Temas de Hoy

Microservicios

Desarrollos Móviles

Microservicios: ¿+Cómo?

12 Factores

Los 12 Factores

“The twelve-factor app” es una **metodología** (quizás algo más) para construir aplicaciones SaaS que:

- Usan formatos **declarativos** para la automatización de la configuración, para minimizar el tiempo y el coste que supone que nuevos desarrolladores se unan al proyecto.

- Tienen un **contrato claro** con el sistema operativo sobre el que trabajan, ofreciendo la **máxima portabilidad** entre los diferentes entornos de ejecución.

Los 12 Factores

- Son apropiadas para **desplegarse** en modernas **plataformas en la nube**, obviando la necesidad de servidores y administración de sistemas.

- **Minimizan las diferencias** entre los entornos de desarrollo y producción, posibilitando un **despliegue continuo** para conseguir la máxima agilidad.

- Y pueden **escalar** sin cambios significativos para las herramientas, la arquitectura o las prácticas de desarrollo.

La metodología “twelve-factor” puede ser aplicada a aplicaciones escritas en cualquier lenguaje de programación, y cualquier combinación de ‘backing services’ (bases de datos, colas, memoria cache, etc).

Origen

Comunidad de desarrolladores, vinculados entre sí mediante la plataforma Heroku.

Heroku es uno de los **PaaS** más utilizados en la actualidad en entornos empresariales por su fuerte enfoque en resolver el despliegue de una aplicación.

Heroku es propiedad de SalesForce.com

12 Factores

I. Código base (Codebase)

Un código base sobre el que hacer el control de versiones y múltiples

Despliegues.

- Relación 1 a 1 con la aplicación.
- Relación 1 a * con los deploy.

12 Factores

II. Dependencias

Declarar y aislar explícitamente las dependencias

- Una aplicación “twelve-factor” no depende nunca de la existencia explícita de paquetes instalados en el sistema.

12 Factores

III. Configuraciones

Guardar la configuración en el entorno

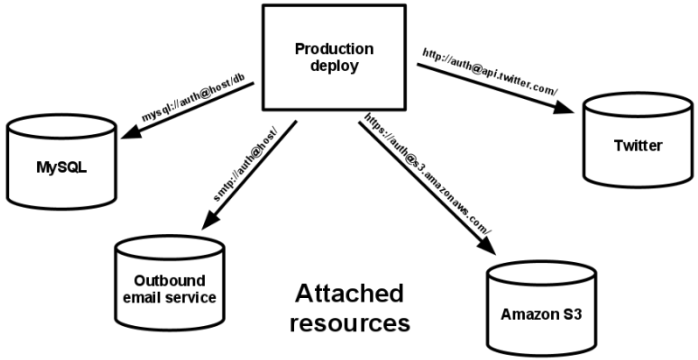
- **Las aplicaciones “twelve-factor” almacenan la configuración en *variables de entorno*** (abreviadas normalmente como *env vars* o *env*).

12 Factores

IV. Backing services

Tratar a los “backing services” como recursos conectables

- **El código de una aplicación “twelve-factor” no hace distinciones entre servicios locales y de terceros.** Para la aplicación, ambos son recursos conectados, accedidos mediante una URL u otro localizador o credencial almacenado en la config.



12 Factores

V. Construir, desplegar, ejecutar

Separar completamente la etapa de construcción de la etapa de ejecución.

- **Las aplicaciones “twelve-factor” hacen una separación completa de las fases de construcción, de distribución y de ejecución.**
- Por ejemplo, es imposible hacer cambios en el código en la fase de ejecución, porque no hay una manera de propagar dichos cambios a la fase de construcción.

12 Factores

VI. Procesos

Ejecutar la aplicación como uno o más procesos sin estado

- **Los procesos “twelve-factor” no tienen estado y son “share-nothing”.**
- Cualquier información que necesite persistencia se debe almacenar en un ‘backing service’ con estado, habitualmente una base de datos.

12 Factores

VII. Asignación de puertos

Publicar servicios mediante asignación de puertos

- **Las aplicaciones “twelve factor” son completamente auto-contenidas** y no dependen de un servidor web en ejecución par a crear un servicio web público.
- Ofrece la posibilidad de que una aplicación pueda llegar a ser un “backing service” para otra aplicación.

