );
    };
})();

// call the function
application.log('Coucou poney!');
```



Opérateur de comparaison

```
// utilisez === et non ==
Ti.API.info(false == 0); // true
Ti.API.info(false == ''); // true
Ti.API.info('' == 0); // true
Ti.API.info(null == 0); // true
Ti.API.info(undefined == false); // true
Ti.API.info(NaN == 0); // true

// attention aux comparaisons de flottants
Ti.API.info(0.3 + 0.4 === 0.7); // false

// attention à typeof
Ti.API.info(typeof null); // 'object'
Ti.API.info(typeof NaN); // 'number'
```

Voir <http://wtfjs.com/> pour plus d'exemples...



```
// écriture standard
if (somecondition === somevalue) {
    var xyz = 'abc';
} else {
    var xyz = '123';
}

// écriture plus compacte - à privilégier quand c'est
possible
var xyz = (somecondition === somevalue) ? 'abc' : '123';
```

- À ne jamais faire :
 - Fichiers de 15000 lignes
 - Absence de point-virgules
 - Noms de variables abscons



CommonJS



- Une initiative de standardisation des inclusions de libs js
 - Mouvement apparu fin 2010
 - <http://commonjs.org/>
- Principe :
 - Les modules exportent une interface publique

```
Module.exports = {
  uppercase: function(word) {
    // return something
  }
}
```

- Introduction d'une fonction « require(module_name) » qui évalue le module, et retourne son API publique

```
// load the module contained in the file my.module.js
var module = require('my.module');
alert(module.uppercase('michel'));
```



Comment versionner et organiser son projet ?



- Possible de travailler avec vos outils habituels :
 - Git
 - Subversion
- Fichiers à ignorer (`.gitignore`) :
 - `.project`
 - `.settings`
 - `build/*`
 - `build.log`
 - `tmp`
 - (si projet Alloy) Resources

Éléments graphiques
Positionner et styler des éléments

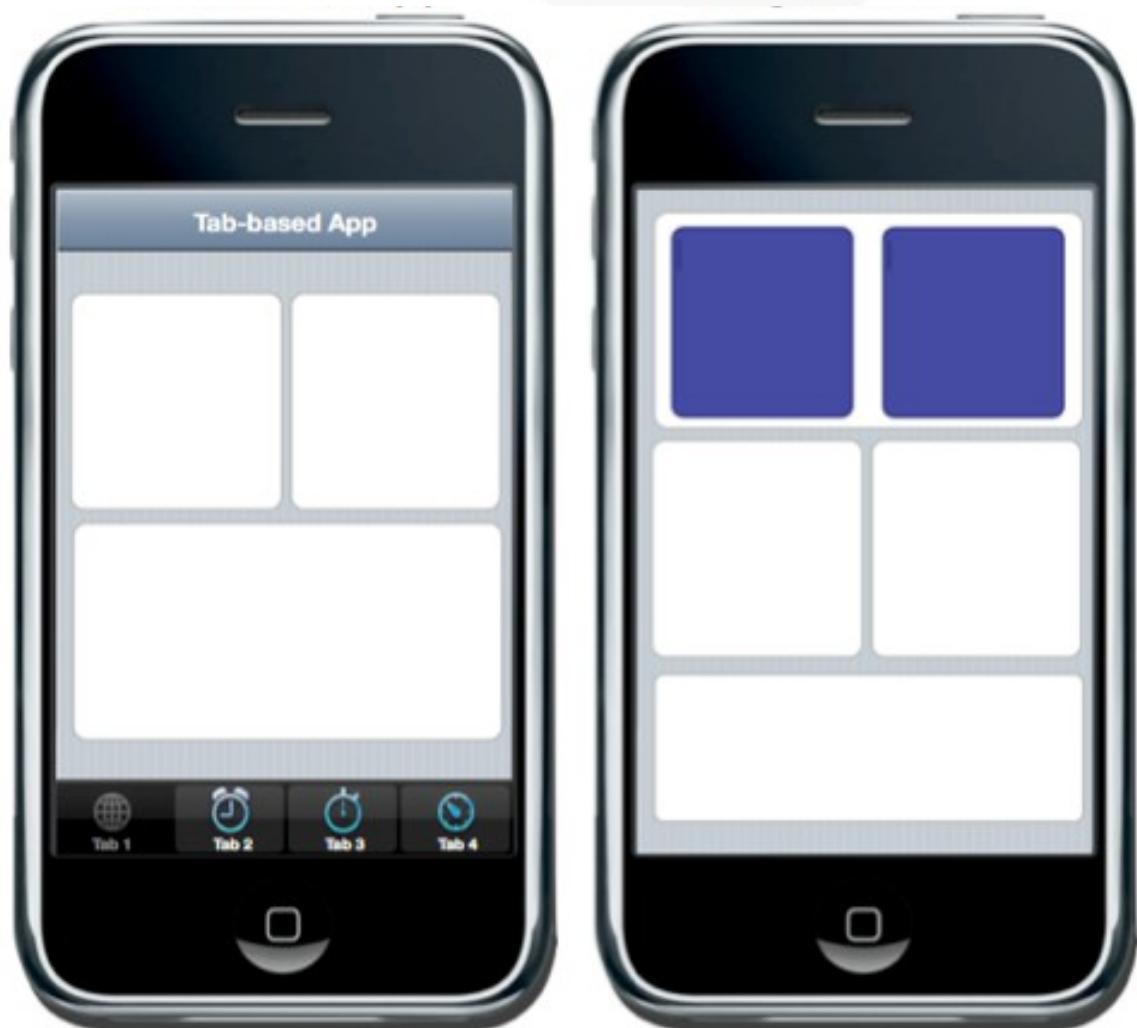




- Deux modes de conception :

- Fenêtres autonomes
- Groupe de « tab »
- (« TabGroup »)

- cf. l'API pour les options





```
// create a window
var win = Titanium.UI.createWindow({
    backgroundColor:'#edeff6',
    title:'Disclaimer',
    barColor:'black'
});

// add a button in it
var button = Titanium.UI.createButton({
    title: 'Close',
    style: Titanium.UI.iPhone.SystemButtonStyle.PLAIN
});
win.setLeftNavButton(b);
button.addEventListener('click', function() {
    win.close();
});

// open the window
win.open({modal:true});
```



```
// create tab group
var tabGroup = Titanium.UI.createTabGroup();
var win1 = Titanium.UI.createWindow({
    title:'Tab 1',
    backgroundColor:'#fff'
});
var tab1 = Titanium.UI.createTab({
    icon:'tab1icon.png',
    title:'Tab 1',
    window:win1
});
var win2 = Titanium.UI.createWindow({
    title:'Tab 2',
    backgroundColor:'#fff'
});
var tab2 = Titanium.UI.createTab({
    icon:'tab2icon.png',
    title:'Tab 2',
    window:win2
});

// add tabs to the group
tabGroup.addTab(tab1);
tabGroup.addTab(tab2);

// open tab group
tabGroup.open();
```



- Au sein d'une fenêtre, la composition graphique de l'application se fait à l'aide de vues :
 - Image view
 - Scroll view
 - Table view
 - Web view
 - Map view
 - Overflow view
 - Dashboard view
- Une vue peut en contenir une autre
- Un peu équivalent aux « div » de HTML

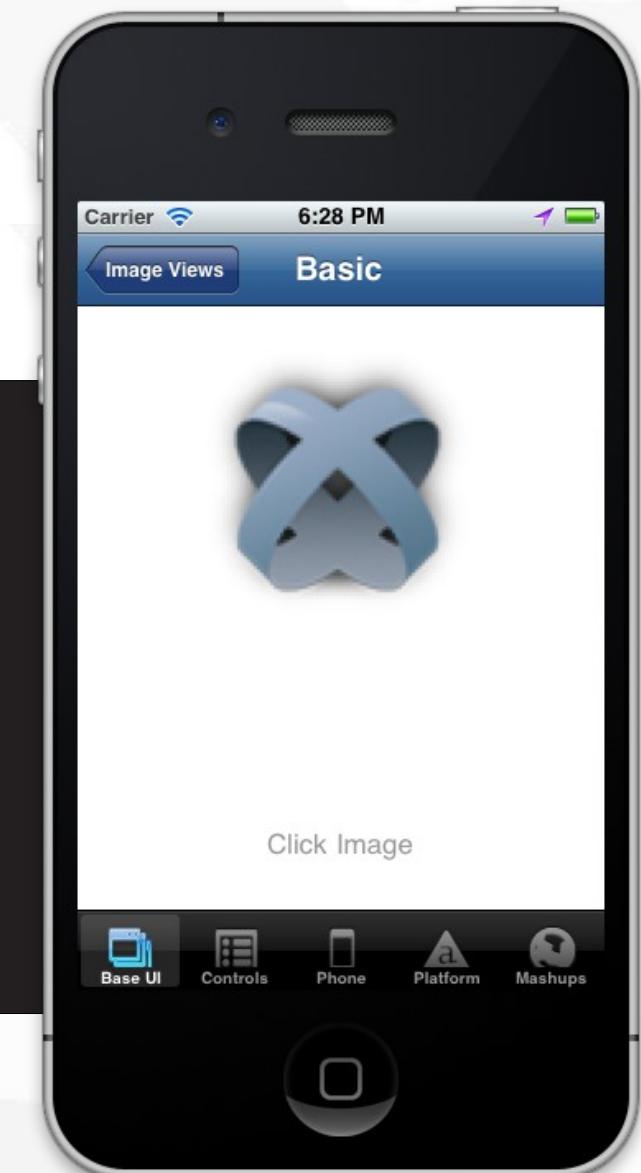


■ Insertion d'images

- Le paramètre « image » peut être une url

```
var win = Titanium.UI.currentWindow;
var imageView = Titanium.UI.createImageView({
    image:'titanium.png',
    width:261,
    height:178,
    top:20
});

imageView.addEventListener('load', function()
{
    Ti.API.info('LOAD CALLED');
});
win.add(imageView);
```





- Adapté aux contenus scrollables de taille variable
 - Affichage d'un pager

```
var view1 = Ti.UI.createView({
    backgroundColor: 'red'
});
var view2 = Ti.UI.createView({
    backgroundColor: 'blue'
});
var view3 = Ti.UI.createView({
    backgroundColor: 'green'
});
var view4 = Ti.UI.createView({
    backgroundColor: 'black'
});

var scrollView = Titanium.UI.createScrollView({
    views: [view1,view2,view3,view4],
    showPagingControl: true,
    pagingControlHeight: 30,
    maxZoomScale: 2.0,
    currentPage: 1
});
win.add(scrollView);
```



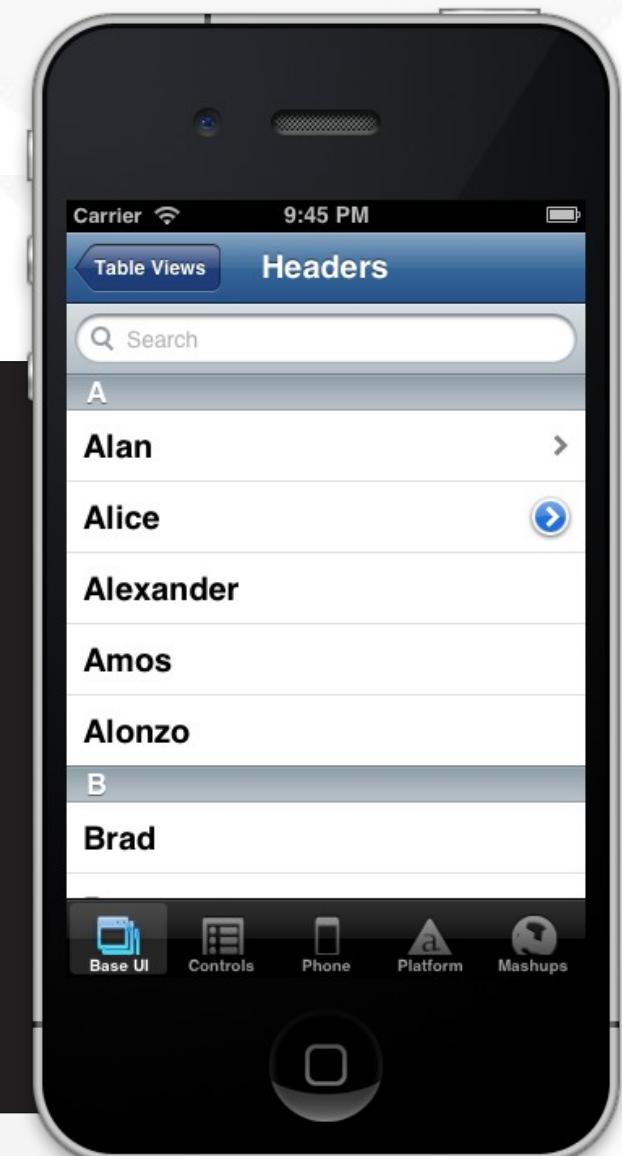


■ Données tabulaires

- Possibilité de headers
- Chaque cellule peut contenir des contrôles

```
var data = [
    {title: 'Alan', hasChild: true, header: 'A'},
    {title: 'Alice', hasDetail: true},
    {title: 'Brad', header: 'B'},
    {title: 'Brenda'},
    {title: 'Callie', header: 'C'},
    {title: 'Chris'},
];
var search = Titanium.UI.createSearchBar({
    showCancel: false
});

