📌 Récapitulatif complet du cours sur Flexbox en React Native

Flexbox est **le système de mise en page principal** en React Native. Il permet de **gérer efficacement l'alignement, la disposition et la répartition des espaces** dans une application mobile.

🔹 1. Comprendre le modèle Flexbox

- **Flexbox fonctionne avec deux axes principaux :**
  - **L'axe principal (main axis)** → Défini par `flexDirection`.
  - **L'axe secondaire (cross axis)** → Perpendiculaire à l'axe principal.
- **Différence avec le web :**
  - En **CSS Web**, `flexDirection` par défaut est `row` (horizontal).
  - En **React Native**, `flexDirection` par défaut est `column` (vertical).

🔹 2. Les propriétés fondamentales de Flexbox

📌 2.1. flexDirection (Définir l'axe principal)

Permet de choisir comment les éléments sont disposés :

- `row` → Les éléments sont alignés **horizontalement**.
- `column` → Les éléments sont alignés **verticalement** (par défaut).
- `row-reverse` → Alignement **inversé horizontalement**.
- `column-reverse` → Alignement **inversé verticalement**.

📌 2.2. justifyContent (Alignement sur l'axe principal)

Définit **comment les éléments sont espacés** sur l'axe principal :

- `flex-start` → Alignés au **début**.
- `flex-end` → Alignés à **la fin**.
- `center` → Alignés **au centre**.
- `space-between` → **Espacement égal** entre les éléments, sans espace aux extrémités.
- `space-around` → **Espacement égal**, avec de l'espace avant et après chaque élément.
- `space-evenly` → **Espacement parfaitement égal** entre tous les éléments.

📌 2.3. alignItems (Alignement sur l'axe secondaire)

Contrôle l'alignement **des éléments dans le sens perpendiculaire à** `flexDirection` :

- `flex-start` → Alignés **au début** de l'axe secondaire.
- `flex-end` → Alignés **à la fin** de l'axe secondaire.
- `center` → Alignés **au centre** sur l'axe secondaire.
- `stretch` → Les éléments prennent **toute la hauteur ou largeur disponible** (si aucune taille fixe n'est définie).

📌 2.4. flex (Gestion de l'espace occupé par les éléments)

Définit la **proportion d'espace qu'un élément occupe** dans son conteneur :

- `flex: 1` → L'élément prend **tout l'espace disponible**.
- `flex: 2` → L'élément prend **deux fois plus d'espace** qu'un élément avec `flex: 1` .
- `flex: 0` → L'élément ne grandit pas et conserve sa taille définie.

🔹 3. Propriétés avancées pour un alignement précis

📌 3.1. alignSelf (Alignement individuel d'un élément)

Permet d'**outrepasser** `alignItems` **pour un seul élément** :

- `auto` → Utilise `alignItems` du parent (comportement par défaut).
- `flex-start` → L'élément s'aligne **au début** de l'axe secondaire.
- `flex-end` → L'élément s'aligne **à la fin** de l'axe secondaire.
- `center` → L'élément est **centré** sur l'axe secondaire.
- `stretch` → L'élément s'étire pour **remplir tout l'espace disponible** sur l'axe secondaire.

📌 3.2. alignContent (Alignement des lignes en cas de flexWrap)

S'applique **uniquement lorsque** `flexWrap: "wrap"` **est activé**, et gère l'alignement **des lignes** d'éléments :

- `flex-start` → Toutes les lignes sont alignées **en haut** du conteneur.
- `flex-end` → Toutes les lignes sont alignées **en bas** du conteneur.
- `center` → Les lignes sont **centrées** dans le conteneur.
- `stretch` → Les lignes s'étirent pour **remplir tout l'espace vertical**.
- `space-between` → Répartit les lignes avec **un espace égal entre elles**.
- `space-around` → Ajoute **de l'espace autour de chaque ligne**.

🔹 4. Astuces et bonnes pratiques avec Flexbox

✅ Bien structurer les écrans avec `flex`

- Utiliser `flex: 1` sur le **conteneur principal** pour qu'il prenne tout l'écran.
- Ajouter `flexDirection` et `justifyContent` pour structurer la mise en page.

✅ Utiliser `alignSelf` pour des ajustements spécifiques

- Si un élément doit être **aligné différemment** des autres, `alignSelf` est la meilleure solution.

✅ Gérer l'adaptabilité avec `flexWrap` et `alignContent`

- Lorsque plusieurs éléments doivent **s'adapter à l'écran**, `flexWrap: "wrap"` permet de les faire passer à la ligne.
- `alignContent` ajuste l'alignement **des groupes de lignes**.

✅ Éviter les valeurs fixes lorsque possible

- Utiliser `flex` plutôt que des `width` et `height` fixes pour un rendu plus fluide.
- `stretch` permet de s'adapter aux différentes tailles d'écran.

✅ Tester sur plusieurs tailles d'écrans

- Utiliser l'outil **"Responsive Design"** dans les émulateurs pour tester différents formats (iPhone, Android, tablettes).

🎯 Conclusion et résumé final

🔹 Propriété	🔹 Effet
`flexDirection`	Définit l'axe principal (`row` , `column` , etc.).
`justifyContent`	Gère l'alignement **sur l'axe principal** (`flex-start` , `center` , `space-between` , etc.).
`alignItems`	Gère l'alignement **sur l'axe secondaire** (`flex-start` , `center` , `stretch` , etc.).
`flex`	Définit la **taille relative** d'un élément par rapport aux autres (`flex: 1` , `flex: 2` , etc.).
`alignSelf`	Change l'alignement **d'un seul élément** indépendamment des autres (`flex-start` , `center` , etc.).
`flexWrap`	Active le passage **automatique à la ligne** (`wrap` , `nowrap`).

🔹 **Propriété**	🔹 **Effet**
`alignContent`	Contrôle l'alignement **des lignes entières** quand `flexWrap` est activé (`space-between` , `center` , etc.).

Grâce à ce récapitulatif, tu as maintenant une **vision complète de Flexbox en React Native** ! 🚀
Entraîne-toi en réalisant **des mises en page concrètes** pour bien assimiler ces concepts. 💡