Séance 09 :

→ REACT.JS – Les fondamentaux

INF37407 – Technologie de l'inforoute

Prof. **Yacine YADDADEN**, Ph. D.

UQAR

Plan

1. Introduction Générale
2. `Map()`, `Filter()` & `Reduce()`
3. Introduction aux composants
4. Introduction au **JSX**
5. Propriétés, États & Arborescence de Composants
6. Navigation & Routage
7. Questions & Discussion

UQAR

Introduction Générale

UQAR

Plan

1. Introduction & contexte
2. Qu’est-ce que **React.js** ?
3. Environnement de développement
4. Première application **React.js** – *To Do List* !
5. Questions et discussion

UQAR

Plan

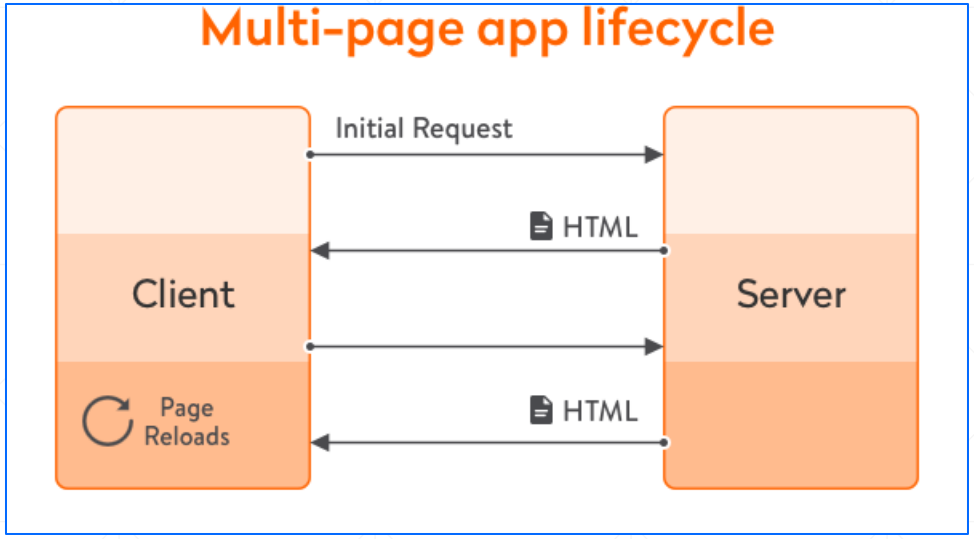
1. **Introduction & contexte**
2. Qu’est-ce que **React.js** ?
3. Environnement de développement
4. Première application **React.js** – *To Do List* !
5. Questions et discussion

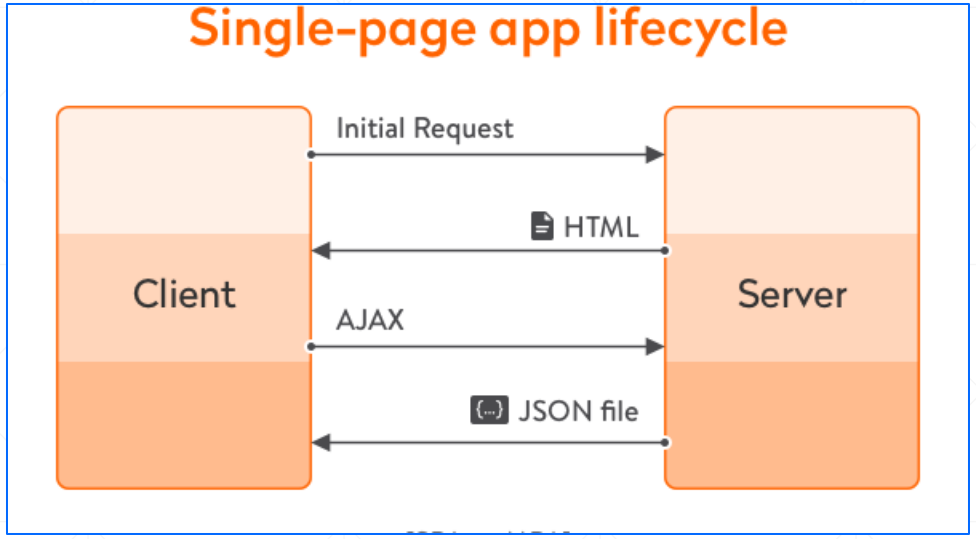
Introduction

- Au cours de ses dernière années, il y a e un énorme engouement pour développement *front-end* en JavaScript,
- L'objectif étant de rendre l'expérience utilisateur plus rapide et plus fluide en limitant au maximum les appels *serveur*,
- Le moyen d'y arriver est de passer par ce qu'on appelle des **SPA** ou *Single Page Application*.

UQAR

MPA *vs.* SPA



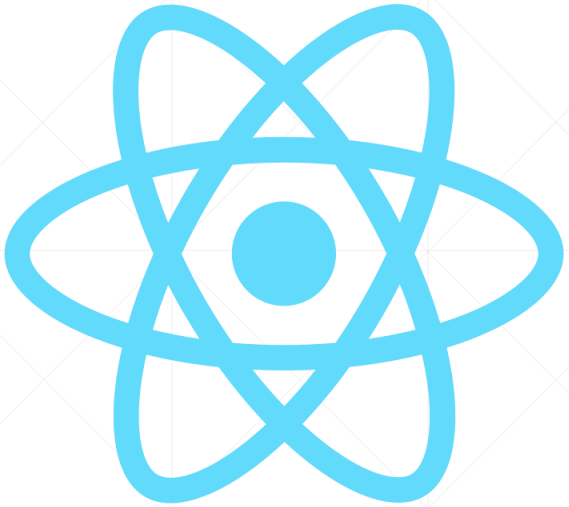


UQAR

Plan

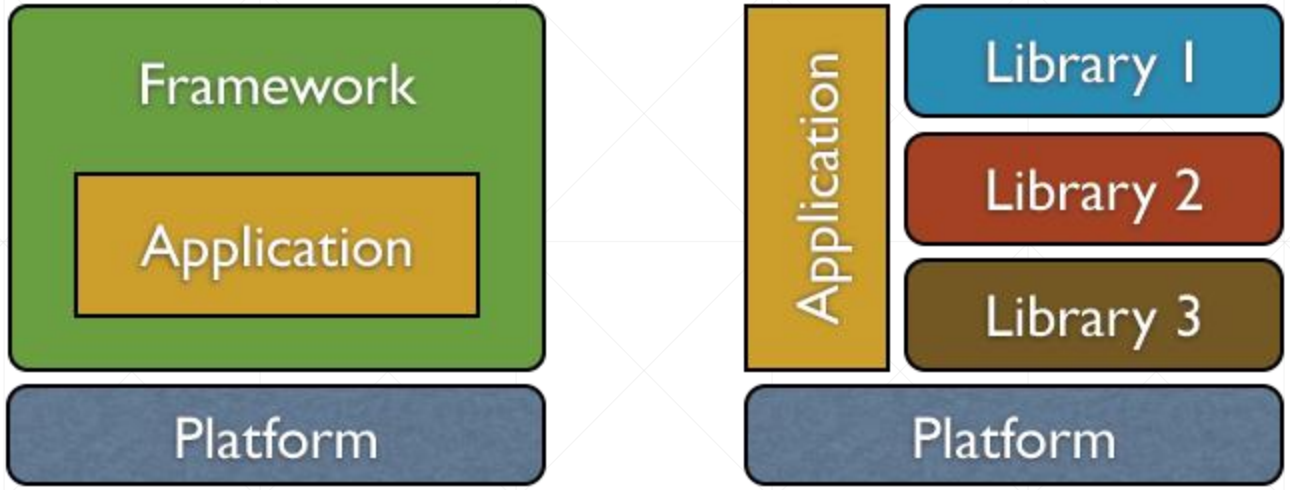
1. Introduction & contexte
2. **Qu’est-ce que React.js ?**
3. Environnement de développement
4. Première application **React.js** – *To Do List* !
5. Questions et discussion

Définition



- **Définition :** *« C'est une **librairie** JavaScript dont l'objectif est de faciliter et d'optimiser le développement des SPA de large envergures. »*,
- Elle a été lancée en 2013 par **Facebook**,
- Elle est *Open Source*,
- Elle est utilisée par des géants du Web : PayPal, Facebook, Airbnb, Apple, …

Librairie *vs.* Framework



UQAR

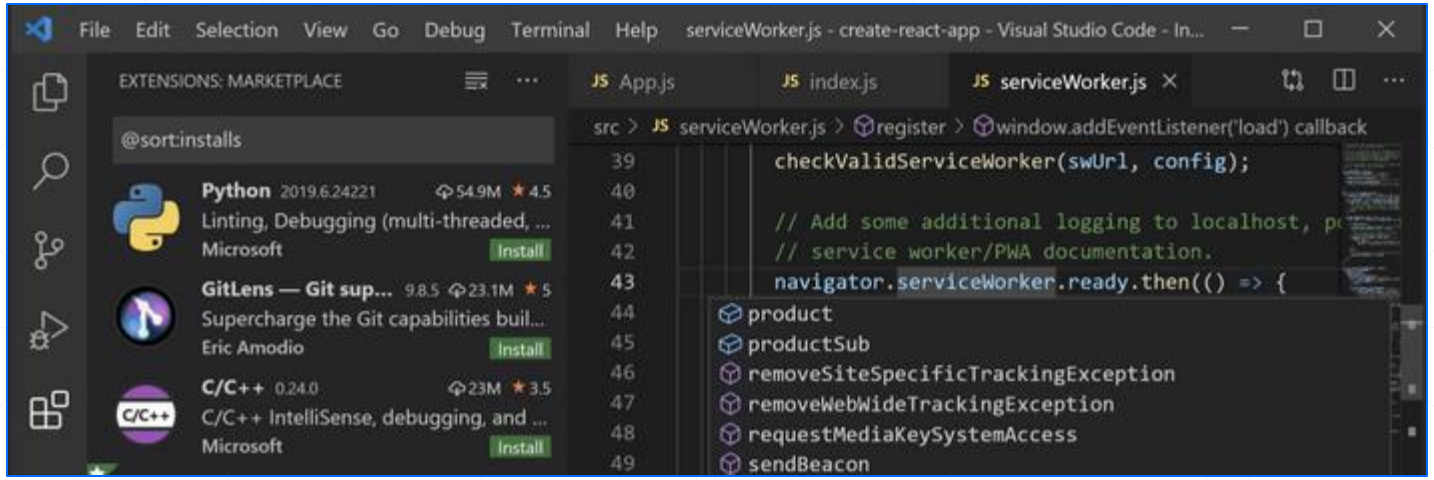
Plan

1. Introduction & contexte
2. Qu’est-ce que **React.js** ?
3. **Environnement de développement**
4. Première application **React.js** – *To Do List* !
5. Questions et discussion

UQAR

Éditeur de code

- Le meilleur choix lorsqu'on développe en **React.js** : **MS Visual Studio Code**,
- Lien de téléchargement : https://code.visualstudio.com/
- Certains paquets permettent de rendre le développement plus aisé.



Paquets complémentaires

- ***HTML CSS Support***, ***Auto Close Tag*** & ***Auto Rename Tag*** pour le **HTML**,
- ***Better comments*** pour mieux organiser les commentaires,
- ***Debugger for Chrome*** ou ***Firefox*** pour avoir les erreur dans la console,
- ***JavaScript (ES6) code snippets*** pour l'autocomplétion en **JavaScript**,
- ***GitLens*** pour faciliter l'utilisation de **Git**,
- ***Live Server*** pour avoir un rafraichissement automatique,
- ***Prettier*** pour le formatage du code,
- ***Material Icon Theme*** pour avoir un thème agréable pour coder.