// create table view
var tableview = Titanium.UI.createTableView({
    data: data,
    search: search,
    filterAttribute: 'title'
});
win.add(tableview);
```





- Les widgets sont les éléments d'interface avec lesquels l'utilisateur peut interagir :

Activity Indicator

Label

Progress bar

Switch

Button bar

Toolbar

Textfield

Textarea

Button

Search bar

Slider

(Date) picker



- Exemple : un bouton

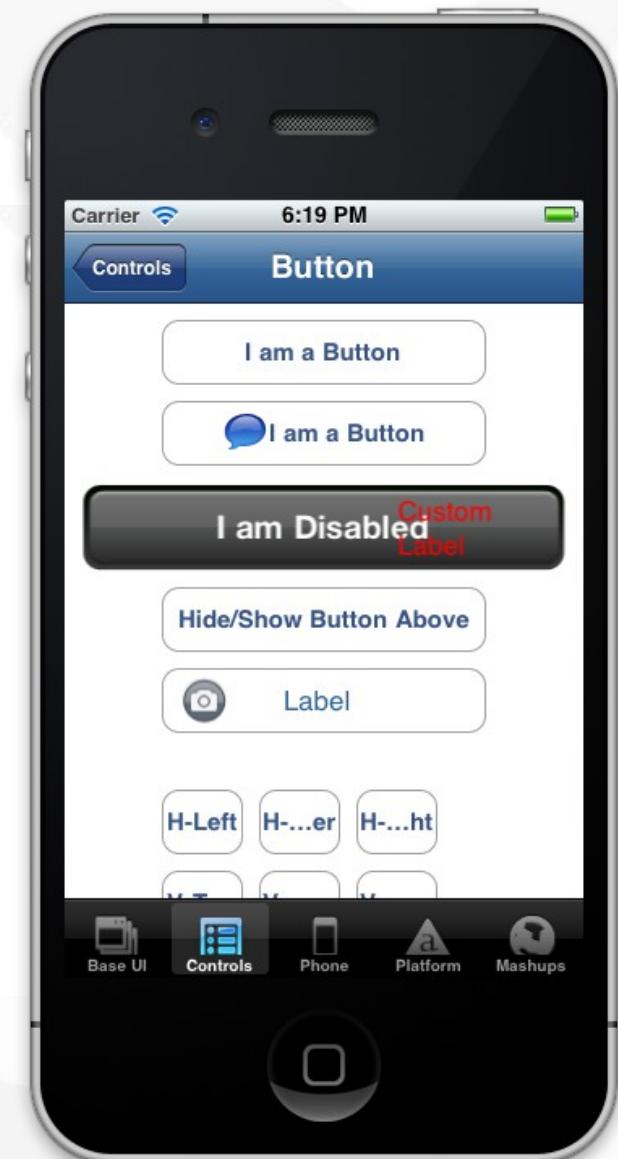
- Constructeur simple ou complexe
 - Traitement par évènements

- Constructeur :

- Ti.UI.createButton()
- API : rien de particulier

- Paramètres :

- enabled, focusable, visible
 - image
 - title
 - etc.





- Les attributs de positionnement :

- bottom / left / right / top
- height / width
- layout
- zIndex

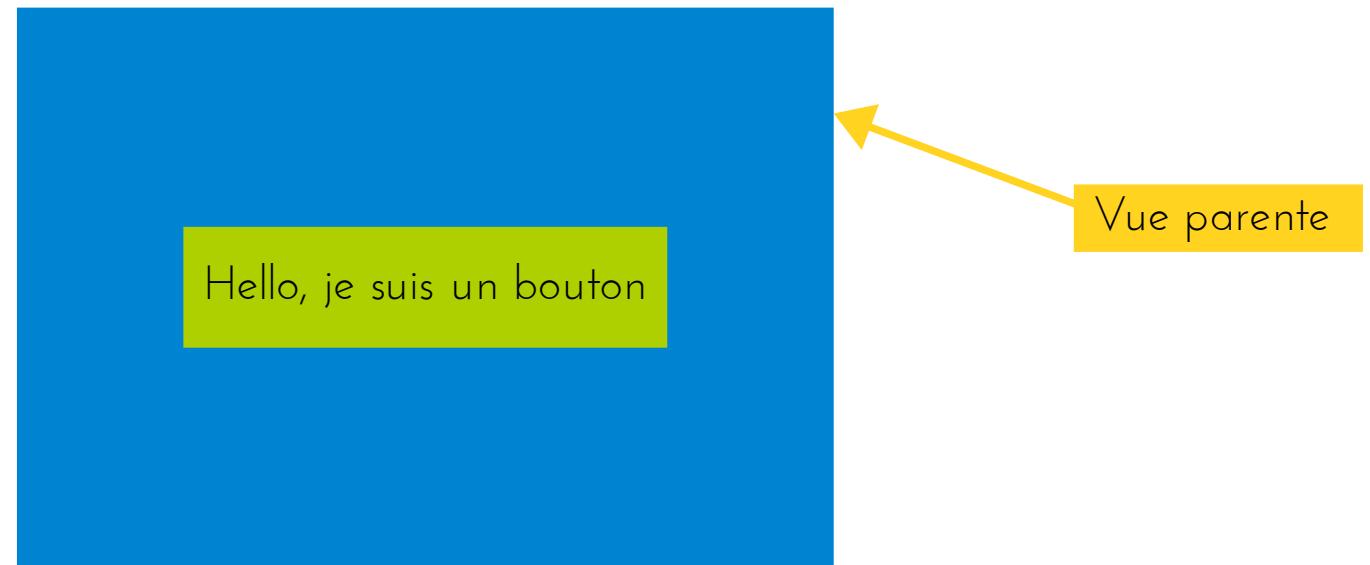
- Des constantes et attributs utiles :

- Ti.UI.FILL / Ti.UI.SIZE
- Ti.UI.View.Rect



Positionner et styles des éléments

- Le positionnement ne se fait pas comme en CSS !
- Par défaut : système de coordonnées absolu depuis le coin haut gauche
- Les enfants d'une vue sont par défaut centrés

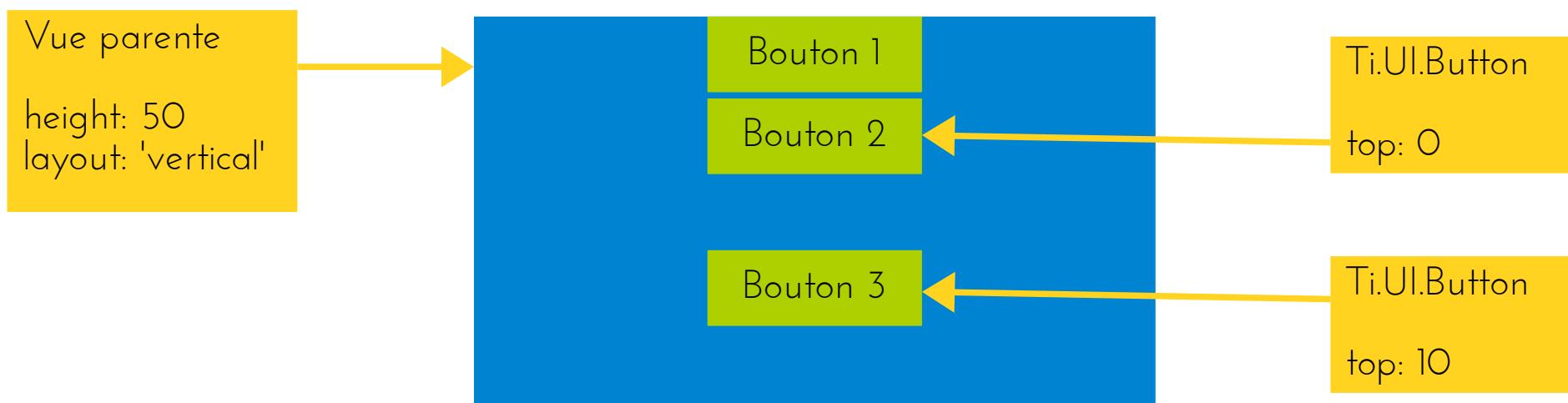


- Les propriétés top / left / bottom / right permettent de positionner les éléments



Positionner et styles des éléments

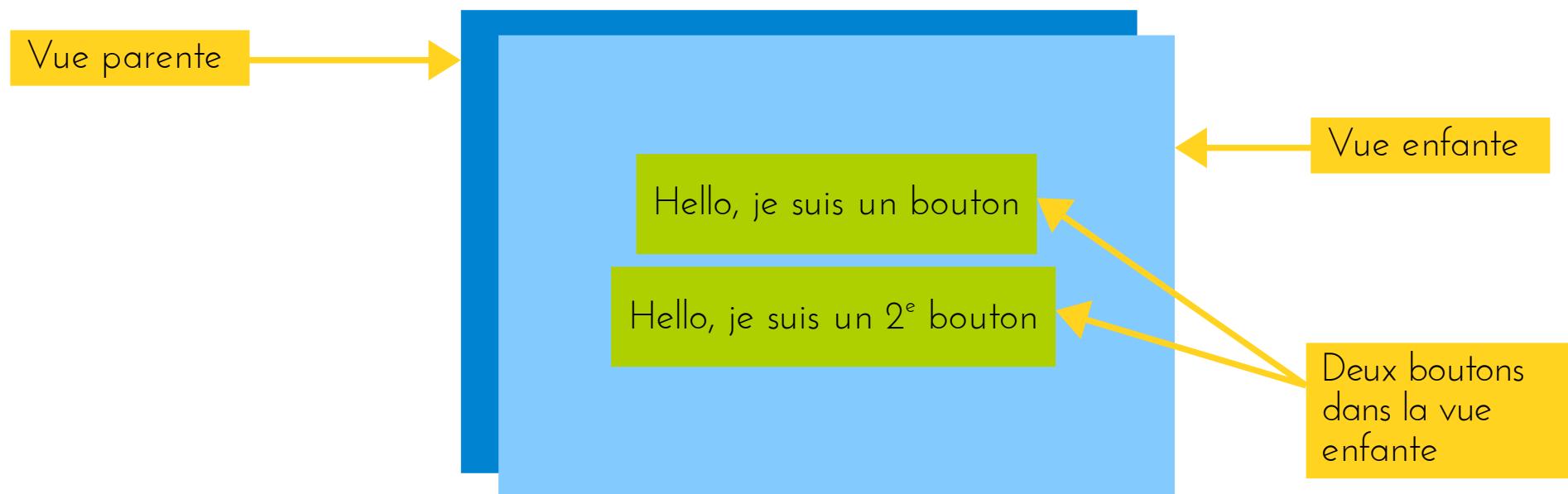
- Positionner des éléments les uns à côtés (ou au dessus) des autres se fait à l'aide de la propriété « layout »





Positionner et styles des éléments

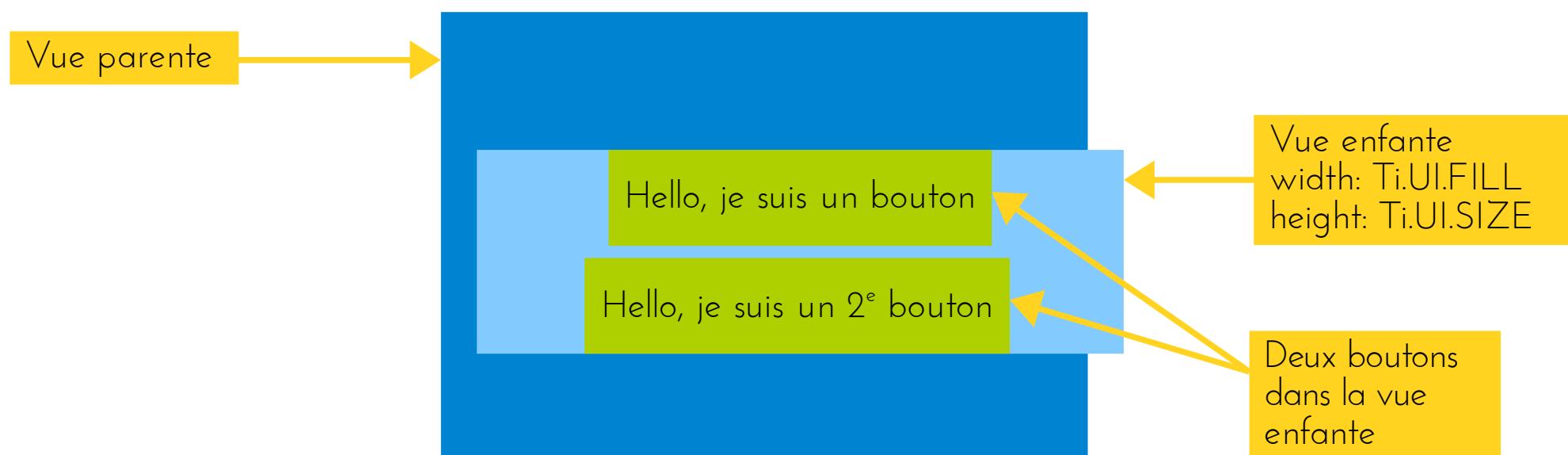
- Les enfants d'une vue (hors widget) utilisent par défaut tout l'espace disponible





Positionner et styles des éléments

- Limiter la taille d'une vue à celle de son contenu se fait à l'aide de Ti.UI.SIZE :





Évènements : émission et réception



Réception d'un événement « local »

```
var button = Titanium.UI.createButton({  
    font: { fontWeight:'bold' },  
    title: 'Don\'t click Me'  
});  
win.add(button);  
  
button.addEventListener('click', function(event) {  
    alert('Hey you clicked me!');  
  
