THE HEARTBEAT OF WOUNDED KNEE NATIVE AMERICA FROM Tutorial

The heartbeat of Wounded Knee Native America from is a phrase that resonates deeply within the history and consciousness of Indigenous peoples in North America. It symbolizes the enduring spirit, resilience, and ongoing struggle of Native Americans, especially in the face of historical trauma, cultural suppression, and modern challenges. Wounded Knee, as a site and symbol, encapsulates a pivotal chapter in Native American history ? one that continues to echo in the cultural, political, and social veins of Indigenous communities today. To understand this heartbeat, we must explore the origins of Wounded Knee, its significance in Native American history, the events that transpired there, and the ongoing legacy that sustains the spirit of Indigenous resistance and renewal.

The Historical Significance of Wounded Knee

Origins of the Site and Initial Conflicts

Wounded Knee is located on the Pine Ridge Indian Reservation in South Dakota. Originally, it was the site of a 1890 massacre where U.S. troops killed hundreds of Lakota men, women, and children during the Ghost Dance movement, a spiritual revival aimed at restoring Native lands and way of life. This tragic event marked a brutal end to a period of escalating tensions between Native nations and the U.S. government.

The Ghost Dance, which spread among various tribes, embodied hopes for a return to indigenous sovereignty and the disappearance of colonial oppression. The U.S. government perceived the movement as a potential uprising, leading to increased military presence and ultimately the massacre at Wounded Knee.

The Symbolic Weight of Wounded Knee in Native American History

Over time, Wounded Knee became a symbol of Native resistance, resilience, and the ongoing struggle for sovereignty. It represents a collective memory of trauma but also a rallying point for activism and cultural revival. The site?s significance extends beyond its tragic past; it embodies the fight for justice, recognition, and the affirmation of Native identity.

The 1890 Wounded Knee Massacre

Events Leading Up to the Massacre

In December 1890, U.S. troops intercepted a group of Lakota Sioux, including Sitting Bull, who was later killed during his arrest, and others participating in the Ghost Dance movement. Fearing an uprising, the military moved in to disarm the group at Wounded Knee Creek.

The Massacre and Its Aftermath

On December 29, 1890, the confrontation turned deadly. A shot was fired, whether accidental or intentional remains debated, and the soldiers opened fire on the unarmed Lakota, killing an estimated 150-300 men, women, and children. The massacre marked the end of the Indian Wars and is often viewed as a symbol of U.S. government brutality against Native peoples.

The aftermath saw a deepening of Native grievances and a renewed sense of injustice that persists through generations.

The Rise of Native Activism and the Legacy of Wounded Knee

The 20th Century: Civil Rights and Indigenous Movements

Following the massacre, Native Americans faced systemic oppression, forced assimilation policies, and marginalization. However, the 20th century witnessed a resurgence of activism:

- 1934: The Indian Reorganization Act aimed to restore some tribal self-governance.
- 1960s-70s: The Red Power movement, inspired by Black civil rights activism, emerged, emphasizing sovereignty and cultural revival.
- 1972: The occupation of Alcatraz Island and the American Indian Movement (AIM) intensified efforts to address injustices.

The 1973 Occupation of Wounded Knee

One of the most significant events in modern Native activism was the 1973 armed occupation of Wounded Knee by AIM members. They sought to protest federal government policies, address treaty rights violations, and oppose corruption within the Bureau of Indian Affairs.

The occupation lasted 71 days, attracting national and international attention. It became a symbol of Native resistance and a call for sovereignty, justice, and cultural preservation.

Impact and Legacy of the 1973 Stand

The Wounded Knee occupation galvanized Indigenous activism, leading to increased awareness of Native issues. It also sparked debates on sovereignty, treaty rights, and the treatment of Native

peoples under U.S. law.

The event solidified Wounded Knee?s place in American history as a symbol of resistance and resilience, inspiring subsequent generations to continue the fight for Indigenous rights.

The Cultural and Spiritual Significance of Wounded Knee

Reclaiming Identity and Tradition

For many Native communities, Wounded Knee is a sacred site representing more than historical trauma; it is a place of spiritual renewal and cultural reclamation. Indigenous leaders and activists often invoke the site to honor ancestors and reaffirm their connection to land and tradition.