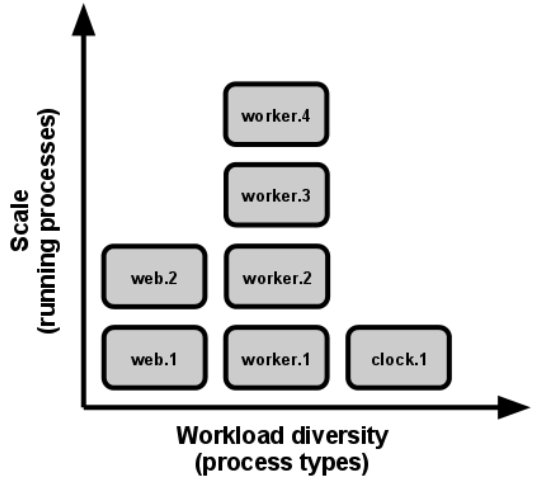
12 Factores

VIII. Concurrencia

Escalar mediante el modelo de procesos

- Los procesos son ciudadanos de primera clase.

- El desarrollador decide que tipo de proceso atiende cada petición.



12 Factores

IX. Desechabilidad

Hacer el sistema más robusto intentando conseguir inicios rápidos y finalizaciones seguras

- **Los procesos de las aplicaciones “twelve-factor” son *desechables*, lo que significa que pueden iniciarse o finalizarse en el momento que sea necesario.**
- Esto permite un escalado rápido y flexible, un despliegue rápido del código y de los cambios de las configuraciones, y despliegues más robustos en producción.

12 Factores

X. Paridad en desarrollo y producción

Mantener desarrollo, preproducción y producción tan parecidos como sea posible

- **Diferencias de tiempo**: Un desarrollador puede estar trabajando en un código durante días, semanas o incluso meses antes de que llegue a producción.
- **Diferencias de personal**: Los desarrolladores escriben el código y los ingenieros de operaciones lo despliegan.
- **Diferencias de herramientas**: Los desarrolladores pueden usar una pila como Nginx, SQLite y OS X, mientras que en producción se usa Apache, MySQL y Linux.

Las aplicaciones “twelve-factor” están diseñadas para hacer despliegues continuos que reducen las diferencias entre los entornos de desarrollo y producción.

12 Factores

XI. Historiales (Logs)

Tratar los historiales como una transmisión de eventos.

- Una aplicación “twelve-factor” nunca se preocupa del direccionamiento o el almacenamiento de sus transmisiones de salida .
- No debería intentar escribir o gestionar ficheros de historial.
- En su lugar, cada proceso en ejecución escribe sus eventos a la salida estándar (o stdout).
- Durante el desarrollo, los desarrolladores verán el flujo en su terminal para observar el comportamiento de la aplicación.

12 Factores

XII. Administración de procesos

Ejecutar las tareas de gestión/administración como procesos que solo se ejecutan una vez

- Ejecutar migraciones de las bases de datos (e.g. manage.py migrate de Django, rake db:migrate de Rails).
- Ejecutar una consola (también conocidas como REPL) para ejecutar código arbitrario o inspeccionar los modelos de la aplicación en una base de datos con datos reales. La mayoría de los lenguajes proporcionan un interprete del tipo REPL si se ejecuta el mismo mandato sin ningún argumento (e.g. python o perl) pero en algunos casos tienen un mandato distinto (e.g. irb en Ruby, rails console en Rails).
- Ejecutar scripts incluidos en el repositorio de la aplicación (e.g. php scripts/fix_bad_records.php).