    // change button's label  
    event.source.title = 'I feel pain';  
});
```

- Évènement attaché à un élément de l'interface
 - Une grande variété d'évènements, varient suivant le widget / la vue
 - cf. documentation d'API pour tous les détails



```
// create an http client
var client = api.createClient();

client.onreadystatechange = function() {
    if (this.readyState == 4) {
        // if the request is successful, send an event
        Ti.App.fireEvent(
            'server.request.result',
            { result: this.responseText }
        );
    }
};
```

- `fireEvent` prend deux arguments :

- Le nom de l'évènement (ce libellé est libre)
- Un tableau de valeurs associés à l'évènement (permet un traitement par les écouteurs)



Réception d'un événement « global »

```
Ti.App.addEventListener('server.request.result', function(event) {  
    my_app.hideIndicator();  
  
    if (event.result) {  
        alert('A valid server response was received: ' + event.result);  
    } else {  
        alert('Something went wrong, apologies for the inconvenience.');  
    }  
});
```

- `addEventListener()` prend deux arguments :
 - Le nom de l'évènement (ce libellé est libre)
 - Une fonction de callback à effectuer à la réception de l'évènement
- Attention :
 - Les évènements sont traités de manière asynchrone (non bloquants)



Accès aux données distantes



- L'accès aux données distantes se fait par le biais de l'API Ti.Network.* :
 - Titanium.Network
 - Titanium.Network.Socket
 - Titanium.Network.Socket.TCP
- Plusieurs protocoles :
 - HTTP (XMLHttpRequest)
 - Bonjour
 - Connexion par socket TCP



```
Ti.API.info('en ligne : ' + (Titanium.Network.online ? 'oui' : 'non'));
Ti.API.info('type de connexion : ' + Titanium.Network.networkTypeName);
Ti.API.info('code du type de connexion : ' + Titanium.Network.networkType);
```

- Attributs de Titanium.Network :

- online : indique si une connexion est disponible ou pas
- networkTypeName : NONE, WIFI, LAN ou MOBILE
- networkType : valeur numérique indiquant le type de réseau



```
var client = Titanium.Network.createHTTPClient();
client.timeout = 20000; // 20 s. timeout

client.onload = function() {
    alert('Résultat reçu : ' + this.responseText);
};

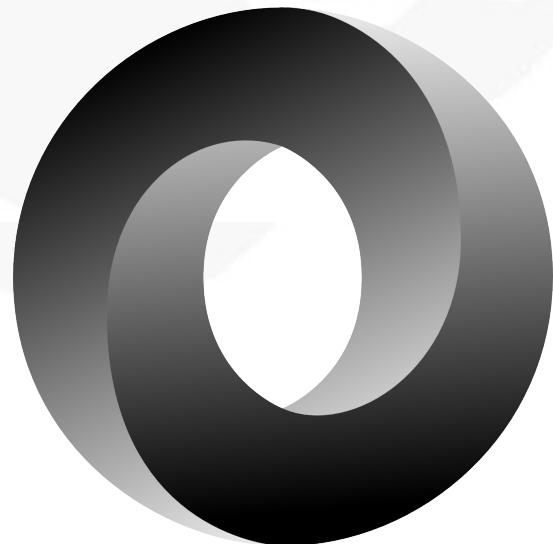
client.open('GET', url);
client.setRequestHeader('User-Agent', 'My super User Agent/1.0');

if ('' !== username && '' !== password) {
    // handle http basic authentication
    auth = 'Basic ' + Titanium.Utils.base64encode(username + ':' +
password);
    client.setRequestHeader('Authorization', auth);
}

client.send(null);
```

- Méthodes utiles :

- abort
- setTimeout
- open, send
- getResponseHeader, setRequestHeader



- Contenu de la réponse HTTP :
 - responseData
 - responseText
 - responseXml
 - status
- Un conseil : APIs mobiles = JSON (pas XML) !
- Autres APIs Titanium en lien avec le réseau :
 - Bonjour : déclaration et recherche de services
 - Sockets TCP (connexions TCP)
 - Autres protocoles possibles

Fahrkarten
Informationen
Reservierungen

Tickets
Information
Reservations



Accéder aux APIs matérielles



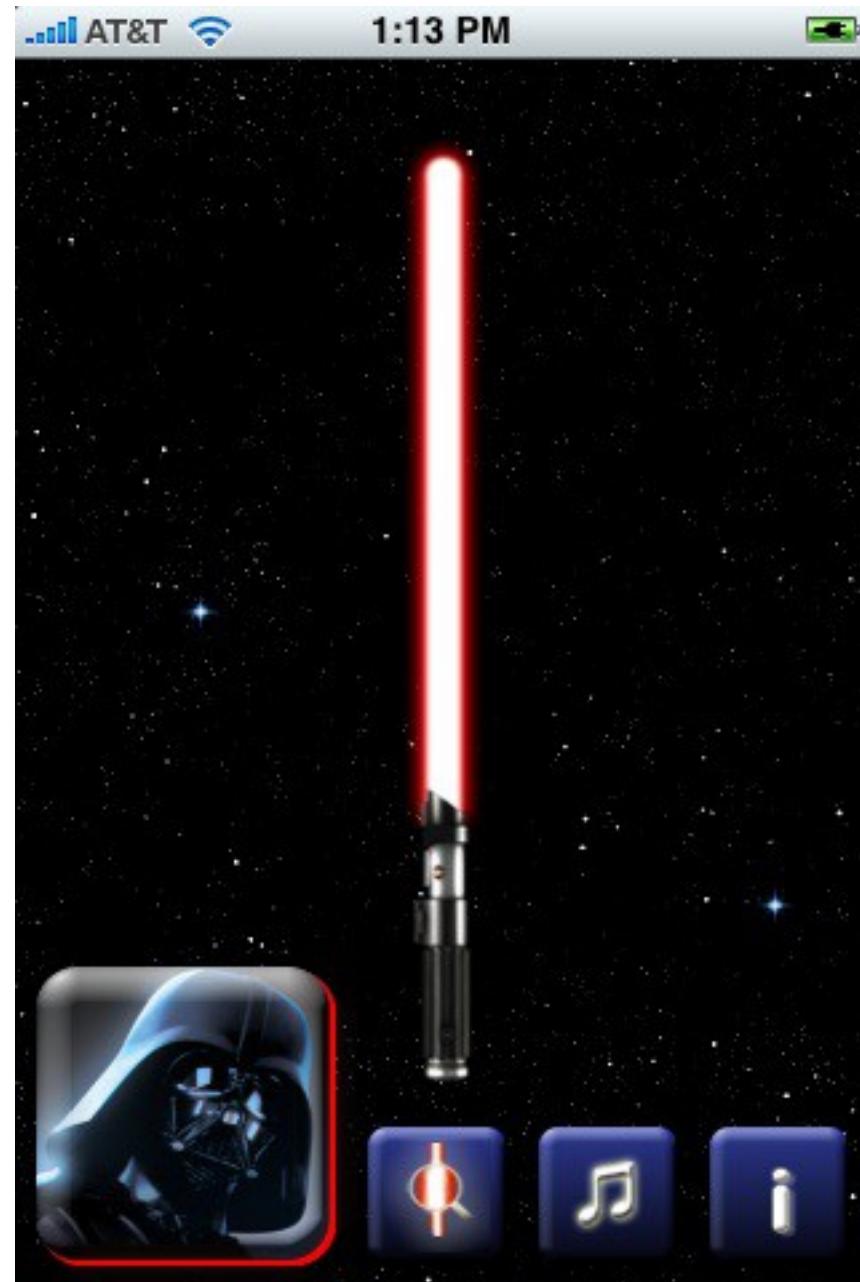
- Objectif : détecter les mouvements

- Pas de méthode : juste un événement « update »

```
Titanium.Accelerometer.addEventListener(  
    'update',  
    function(e) {  
        Ti.API.debug("x: " + e.x + ", y: " + e.y + ", z: "+ e.z);  
  
        // compute change since last time  
        // if more than a value,  
        // display some light and make some noise  
  
        // you have a light saber  
    }  
);
```

- Ne pas oublier de se désabonner (consommateur) :

```
// remove the listener  
Titanium.Accelerometer.removeEventListener('update', oldFunction);  
  
// add it back  
Titanium.Accelerometer.addEventListener('update', oldFunction);
```





- Pratique pour sauver des données (cache, sauvegarde locale de documents, etc.)
- **Ti.Filesystem.File**
 - Fichiers :
 - `createFile()`, `deleteFile()`, `read()`, `write()`, `rename()`
 - `extension()`, `nativePath()`
 - Accesseurs : `exists()`, `readonly()`, `hidden()`, `writeable()`, `symbolicLink()`
 - Dossiers : `createDirectory()`, `deleteDirectory()`, `getDirectoryListing()`
- Informations au sujet du filesystem :
 - `spaceAvailable()`



- **Titanium.UI.Clipboard**
 - `getText()` / `setText()` / `hasText()`
 - `getData()` / `setData()` / `hasData()`
- **Support de plusieurs mime types**
 - Plusieurs clipboard en parallèle
 - Que sur Iphone
 - Texte uniquement sur Android



■ Titanium.Media

- accès aux interactions de type « Média »
 - Enregistrement / lecture de photos et vidéos
 - Enregistrement / lecture de sons ou de morceaux de musique
- Caméra :
 - que sur un vrai device, pas en simulateur
 - showCamera()
 - callbacks
 - mediaTypes
 - SaveToPhotoGallery
 - allowEditing
 - showControls
 - Overlay
 - takePicture()
 - hideCamera()
 - SaveToPhotoGallery()





```
var win = Titanium.UI.currentWindow;
Titanium.Media.showCamera({
    success: function(event) {
        var image = event.media;

        if (event.mediaType == Ti.Media.MEDIA_TYPE_PHOTO) {
            var imageView = Ti.UI.createImageView({
                width: win.width,
                height: win.height,
                image: event.media
            });
            win.add(imageView);
        } else {
            alert('Take a picture. Illegal type ' + event.mediaType);
        }
    },
    cancel: function() {},
    error: function(error) {
        // show an error message
        // test error code Titanium.Media.NO_CAMERA
    },
    saveToPhotoGallery: true,
    mediaTypes: [Ti.Media.MEDIA_TYPE_VIDEO, Ti.Media.MEDIA_TYPE_PHOTO]
});
```



- Titanium.Geolocation

- `getCurrentPosition()`
- `getCurrentHeading()` : direction

- Fonctionnalités de geocoding :

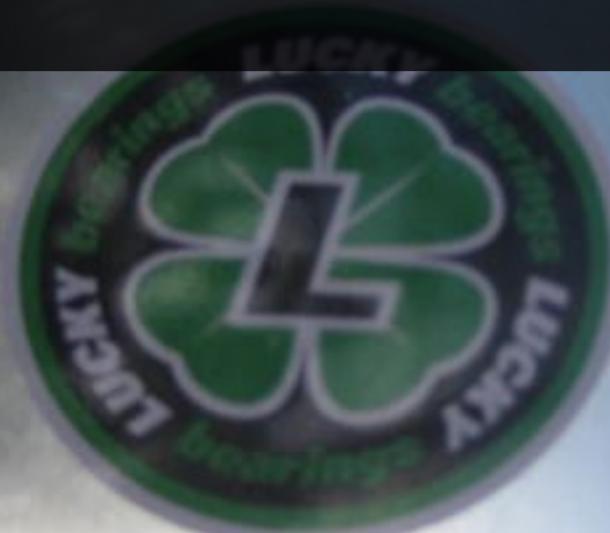
- `ForwardGeocoder()` adresse → coordonnées
- `ReverseGeocoder()` coordonnées → adresse
- Pratique pour du guidage

- Évènements : « location » et « heading »

- Attention aux performances (envoyés en continu), cf. paramètres `locationFilter` et `headingFilter`



Alloy, le framework MVC officiel pour Titanium



TITANIUM
LIGHTER FASTER STRONGER



- Quelques limitations néanmoins :
 - Pour le développeur, Titanium est une plateforme
 - Pas un framework au sens RAD / MVC
 - UI écrite en JS = aïe
- Plateforme géniale mais :
 - Peu contraignante
 - Bonnes pratiques : selon le développeur
- De nombreux frameworks pseudo-MVC
 - Manque de standardisation des savoir-faire
 - Contributions hétérogènes



Les frameworks existants...

- IQ SDK - <https://github.com/dev-iqe/iq-sdk>
 - EnJS + Joose + classes custom
 - Aucune communauté
- Extanium - <http://extanium.org>
 - Extjs in Titanium
 - V 0.2.1, rien depuis juin 2011. Aucune communauté
- TiMVC - <http://timvc.com/>
 - Ti.include(), pas d'utilisation de CommonJS
 - Aucune communauté
- Kranium - <http://kraniumjs.com/>
 - Assemblage de briques
 - V 0.1.3, rien depuis septembre 2011. Aucune communauté



- Les frameworks existants...

- IQ SDK - <https://github.com/dev-iQue/iqSDK>

- EnJS + Joose + classes custom
 - Aucune communauté

- Extanium - <http://extanium.org>

- Extjs in Titanium
 - V 0.2.1, rien depuis juin 2011.
 - Aucune communauté

- TiMVC - <http://timvc.com/>

- Ti.include(), pas d'utilisation de CommonJS
 - Aucune communauté

- Kranium - <http://kraniumjs.com/>

- Assemblage de briques
 - V 0.1.3, rien depuis septembre 2011.
 - Aucune communauté

Seriously.
WTF???



- Alloy

- Open Source, mené par Appcelerator
- Apparu mi 2012 (première bêta en juin)
- Utilise des composants ouverts (projets tiers)
- S'appuie sur node.js, disponible par npm
- Déjà bien documenté
- GitHub : ~800 stars, ~600 forks
- Test :
http://docs.appcelerator.com/titanium/latest/#!/guide/Alloy_Quick_Start
ou
<https://github.com/appcelerator/alloy/tree/master/test/apps>



- Pré-requis pour jouer :

- node.js

- Installation :

