

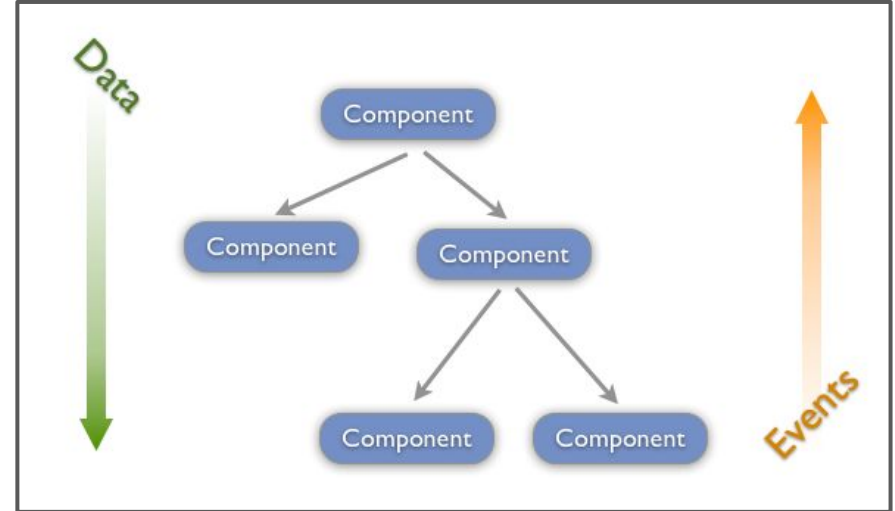
Interaction entre les composants React

Les composants React avancé

Communication entre composant

Pour rendre possible la communication entre composants, il est possible de :

- Envoyer des données au composant pour **descendre** dans l'arborescence.
- Utiliser un callback sur le composant (*mécanisme d'événement*) pour **monter** dans l'arborescence.



Ce chapitre n'aborde pas les concepts avancés tels que : Redux, le Context, ...

Depuis le Parent vers l'Enfant

L'envoi de données vers un composant enfant est réalisable en utilisant uniquement le système de « props » de React.

- Dans le parent : En JSX, utiliser les attributs du composant.

```
<Demo data={donneeEnvoyee} />
```

- Dans l'enfant : Récupérer les données via les « props » du composant.

```
const donneeRecu = props.data;
```

Depuis l'Enfant vers le Parent

L'envoi d'événement (avec ou sans données) vers un composant parent est possible en définissant un callback dans les « props » du composant enfant.

- Dans le parent : En JSX, ajouter un callback sur l'attribut utilisé en tant qu'event.

```
<Demo onResult={handleFunction} />
```

- Dans l'enfant : Déclencher le "callback" depuis les « props » du composant.

```
props.onResult(donneeEnvoyee)
```

Pour éviter une erreur lorsqu'un événement est déclenché et qu'aucun callback n'a été défini sur composant, il est possible d'ajouter une fonction vide (NOOP) en utilisant la propriété « defaultProps » du composant.

Exemple d'un événement d'un composant

```
import PropTypes from 'prop-types';

const ReponseEvent = function (props) {
  const sendResponse = () => {
    props.onReponse(42);
  };

  return (
    <button onClick={sendResponse}> Envoyer une réponse... </button>
  );
};

ReponseEvent.defaultProps = {
  onReponse: () => { }
};

ReponseEvent.propTypes = {
  onReponse: PropTypes.func
};

export default ReponseEvent;
```

Entre deux composants frères

Pour permettre une communication bidirectionnelle entre 2 composants frères, il est possible d'utiliser le composant parent en tant qu'intermédiaire.

Cette stratégie consiste à réaliser une liaison "Enfant > Parent" et "Parent > Enfant" vers chaque composant, et leur permettre de communiquer à l'aide de fonctions.

```
<div>
  <SiblingA
    data={valueStateA}
    onEvent={eventSiblingA} />
  <SiblingB
    data={valueStateB}
    onEvent={eventSiblingB} />
</div>
```

Exercice • Interaction entre composant

Les composants React avancé

Exercice

1. Créer une application « Todo list » constitué des éléments suivants :
 - Un formulaire pour ajouter une nouvelle tâche à effectuer.
 - Une liste de tâches, qui permet les actions suivantes :
 - Terminer une tâche.
 - Supprimer une tâche
2. Une tâche contient les informations suivantes :
 - Un nom (*Obligatoire*) et une description (*Optionnel*)
 - Une priorité (*Basse / Normal / Urgent*)
 - Une complétion (*Une valeur booléen*)

Exercice

3. Ajouter une validation au formulaire
 - Champs manquants en rouge.
4. Ajouter de la couleur aux tâches
 - Urgente en rouge
 - Terminée en gris (Rayure diagonal)
5. (BONUS) Ajouter des filtres à la liste
 - En cours
 - Urgentes
 - Terminées

Ajouter une nouvelle tâche

Nom

Description

Priorité Normal ▼ Ajouter

Liste des tâches

Acheter du café (Urgent) Terminer

Supprimer

Réaliser l'exercice Terminer

Créer l'application « Todo List » Supprimer

Tâche terminée Terminer

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Maecenas scelerisque nisi sed. Supprimer