REACT MASTERCLASS

La Révolution de l'Interface Utilisateur : Du Virtuel DOM aux Server Components. Une exploration complète et didactique.

INGÉNIERIE WEB MODERNE

01. Philosophie & Concepts

Comprendre pourquoi React a redéfini les standards du développement web moderne.

QU'EST-CE QUE REACT ?

Une Bibliothèque, pas un Framework

Créé par **Facebook en 2013**, React se concentre exclusivement sur la couche d'affichage (le View du modèle MVC).

📦 Il offre une flexibilité totale en s'associant à l'écosystème de votre choix.

L'Approche Écosystème

À l'inverse de frameworks monolithiques, React favorise une intégration à la carte : routage, state management et outils de build.

🔄 Utilisé par Netflix, Airbnb, Instagram et des millions de développeurs.

LE PARADIGME DÉCLARATIF

Impératif vs Déclaratif

Impératif : Écrire chaque étape de manipulation du DOM (ex: Vanillla JS `appendChild`).

⚠️ Complexe à maintenir et propice aux bugs.

L'Élégance Déclarative

Déclaratif : Décrire simplement l'état final attendu pour votre interface graphique.

🪄 React se charge d'orchestrer efficacement les mises à jour physiques.

LE TOUT-COMPOSANT

Réutilisabilité

Découper l'interface en blocs indépendants et autonomes pour éviter toute duplication de code.

Encapsulation

Chaque composant gère sa propre logique métier, son style graphique et son cycle de vie interne.

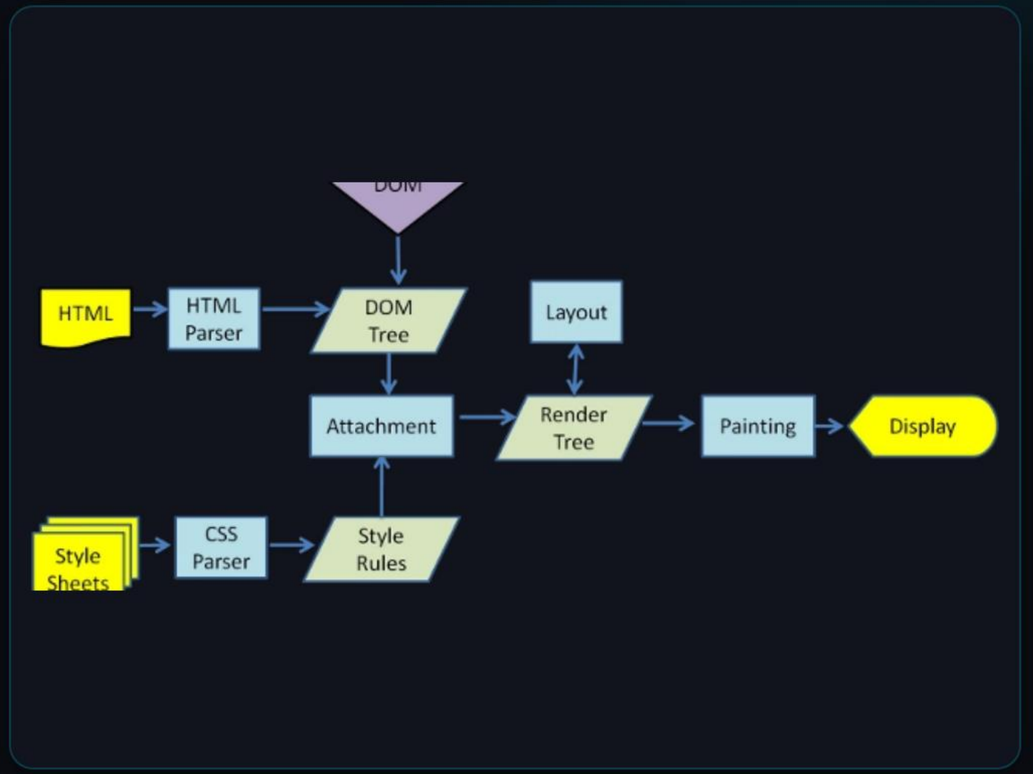
Composabilité

Assembler de petits composants simples pour ériger des architectures logicielles complexes et robustes.

LE VIRTUAL DOM EXPLIQUÉ

Modifier le DOM d'un navigateur est une opération matériellement lourde. React résout ce problème avec le **Virtual DOM** :

- **Copie en mémoire :** React maintient une représentation légère du DOM.
- **Diffing :** Lors d'un changement d'état, il compare l'ancien et le nouveau DOM virtuel.
- **Reconciliation :** Seuls les éléments modifiés sont appliqués au vrai DOM.