```
$ sudo npm install -g alloy
```



- Un outil en ligne de commande :

- new : initialise un projet alloy
 - compile : compile le projet pour cible « webapp »
 - extract-i18n : extrait les nouvelles chaînes i18n du code
 - generate : génère du code (contrôleur / modèle)



Architecture d'un projet alloy

app/ ————— dossier de code lié à alloy

build/

i18n/

modules/

plugins/

Resources/

manifest

tiapp.xml

dossier de code lié à alloy

Dossiers et fichiers habituels
d'un projet Titanium

Ne codez plus dans Resources !



Architecture d'un projet alloy

```
app/  
  assets/ ——————  
  controllers/  
  lib/  
  migrations/  
  models/  
  styles/  
  views/  
  widgets/  
  alloy.jmk  
  alloy.js  
  README  
  config.json
```

Les assets (images, sons, etc.)
de votre projet.

Déployé dans Resources/ à la
compilation



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/
styles/
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Les contrôleurs, ie. Les articulations de votre application



app/
assets/
controllers/
lib/ —————
migrations/
models/
styles/
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Les librairies CommonJS :

- Vos librairies métier
- Des libs externes (lib/vendor)

Dossier à créer à la main

Chargement par : require('library')



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/
styles/
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Les classes de migration de vos modèles de données.

Dossier à créer à la main



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/ ——————
styles/
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Les classes du modèle de données.



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/
styles/ ——————
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Les styles associés aux vues.
Fichiers .tss (Titanium
StyleSheet)
Syntaxe similaire à CSS



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/
styles/
views/ ——————
widgets/
alloy.jmk
alloy.js
README
config.json

Les vues de l'application : des templates au format XML

Plus besoin de Ti.UI.create*



Architecture d'un projet alloy

```
app/  
  assets/  
  controllers/  
  lib/  
  migrations/  
  models/  
  styles/  
  views/  
  widgets/  
    alloy.jmk  
    alloy.js  
  README  
  config.json
```

Les widgets propres à votre application (ie., non fournis par alloy).

Les widgets sont des composants à assembler pour créer l'application.

Dossier à créer à la main



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/
styles/
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Fichier type makefile, pour définir des opérations pre- et post- compilation :

- Changer la configuration
- Logger des informations
- Déployer sur un serveur
- etc.



Architecture d'un projet alloy

app/
assets/
controllers/
lib/
migrations/
models/
styles/
views/
widgets/
alloy.jmk
alloy.js —————
README
config.json

Bootstrap / initialisation
Code exécuté au lancement de
l'application (=Resources/app.js)



Architecture d'un projet alloy

```
app/  
assets/  
controllers/  
lib/  
migrations/  
models/  
styles/  
views/  
widgets/  
alloy.jmk  
alloy.js  
README —————  
config.json
```

Littérature





Architecture d'un projet alloy

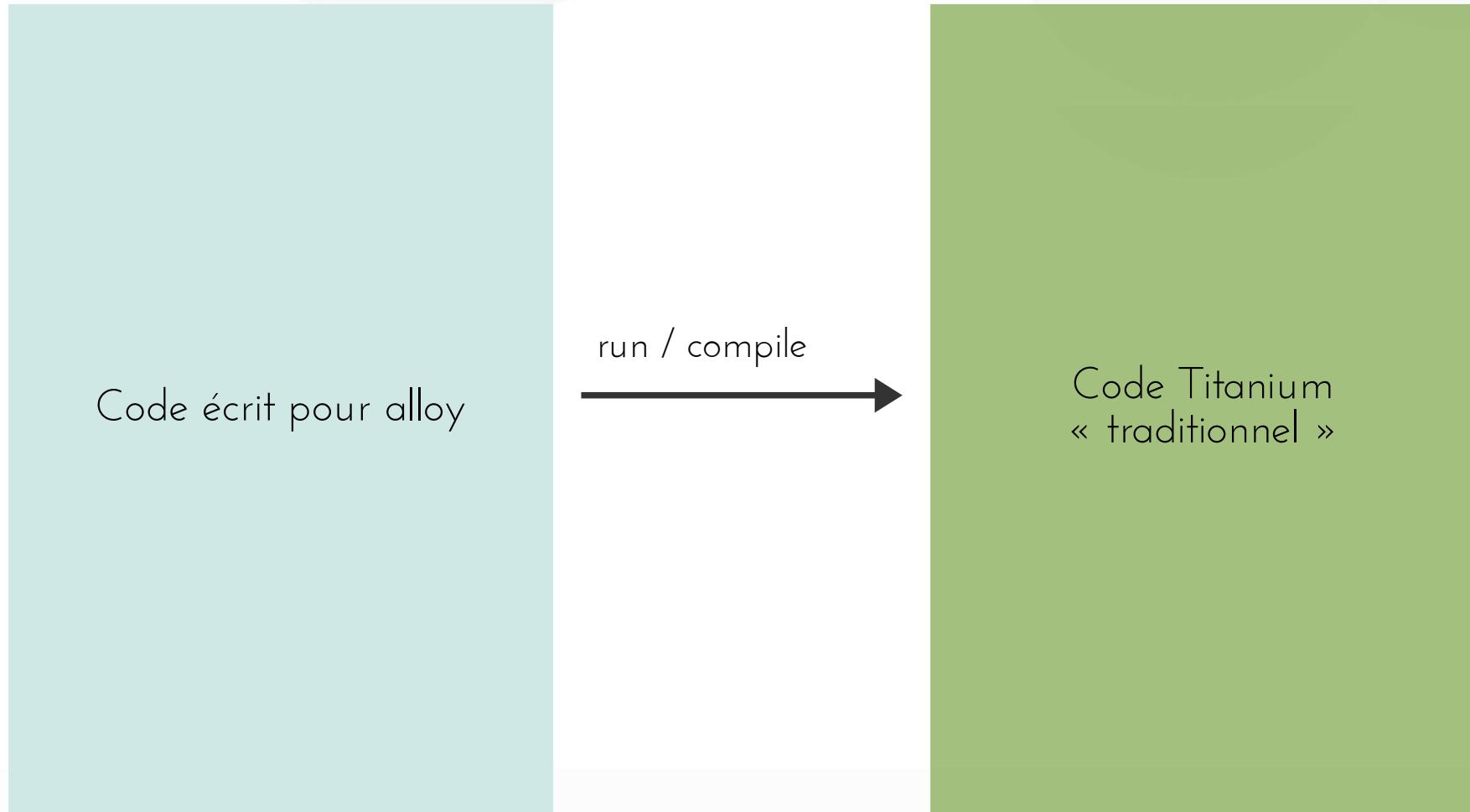
app/
assets/
controllers/
lib/
migrations/
models/
styles/
views/
widgets/
alloy.jmk
alloy.js
README
config.json

Déclaration de constantes et paramètres d'exécution (urls, variables, etc.).

Déclaration des dépendances de l'application envers des widgets.

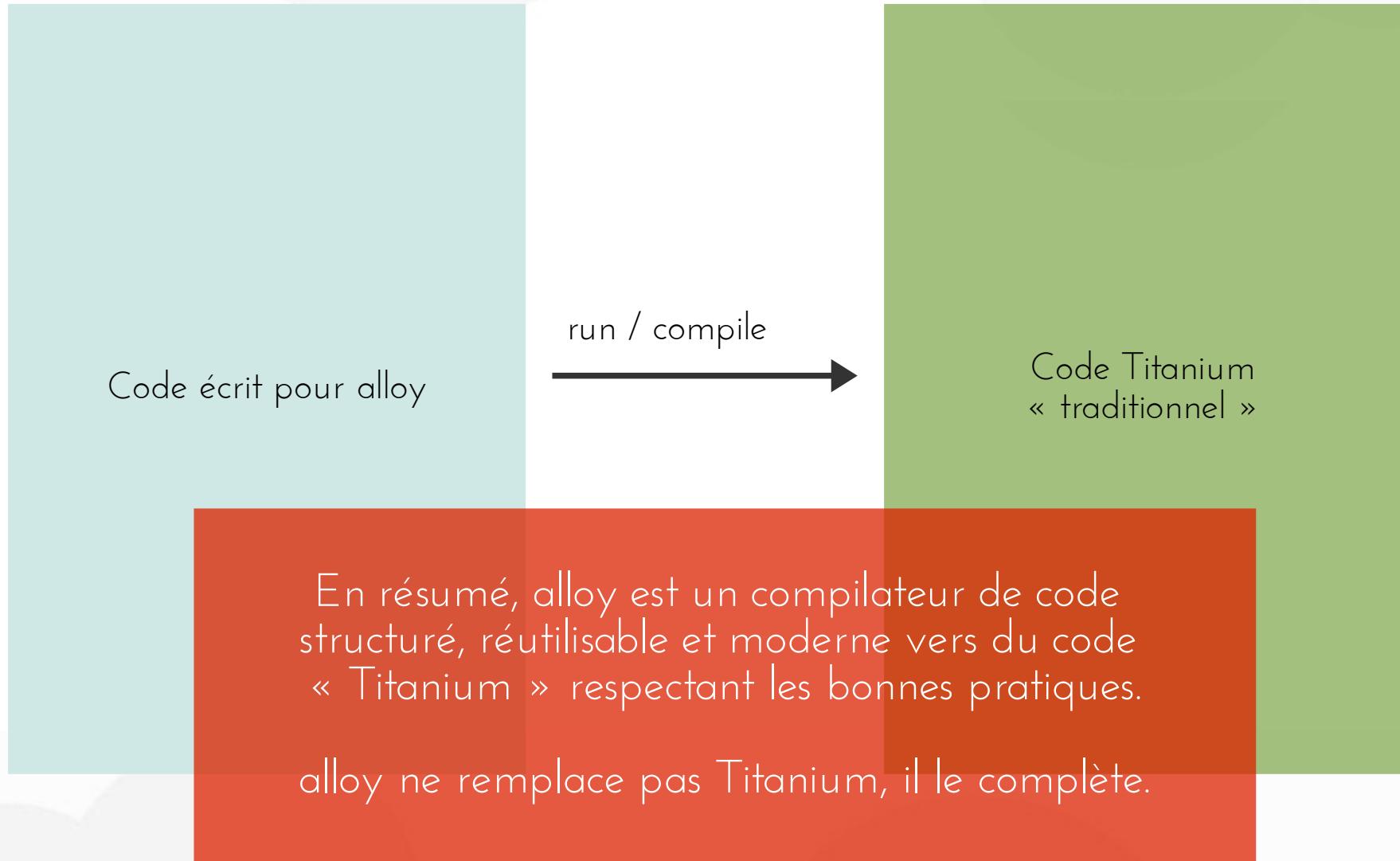


Principe de fonctionnement



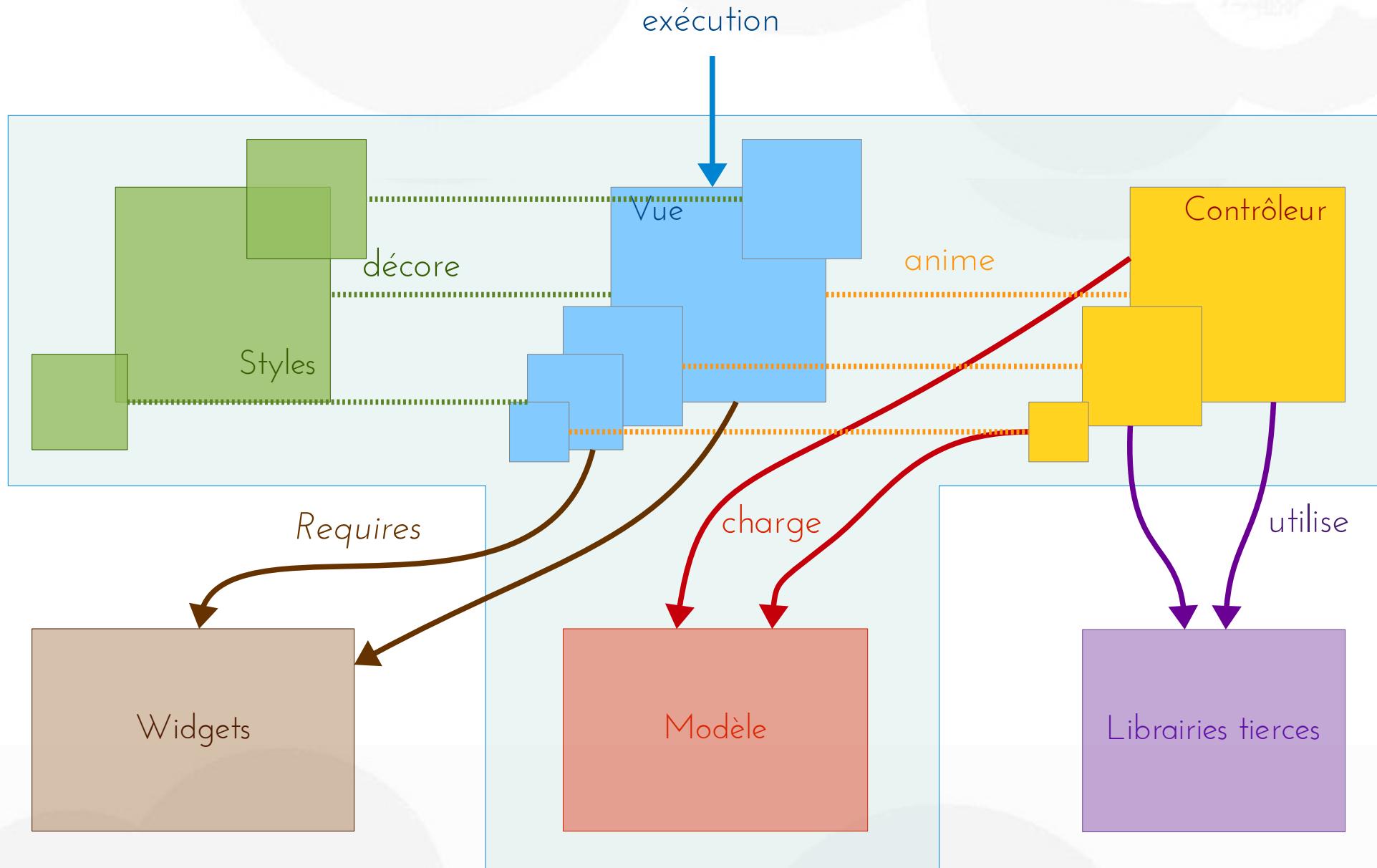


Principe de fonctionnement





Architecture d'une application alloy





- Les vues sont des fichiers XML
- Au moins une vue (le premier « écran ») :
`index.xml`
- Les vues ne définissent pas l'apparence de l'application mais seulement son organisation
- Les styles `.tss` permettent de styler les vues (analogie à HTML et CSS)



- Exemple de vue principale :