Installation de `Node.js`

- **Définition :** *« C'est une plateforme logicielle libre en JavaScript. Elle permet de pouvoir exécuter du code JavaScript en dehors du Navigateur Web. Elle est généralement utilisée côté serveur ou pour émulé un serveur. »*,
- Afin de pouvoir l'installer :
  1. Se rendre au site Web : https://nodejs.org/en/download/,
  2. Télécharger la dernière version **LTS** *(stable)*,
  3. Lancer l'exécutable et suivre les étapes.
- Afin de vérifier que c'est bien installé :
  1. Ouvrir la ligne de commande,
  2. Lancer la commande **`node -v`**

Résultat


yacin
yacin@NEBUCHADNEZZAR ~ [06:32]
❯ npm -v
6.13.4
yacin@NEBUCHADNEZZAR ~ [06:32]
❯

UQAR

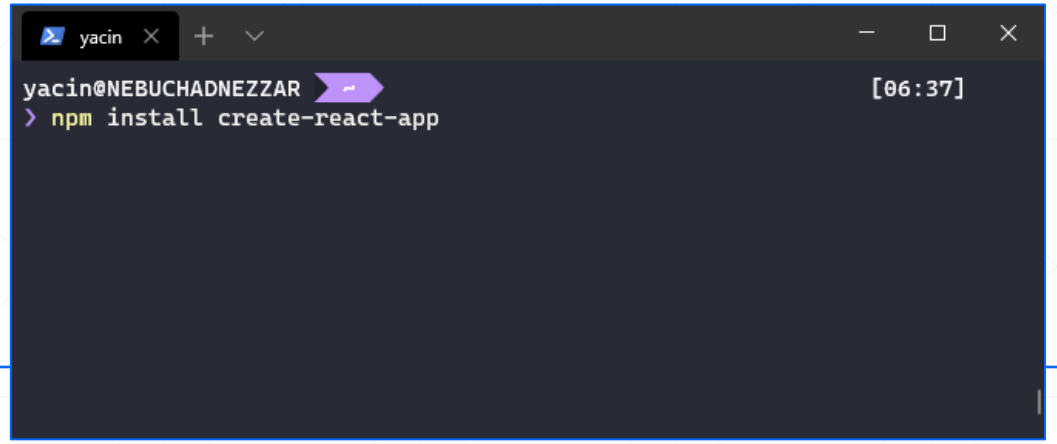
Qu’est-ce que npm ?

- **Définition :** *« pour Node Package Manager est gestionnaire de paquets intégré à `Node.js` et permet d’installer très facilement différent paquets à l’aide d’une simple commande. »*,
- **Utilisation :**
  - Installation d’un paquet de façon *globale* : **`npm install -g`** *`[ nom_du_paquet ]`*
  - Installation d’un paquet de façon *locale* : **`npm install`** *`[ nom_du_paquet ]`*
- **Alternative :**
  - Il est également possible d’utiliser **`yarn`** après l’avoir installé,
  - Installation : **`npm install -g yarn`**
  - Utilisation – ajout : **`yarn add`** *`[ nom_du_paquet ]`*
  - Utilisation – suppression : **`yarn remove`** *`[ nom_du_paquet ]`*

UQAR

Gestionnaire de projet React.js

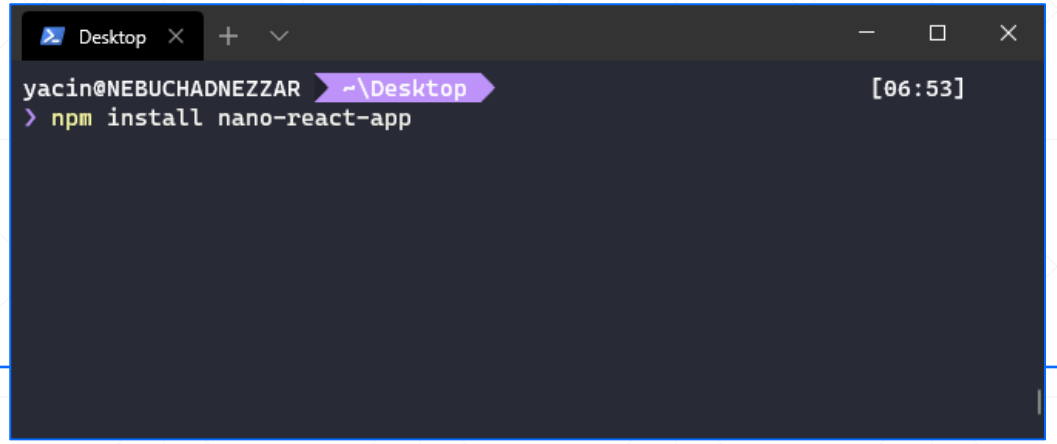
- Il existe un package dans **`Node.js`** qui permet la gestion et la création automatique de projet de type **React.js** qui est basé sur **Webpack**,
- Afin de pouvoir l'installer, il faut suivre les étapes suivantes :
  1. Ouvrir la ligne de commande,
  2. Lancer la commande : **`npm install create-react-app`**



UQAR

Gestionnaire de projet React.js – minimal

- Il existe un package dans **`Node.js`** qui permet la gestion et la création automatique de projet de type **React.js** qui est basé sur **Vite**,
- Afin de pouvoir l'installer, il faut suivre les étapes suivantes :
  1. Ouvrir la ligne de commande,
  2. Lancer la commande : **`npm install nano-react-app`**



UQAR

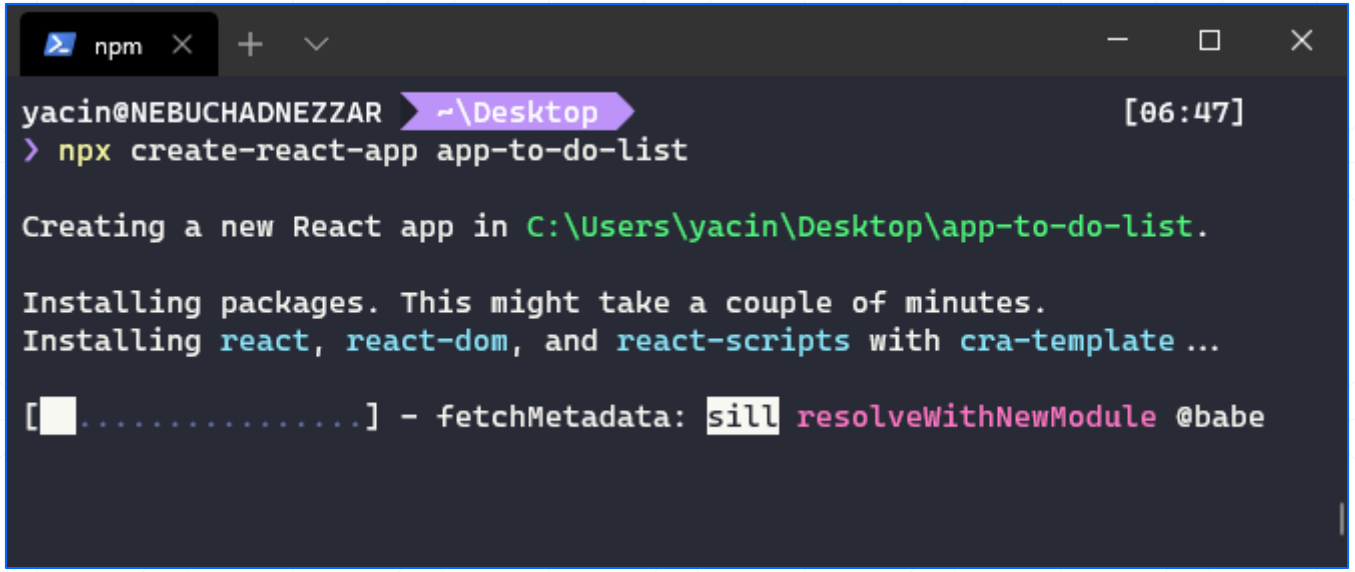
Plan

1. Introduction & contexte
2. Qu’est-ce que **React.js** ?
3. Environnement de développement
4. **Première application React.js – *To Do List* !**
5. Questions et discussion

Création d’un nouveau projet

La création d’un nouveau projet **React.js** se fait à l’aide de la commande :

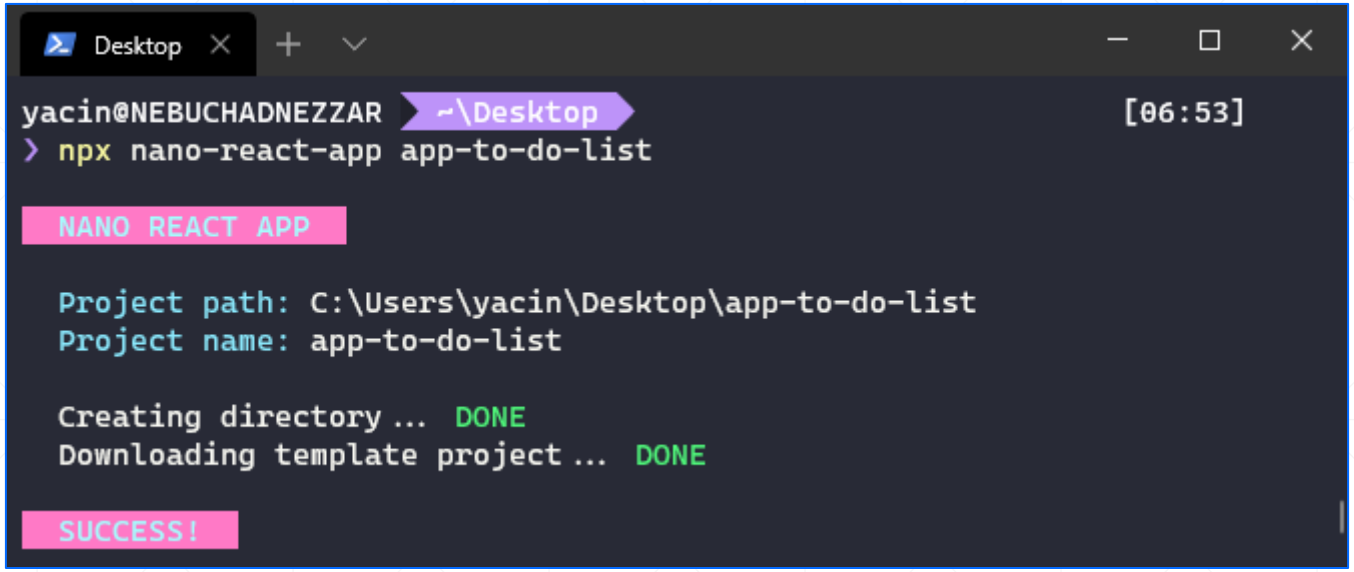
`npx create-react-app` *`[ nom_du_projet_react ]`*



Création d'un nouveau projet – minimal

La création d'un nouveau projet **React.js** se fait à l'aide de la commande :