JSX : HTML DANS LE JS

Une Extension de Syntaxe

JSX n'est ni du HTML, ni une chaîne de caractères. C'est une extension syntaxique de JavaScript.

Le compilateur (Babel / SWC) transforme le JSX en appels de fonctions classiques `React.createElement()`.

🛡️ Protection native contre les injections de code (XSS).

```
const Element = () ⇒ { const user = "Didactique"; return (  className="card"> Bonjour {user} Le futur du web est ici.
```

02. Prérequis & Setup

Du poste de travail local au déploiement d'une structure projet moderne.

LES PRÉREQUIS DU DÉVELOPPEUR

- **Node.js & NPM :** Nécessaire pour installer les dépendances et exécuter le serveur local (Recommandé: Node 20+ LTS).
- **JavaScript Moderne (ES6+) :** Maîtriser l'affectation par décomposition (destructuring), les fonctions fléchées, les modules ES, et les méthodes de tableaux (`map`, `filter`).
- **Le Gestionnaire de Packages :** Utilisation recommandée de **pnpm** ou **npm** pour la gestion des paquets de production.

VITE.JS : LE STANDARD LOCAL

Pourquoi Vite.js ?

Vite a remplacé le vieillissant `create-react-app`.

Il s'appuie sur les modules ES natifs du navigateur pour offrir un démarrage instantané et un rechargement à chaud (HMR) ultra-rapide.

⚡ Temps de

Commandes de génération rapide :

```
# Initialiser un projet avec Vite npm create vite@latest mon-app --template react-ts # Naviguer et installer les dépendances cd
```

REACT AU-DELÀ DU SINGLE-PAGE

Next.js

Le framework d'entreprise recommandé par l'équipe React pour le rendu hybride (SSR, SSG) et le routage par fichiers.

Remix

Framework moderne axé sur les standards du web, la gestion fluide des formulaires et une gestion optimisée du cache.

React Native

Permet d'utiliser le paradigme React pour compiler de véritables applications mobiles natives iOS et Android.

ANATOMIE D'UN PROJET VITE

Voici la structure clé d'un projet React moderne généré sous Vite.js :

```
mon-app/ ├─ node_modules/ ├─ public/ ├─ src/ | ├─ assets/ | ├─ components/ | ├─ App.tsx | ├─ ind
```

- **src/ :** Dossier contenant l'intégralité du code source.
- **main.tsx :** Le point d'entrée qui monte l'application dans le DOM réel.
- **App.tsx :** Le composant racine global

03. Vos Premiers Pas

Apprendre l'écriture didactique de composants réutilisables.

VOTRE PREMIER COMPOSANT

Composant Fonctionnel
En modern React, un composant est simplement une fonction JavaScript qui retourne du JSX.

La première lettre du nom du composant doit impérativement être en **majuscule** pour être identifiée par React.

```
import React from 'react'; export const WelcomeBanner = () ⇒ { return (  className="banner"> Didactique & Moderne Bienvenue d
```

PROPS : PARAMÉTRER LE RENDU

Flux de Données Unidirectionnel

Les **Props** sont les paramètres passés à un composant parent vers un enfant.

Elles sont **immuables** (en lecture seule) pour garantir la prévisibilité du rendu graphique.

↓ Les données descendent de

```
// Déclaration de l'enfant avec Props const UserCard = ({ name, role }) ⇒ { return (  {name} Rôle : {role}  ); }; // Utilisat
```

STATE : LA MÉMOIRE INTERNE

Rendre l'interface réactive

Le **State** (État) représente les données qui peuvent évoluer au fil du temps dans votre composant.

Chaque modification de l'état déclenche automatiquement un **nouveau rendu** du composant et de ses enfants.

```
import { useState } from 'react'; const Counter = () ⇒ { // Déclaration du state const [count, setCount] = useState(0); ret
```

On utilise pour

GESTION DES ÉVÉNEMENTS

Événements Synthétiques

React utilise un système d'événements standardisé (SyntheticEvents) enveloppant le comportement natif des navigateurs.

Les noms d'événements s'écrivent en **camelCase** (ex: `onClick`, `onChange`, `onSubmit`).

</> On passe

```
const FormButton = () ⇒ { const handleClick = (e) ⇒ { e.preventDefault(); alert("Action interceptée !"); }; return (  onCl
```

04. Les Hooks Fondamentaux

Manipuler l'état et les effets secondaires de manière purement fonctionnelle.

LA RÉVOLUTION DES HOOKS

Règles Strictes

Les Hooks doivent impérativement être appelés au niveau racine de vos fonctions. Jamais dans des boucles ou des conditions.