```
<Alloy>
  <Window class="container">
    <Label id="label" onClick="doClick">Hello, World</Label>
  </Window>
</Alloy>
```

<Alloy> conteneur de l'application

<Window> : Préfixé par Ti.UI.
Attribut ns="..."
<Label> : Pour label, etc.



- Exemple de vue principale :

```
<Alloy>
  <Window class="container">
    <Label id="label" onClick="doClick">Hello, World</Label>
  </Window>
</Alloy>
```

Support de classes et d'identifiants

Possible de gérer des évènements



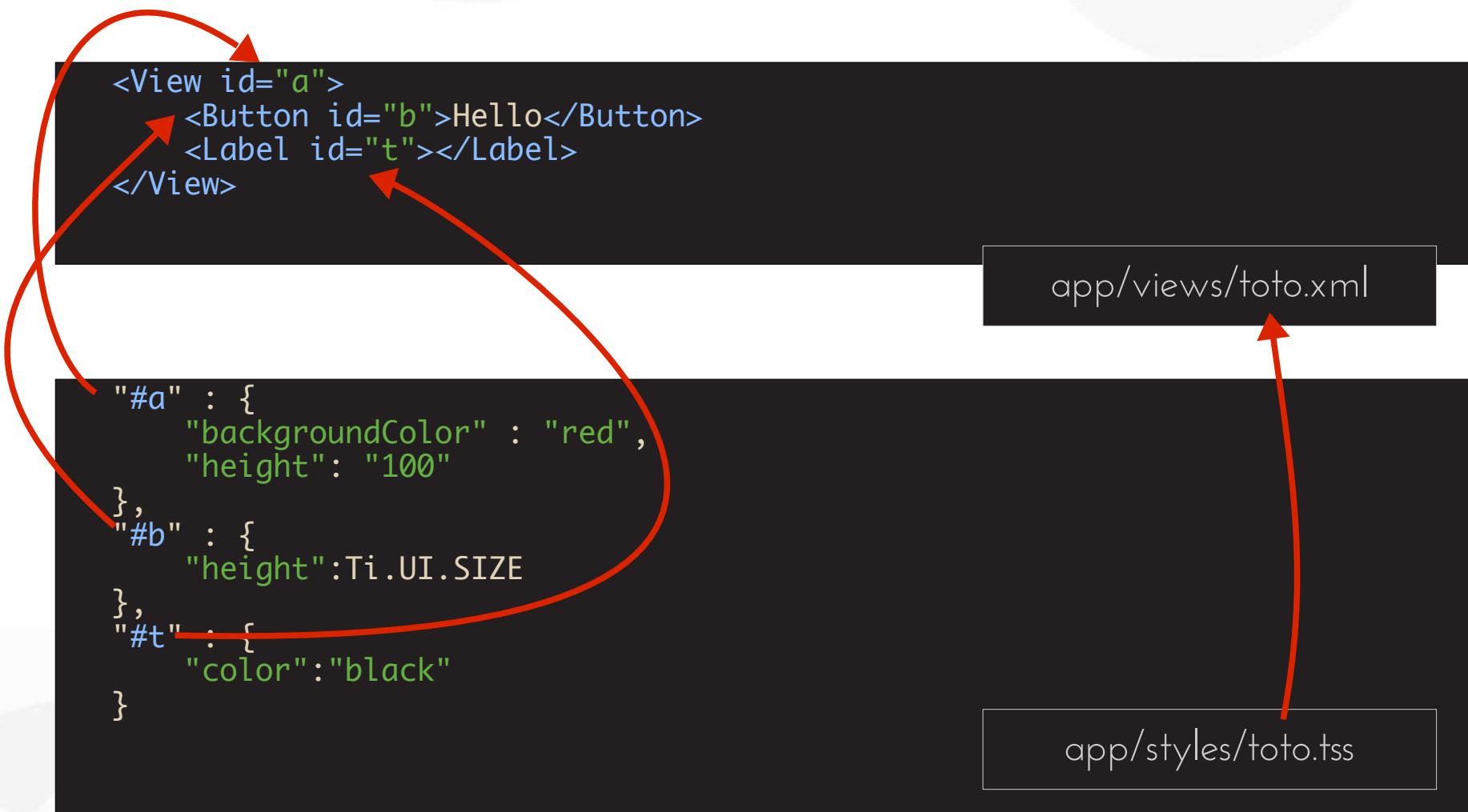
- Inclusion de vues

```
<Alloy>
  <Require src="header" id="header"/>
  <Require src="home" id="content"/>
  <Require src="menu" id="menu"/>
</Alloy>
```

- Mot clé « Require »
- Possibilité de diviser les vues
 - Hiérarchie
 - Réutilisation de vues partielles



- Les styles sont définis dans des fichiers .tss
 - nom_de_vue.xml ↔ nom_de_vue.tss



- On peu appliquer un style de manière globale en créant un thème graphique
- Dans app/themes/NOM_DU_THEME
 - Sous-dossiers :
 - Styles/
 - Assets/
 - NOM_DU_THEME doit être déclaré dans le config.json :

```
{  
    "global": {  
        "foo": "bar",  
        "theme": "poneyClub"  
    },  
    "env:development": {},  
    "env:test": {},  
    "env:production": {},  
    "os:android": {},  
    "os:ios": {},  
    "dependencies": {}  
}
```



Les contrôleurs, un peu d'activité en vue...

- Même principe pour les contrôleurs :
 - nom_de_vue.xml ↔ nom_de_vue.js

```
<View id="a">
  <Button id="b">Hello</Button>
  <Label id="t" onClick="doClick">Michel</Label>
</View>
```

app/views/toto.xml

```
$.a.backgroundColor = 'blue';

$.b.addEventListener('click', function(e){
  $.t.text = 'Gérard';
});

function doClick(e) {
  $.t.color = 'green';
}
```

app/controllers/toto.js



Les contrôleurs, un peu d'activité en vue...

- Même principe pour les contrôleurs :

- nom_de_vue.xml ↔ nom_de_vue.js



```
$.a.backgroundColor = 'blue';
```

```
$.b.addEventListener('click', function(e){
    $.t.text = 'Gérard';
});
```

```
function doClick(e) {
    $.t.color = 'green';
}
```

app/views/toto.xml

Ajout de comportement par écouteur d'évènement

app/controllers/toto.js



Instancier un contrôleur depuis un autre

- Un contrôleur peut en instancier un autre :

```
var openListPage = function(e) {
    var page = Alloy.createController('foo/listWindow').getView();
    page.open();
};
```

- Passage de paramètres :

```
var openListPage = function(e) {
    var page = Alloy.createController('foo/listWindow', {
        foo: 'coucou',
        bar: 'poney'
    }).getView();

    page.open();
};
```

app/controllers/toto.js



Instancier un contrôleur depuis un autre

- Récupération des arguments :

app/controllers/foo/listWindow.js

```
var args = arguments[0] || {};
var foo = args.foo; // attention, potentiellement indéfini !
var poney = args.poney || 'cheval';
```

- Très pratique pour découper son application
 - Évite d'employer Ti.UI.create* dans les contrôleurs
 - Bonne pratique : tout élément de vue est créé dans un template XML, jamais en js



Créer des widgets Alloy



Les widgets, un soupçon de réutilisabilité

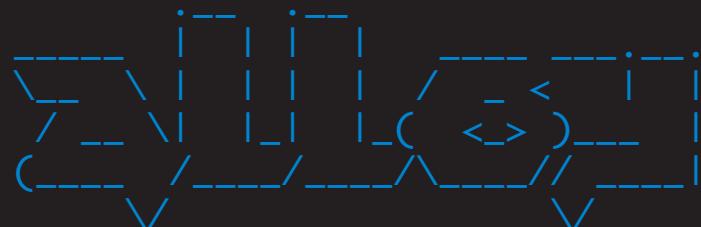
- Un widget :
 - un module embarquant sa propre logique MVC
 - un package réutilisable, autonome
 - peut être importé dans toute application Titanium Alloy
 - peut fournir des vues, ou simplement des fonctionnalités



Les widgets, un soupçon de réutilisabilité

- Toute partie « générique » d'une application devrait être développée sous la forme d'un widget
 - Datepicker
 - Formulaire de connexion Oauth
 - etc.
- Créer un widget est simple :

```
$ alloy generate widget com.picmybox.hellobutton
```



Alloy by Appcelerator. The MVC app framework for Titanium.

```
[INFO] Generated widget named com.picmybox.hellobutton
```



- Le code est généré dans app/widgets/XXX
- Un widget se présente comme une application à part entière
 - Vue
 - Contrôleur
 - Modèles
 - Librairies
 - widget.json décrit le widget
- Le widget doit être référencé dans le config.json du projet !



- Emploi d'un widget dans une vue :

```
<Alloy>
  <Window id="coucou">
    <Require type="widget" src="com.jolicode.helloButton" id="poney"/>
  </Window>
</Alloy>
```

- Un widget peut embarquer sa propre logique
 - On peut la surcharger localement dans l'application
 - S'il y en a, les assets seront copiés dans **Ressource** au moment de la compilation



- Un widget peut exposer une API publique vers l'application :

```
// widget.js controller file  
exports.doSomething = function()  
{  
    alert('Coucou poney');  
}
```

Mot-clé **exports** définition d'une API publique (bisous CommonJS)