Aplicaciones Móviles

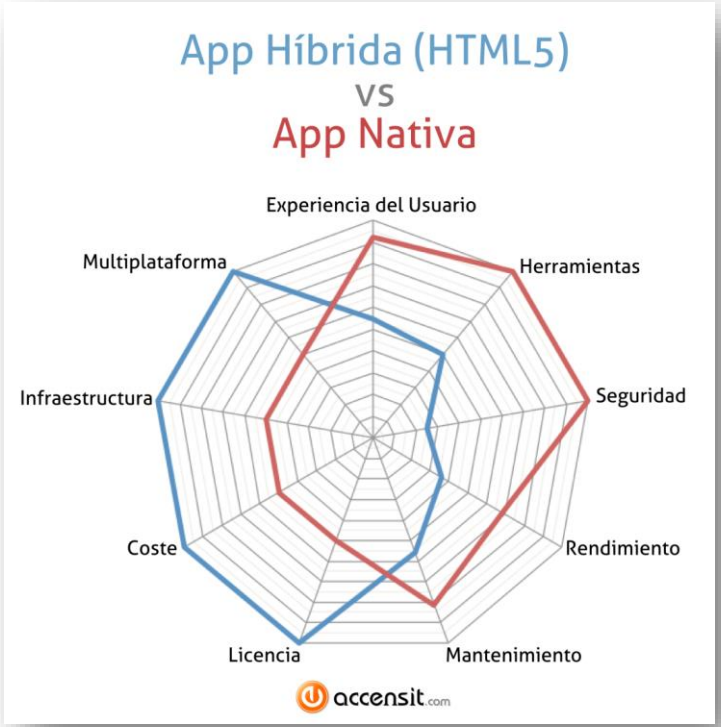
Tipos de Desarrollos Móviles

- **Nativas**
  - Este tipo de desarrollos son hechos de forma específica para un determinado sistema operativo.
- **Web**
  - Aplicaciones que corren sobre el navegador web de nuestro dispositivo móvil.
- **Híbridas**
  - Este tipo de aplicaciones son una combinación de ambas: **Nativas + Web**.

¿Y cuál conviene?

¿Y cuál conviene?

¿Y cuál conviene?

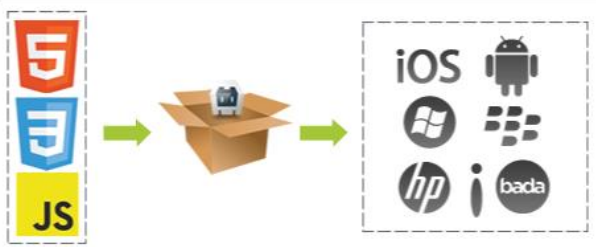


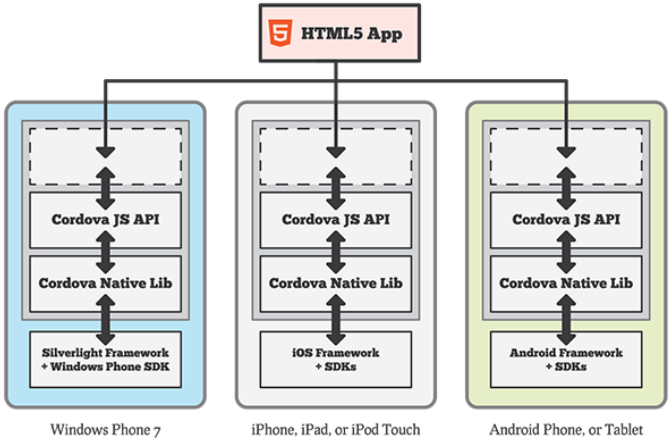
Comparación

	Nativa	Web App	Híbrida
App Features			
Gráficos	API Nativa	HTML, Canvas, SVG	HTML, Canvas, SVG
Rendimiento	Muy rápido	Lento	Rápido
Look & feel	Nativo	Emulado	Emulado
Distribución	Store	Web	Store
Acceso dispositivo			
Cámara	Sí	No	Sí
Notificaciones	Sí	No	Sí
Contactos, calendario	Sí	No	Sí
Almacenamiento Offline	Almacenamiento seguro de ficheros	El que permita el navegador: - Cache - IndexedDb - SQL compartida	Almacenamiento seguro de ficheros, SQL compartida.
Geolocalización	Sí	Sí	Sí
Gestos			
Swipe	Sí	Sí	Sí
Pinch, spread	Sí	No	Sí
Conectividad	Online y offline	Mayormente online	Online y offline
Conocimientos necesarios	ObjectiveC, Java	HTML5, CSS, Javascript	HTML5, CSS, Javascript

Desarrollos Móviles con HTML5

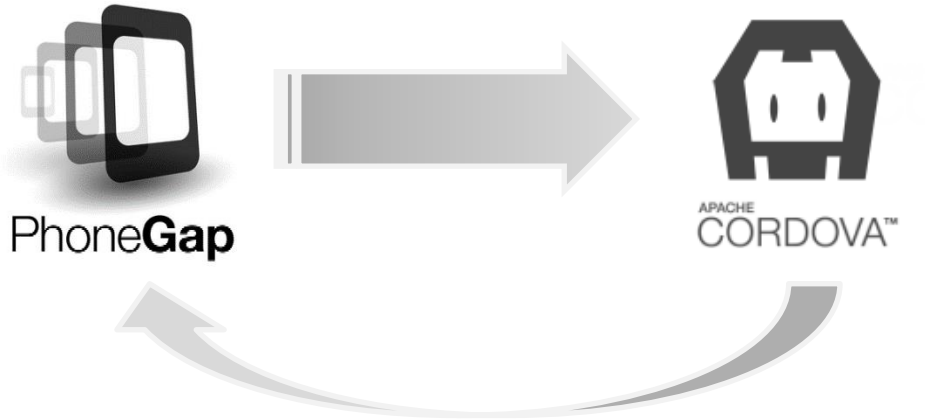
- **Apache Cordova** es una plataforma que permite construir aplicaciones nativas utilizando **HTML**, **CSS** y**JavaScript**