The Role of Art, Music, and Rituals

Cultural expressions serve as vital tools for healing and resistance:

- Storytelling and oral histories keep the memory of Wounded Knee alive.
- Music and dance, including traditional songs, serve as acts of defiance and cultural preservation.
- Rituals and ceremonies reaffirm spiritual ties and resilience.

Reconciliation and Healing Initiatives

Efforts have been made to promote healing:

- Memorial services and commemorations at Wounded Knee.
- Educational programs to teach about the site?s history.
- Initiatives to restore and preserve sacred lands.

These acts help sustain the heartbeat of Native America, fostering resilience and cultural vitality.

Contemporary Challenges and the Ongoing Heartbeat

Legal and Political Struggles

Native communities continue to face challenges related to sovereignty, land rights, and resource control. Issues such as pipeline construction near sacred sites, land dispossession, and treaty violations remain pressing concerns.

Environmental and Social Justice

Environmental degradation and social disparities impact Native communities' health, education, and economic stability. Activism around issues like climate change and environmental justice is rooted in the legacy of resistance symbolized by Wounded Knee.

Revitalization Movements

Modern movements focus on:

- Language revitalization programs.
- Land reclamation efforts.
- Cultural education and youth engagement.

These initiatives keep the heartbeat of Wounded Knee alive, ensuring that Indigenous voices and traditions thrive.

The Future of Native America and the Spirit of Wounded Knee

Continuing the Legacy of Resistance and Resilience

The spirit of Wounded Knee endures through the activism, cultural revival, and spiritual strength of Native peoples. It reminds us that the struggle for justice, recognition, and self-determination is ongoing.

Building Bridges and Fostering Understanding

Education and dialogue are crucial for fostering respect and reconciliation. Recognizing the historical significance of Wounded Knee and its symbolism helps promote awareness and allies in the fight for Indigenous rights.

Hope and Renewal

Despite historical trauma and ongoing challenges, the heartbeat of Wounded Knee is a testament to resilience. It beats strongly in the cultural practices, activism, and dreams of Native communities striving for a just and equitable future.

In conclusion, the heartbeat of Wounded Knee Native America from its tragic origins to its role as a symbol of resistance and renewal encapsulates the enduring spirit of Indigenous peoples. It reminds us of the importance of remembering history, honoring culture, and supporting ongoing struggles for

justice. Wounded Knee is not merely a site of past trauma but a living symbol of resilience that continues to inspire and energize Native communities across generations.

The Heartbeat of Wounded Knee Native America: An In-Depth Exploration

Introduction: Understanding the Significance of Wounded Knee in Native American History

The phrase "Wounded Knee" resonates profoundly within the collective consciousness of Native American communities and beyond. It encapsulates a pivotal moment in U.S. history?a tragic event symbolizing resistance, trauma, resilience, and the ongoing quest for justice. To truly grasp the heartbeat of Wounded Knee in the context of Native America, we must journey through its historical roots, cultural significance, and contemporary implications, exploring how this event continues to influence Native identities and movements today.

The Historical Context of Wounded Knee

Origins of the Wounded Knee Massacre

The massacre at Wounded Knee occurred on December 29, 1890, on the Pine Ridge Reservation in South Dakota. It was the culmination of decades of conflict between Native tribes and U.S. government policies aimed at assimilating or eradicating indigenous cultures.

Key factors leading to the massacre include:

- The decline of the Sioux "Ghost Dance" movement: A spiritual revival that promised renewal and hope for Native communities threatened by displacement and cultural suppression.
- U.S. military interventions: The military sought to suppress the Ghost Dance movement, viewing it as a threat to control.
- Tensions over land and sovereignty: The ongoing loss of land, broken treatie

QuestionAnswer What is the significance of 'The Heartbeat of Wounded Knee' in Native American history? 'The Heartbeat of Wounded Knee' is a documentary film that explores the history, struggles, and resilience of Native American communities, especially focusing on events like the Wounded Knee Massacre and contemporary issues faced by Indigenous peoples. Who directed 'The Heartbeat of Wounded Knee'? The documentary was directed by Steve Hinshaw and Jack Pettibone Riccobono, aiming to shed light on Native American experiences and activism. How does 'The Heartbeat of Wounded Knee' portray Native American activism? The film highlights Native activism by documenting protests, movements for sovereignty, and efforts to preserve cultural identity, emphasizing the ongoing fight for Native rights. What historical events are covered in 'The Heartbeat of Wounded Knee'? The documentary covers key events such as the Wounded Knee