`npx nano-react-app` *`[ nom_du_projet_react ]`*



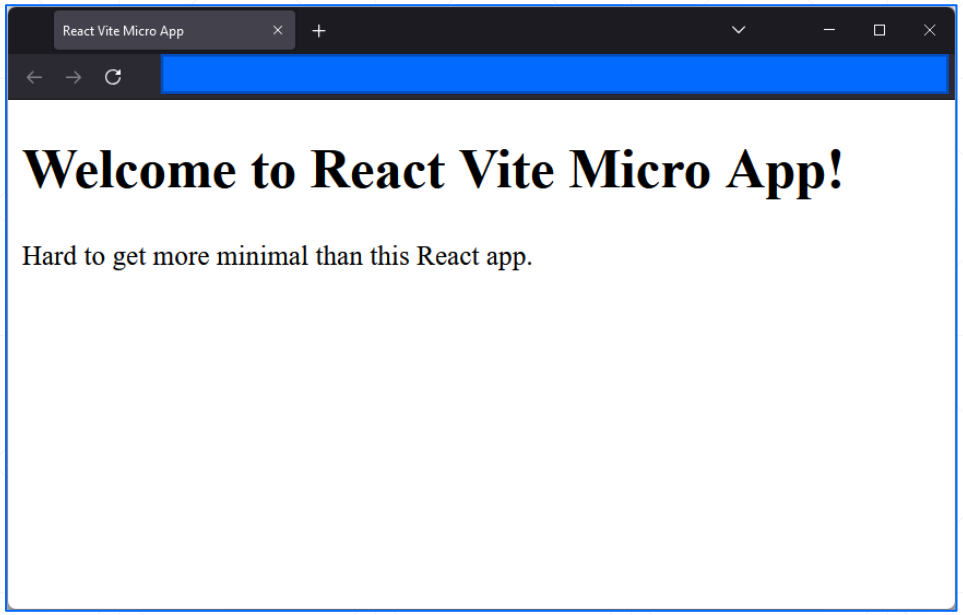
UQAR

Lancer le projet

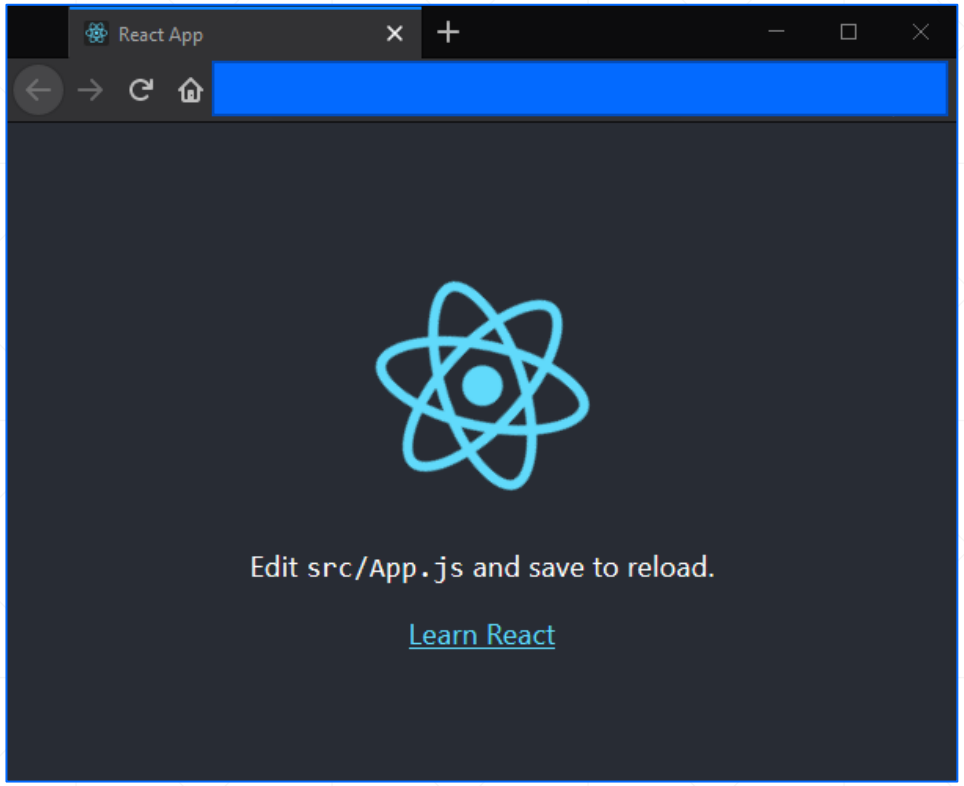
- Afin de lancer le projet, il faut exécuter les commandes suivantes :
  1. **`cd`** *`[ nom_du_projet_react ]`*
  2. **`npm install`**
  3. **`npm start`**

- Ça va permettre d'émuler un *serveur local* pour lancer l'application sur :
  a. Version *complète* : **`http://localhost:3000`**
  b. Version *minimale* : **`http://localhost:5173`**
- Il est possible d'avoir du *live reload* ou un *rafraichissement automatique*.

Résultats



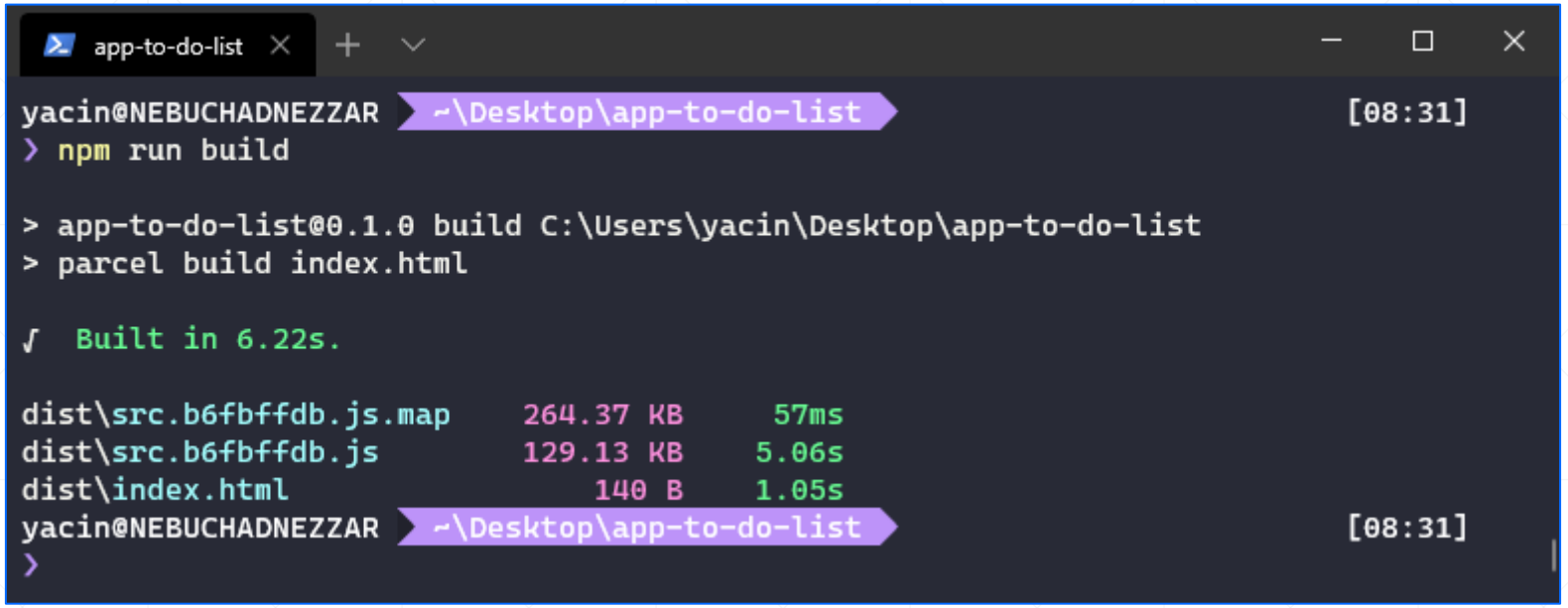
Version minimale



Version complète

Compilation

Afin de compiler un projet **React.js**, il faut exécuter la commande : **`npm run build`**



UQAR

Plan

1. Introduction & contexte
2. Qu’est-ce que **React.js** ?
3. Environnement de développement
4. Première application **React.js** – *To Do List* !
5. **Questions et discussion**

UQAR

Map(), Filter() & Reduce()

UQAR

Plan

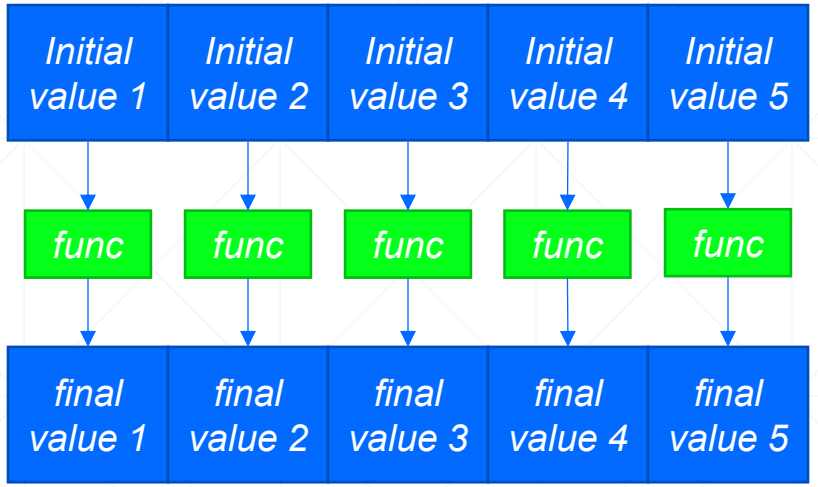
1. Introduction & contexte
2. Introduction à `map()`
3. Initiation à `filter()`
4. Introduction à `reduce()`
5. Questions et discussion

Introduction & contexte

- Traitement des tableaux et listes de données,
- Généralement, utilisation de simple boucles (`for`, `while`, …),
- Utilisation de la **programmation fonctionnelle** (*plus optimisée*),
- En JavaScript (ES6), il y a différentes fonctions :
  - `map()`,
  - `filter()`,
  - `reduce()`.

Explication : `map()`

- **Objectif :** *Elle est similaire à* ***`ForEach()`*** *sauf qu'elle applique une fonction flèche à chaque élément.*
- **Graphique :**

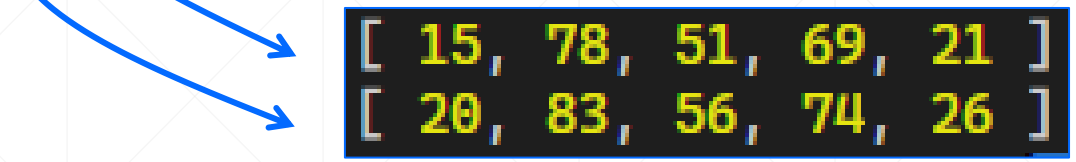


Code : map()

- **Code :**

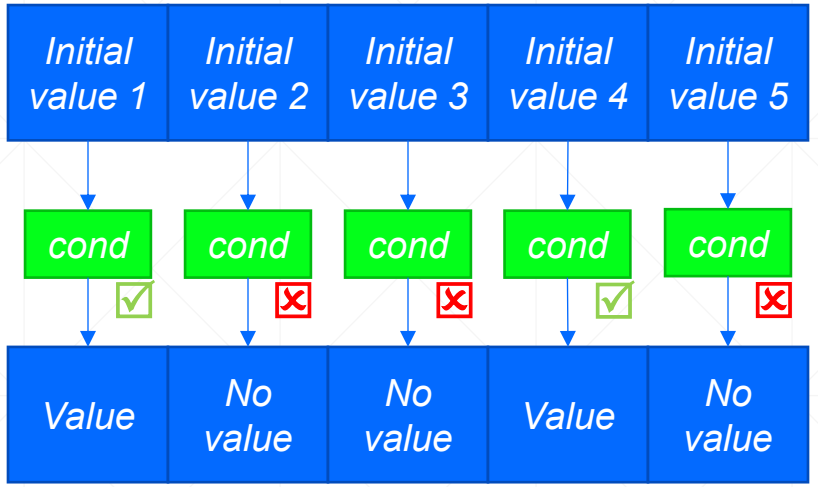
```
let in_arr = [15, 78, 51, 69, 21];
let out_arr = in_arr.map(elt => (elt + 5));
```