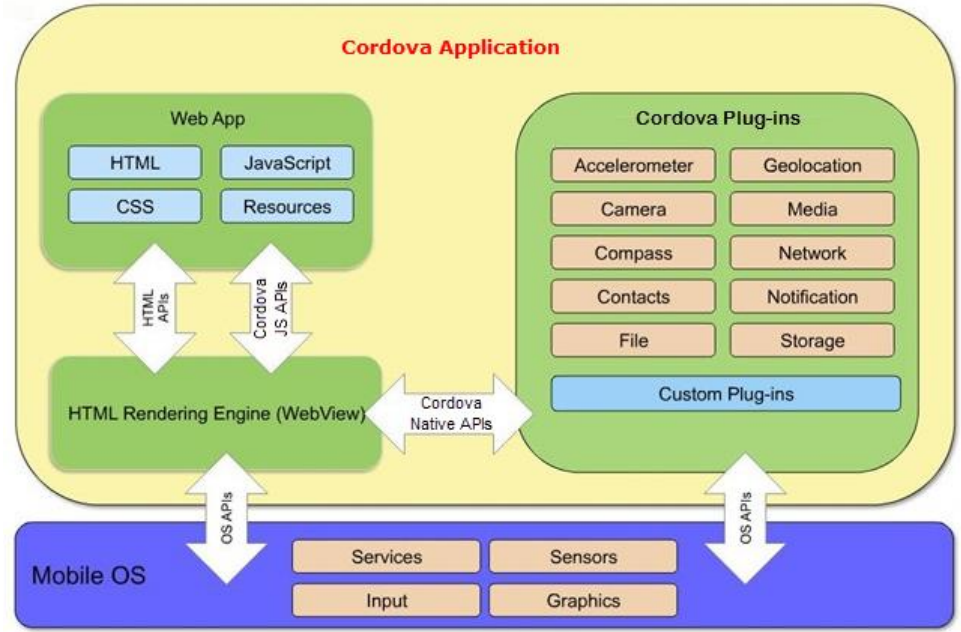


Historia

- Apache Cordova originalmente fue **Phonegap** construido por Nitobi
- Open-source & free software(MIT License), actualmente Apache License.
- Nitobi fue adquirida por Adobe. Entonces dono el código base de PhoneGap a la Apache Software Foundation (ASF)
- PhoneGap es actualmente un producto de Adobe. Es una **distribución de Apache Cordova**.