Massacre of 1890, the American Indian Movement's activities, and contemporary issues like land rights and cultural preservation. Why is Wounded Knee a symbol in Native American history? Wounded Knee symbolizes both a tragic massacre in 1890 and a site of resistance and activism, representing the resilience and ongoing struggles of Native peoples. How has 'The Heartbeat of Wounded Knee' impacted awareness about Native American issues? The film has increased awareness by educating audiences on Native history, contemporary challenges, and the importance of Indigenous sovereignty and cultural preservation. In what ways does 'The Heartbeat of Wounded Knee' address cultural identity? The documentary emphasizes the importance of cultural traditions, language, and spirituality in maintaining Native identity amidst historical and ongoing challenges. What role does the American Indian Movement play in the documentary? The film features the American Indian Movement as a pivotal force in Native activism, highlighting their efforts to achieve sovereignty, rights, and recognition. How can viewers learn more about Native American history through 'The Heartbeat of Wounded Knee'? The documentary serves as an educational tool, providing historical context and personal stories that deepen understanding of Native American experiences and resilience. Where can I watch 'The Heartbeat of Wounded Knee'? The film is available on various streaming platforms, including official websites, educational platforms, or through purchase or rental on major digital services like Amazon Prime or iTunes.

Related keywords: Native American history, Wounded Knee massacre, Sioux Nation, American Indian movement, indigenous rights, Native American activism, Sitting Bull, Lakota Sioux, 1890 massacre, Native American sovereignty

REACT NATIVE BY EXAMPLE NATIVE MOBILE DEVELOPMENT

React Native by Example Native Mobile Development

In the rapidly evolving world of mobile app development, choosing the right framework is crucial for delivering high-quality, performant, and maintainable applications. **React Native by example native mobile development** has emerged as a popular choice among developers seeking to build cross-platform apps with a near-native user experience. This article explores React Native through practical examples, highlighting its core concepts, best practices, and real-world applications to help you harness its full potential.

What Is React Native?

React Native is an open-source framework developed by Facebook that enables developers to build mobile applications using JavaScript and React. Instead of writing separate codebases for iOS and Android, React Native allows you to write a single codebase that compiles into native components

for both platforms.

Key Features of React Native

- Cross-Platform Development: Write once, run anywhere on iOS and Android.
- Native Performance: Uses native components, ensuring smooth animations and interactions.
- Hot Reloading: Instantly see changes without rebuilding the app.
- Large Community and Ecosystem: Extensive libraries, tools, and community support.

Getting Started with React Native

Before diving into examples, ensure you have the necessary environment set up.

Prerequisites

- Node.js and npm installed
- React Native CLI installed globally (`npm install -g react-native-cli`)
- Xcode (for iOS development)
- Android Studio (for Android development)

Creating Your First React Native App

```bash

npx react-native init MyFirstApp

cd MyFirstApp

npx react-native run-ios For iOS

npx react-native run-android For Android

```

This scaffolds a new project and launches the app on your emulator or device.

Core Concepts of React Native

Understanding React Native's core concepts is essential for effective development.

Components

React Native apps are built with components, which are reusable building blocks.

State and Props

- Props: Input parameters passed to components.
- State: Internal data that can change over time, affecting rendering.

Styling

React Native uses JavaScript objects for styling, similar to CSS but with some differences.

React Native by Example: Building a Simple App

Let's walk through a practical example: creating a simple counter app.