- **Résultat :**



Explication : `filter()`

- **Objectif :** *Elle permet de sélectionner un certain nombre d'éléments à partir d'une liste en respectant une condition spécifique.*
- **Graphique :**



Code : filter()

- **Code :**

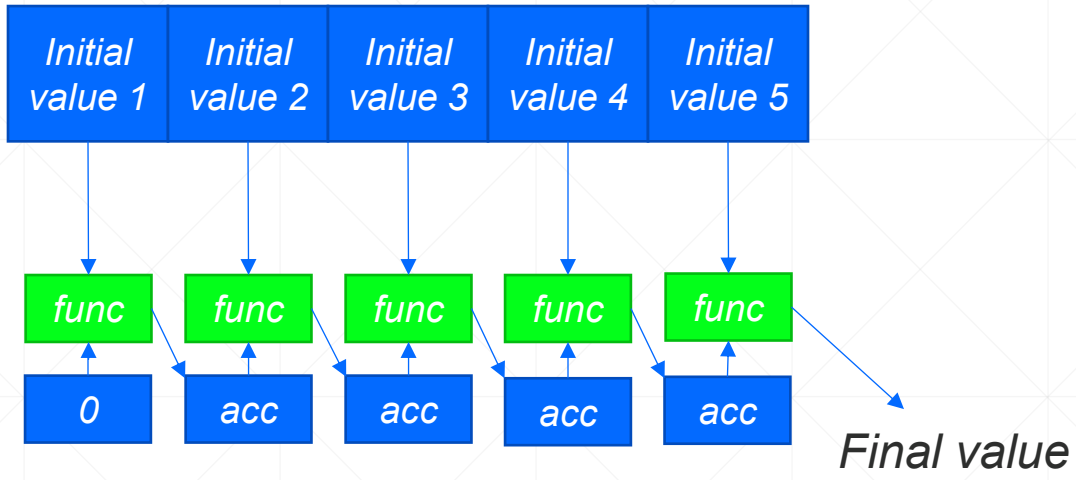
```
let in_arr = [15, 78, 51, 69, 21];
let out_arr = in_arr.filter(elt => (elt > 30));
```

- **Résultat :**

```
[ 15, 78, 51, 69, 21 ]
[ 78, 51, 69 ]
```

Explication : reduce()

- **Objectif :** *Elle contient deux paramètre l'un est la **valeur courante** et l'autre est la **valeur accumulée**. Le résultat retourné est la valeur accumulée.*
- **Graphique :**



UQAR

Code : reduce()

- **Code :**

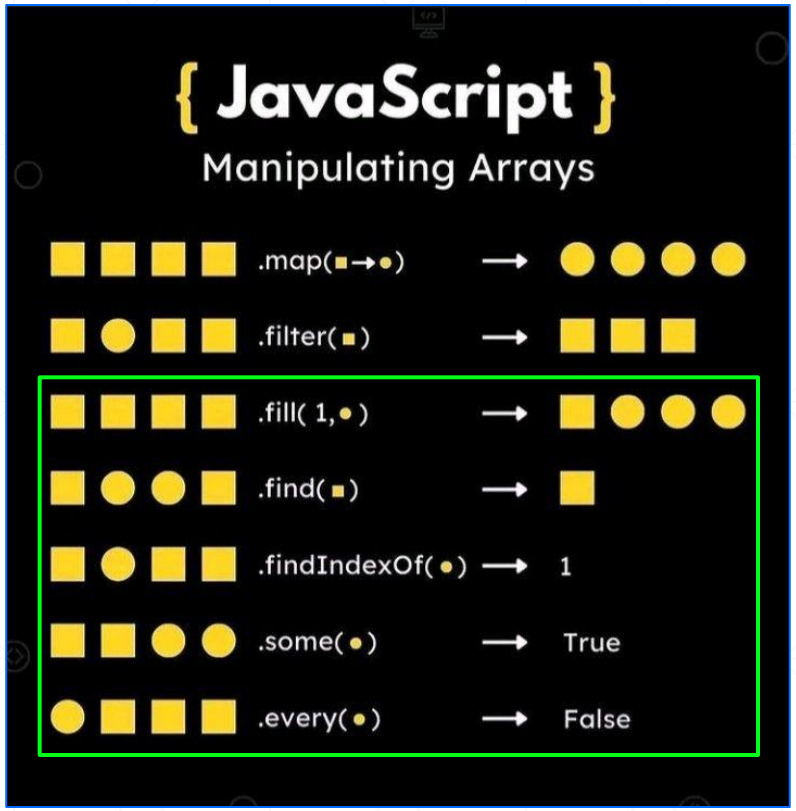
```
let in_arr = [15, 78, 51, 69, 21];
let out_val = in_arr.reduce((acc, elt) => acc + elt);
```

- **Résultat :**

```
[ 15, 78, 51, 69, 21 ]
234
```

UQAR

Autres méthodes



UQAR

Introduction aux composants

UQAR

Plan

1. Faire du pure **React.js**
2. Le **DOM** vs. **DOM** *Virtuel*
3. Élément **React.js**
4. Qu’est-ce que **ReactDOM** ?
5. Inclure des données *(Élément **React.js**)*
6. Composants **React.js**
7. Génération du rendu
8. Questions et discussion

UQAR

React.js – Le minimum requis

- Afin de faire du développement **React.js** il faut avoir :
  - **React**,
  - **ReactDOM**.

- **React :** *est utilisée pour la création des vues.*
- **ReactDOM :** *est utilisée pour faire le rendu des vues.*

UQAR

React.js – Structure de la page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>React Minimal</title>
</head>
<body>
    <!-- c'est là que va être contenu notre rendu -->
    <div id="react-container"></div>

    <!-- c'est les deux bibliothèques nécessaires -->
    <script crossorigin src="https://unpkg.com/react@16.7.0/umd/react.production.min.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@16.7.0/umd/react-dom.production.min.js"></script>

    <script>
        /* c'est ici qu'on met notre code */
    </script>
</body>
</html>
```

IMPORTANT : *pour les paquets téléchargés à partir de https://cdnjs.com, il faut utilise la version **UMD** (Universal Module Definition).*

Le **DOM** vs. **DOM *Virtuel***

- **DOM :** *« Le Modèle Objet de Document (**DOM: Document object Modéle**) est la représentation de donnée d'objets qui comprend la structure et le contenu d'un document web, nous allons présenter brièvement le DOM. Nous verrons comment le DOM représente un document HTML ou XML en mémoire et comment vous utilisez les API pour créer du contenu web et des applications. »*[1]
- **DOM *Virtuel* :** *« Le DOM virtuel (VDOM) est un concept de programmation dans lequel une représentation idéale, ou « virtuelle », d'une interface utilisateur (UI) est conservée en mémoire et synchronisée avec le DOM « réel » par une bibliothèque telle que ReactDOM. »* [2]

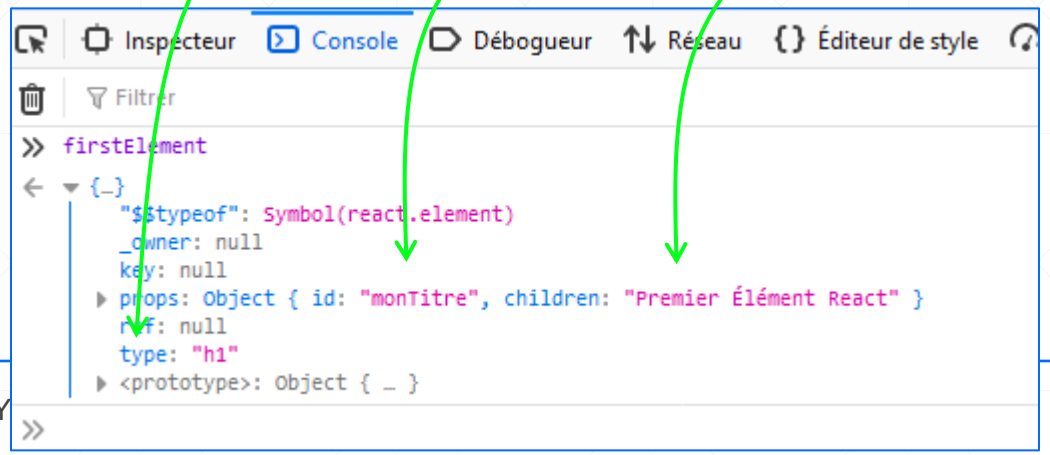
[1] https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model/Introduction
[2] https://fr.reactjs.org/docs/faq-internals.html

Élément React.js

- **Définition :** *Si le **DOM** est composé d'**éléments DOM**, alors **ReactDOM** est composé d'**éléments React**. Ces derniers sont des instructions ou une description sur comment les **éléments DOM** devraient être après le rendu.*
- **Exemple :**

```
let firstElement = React.createElement("h1", {id: "monTitre"}, "Premier Élément React");
```

- **Dans la console :**



UQAR

Qu'est-ce que **ReactDOM** ?

- **Définition :** *c'est un ensemble d'outils permettant de faire le rendu ou l'affichage des éléments React sur le navigateur Web.*
- **Génération d'un rendu :**

```
<!-- ? c'est là que va être contenu notre rendu -->
<div id="react-container"></div>

<!-- c'est les deux bibliothèques nécessaires -->
<script crossorigin src="https://unpkg.com/react@16.7.0/umd/react.production.min.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16.7.0/umd/react-dom.production.min.js"></script>

<script>
    /* c'est ici qu'on met notre code */
    let firstElement = React.createElement("h1", {id: "monTitre"}, "Premier Élément React");