```
// application controller  
$.foo.doSomething();
```

Appel de la méthode par le biais du nom du Widget



Au sujet des widgets...

- Liste de widgets existants...
 - <http://alloylove.com/>
 - <http://gitt.io/>

The screenshot shows the homepage of the Alloy website. The main heading is "Get your  love." Below the main title is a search bar with the placeholder "I need ...". At the bottom, there are two cards: one for "ACS Sync" and one for "Acs Login".

ACS Sync

New and improved ACS Alloy Module

adapter
Aaron Saunders

Acs Login

An ACS login widget for Titanium

widget



Ressources et lecture...



- Titanium...

- <http://developer.appcelerator.com/blog/feed>
- <http://cssgallery.info/feed/>
- <http://guilherme.pro/feed/>
- <http://blog.mattapperson.com/feed.xml>
- <http://feeds.feedburner.com/TitaniumNinja>
- <http://www.vancelucas.com/feed/>
- <http://www.tryexcept.com/feed>
- <http://roguesynaptics.com/rss>
- <http://fokkezb.nl/feed/>
- <http://gitt.io/>
- <http://www.tidev.io/>

- Javascript...

- <http://rss.badassjs.com/>
- <http://feeds.feedburner.com/JohnResig>



- Où trouver de la documentation ?

- Documentation d'API + documentation de référence
<http://docs.appcelerator.com/titanium/latest/>
- Exemple des APIs : KitchenSink
<http://github.com/appcelerator/KitchenSink>



This is the end

Questions ?



- Une application pour trouver des Vélib's
 - Utiliser l'API de JC Decaux : <https://developer.jcdecaux.com>
- Features attendues :
 - Affichage des stations sur la carte
 - Possibilité de se localiser
 - Chercher une adresse précise
 - Voir le détail d'une station



Pour démarrer / aujourd'hui





Pour démarrer / aujourd'hui





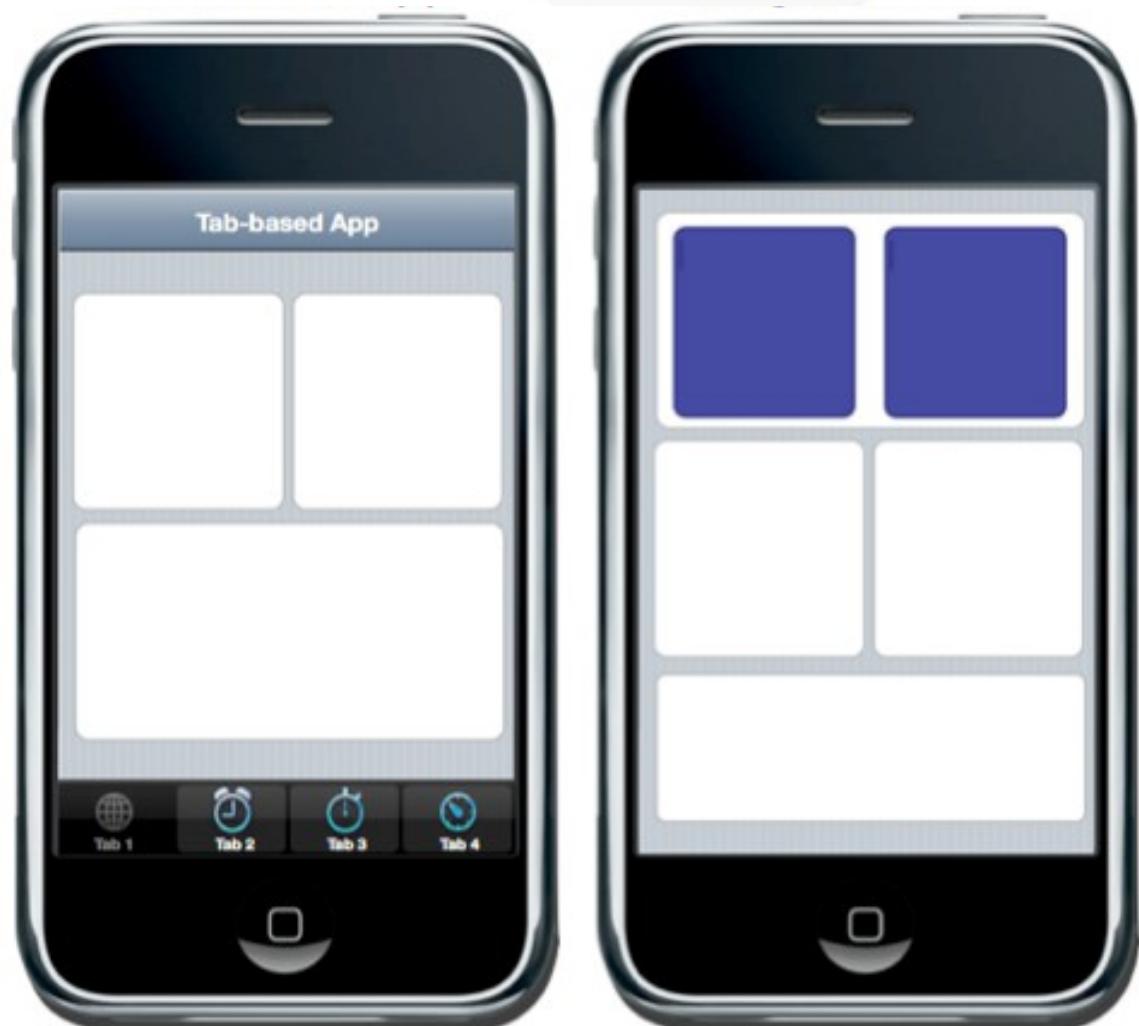
Annexe - Exemples d'éléments graphiques
fenêtres, vues, widgets, etc.



- Deux modes de conception :

- Fenêtres autonomes
- Groupe de « tab »
- (« TabGroup »)

- cf. l'API pour les options





```
// create a window
var win = Titanium.UI.createWindow({
    backgroundColor:'#edeff6',
    title:'Disclaimer',
    barColor:'black'
});

// add a button in it
var button = Titanium.UI.createButton({
    title: 'Close',
    style: Titanium.UI.iPhone.SystemButtonStyle.PLAIN
});
win.setLeftNavButton(b);
button.addEventListener('click', function() {
    win.close();
});

// open the window
win.open({modal:true});
```



```
// create tab group
var tabGroup = Titanium.UI.createTabGroup();
var win1 = Titanium.UI.createWindow({
    title:'Tab 1',
    backgroundColor:'#fff'
});
var tab1 = Titanium.UI.createTab({
    icon:'tab1icon.png',
    title:'Tab 1',
    window:win1
});
var win2 = Titanium.UI.createWindow({
    title:'Tab 2',
    backgroundColor:'#fff'
});
var tab2 = Titanium.UI.createTab({
    icon:'tab2icon.png',
    title:'Tab 2',
    window:win2
});

// add tabs to the group
tabGroup.addTab(tab1);
tabGroup.addTab(tab2);

// open tab group
tabGroup.open();
```



- Au sein d'une fenêtre, la composition graphique de l'application se fait à l'aide de vues :
 - Image view
 - Scroll view
 - Table view
 - Web view
 - Map view
 - Overflow view
 - Dashboard view
- Une vue peut en contenir une autre
- Analogie à Java Swing...

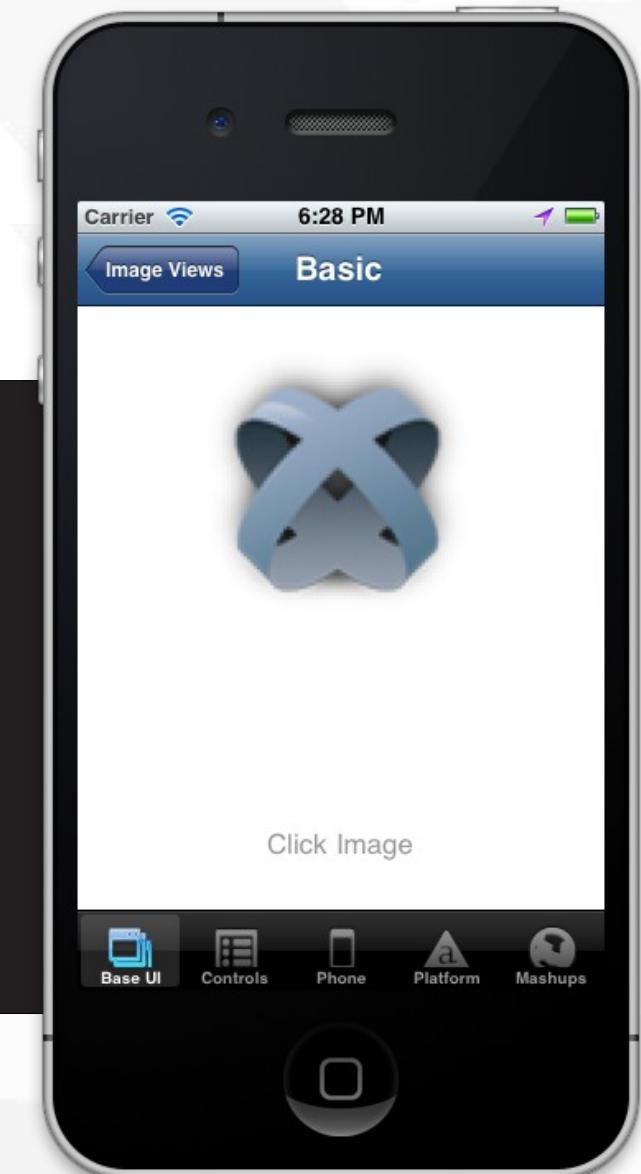


■ Insertion d'images

- Le paramètre « image » peut être une url

```
var win = Titanium.UI.currentWindow;
var imageView = Titanium.UI.createImageView({
    image:'titanium.png',
    width:261,
    height:178,
    top:20
});

imageView.addEventListener('load', function()
{
    Ti.API.info('LOAD CALLED');
});
win.add(imageView);
```





- Adapté aux contenus scrollables de taille variable
 - Affichage d'un pager

```
var view1 = Ti.UI.createView({
    backgroundColor: 'red'
});
var view2 = Ti.UI.createView({
    backgroundColor: 'blue'
});
var view3 = Ti.UI.createView({
    backgroundColor: 'green'
});
var view4 = Ti.UI.createView({
    backgroundColor: 'black'
});

var scrollView = Titanium.UI.createScrollView({
    views: [view1,view2,view3,view4],
    showPagingControl: true,
    pagingControlHeight: 30,
    maxZoomScale: 2.0,
    currentPage: 1
});
win.add(scrollView);
```



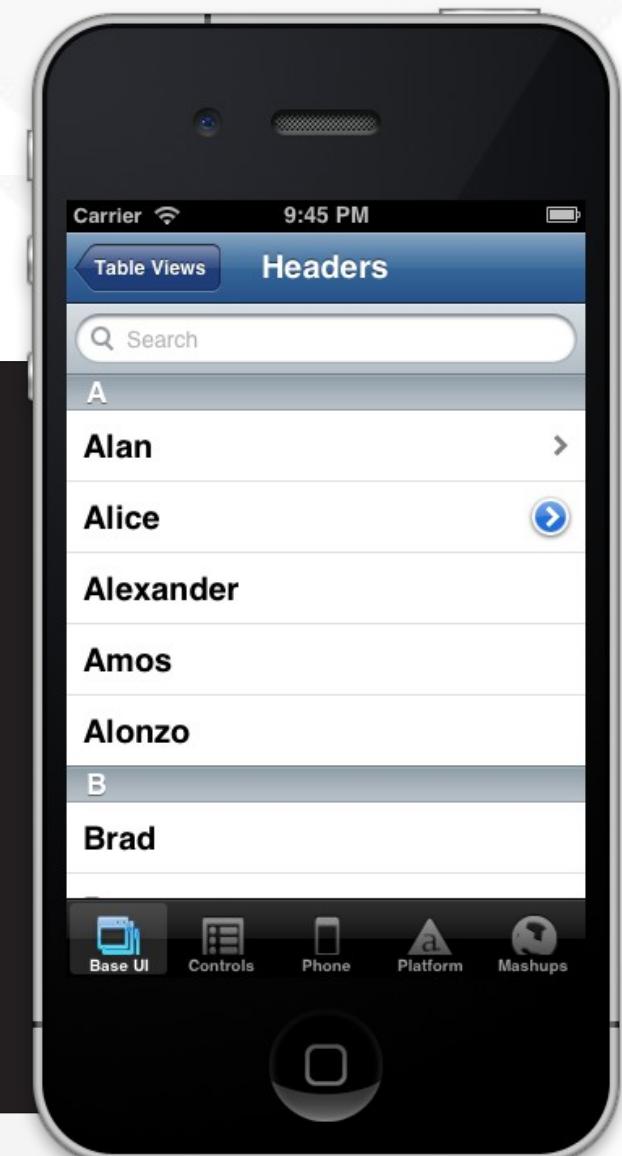


■ Données tabulaires

- Possibilité de headers
- Chaque cellule peut contenir des contrôles

```
var data = [
    {title: 'Alan', hasChild: true, header: 'A'},
    {title: 'Alice', hasDetail: true},
    {title: 'Brad', header: 'B'},
    {title: 'Brenda'},
    {title: 'Callie', header: 'C'},
    {title: 'Chris'},
];
var search = Titanium.UI.createSearchBar({
    showCancel: false
});

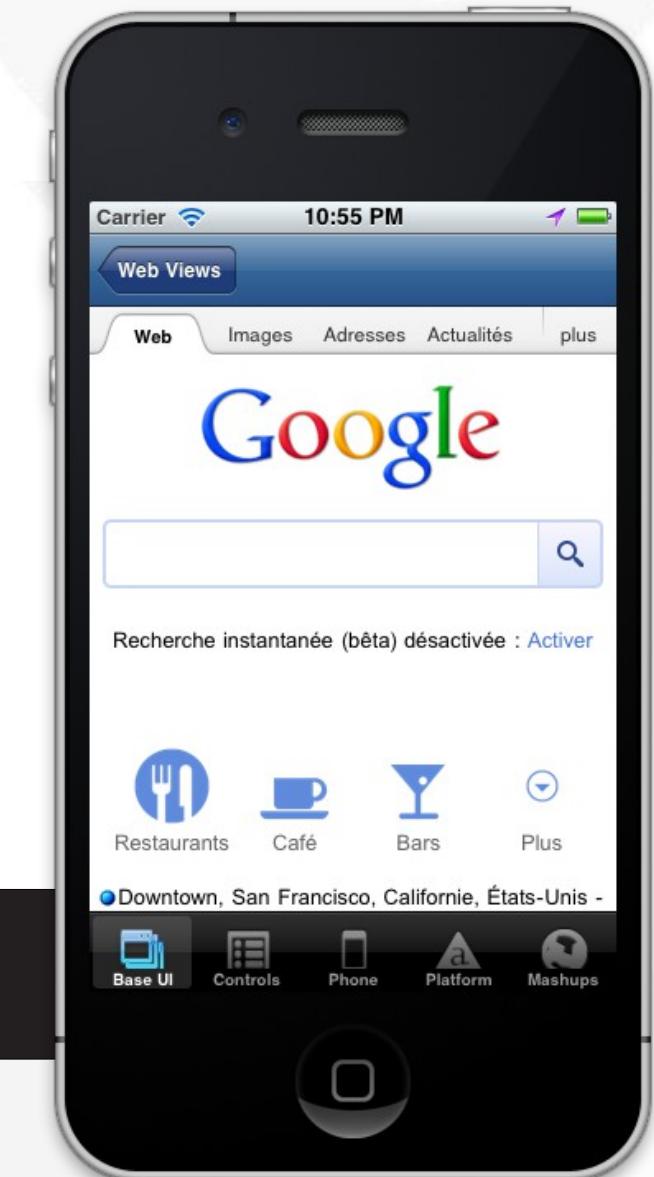
// create table view
var tableview = Titanium.UI.createTableView({
    data: data,
    search: search,
    filterAttribute: 'title'
});
win.add(tableview);
```





- Vue de navigateur Web
 - Plusieurs mime-type possibles
 - Contenu local ou distant
- Méthodes de manipulation
 - goBack(), goForward()
 - reload(), stopLoading()
 - setBasicAuthentication()