Step 1: Create the Counter Component

```jsx
import React, { useState } from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';
const Counter = () => {
const [count, setCount] = useState(0);
return (
Count: {count}
 setCount(count + 1)} />
 setCount(count - 1)} />
);
};
```

```
const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

},

counterText: {

fontSize: 24,

marginBottom: 20,

},

});

export default Counter;
```

Step 2: Include Counter in App.js

```jsx
import React from 'react';

import { SafeAreaView } from 'react-native';

import Counter from './Counter';

const App = () => (

);

export default App;
```

```

This example demonstrates state management, component structure, and styling.

## Handling Navigation in React Native

Most apps require navigation between screens. React Native provides several libraries, with React Navigation being the most popular.

Setting Up React Navigation

```bash

npm install @react-navigation/native

npm install react-native-screens react-native-safe-area-context

npm install @react-navigation/native-stack

```

Example: Basic Stack Navigation

```jsx

import React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createNativeStackNavigator } from '@react-navigation/native-stack';

import HomeScreen from './HomeScreen';

import DetailsScreen from './DetailsScreen';

const Stack = createNativeStackNavigator();

const App = () => (

);

export default App;

```

This setup creates a simple navigation flow between two screens.

Working with Native Modules

React Native allows integration with native code for functionalities not available in JavaScript.

Using Native Modules

- Linking Native Libraries: Use `react-native link` or auto-linking.
- Creating Custom Native Modules: Write platform-specific code in Java or Swift/Objective-C.

Example: Accessing Device Camera

Libraries like `react-native-camera` simplify camera access.

```bash

npm install react-native-camera

```

```jsx

import { RNCamera } from 'react-native-camera';

const CameraScreen = () => (

style={{ flex: 1 }}

type={RNCamera.Constants.Type.back}

/>

);

```

This example highlights how to incorporate native device features.

State Management in React Native

Managing complex state is vital for scalable apps.

Popular State Management Libraries

- Redux: Predictable state container.
- MobX: Simple, scalable reactive state management.
- Context API: Built-in React solution for smaller apps.

Example: Using Redux

```bash

npm install redux react-redux

```

Create a store, define actions, reducers, and connect components accordingly.

Deploying React Native Apps

Once your app is ready, deployment involves building release versions for both platforms.

Building for iOS

- Use Xcode's Archive feature.
- Upload to App Store via Application Loader or Transporter.

Building for Android

```bash

cd android

./gradlew assembleRelease

```

Generate APK or App Bundle for distribution via Google Play.

Best Practices for React Native Development

To ensure your React Native projects are maintainable and performant, follow these best practices:

- Component Reusability: Break UI into small, reusable components.
- Optimize List Rendering: Use `FlatList` and `SectionList` for large data sets.
- Manage State Efficiently: Keep state local where possible; lift up or use global stores as needed.
- Use Native Modules Wisely: Only when necessary, to avoid performance bottlenecks.
- Test Thoroughly: Write unit and integration tests.
- Keep Dependencies Updated: Regularly update libraries to benefit from bug fixes and improvements.

Popular Libraries and Tools in React Native Ecosystem

Enhance your development process with these libraries:

- React Navigation: Navigation management.
- Axios: HTTP client for API calls.
- React Native Paper: UI components following Material Design.
- Redux Toolkit: Simplifies Redux setup.
- Detox: End-to-end testing framework.

Real-World Applications of React Native

React Native powers many successful apps across various industries:

Examples of Companies Using React Native

- Facebook: Initial creator of React Native.
- Instagram: React Native parts for certain features.
- Tesla: Mobile app built with React Native.
- Shopify: Cross-platform mobile app.
```

- Walmart: Rewritten using React Native for better performance.

Benefits Observed

- Faster development cycles.
- Code sharing across platforms.
- Access to native device features.
- Improved user experience with smooth animations.

Future of React Native

React Native continues to evolve with improvements like:

- Fabric Renderer: For better performance and interoperability.
- Turbo Modules: Lazy loading native modules.
- React Native for Windows + macOS: Extending to desktop applications.
- Better Developer Tooling: Enhanced debugging and profiling.

Staying updated with the React Native ecosystem ensures your skills remain relevant and your apps competitive.

Conclusion

React Native by example native mobile development offers a compelling solution for developers aiming to build high-performance, cross-platform apps efficiently. By leveraging React Native?s component-based architecture, native modules, and rich ecosystem, you can create feature-rich applications that deliver seamless user experiences on both iOS and Android.

Whether you're starting with simple components or managing complex navigation and native integrations, React Native provides the tools and flexibility needed. Embrace best practices, explore community resources, and continuously experiment with real-world examples to master React Native development.

Embark on your React Native journey today and transform your ideas into native-quality mobile applications faster and smarter than ever before.