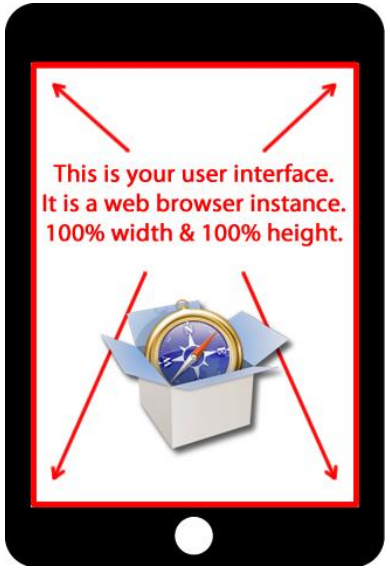


Arquitectura de Cordova



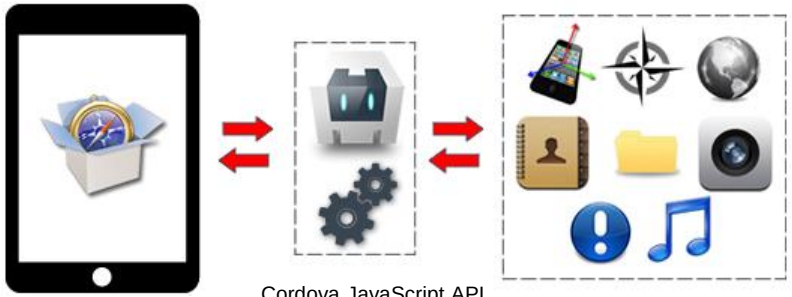
User Interface

- Se utiliza HTML, CSS, and JavaScript.
- El UI layer es un web browser en modo pantalla completa.
- La “web view” utilizada por la aplicación es la misma utilizada nativamente por el sistema operativo.
  - iOS: **Objective-C UIWebView class**
  - Android: **android.webkit.WebView**
  - WP7: **WebBrowser**
  - WP8: WebBrowser control (Internet Explorer 10)
  - BlackBerry: **WebWorks framework**

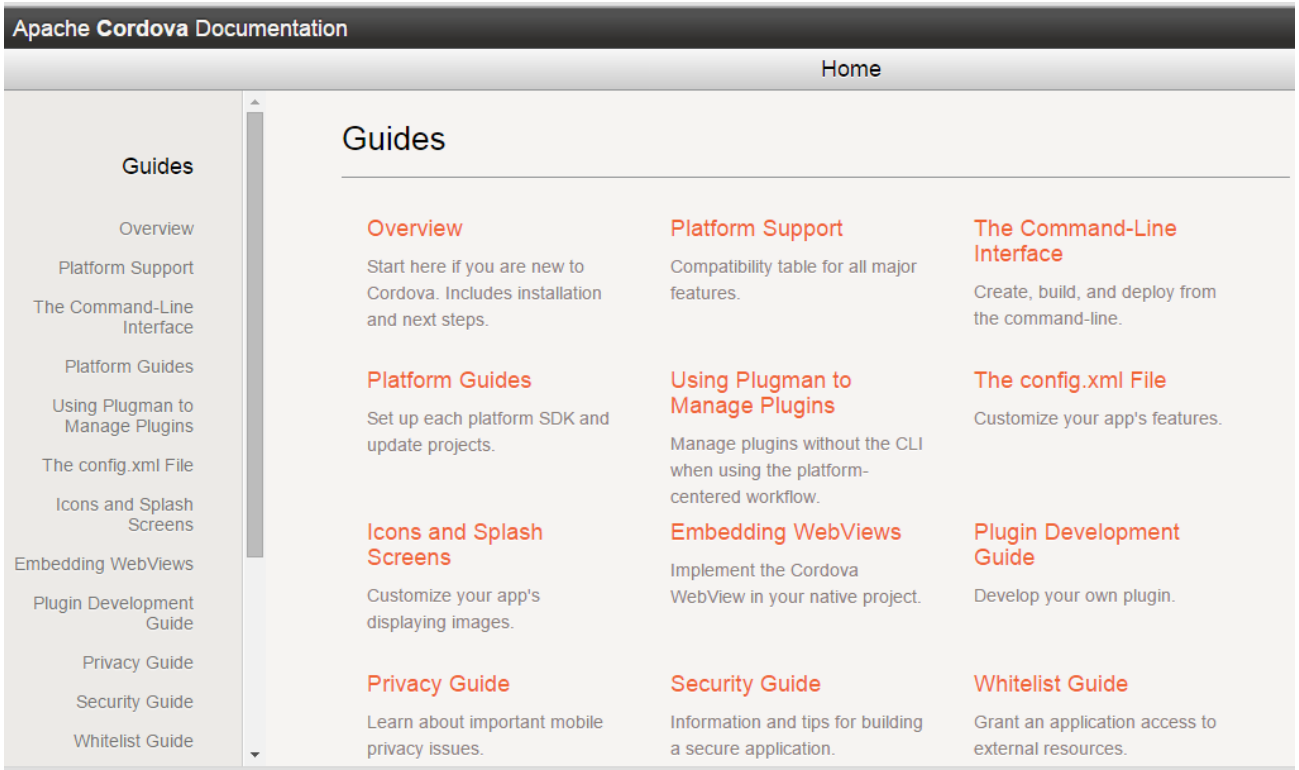


Apache Cordova API

- Provee una API
  - Permite **acceder a las funcionalidades nativas del dispositivo utilizando Javascript.**
  - APIs for Accelerometer, Camera, Compass, Media, FileSystem, etc.
  - Extendable using native plug-in
- docs.phonegap.com