```
webview = Ti.UI.createWebView();
webview.url = 'http://www.google.com/';
win.add(webview);
```

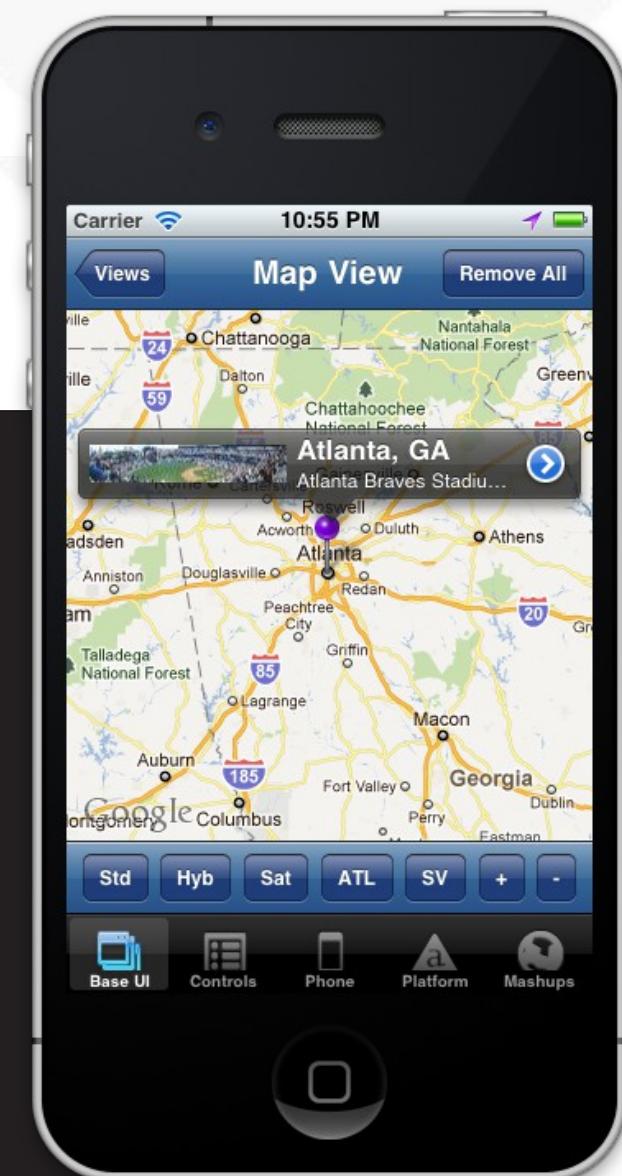




- Affichage de cartes Google Maps

- Ajout d'annotations
- Attention, différences entre Android et iOS
- Module natif à activer : ti.maps

```
var apple = Titanium.Map.createAnnotation({  
    latitude: 37.33168900,  
    longitude: -122.03073100,  
    title: 'Apple',  
    subtitle: 'Cupertino, CA',  
});  
var mapview = Titanium.Map.createView({  
    mapType: Titanium.Map.STANDARD_TYPE,  
    region:{  
        latitude: 33.74511,  
        longitude: -84.38993,  
        latitudeDelta: 0.5,  
        longitudeDelta: 0.5  
    },  
    regionFit: true,  
    userLocation: true,  
    annotations: [apple]  
});  
  
mapview.selectAnnotation(apple);  
win.add(mapview);
```





■ Carousel d'images

- iOS uniquement
- Images locales ou distantes
- setImage() pour changer l'image à une position donnée

```
var images = [  
    { image: 'http://domain.tld/1.jpg', width: 225, height: 225 },  
    { image: 'http://domain.tld/2.jpg', width: 225, height: 225 },  
    { image: 'http://domain.tld/3.jpg', width: 225, height: 225 },  
    { image: 'http://domain.tld/4.jpg', width: 225, height: 225 }  
];  
var view = Titanium.UI.iOS.createCoverFlowView({  
    images: images,  
    backgroundColor: '#000'  
});  
win.add(view);
```





Dashboard view

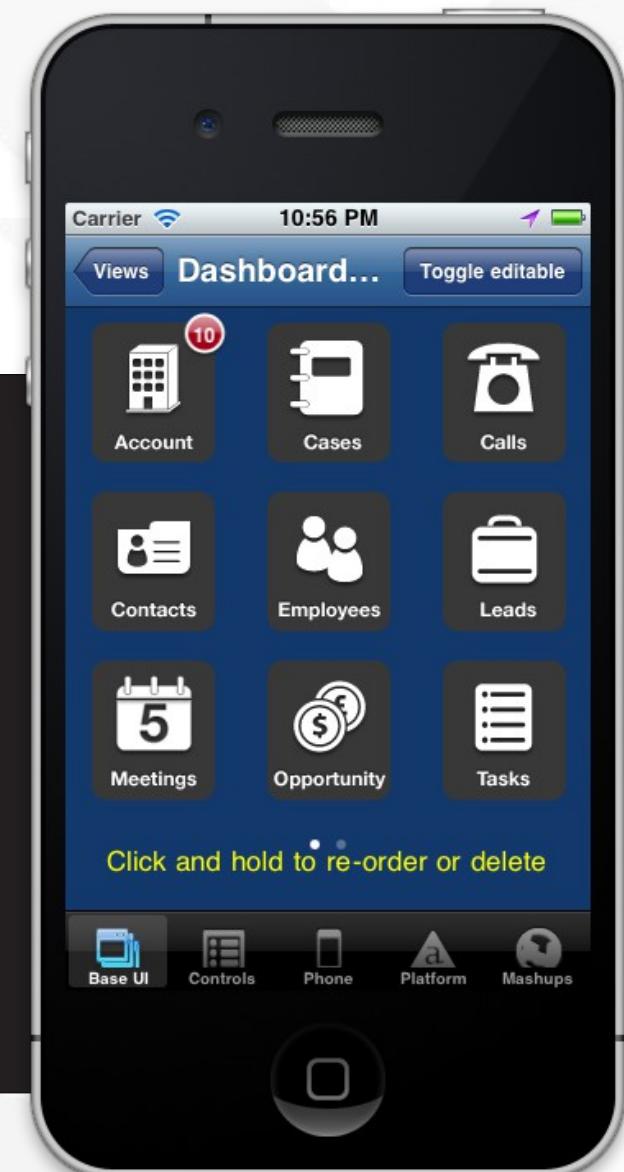
- Tableau de bord scrollable
 - IOS uniquement

```
var data = [];
var labels = ['account','calls','contacts','tasks'];

for (var c=0; c<labels.length; c++)
{
    var item = Titanium.UI.createDashboardItem({
        image:'../images/dashboard/'+labels[c]+'_off.png',
        selectedImage:'../images/dashboard/'+labels[c]+'_on.png',
        label:labels[c]
    });