React Native by Example: Native Mobile Development

In the rapidly evolving landscape of mobile application development, the pursuit of efficiency,

performance, and user experience has driven developers to explore diverse frameworks and tools. Among these, React Native by Example has emerged as a compelling approach, bridging the gap between web development and native mobile app creation. This investigative review delves into the intricacies of React Native, examining its architecture, advantages, limitations, and practical applications through illustrative examples, positioning it as a pivotal technology in the native mobile development arena.

Understanding React Native: An Overview

React Native, developed by Facebook and released in 2015, is an open-source framework that enables developers to build mobile applications using JavaScript and React. Unlike traditional native development, which requires platform-specific languages like Swift or Kotlin, React Native allows for a shared codebase across iOS and Android, significantly reducing development time and effort.

Core Principles of React Native

- JavaScript & React: Utilizes JavaScript and React components to define UI and logic.
- Native Components: Translates React components into native UI elements, ensuring performance and look-and-feel consistency.
- Bridge Architecture: Facilitates communication between JavaScript code and native modules via a bridge, enabling complex functionalities.

React Native by Example: Practical Insights into Native Development

To truly understand React Native?s potential, examining concrete examples of native development tasks implemented via React Native is essential. These examples highlight how React Native simplifies complex native functionalities while maintaining high performance.

1. Building a Basic UI: Cross-Platform Button

Native Approach:

- iOS (Swift): Using UIButton and constraints.
- Android (Kotlin): Using Button and layout parameters.

React Native Example:

```javascript
```

```
import React from 'react';

import { Button, View, StyleSheet } from 'react-native';

const App = () => (

 alert('Button Pressed!')} />

);

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

},

});

export default App;
```

Insight: A single React Native component creates a native button on both iOS and Android, reducing platform-specific code.

2. Accessing Device Hardware: Camera Integration

Native Approach:

- iOS: Using AVFoundation framework.
- Android: Using Camera2 API.

React Native Example:

Using a third-party library like `react-native-camera` simplifies hardware access.

```javascript
import React, { useRef } from 'react';
import { View, StyleSheet } from 'react-native';
import { RNCamera } from 'react-native-camera';
const CameraScreen = () => {
const cameraRef = useRef(null);
const takePicture = async () => {
if (cameraRef.current) {
const options = { quality: 0.5, base64: true };
const data = await cameraRef.current.takePictureAsync(options);
console.log(data.uri);
}
};
return (

ref={cameraRef}
style={styles.preview}
type={RNCamera.Constants.Type.back}
captureAudio={false}
/>
```

```
);

};

const styles = StyleSheet.create({

container: { flex: 1 },

preview: { flex: 1, justifyContent: 'flex-end', alignItems: 'center' },

});

export default CameraScreen;
```

Insight: React Native?s ecosystem allows easy integration with native modules, abstracting complex platform-specific code.

3. Handling Background Processes: Push Notifications

Native Approach:

- Implemented via platform-specific SDKs and background services.

React Native Example:

Using `react-native-push-notification` to manage notifications.

```javascript
import PushNotification from 'react-native-push-notification';

// Configure notifications

PushNotification.configure({

onNotification: function (notification) {
```

```
console.log("Notification received:", notification);

},

requestPermissions: true,

});

// Send a local notification

PushNotification.localNotification({

title: "Hello",

message: "This is a test notification",

});
```

Insight: React Native offers streamlined APIs and community packages for background and notification functionalities that typically require native coding.

Advantages of React Native in Native Mobile Development

- Cross-Platform Compatibility

- Single codebase for iOS and Android.
- Consistent UI across platforms with platform-specific customizations when necessary.

- Accelerated Development Cycle

- Faster prototyping and iteration.
- Hot Reload feature for real-time updates without rebuilding.

- Large Ecosystem and Community Support

- Extensive libraries and third-party modules.
- Active community forums and documentation.

- Cost-Effectiveness

- Reduced development and maintenance costs.
- Easier onboarding for web developers familiar with React.

- Near-Native Performance

- Native components for UI elements.
- Bridge architecture enables performance-intensive features.

Limitations and Challenges of React Native

While React Native offers numerous benefits, it also presents some limitations that merit critical assessment.

- Performance Bottlenecks

- For highly complex or graphics-intensive apps (e.g., games, AR), native development may outperform React Native.