    ReactDOM.render(firstElement, document.getElementById("react-container"));
</script>
```

UQAR

Inclure des **enfants** *(Children)*

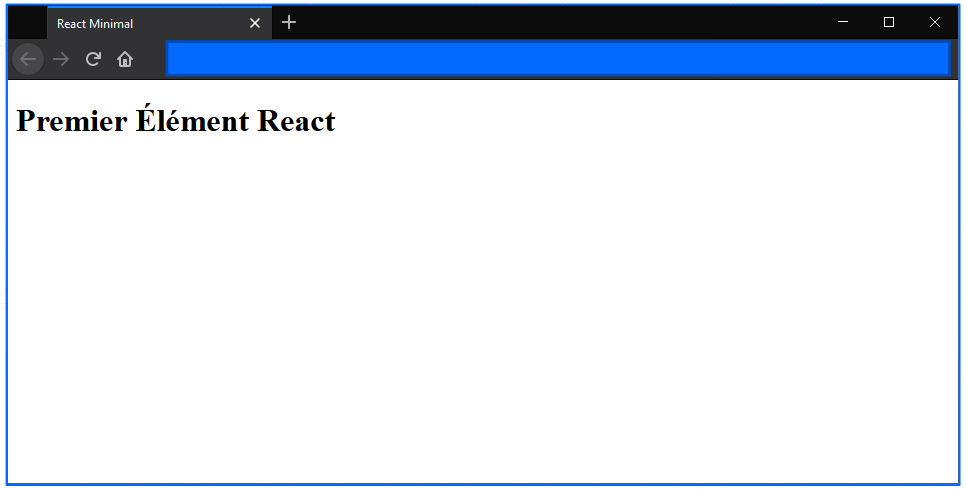
- Il est possible d'inclure plusieurs **sous-éléments React** au sein d'un élément,
- Ça permet de générer une certain *arborescence* telle que celle du **DOM**.

*Sur React.js on n'utilise pas le mot clé **`class`** pour désigner une classe pour un composant HTML car il est déjà réservé, on utilise **`className`** à la place.*

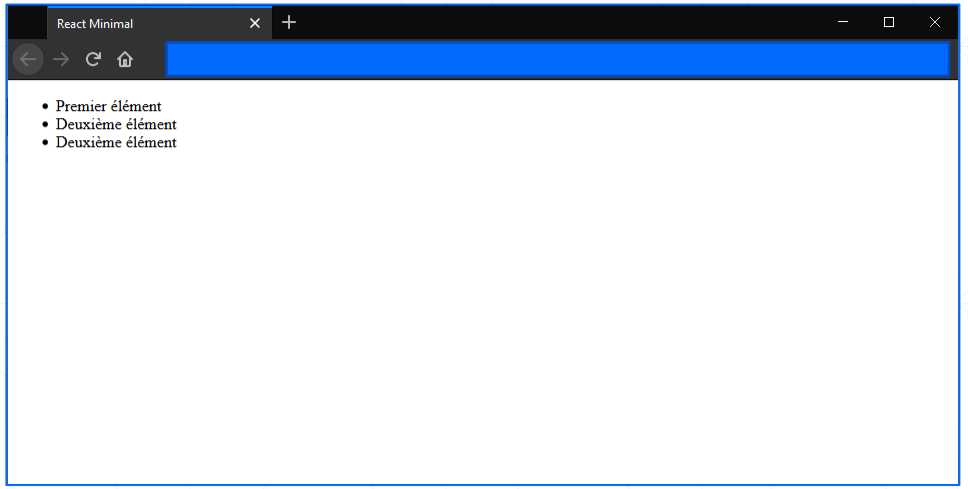
- **Exemple :**

```
let firstElement = React.createElement(
"ul",
{ className: "Liste" },
React.createElement("li", null, "Premier item"),
React.createElement("li", null, "Deuxième item"),
React.createElement("li", null, "Troisième item"),
);
```

Qu'est-ce que **ReactDOM ? [Résultat]**



Premier exemple



Deuxième exemple (arborescence)

UQAR

Inclure des données *(Élément React.js)*

- **Problème :** *Que faire lorsqu'on a des données dynamiques (qui varient) ?*
- **Solution :** *Faire appel aux fonctionnalités de JavaScript.*
- **Exemple :**

```
let dataItems = ["premier item", "deuxième item", "troisième item"];

let firstElement = React.createElement(
"ul",
{ id: "maListe" },
dataItems.map((item) => React.createElement("li", {}, item))
);
```

Les données dans une liste

*Utilisation de **map()** pour parcourir la liste*

Résultat : *Le même que précédemment*

Composants React.js

- **Définition :** *Il représente une partie de l'**UI** (Interface Utilisateur), il peut inclure plusieurs éléments **HTML**.*
- **Intérêts :**
  - *Faire en sorte d'avoir des éléments réutilisables,*
  - *Permettre d'avoir un code plus clair & concis.*
- **Trois différentes méthodes :**
  - Utilisation de `createClass` *(obsolète).*,
  - Utilisation de *classes* **ES6**,
  - Utilisation de ***stateless functions component*** *(fonctions flèches).*

Composants React.js – Classes ES6

- **Déclaration :**

Déclaration du composant

```
class ListItems extends React.Component {

render() {
    return React.createElement(
    "ul",
    { className: "maListe" },
    this.props.Items.map((item) => React.createElement("li", {}, item))
    );
}
}
```

Génération du rendu du composant

Inclure des données

Faire appel au composant

- **Appel :**

```
ReactDOM.render(
React.createElement(ListItems, { Items: dataItems }, null),
document.getElementById("react-container")
);
```

Composants React.js – Fonctions flèches

- **Déclaration :**

Les données d'entrée

```
const ListItems = (props) =>
React.createElement(
    "ul",
    { className: "maListe" },
    props.Items.map((item) => React.createElement("li", {}, item))
);
```

Déclaration du composant

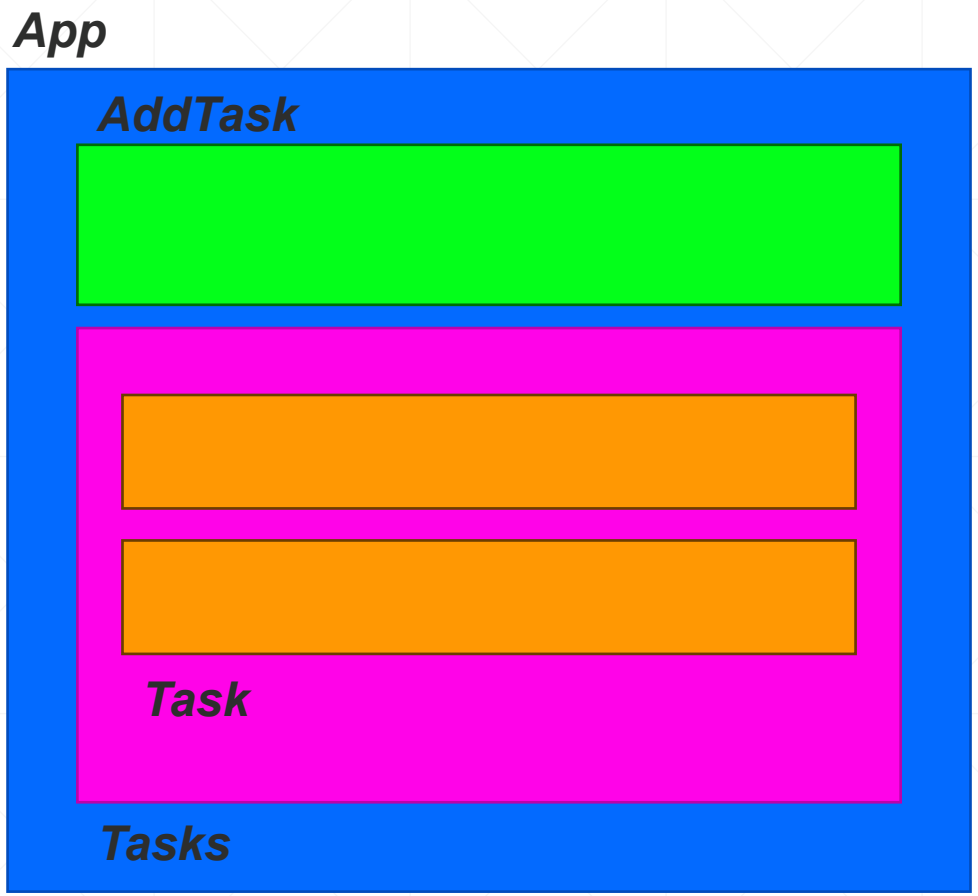
Inclure des données

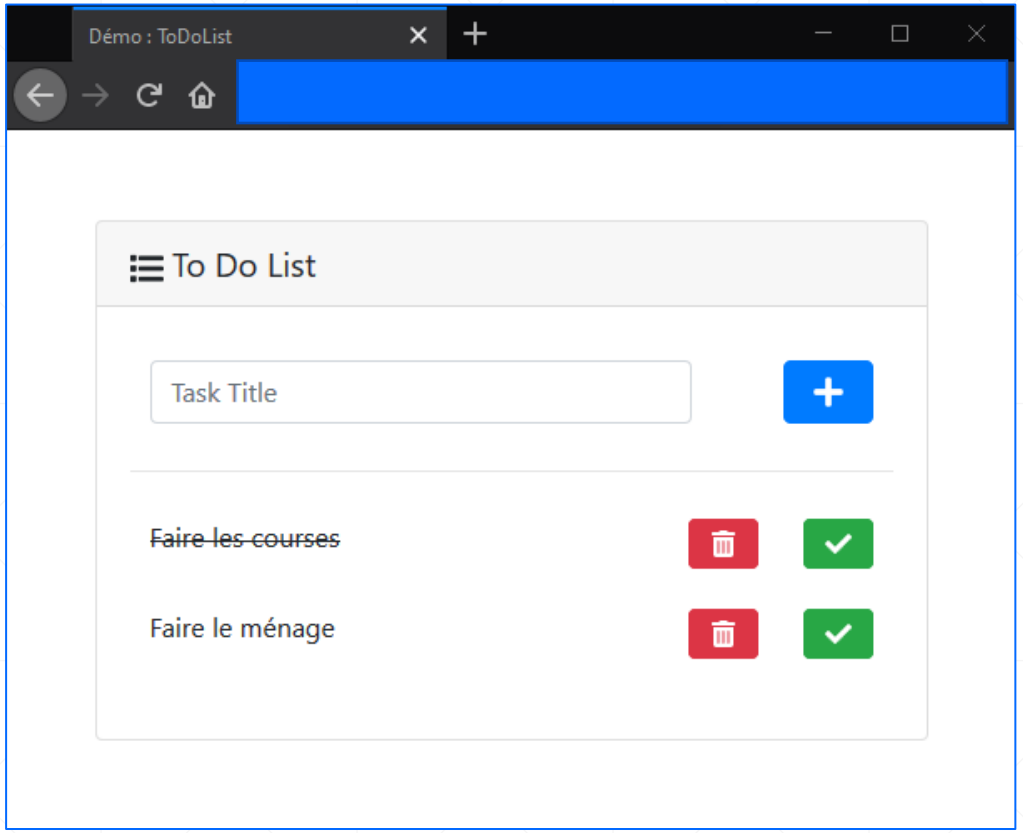
Faire appel au composant

- **Appel :**

```
ReactDOM.render(
React.createElement(ListItems, { Items: dataItems }, null),
document.getElementById("react-container")
);
```

Composants React.js *(ToDoList)*





UQAR

Génération du rendu

- La librairie **React.js** permet une séparation entre *données* et *logique*,
- La *modification des données* implique la *modification* du **DOM**,
- **Problème :** *la régénération totale du **DOM** peut causer des lenteurs.*
- **Solution :**
  - **React.js** *permet une mise à jour partielle du **DOM**,*
  - *Celle-ci ne cible que les éléments touchés par les modifications,*
  - *Ça permet d'avoir plus de fluidité et de rapidité.*

UQAR

Introduction au JSX

UQAR

Plan

1. Introduction & Contexte
2. Qu’est-ce que le **JSX** ?
3. Introduction à **`Babel.js`**
4. Le **`Webpack.js`** *vs.* **`Vite.js`**
5. Concrètement dans **React.js**
6. Questions et discussion

Introduction & Contexte

- Dans le cours précédent, nous avons vu création des **éléments React** avec :

```
React.createElement()
```

- On remarque c'est un peu **lourd** et **fastidieux** de construire toutes une interface avec ça,
- Un moyen plus simple serait de pouvoir utiliser la **syntaxe HTML** directement dans le JavaScript,
- L'idéal serait de pouvoir construire *l'arborescence de l'interface* en passant par des **balises**.

UQAR

Qu’est-ce que le JSX ?

- **Définition :** *Le **JSX** ou **J**ava**S**cript **X**ML est une **extension syntaxique** du JavaScript permettant à ce dernier de pouvoir utiliser du code HTML directement dans le code JavaScript.*
- Il a été lancé par les équipe **React** de **Facebook** avec l’objectif principal d’avoir un *code plus concis* lors de la construction du **DOM** *virtuel*.