De que se trata…



Camera

```
navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
    destinationType: Camera.DestinationType.DATA_URL
});

function onSuccess(imageData) {
    var image = document.getElementById('myImage');
    image.src = "" + imageData;
}

function onFail(message) {
    alert('Failed because: ' + message);
}
```

Acelerometro

```
function onSuccess(acceleration) {
    alert('Acceleration X: ' + acceleration.x + '\n' +
          'Acceleration Y: ' + acceleration.y + '\n' +
          'Acceleration Z: ' + acceleration.z + '\n' +
          'Timestamp: '      + acceleration.timestamp + '\n');
};

function onError() {
    alert('onError!');
};

navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
```

GPS

```
navigator.geolocation.getCurrentPosition(geolocationSuccess,
                                         [geolocationError],
                                         [geolocationOptions]);
```

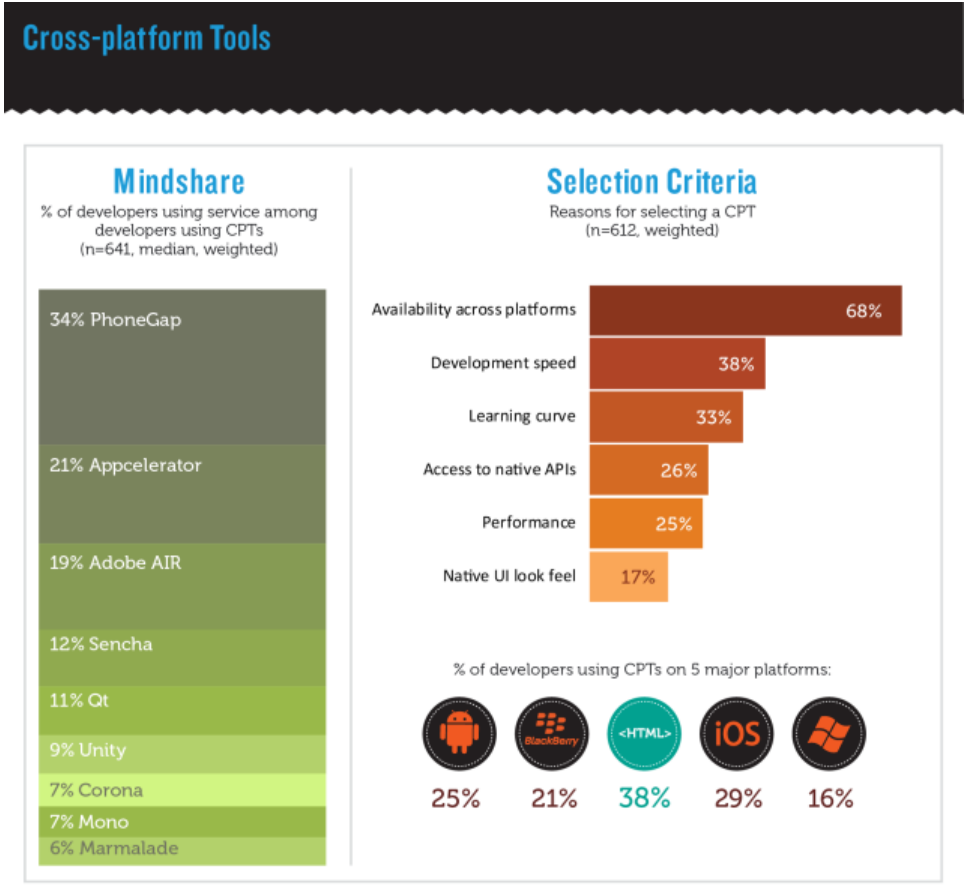
```
var onSuccess = function(position) {
    alert('Latitude: '          + position.coords.latitude          + '\n' +
          'Longitude: '         + position.coords.longitude         + '\n' +
          'Altitude: '          + position.coords.altitude          + '\n' +
          'Accuracy: '          + position.coords.accuracy          + '\n' +
          'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
          'Heading: '           + position.coords.heading           + '\n' +
          'Speed: '             + position.coords.speed             + '\n' +
          'Timestamp: '         + position.timestamp                + '\n');
};

// onError Callback receives a PositionError object
//
function onError(error) {
    alert('code: '    + error.code    + '\n' +
          'message: ' + error.message + '\n');
}
```

¿Alternativas?

- Por todos lados…
- http://mobile-frameworks-comparison-chart.com/

¿Cuál elegían en el 2013?



¿Preguntas?

¡Hasta la próxima Semana!