- Native Module Dependency

- Access to new or advanced native features often requires writing custom native modules, which reintroduces native coding.

- Platform Discrepancies

- Despite the shared codebase, platform-specific behaviors can cause inconsistencies requiring

additional handling.

- Fragmentation of Ecosystem

- Variability in third-party libraries? quality and maintenance status.

- Debugging Complexity

- Debugging across the JavaScript bridge can be challenging, especially for native crashes.

React Native in Practice: Case Studies and Industry Adoption

Case Study 1: Facebook

- Original developer of React Native, uses it extensively for internal apps and some features.

Case Study 2: Instagram

- Incorporated React Native into existing native apps to accelerate development.

Case Study 3: Airbnb (formerly)

- Used React Native for their app, but later transitioned away due to performance issues and ecosystem limitations.

Industry Trends:

- Growing adoption in startups and mid-sized companies.
- Larger enterprises are cautious, often adopting React Native selectively or as part of a hybrid approach.

The Future of React Native in Native Mobile Development

React Native continues to evolve, with key developments including:

- Fabric Renderer: Enhances performance and UI responsiveness.
- TurboModules: Improves communication speed between JavaScript and native modules.
- Community-driven Innovations: New libraries and integrations expanding capabilities.

As the ecosystem matures, React Native?s role as a bridge between web and native development is poised to strengthen, making it an increasingly vital tool in the modern developer?s arsenal.

Conclusion: Is React Native the Right Choice for Native Mobile Development?

React Native by Example demonstrates that it is a powerful framework capable of producing high-quality, performant mobile applications with a shared codebase. Its ability to simplify cross-platform development, coupled with a vibrant ecosystem, makes it an attractive choice for many projects. However, developers must weigh its limitations against project requirements, especially where native performance and platform-specific features are critical.

In summary, React Native offers a pragmatic compromise?balancing native-like performance with development efficiency. Its ongoing evolution signals a promising future, making it a compelling option for organizations aiming to deliver robust mobile experiences without the overhead of maintaining separate native codebases.

Final Thoughts

As mobile applications become increasingly complex and user expectations rise, frameworks like React Native exemplify the innovative approaches shaping native mobile development. By leveraging example-driven insights, this review underscores the importance of understanding both its capabilities and constraints?empowering developers and organizations to make informed decisions in their mobile strategy.

QuestionAnswer What are the main advantages of using React Native for native mobile app development? React Native allows developers to build cross-platform mobile apps using JavaScript and React, enabling code reuse across iOS and Android. It offers near-native performance, a large community, reusable components, hot-reloading for faster development, and access to native device features through a rich set of APIs. How does React Native facilitate native module integration for advanced functionalities? React Native provides a bridge that allows developers to write native modules in Java, Objective-C, or Swift, which can then be invoked from JavaScript. This enables integration of platform-specific features and performance-critical code, ensuring seamless native functionality within the React Native app. Can you give an example of how to create a simple React

Native component? Certainly! Here's a basic example: ```jsx import React from 'react'; import { View, Text, StyleSheet } from 'react-native'; const HelloWorld = () => (Hello, React Native!); const styles = StyleSheet.create({ container: { flex: 1, justifyContent: 'center', alignItems: 'center', }, text: { fontSize: 20, fontWeight: 'bold', }, }); export default HelloWorld; ``` What are some common challenges developers face when using React Native for native mobile development? Common challenges include managing native dependencies and modules, handling performance issues with complex or large apps, ensuring consistent UI across different devices and OS versions, debugging native code, and keeping up with rapid updates to React Native and related libraries. How does React Native compare to other cross-platform frameworks like Flutter or Xamarin? React Native uses JavaScript and React, making it accessible for web developers and offering a large ecosystem. Flutter, using Dart, provides highly performant, visually rich UIs with a single codebase, but has a smaller community. Xamarin, based on C, integrates well with Microsoft tools but may have a steeper learning curve for non-.NET developers. The choice depends on project requirements, developer expertise, and desired performance and UI fidelity.

Related keywords: React Native, mobile app development, cross-platform development, JavaScript mobile apps, native mobile apps, React Native tutorials, mobile development examples, React Native components, native mobile UI, mobile app coding

Read the full article online: REACT NATIVE BY EXAMPLE NATIVE MOBILE DEVELOPMENT