```
<ul>
    <li>premier item</li>
    <li>deuxième item</li>
    <li>troisième item</li>
    <li>quatrième item</li>
</ul>
```

Code **JSX** pour une *liste non ordonnée*

UQAR

Plus de détails sur JSX

- Il permet de créer une *arborescence* telle que celle du **DOM** :
- Il faut utiliser ***className*** à la place de ***class***,
- Il est possible d'insérer du **code JavaScript** :

```
<ListElements>
    <Element {this.props.titre} />
</ListElements>
```

```
<ListElements>
    <Element />
    <Element />
    <Element />
    <Element />
</ListElements>
```

- Afin d'afficher une liste de **façon dynamique** :

```
let dataElement = ["premier item", "deuxième item", "troisième item", "quatrième item",];
<ul>
    {this.props.dataElement.map((item, i) => <li key={i}>{item}</li> )}
</ul>
```

Introduction à Babel.js

- **Définition :** *C'est une bibliothèque Open Source JavaScript servant de transcompilateur permettant transformer un code initial en ES6 (et supérieur) ou incluant du JSX en un code final en ES5 compatible et utilisable avec la plupart des navigateurs Web.*
- C'est un composant indispensable quand on fait du développement en **React.js**
- Afin de pouvoir l'utiliser, il faut commencer par l'inclure :

```
<script
    crossorigin
    src="https://unpkg.com/babel-standalone@6.26.0/babel.min.js"
></script>
```

Introduction à Babel.js *(Suite)*

- Ensuite, il faut mettre le code à l'intérieur de :

*Spécifier qu'on va utiliser du code **JSX***

*Code **JSX***

```
<script type="text/babel">

    ReactDOM.render(
        <h1>Mon Titre en JSX</h1>,
        document.getElementById("react-container")
    );
</script>
```

- Résultat :

Navigateur Web

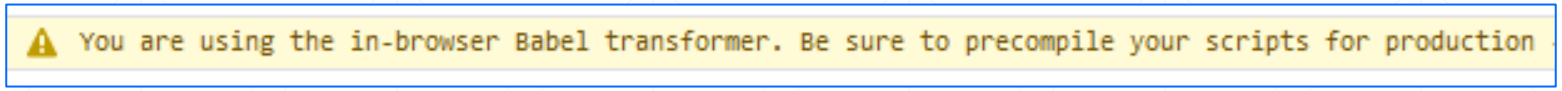
Mon Titre en JSX

```
<body>
  <div id="react-container">
    <h1>Mon Titre en JSX</h1>
  </div>
  <script crossorigin="" src="https://unpkg
```

Inspecteur de code

Introduction à `Babel.js` *(Suite)*

- Il ne faut pas utiliser ça en production, car ce n'est pas optimisé de faire la *transcompilation* à chaque fois :



Au niveau de la console

- Afin d'optimiser la version de production, il existe ce qu'on appelle des *module bundler*.

Le `Webpack.js` *vs.* `Vite.js`

- **Définition :** `Webpack.js` *et* `Vite.js` *sont des modules bundler ou en français des regroupeurs de modules. Il permettent de rassembler tout les fichiers statiques du projet (CSS, JavaScript & images) et de générer un seul fichier.*
- Il apportent deux principaux avantages :
  - La **rapidité d'accès** *(optimisation côté réseau)*,
  - La **modularité** *(gestion par bouts de code)*.
- **`create-react-app`** utilise **`Webpack.js`**,
- **`nano-react-app`** utilise **`Vite.js`**.
- Le premier (**`Webpack.js`**) sera plus optimal pour les projets de grande envergure tandis que le deuxième (**`Vite.js`**) est léger et rapide le rendant adéquat pour les petit projets.

Concrètement dans React.js

- **Déclaration :**

```
const ListItem = ({ Title, Items }) => (
<div className="liste">
    <h1>{Title}</h1>
    <ul>
    {Items.map((item, i) => ( <li key={i}>{item}</li> ))}
    </ul>
</div>
);
```

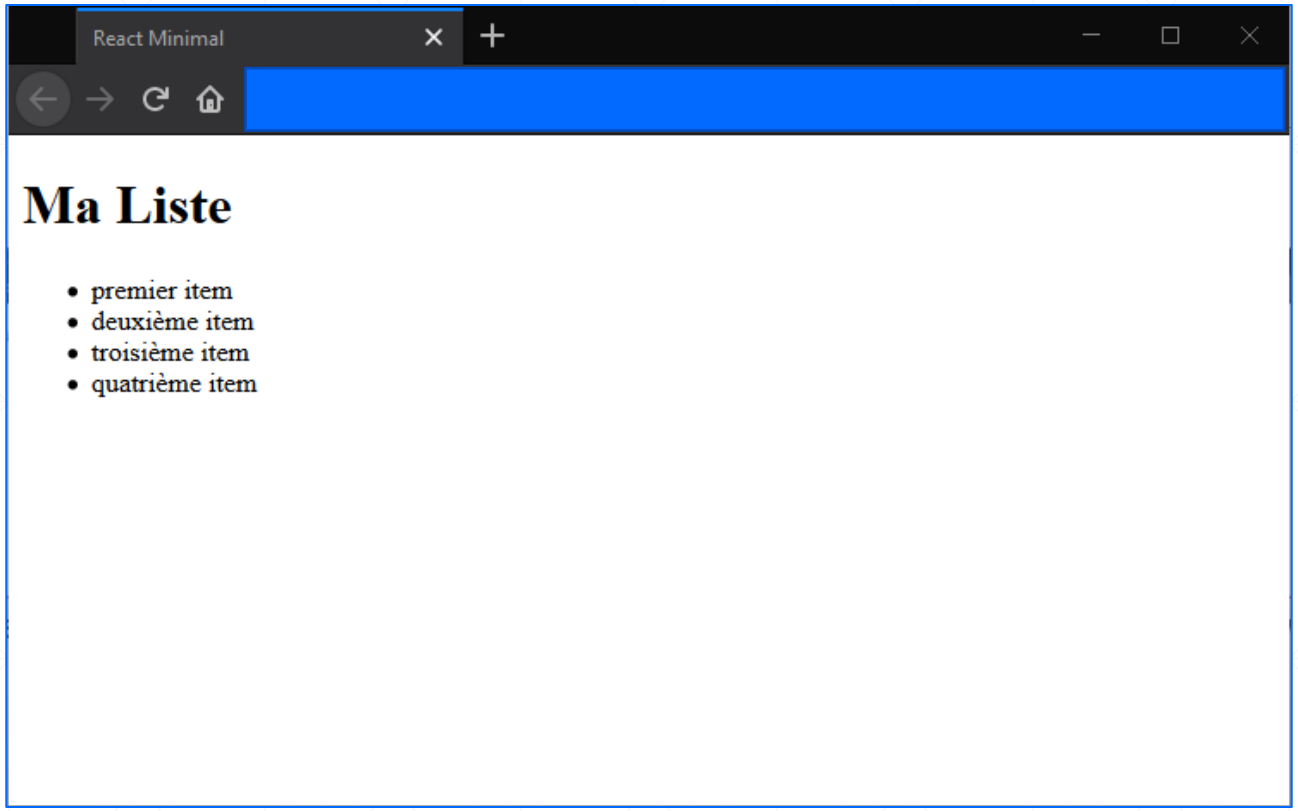
- **Appel :**

Faire appel au composant

Inclure des données

```
ReactDOM.render(
<ListItem Title="Ma Liste" Items={dataElement} />,
document.getElementById("react-container")
);
```

Concrètement dans React.js [Résultat]



UQAR

Propriétés, États & Arborescence de Composants

UQAR

Plan

1. Les propriétés ou **`props`**
2. Les références ou **`refs`**
3. Flux de données inverse
4. Les états ou **`states`**
5. Questions et discussion

Les propriétés ou props

- **Définition :** *Ils sont considérés comme des entrées, des arguments ou paramètres liés aux composants.*
- **Objectif :** *Le principal objectif est de pour voir transmettre des données aux composant de façon efficace.*
- **Exemple** *(tiré de la ToDoList)* **:**

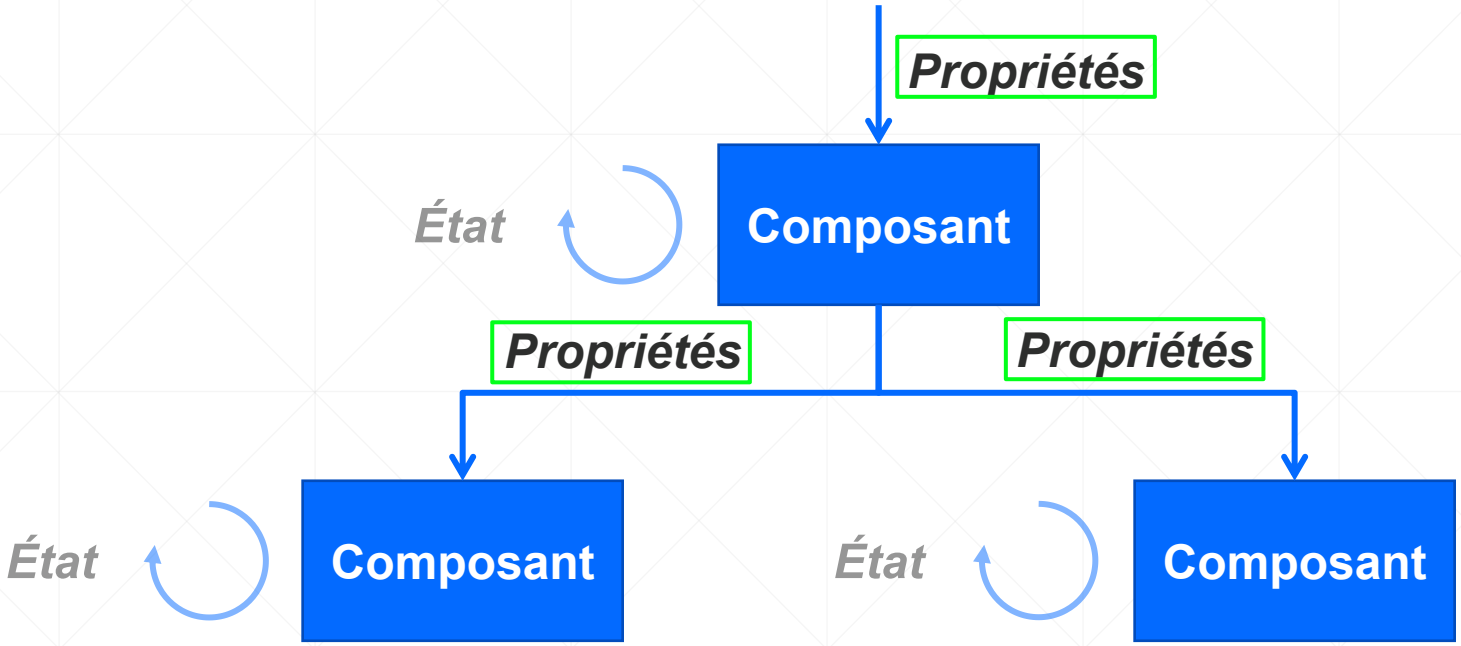
Propriétés

```
const Task = ({ title }) => (
    // code composant
);
```

```
const Task = ( props ) => (
    // code composant
);
```

UQAR

Les propriétés ou props



Validation des propriétés

- **Problématique :** *Le langage JavaScript n'est pas typé et par conséquent beaucoup de erreurs et bugs sont liés au type de variable utilisé.*
- **Solution :** *React.js fournit des outils afin d'effectuer la validation des propriétés avant leur utilisation dans des composants.*

Type	Validateur
Tableau	`PropTypes.array`
Booléen	`PropTypes.bool`
Fonctions	`PropTypes.func`
Nombres	`PropTypes.number`
Objets	`PropTypes.object`
Chaînes de caractères	`PropTypes.string`

