Les Balises en React Native

En **React Native**, les **"balises"** sont en réalité des **composants** JavaScript qui servent à structurer et à afficher l'interface utilisateur. Contrairement à HTML qui utilise des balises comme `<div>`, `<span>`, `<p>`, etc., React Native utilise des **composants React** qui sont écrits en JavaScript et qui ont des noms spécifiques. Ces composants sont utilisés pour créer l'interface mobile, que ce soit pour la mise en page, l'affichage du texte, la gestion des entrées, etc.

Voici un **cours détaillé** sur les principaux **composants de base en React Native**, que l'on peut assimiler à des "balises" :

---

1. Composant `View`

Le composant **`View`** est l'équivalent d'un conteneur dans React Native. Il sert à encapsuler d'autres composants et à organiser l'interface utilisateur. Il fonctionne comme une balise `<div>` en HTML.

Exemple d'utilisation de `View` :

```
import React from 'react';
import { View, Text } from 'react-native';

const App = () => {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center'
}}>
      <Text>Bienvenue dans React Native !</Text>
    </View>
  );
};

export default App;
```

Explication :

- **`View`** : Crée un conteneur pour d'autres composants.
- **`style`** : Utilisé pour appliquer des styles, comme `flex`, `justifyContent`, `alignItems` pour la disposition des éléments.

2. Composant `Text`

Le composant **`Text`** est utilisé pour afficher du texte dans React Native. Contrairement aux balises HTML comme `<p>` ou `<span>`, vous devez utiliser **`Text`** pour toute gestion de texte.

Exemple d'utilisation de `Text` :

```
import React from 'react';
import { Text, View } from 'react-native';
```

```
const App = () => {
  return (
    <View>
      <Text style={{ fontSize: 20, color: 'blue' }}>Hello, React
Native!</Text>
    </View>
  );
};

export default App;
```

Explication :

- **`Text`** : Utilisé pour afficher du texte.
- Vous pouvez appliquer des styles comme **`fontSize`**, **`color`**, etc., pour modifier l'apparence du texte.

3. Composant `Image`

Le composant **`Image`** permet d'afficher des images dans l'application. Vous pouvez soit charger des images locales, soit des images externes à partir d'une URL.

Exemple d'utilisation de `Image` :

```
import React from 'react';
import { Image, View } from 'react-native';

const App = () => {
  return (
    <View>
      <Image
        source={require('./assets/logo.png')} // Image locale
        style={{ width: 200, height: 200 }}
      />
    </View>
  );
};

export default App;
```

Explication :

- **`Image`** : Affiche une image.
- **`source`** : Définit la source de l'image, soit une image locale avec `require`, soit une URL distante.
- **`style`** : Applique des dimensions ou autres styles (comme `resizeMode`, `borderRadius`).

4. Composant `TextInput`

Le composant **`TextInput`** permet de capturer des entrées de texte de l'utilisateur, comme un champ de formulaire dans une application web.

Exemple d'utilisation de `TextInput` :

```
import React, { useState } from 'react';
import { TextInput, View, Text } from 'react-native';

const App = () => {
  const [text, setText] = useState('');

  return (
    <View>
      <TextInput
        style={{ height: 40, borderColor: 'gray', borderWidth: 1 }}
        placeholder="Entrez du texte"
        onChangeText={(newText) => setText(newText)}
        value={text}
      />
      <Text>Vous avez écrit : {text}</Text>
    </View>
  );
};

export default App;
```

Explication :

- **`TextInput`** : Crée un champ de texte pour saisir des données.
- **`onChangeText`** : Permet de capturer la saisie de l'utilisateur et de la mettre à jour dans l'état avec **`useState`**.
- **`value`** : Lier la valeur du champ au state.

5. Composant `Button`

Le composant **`Button`** permet d'afficher un bouton sur lequel l'utilisateur peut cliquer pour effectuer une action. Ce bouton est similaire aux boutons HTML `<button>`.

Exemple d'utilisation de `Button` :

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

const App = () => {
  return (
    <View>
      <Button
        title="Appuyez ici"
        onPress={() => alert('Bouton appuyé')}
      />
    </View>
  );
};

export default App;
```

Explication :

- **`Button`** : Crée un bouton avec un texte.
- **`title`** : Définit le texte du bouton.

- **`onPress`** : Définit une fonction qui sera exécutée lorsque l'utilisateur appuie sur le bouton.

6. Composant `ScrollView`

Le composant **`ScrollView`** permet de rendre un contenu qui dépasse l'écran défilable. C'est utile pour afficher une grande quantité de contenu dans une interface mobile.

Exemple d'utilisation de `ScrollView` :