    data.push(item);
}

var dashboard = Titanium.UI.createDashboardView({
    data:data
});
```





- Les widgets sont les éléments d'interface avec lesquels l'utilisateur peut interagir :

Activity Indicator

Label

Progress bar

Switch

Button bar

Toolbar

Textfield

Textarea

Button

Search bar

Slider

(Date) picker



Activity Indicator

- Indicateur d'activité
 - Positionnable un peu partout dans l'interface
- Constructeur :
 - `Ti.UI.createActivityIndicator()`
- API :
 - `show()`
 - `Hide()`
- Paramètres :
 - `color`
 - `font`
 - `message`
 - `style`





(Date) Picker

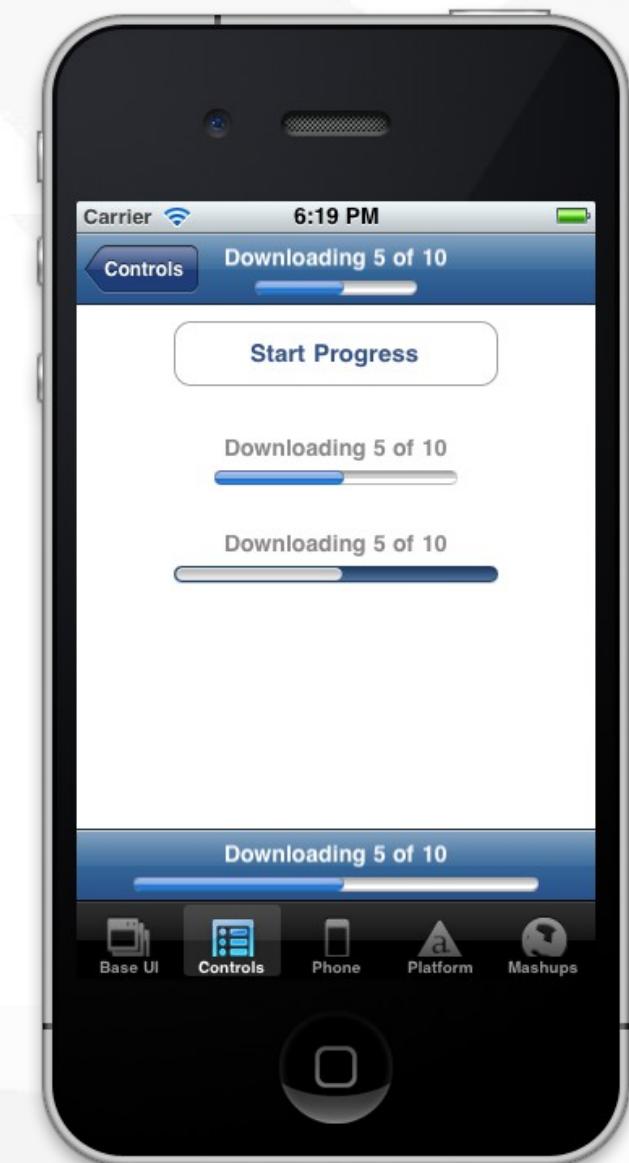
- Liste de sélection
 - Une ou plusieurs colonnes
- Constructeur :
 - Ti.UI.createPicker()
- API :
 - getSelectedRow(), setSelectedRow()
 - reloadColumn()
- Paramètres :
 - columns
 - locale
 - type
 - value
 - etc.





Progress bar

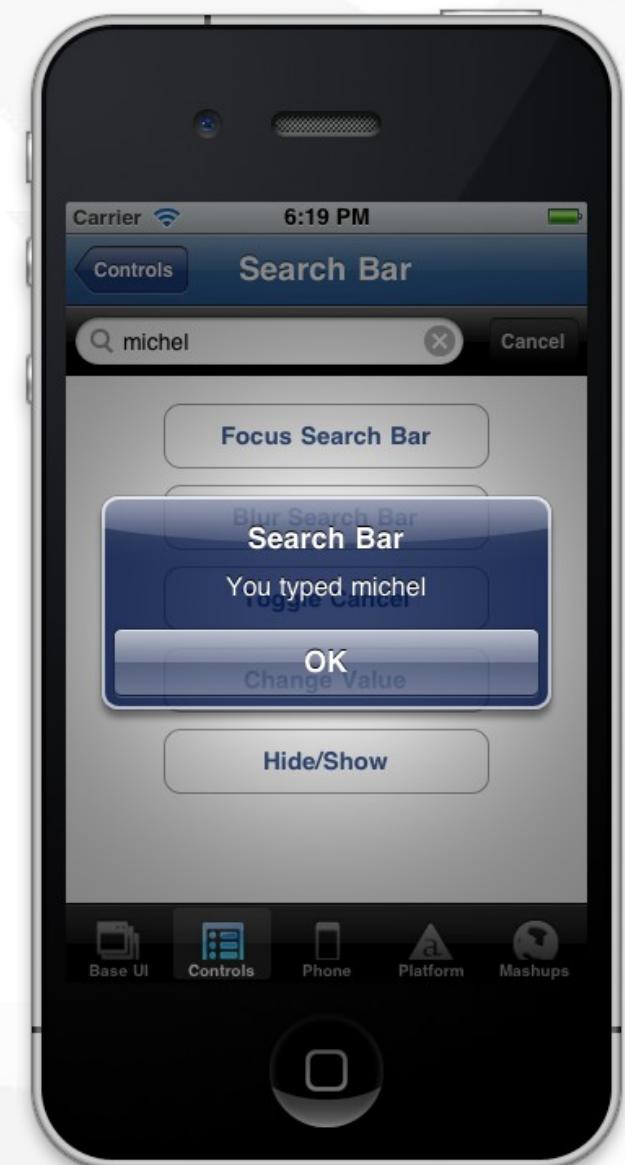
- Indicateur de progression
 - Positionnable un peu partout dans l'interface
- Constructeur :
 - Ti.UI.createProgressBar()
- API : aucune méthode
- Paramètres :
 - color
 - font
 - max / min
 - message
 - style
 - value





Search bar

- Barre de recherche
 - Actions de filtrage / recherche par évènements
- Constructeur :
 - `Ti.UI.createSearchBar()`
- API :
 - `blur()`
 - `focus()`
- Paramètres :
 - `showCancel`
 - `hintText`
 - `title`
 - etc.





Button bar

- Barre de boutons

- Positionnable un peu partout dans l'interface
- Ajouts de labels simple ou complexe (boutons stylisés)

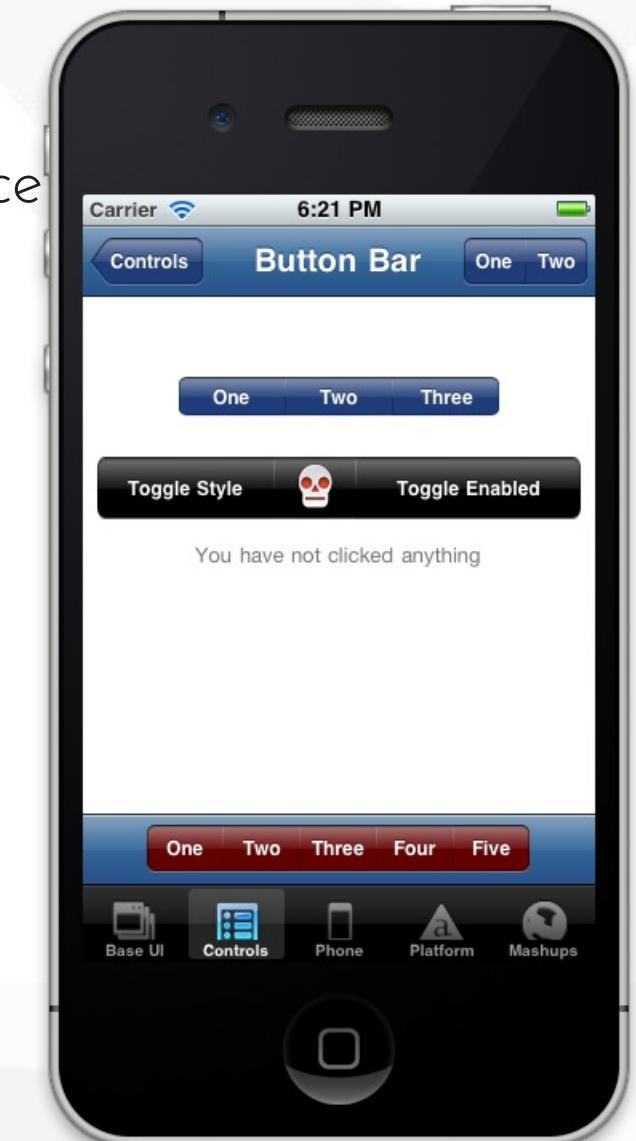
- Constructeur :

- Ti.UI.createButtonBar()

- API : rien de particulier

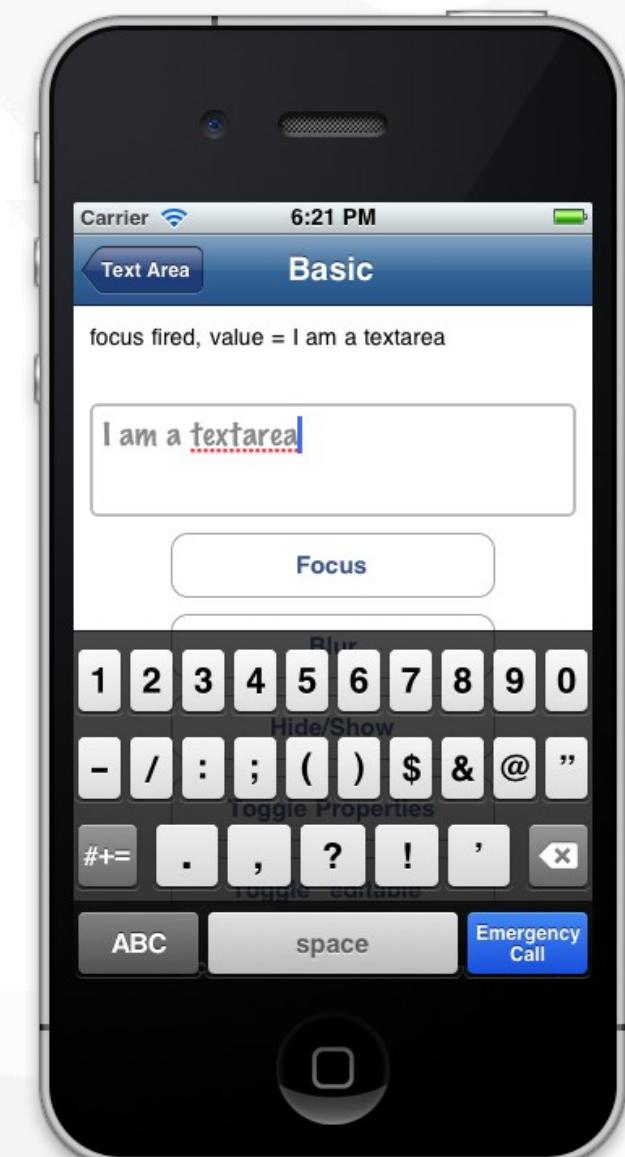
- Paramètres :

- labels
- style



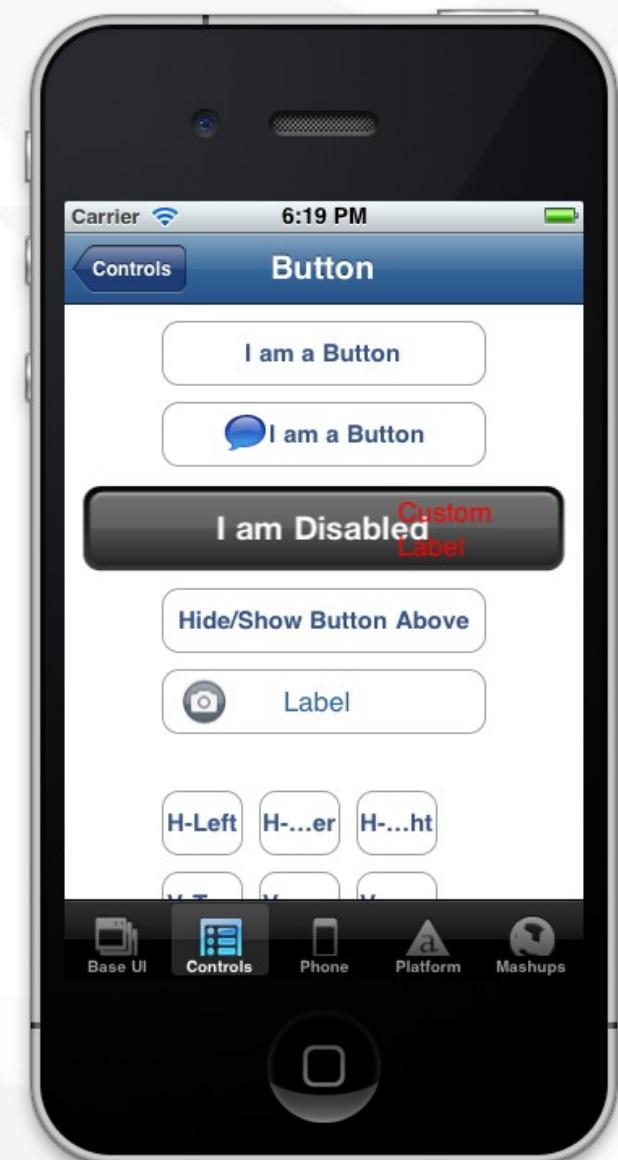


- Zone de saisie texte
 - Équivalent au textarea HTML
- Constructeur :
 - Ti.UI.createTextarea()
- API :
 - blur(), focus()
 - hasText()
- Paramètres :
 - autoLink
 - editable, enabled, focusable
 - value
 - etc.





- Bouton
 - Constructeur simple ou complexe
 - Traitement par évènements
- Constructeur :
 - Ti.UI.createButton()
- API : rien de particulier
- Paramètres :
 - enabled, focusable, visible
 - image
 - title
 - etc.





- Barre de boutons / outils
 - Positionnable en bas de fenêtre
 - Typiquement : boutons d'upload, de rechargement, etc.
- Constructeur :
 - `Ti.UI.createToolbar()`
- API : rien de particulier
- Paramètres : rien de particulier





- **Texte simple**

- Positionnable un peu partout dans l'interface
- Très utilisé pour personnaliser le look par défaut d'autres widgets

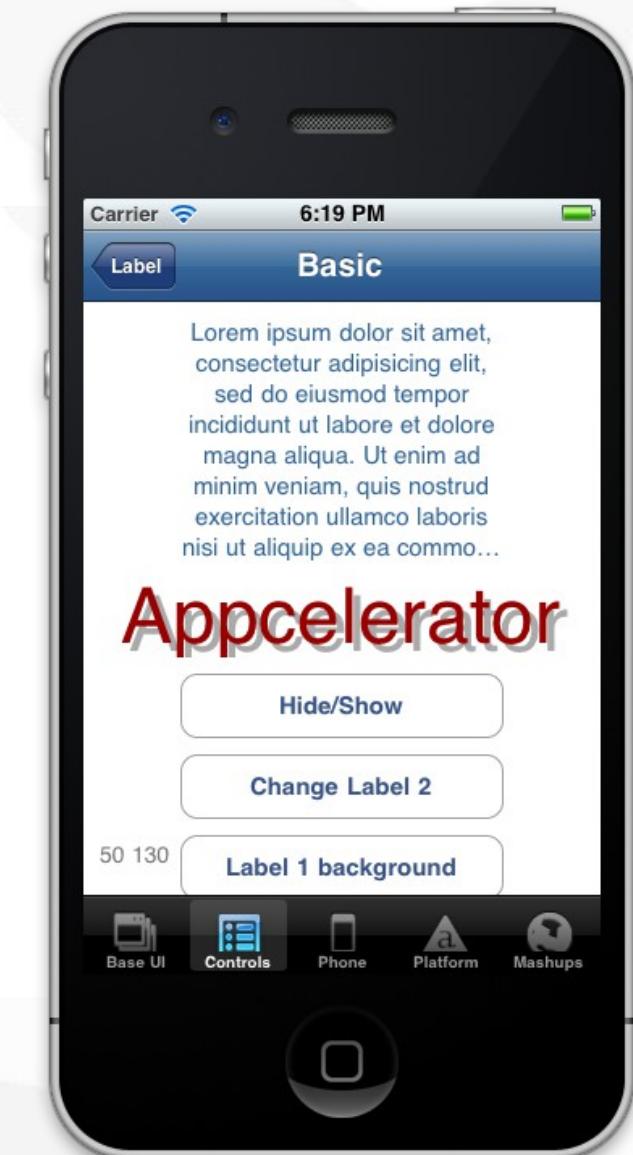
- **Constructeur :**

- `Ti.UI.createLabel()`

- API : rien de particulier

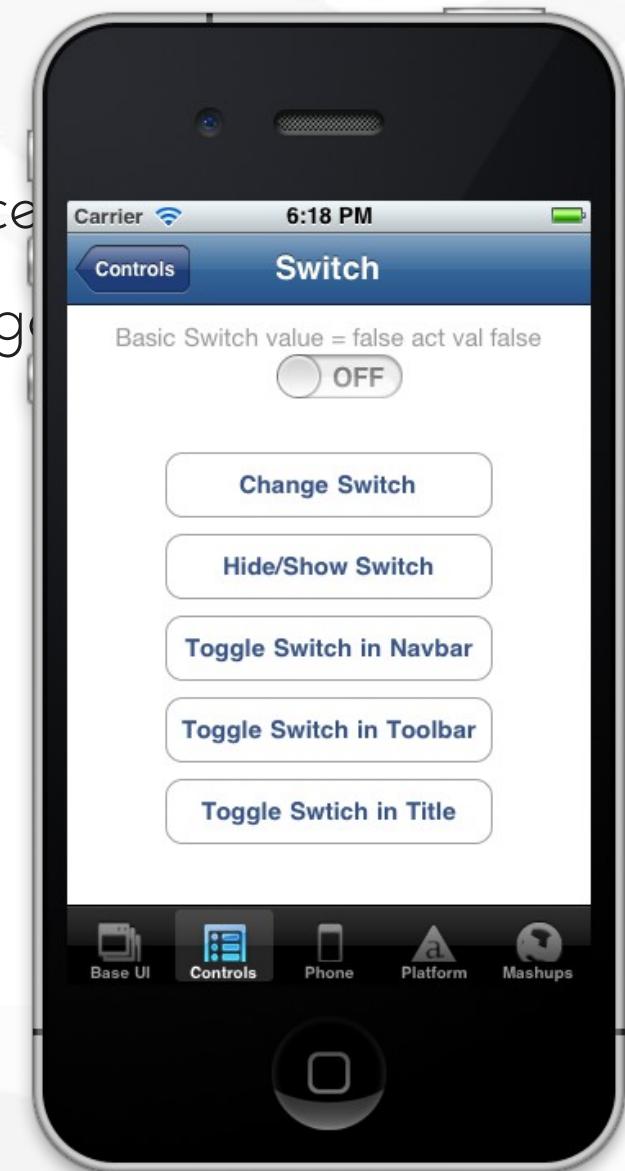
- **Paramètres :**

- plein ! cf. API





- Bouton à deux états
 - Positionnable un peu partout dans l'interface
 - Pratique pour activer / désactiver un réglage
- Constructeur :
 - Ti.UI.createSwitch()
- API : rien de particulier
- Paramètres remarquables :
 - title, titleOn, titleOff
 - value (true ou false)





- Barre de réglage précis
 - gradateur de réglage
 - Indication de pourcentages, ou de valeurs arbitraires
- Constructeur :
 - Ti.UI.createSlider()
- API : rien de particulier
- Paramètres :
 - min, max
 - leftTrackImage, rightTrackImage,
 - thumbImage,
 - value