```
npm install prop-types
```

Les propriétés – *Validation*

- **Important :** *Elle se fait en dehors de la déclaration du composant.*
- **Prérequis :** *Il faut importer*

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

Déclaration

```
const Task = ({ title, status }) => (
    // code composant
);
```

Booléen

Chaîne de Caractères

Validation

```
Task.propTypes = {
  title: PropTypes.string.isRequired,

  status: PropTypes.bool,
};
```

Obligation de fournir une valeur

Les propriétés – *Valeurs par défaut*

- **Important :** *Elle se fait en dehors de la déclaration du composant.*

Déclaration

```
const Task = ({ title, status }) => (
    // code composant
);
```

Valeur par défaut

```
Task.defaultProps = {
  title: "Ma Tâche",

  status: false,
};
```

Les références ou refs – Fonctions flèches

- **Définition :** *C'est une fonctionnalité permettant à un composant de communiquer avec les autres composants enfants. Elles sont souvent utilisé dans le cadre des formulaires.*

Déclaration de la variable

Déclaration de la fonction

Récupération de la valeur

Lier la variable avec une référence

Appel de la fonction

```
const AddTask = () => {
  let _titre;

  const add = () => {
    alert(`Les titre est ${_titre.value}`);
  }

  return (
    <div>
      <input ref={input => _titre = input} type="text" />

      <button onClick={add}>Ajouter</button>
    </div>
  )
};
```

Les références ou refs – Classes ES6

Lier la méthode au composant

Déclaration de la méthode

Lier la variable avec une référence

Appel de la fonction

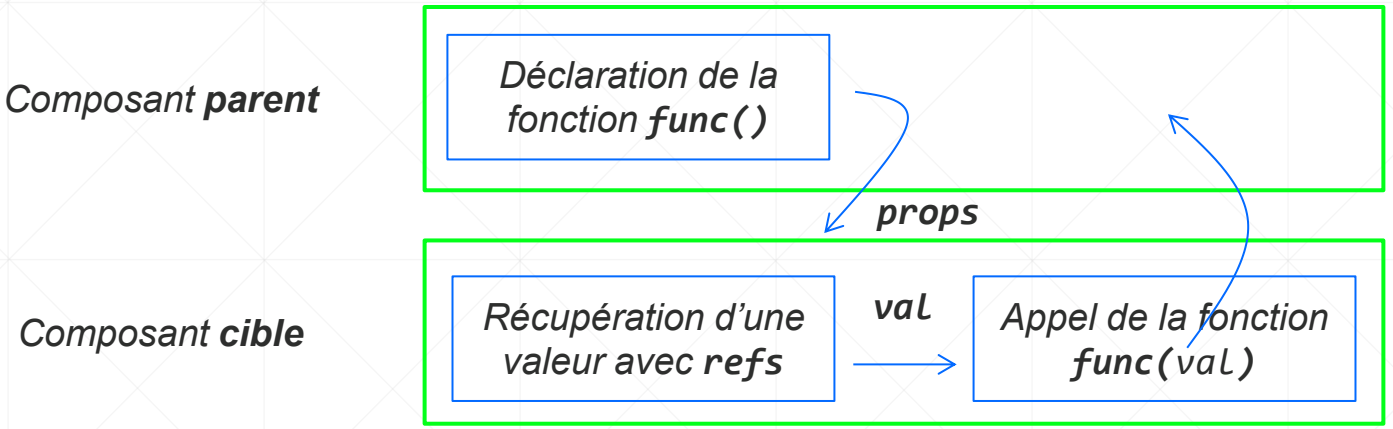
```
class AddTask extends React.Component {
  constructor(props) {
    super(props);
    this.add = this.add.bind(this);
  }

  add() {
    const { _titre } = this.refs;
    alert(`Les titre est ${_titre.value}`);
  }

  render() {
    return (
      <div>
        <input ref="_titre" type="text" />
        <button onClick={this.add}>Ajouter</button>
      </div>
    );
  }
}
```

Flux de données inverse

- **Problématique :** *On a vu comment récupéré une valeur d'un champs de formulaire, mais comment le transférer la valeur vers un composant parent par exemple ?*
- **Solution :** *Il est possible de transférer la valeur en utilisant un propriété de type fonction qui sera transmise au composant cible par son parent.*



Flux de données inverse – Classes ES6

*Composant **Parent***

```
affichage(titre) {
  alert(`Les titre est ${titre}`);
}
```

```
<AddTask onNewTask={affichage} />
```

```
this.props.onNewTask(_titre.value);
```

*Composant **Cible***

Flux de données inverse – Fonctions flèches

*Composant **Parent***

```
affichage(titre) {
  alert(`Les titre est ${titre}`);
}
```

```
<AddTask onNewTask={affichage} />
```

```
props.onNewTask(_titre.value);
```

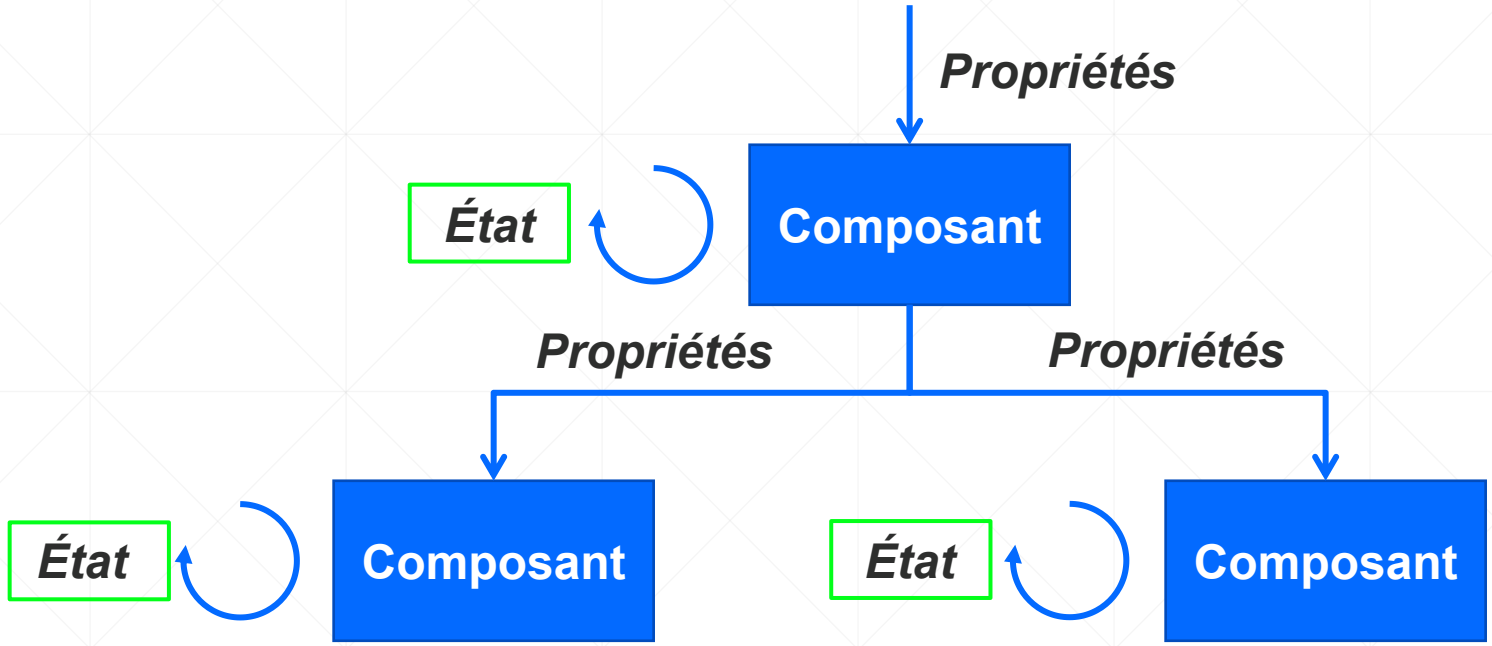
*Composant **Cible***

UQAR

Les états ou states

- **Contexte :** *Les propriétés permettent une certaine gestion des données, mais elle sont immuables (ne changes pas) après la génération du composant.*
- **Problématique :** *Comment gérer des données changeantes au sein d'un composant ?*
- **Solution :** ***React.js*** *fournit un option permettant de gérer les données dynamiques. Les changements sont automatiquement détectés et une mise à jour de l'interface est appliqué en conséquence.*
- **Important :** *Les états ne peuvent être utilisés qu'avec les* ***Classes ES6****, les fonctions flèches ne sont pas compatibles d'où le nom :* ***stateless component*** *ou* ***composant sans état****.*
- **Objectif :** *Centraliser les données de l'application.*

Les états ou states



UQAR

Méthodes liées aux états

```
constructor(props) {
  super(props);
  this.state = { key: value };
}
```

Initialisation de l'état