```
import React from 'react';
import { ScrollView, Text, View } from 'react-native';

const App = () => {
  return (
    <ScrollView>
      <View style={{ padding: 20 }}>
        <Text>Élément 1</Text>
        <Text>Élément 2</Text>
        <Text>Élément 3</Text>
        <Text>Élément 4</Text>
        <Text>Élément 5</Text>
      </View>
    </ScrollView>
  );
};

export default App;
```

Explication :

- **`ScrollView`** : Rendre un contenu défilable.
- C'est utile pour afficher des listes ou des contenus longs.

7. Composant `FlatList`

Le composant **`FlatList`** est utilisé pour afficher des listes de données de manière optimisée. Il ne rend que les éléments visibles à l'écran, ce qui améliore les performances pour les longues listes.

Exemple d'utilisation de `FlatList` :

```
import React from 'react';
import { FlatList, Text, View } from 'react-native';

const App = () => {
  const data = [
    { key: 'Élément 1' },
    { key: 'Élément 2' },
    { key: 'Élément 3' },
    { key: 'Élément 4' },
    { key: 'Élément 5' },
  ];

  return (
```

```
    <FlatList
      data={data}
      renderItem={({ item }) => <Text>{item.key}</Text>}
    />
  );
};

export default App;
```

Explication :

- **`FlatList`** : Optimisé pour les longues listes de données.
- **`renderItem`** : Fonction qui détermine la façon dont chaque élément de la liste est affiché.

8. Composant `TouchableOpacity`

Le composant **`TouchableOpacity`** est utilisé pour rendre un élément interactif (comme un bouton personnalisé). Il applique un effet d'opacité lorsque l'élément est pressé.

Exemple d'utilisation de `TouchableOpacity` :

```
import React from 'react';
import { TouchableOpacity, Text, View } from 'react-native';

const App = () => {
  return (
    <View style={{ justifyContent: 'center', alignItems: 'center', flex: 1
}}>
      <TouchableOpacity onPress={() => alert('Pressé')}>
        <Text style={{ fontSize: 20, color: 'blue' }}>Appuyez ici</Text>
      </TouchableOpacity>
    </View>
  );
};

export default App;
```

Explication :

- **`TouchableOpacity`** : Crée un élément cliquable avec un effet d'opacité lors du clic.
- **`onPress`** : Permet de spécifier l'action à effectuer lorsque l'élément est pressé.

Résumé des Balises de Base en React Native :

1. **`View`** : Conteneur de base pour organiser les autres composants.
2. **`Text`** : Affiche du texte dans l'application.
3. **`Image`** : Affiche une image.
4. **`TextInput`** : Permet à l'utilisateur de saisir du texte.
5. **`Button`** : Crée un bouton cliquable.
6. **`ScrollView`** : Permet de faire défiler un contenu long.
7. **`FlatList`** : Optimisé pour afficher des listes de données longues.

8. **`TouchableOpacity`** : Crée un élément interactif avec un effet d'opacité.

Ces composants permettent de structurer et d'interagir avec l'interface utilisateur dans une application mobile React Native, ce qui rend la création d'applications mobiles riche et interactive.

🛠️ Atelier : Mini-App "Profil Utilisateur"

🎯 Objectif

Créer une application React Native qui affiche le profil d'un utilisateur (image, nom, description, liste de hobbies), avec la possibilité de :

- Modifier le nom via un **TextInput**.
- Afficher la liste des hobbies avec **FlatList**.
- Cliquer sur un bouton **TouchableOpacity** pour ajouter un nouveau hobby.
- Rendre le tout défilable avec **ScrollView**.

📌 Consignes

1. **Structure de base**
   - Crée un composant `App.js` avec un conteneur `<View>`.
   - Ajoute un titre avec `<Text>` : `"Mon Profil"`.
2. **Image de profil**
   - Ajoute une **Image** (locale ou depuis une URL).
   - Style : ronde (utilise `borderRadius`).
3. **Nom de l'utilisateur**
   - Utilise `<Text>` pour afficher un nom (par défaut : "Utilisateur Anonyme").
   - Ajoute un **TextInput** pour modifier ce nom (stocke-le dans un `useState`).
4. **Description**
   - Ajoute un petit texte descriptif (ex : "Développeur passionné par React Native").
5. **Liste de hobbies**
   - Utilise une **FlatList** pour afficher une liste de hobbies (par ex. : "Lecture", "Sport", "Voyage").
   - Chaque hobby est affiché dans un `<Text>` stylisé.
6. **Bouton Ajouter un Hobby**
   - Utilise **TouchableOpacity** avec un style personnalisé (fond bleu, texte blanc).
   - Quand on appuie dessus, un nouveau hobby `"Nouveau Hobby"` est ajouté à la liste.
7. **ScrollView**
   - Si le contenu devient trop long, rends-le défilable avec `<ScrollView>`.