```
componentDidMount() {
  // code
}
```

Action lors du montage du composant

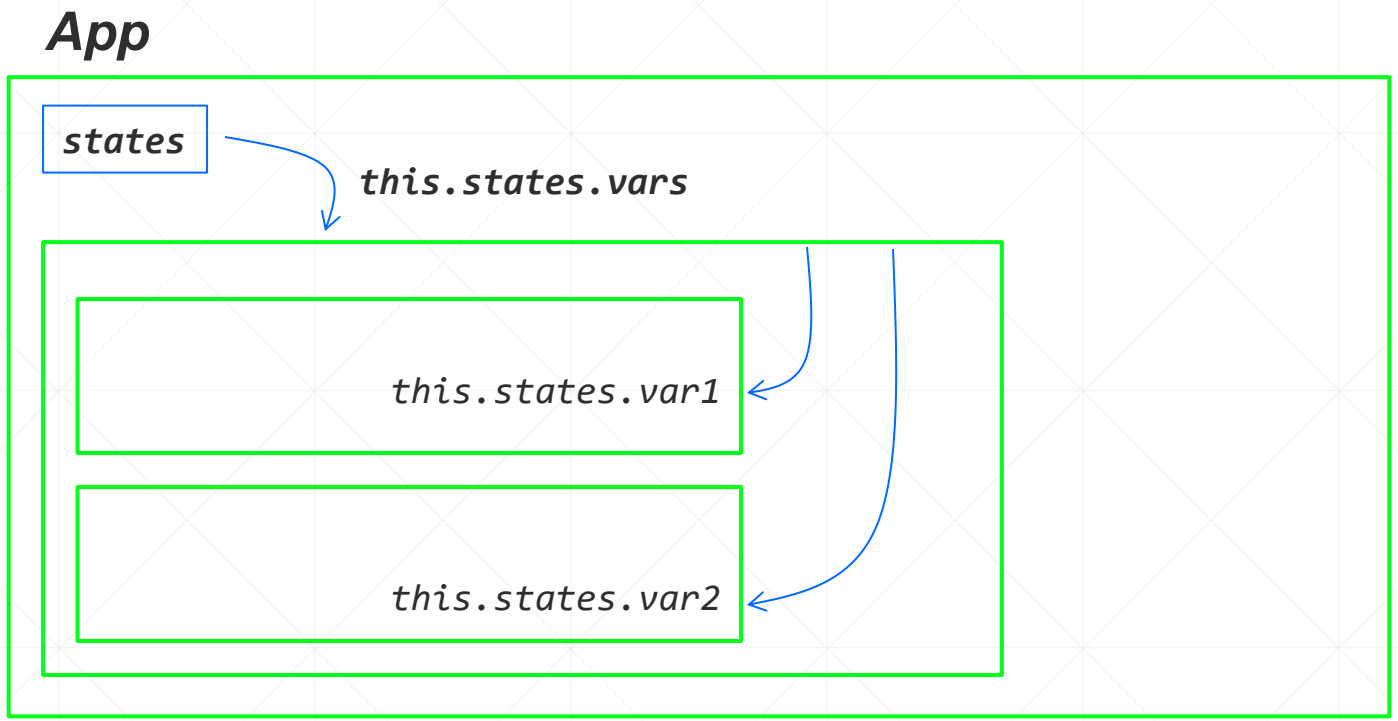
```
componentWillUnmount() {
  // code
}
```

Action lors du démontage du composant

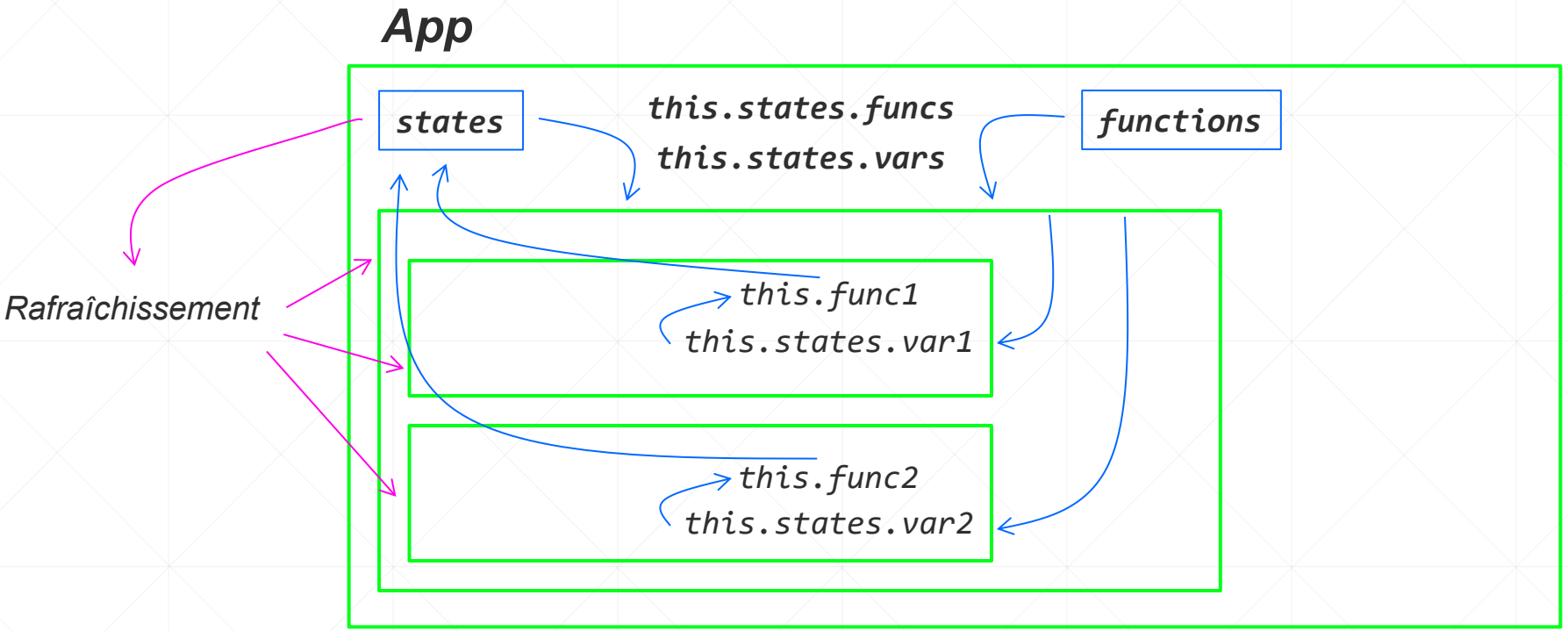
```
changementEtat() {
  this.setState({ key: newValue });
}
```

Changement de l'état

Changement des états – *vers le bas*



Changement des états – *vers le haut*



UQAR

Navigation & Routage

UQAR

Plan

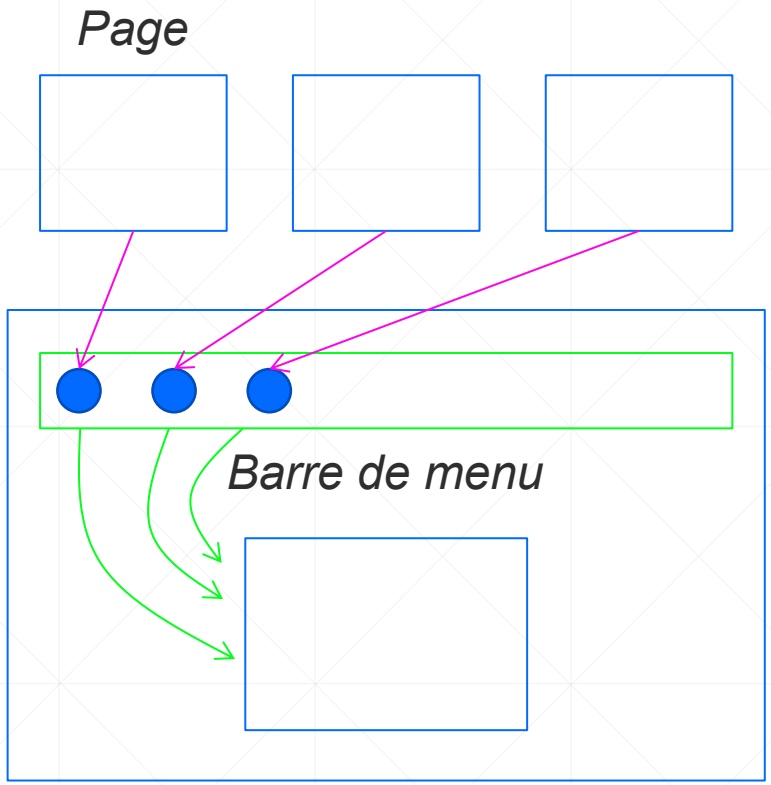
1. Introduction & Contexte
2. Installation de **`react-router`**
3. Simple mise en application
4. Les propriétés de routage
5. Utilisation de *paramètres*
6. Ajouter une barre de menu
7. Questions et discussion

UQAR

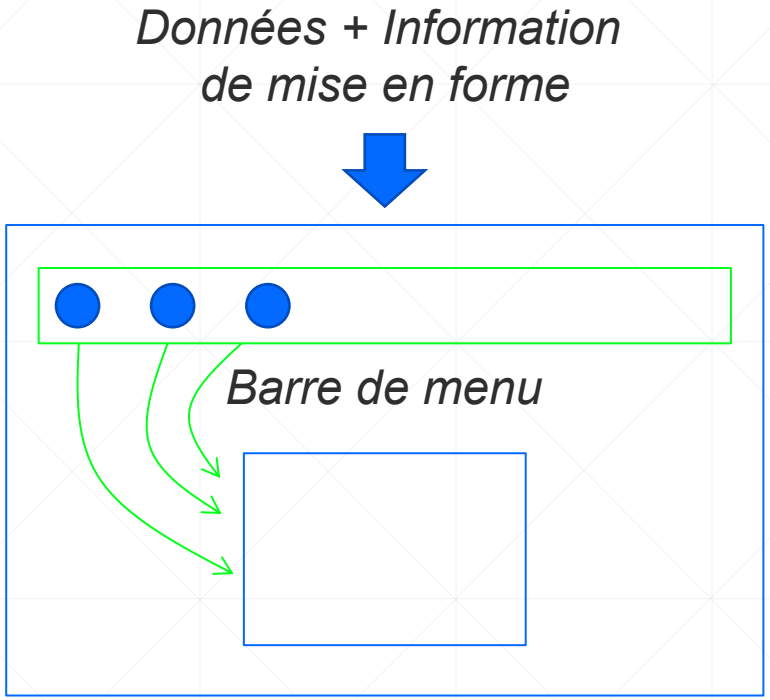
Introduction & Contexte

- **Cas classique :**
  - *Les applications Web classiques sont multipages,*
  - *Chaque requête avec une certaine URL permet de récupérer une page spécifique.*
- **Cas des SPA :**
  - *Les **SPA** génèrent le rendu de façon dynamique avec JavaScript,*
  - *C'est un uniquement les données qui sont mises à jour à partir du serveur.*
- **Problématique :** *Comment implémenter un système de navigation dans le cas d'une **SPA** en **React.js**?*
- **Solution :** *Utilisation du module **`react-router`**.*

Introduction & Contexte



Cas Classique



Cas **SPA**

Installation de `react-router`

Afin d'installer **`react-router`**, ils suffit de lancer la commande :

```
npm install react-router-dom
```

Simple mise en application

Afin de créer un simple système de routage, on va suivre les étapes suivantes :

1. Import les bons composants :

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

2. Création des différents composants ou pages cibles,

3. Ajouter des *routes* :

Chemin en URL

Composant cible

Containeur de routes

Route

```
<BrowserRouter>
  <Routes>
    <Route exact path="/" element={<Home />} />
    <Route path="/products" element={<Products />} />
  </Routes>
</BrowserRouter>;
```

UQAR

Les propriétés de routage

`react-router` offre la possibilité de gérer les erreurs dans l'**URL** :

1. Import les bons composants :

```
import { BrowserRouter, Route, Routes } from "react-router-dom";
```

2. Ajouter un composant pour gérer l'erreur (*404 not found*) :

3. Ajouter la *route* :

```
<BrowserRouter>
  <div>
    <Routes>
      <Route exact path="/" element={<Home />} />
      <Route path="/products" element={<Products />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
  </div>
</BrowserRouter>;
```

```
{props.location.pathname}
```

Propriété sur l'URL entré par l'utilisateur

UQAR

Utilisation de *paramètres*

Il est possible de récupérer des paramètres à partir d'une URL :

1. Dans la définition de la route :

```
<Route path="/products/:name/:city" element={<Products />} />
```

1er paramètre *2ème paramètre*

2. Dans l'URL saisir : …/products/**yacine**/**quebec**

3. Afin de récupérer la valeur dans le composant :

```
{props.match.params.name}
```

```
{props.match.params.city}
```

UQAR

Ajouter une barre de menu

Afin d'ajouter une barre de menu pour faciliter la navigation :

1. Il faut qu'elle soit à l'intérieur de : `<BrowserRouter>`
2. Importer le composant à partir **`react-router`** :

```
import { Link } from "react-router-dom";
```

3. Sur chaque élément du menu :

```
<ul>
  <li>
    <Link to="/">Accueil</Link>
  </li>
  <li>
    <Link to="/Products">Produits</Link>
  </li>
</ul>
```

UQAR

Questions & Discussion

UQAR

Bibliographie

1. Templier, Thierry & Gougeon, Arnaud (2007). JavaScript pour le Web 2.0 Programmation objet, DOM, Ajax, Prototype, Dojo, Script.aculo.us, Rialto. Éditions Eyrolles.
2. Porteneuve, Christophe (2008). Bien développer pour le Web 2.0 : Bonnes pratiques Ajax. Éditions Eyrolles.
3. Engels, Jean (2012). HTML5 et CSS3 : Cours et exercices corrigés. Éditions Eyrolles.
4. Martin, Michel (2014). HTML5, CSS3 & jQuery : Créez votre premier site web. Éditions Pearson.