Zéro Classes

Ils permettent de profiter de toutes les fonctionnalités de React (état, cycle de vie) sans écrire une seule classe JavaScript.

Partage de Logique

Facilité extrême pour extraire et réutiliser de la logique métier sous forme de fonctions personnalisées (Custom Hooks).

USESTATE : FOCUS DIDACTIQUE

Comment ça marche ?
L'appel à `useState(valeurInitiale)` renvoie un tuple contenant :

- **La variable :** La valeur courante de l'état.
- **Le setter :** La fonction pour modifier cette valeur.

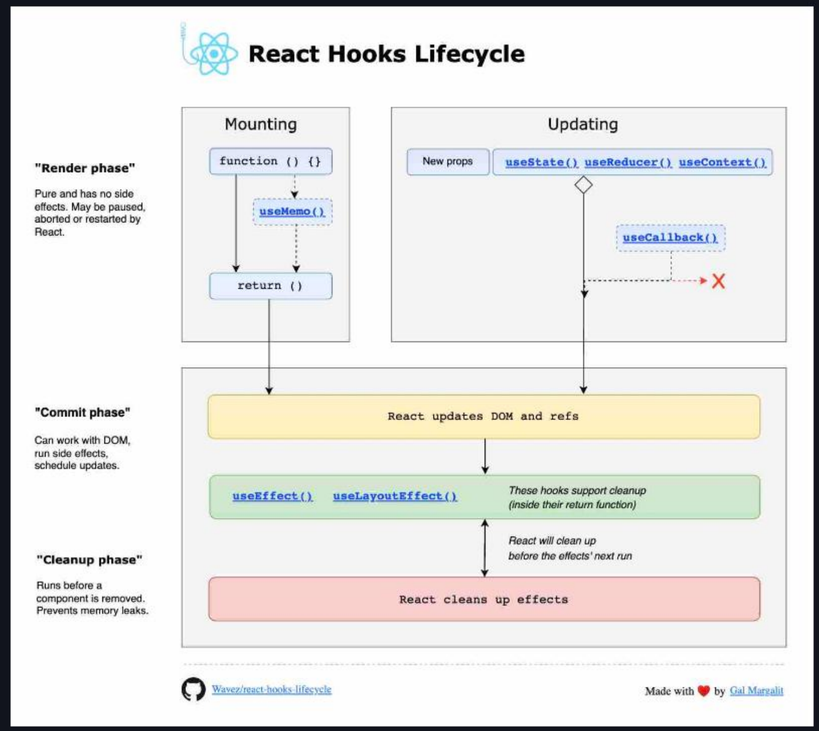
⚠ Ne modifiez jamais l'état directement ! Utilisez toujours le setter.

```
import { useState } from 'react'; export const ToggleComponent = () ⇒ { const [isOpen, setIsOpen] = useState(false
```

USEEFFECT : EFFETS SECONDAIRES

Pour interagir avec le monde extérieur (requêtes API, abonnements, timers), on s'appuie sur le cycle de vie de **useEffect** :

- **Tableau de dépendances vide `[]` :** S'exécute une seule fois au montage du composant.
- **Avec dépendances `[dep]` :** S'exécute à chaque fois que la variable spécifiée change.
- **Fonction de nettoyage (Cleanup) :** Retournée pour nettoyer les abonnements (ex: `clearInterval`).



USECONTEXT : ÉVITER L'ENFER

Le problème : Prop Drilling

Devoir faire transiter une donnée (ex: le thème ou l'utilisateur connecté) à travers 10 niveaux de composants intermédiaires inutiles.

Rend le code verbeux et difficile à factoriser.

La solution : React Context

Permet de créer un conteneur global accessible instantanément par n'importe quel composant de l'arbre, sans intermédiaire.

Recommandé pour les préférences ou états globaux.

CRÉER UN HOOK CUSTOMISÉ

Factoriser la Logique

Un Custom Hook est une fonction dont le nom commence par **use** et qui appelle d'autres hooks.

Il permet d'isoler des règles métier (ex: appels réseau, géolocalisation) pour les réutiliser dans toute l'application.

```
function useToggle(initialVal = false) { const [value, setValue] = useState(initialVal); const toggle = () ⇒ setValue(!value)
```

05. Formulaires & Données

Gérer la saisie utilisateur et intégrer des sources d'informations distantes.

CONTROLLED VS UNCONTROLLED

Composant Contrôlé (Recommandé)
React est l'unique source de vérité. L'état synchronise chaque frappe clavier via la propriété `value` et l'événement `onChange`.

✓ Permet la validation instantanée des champs de saisie.

```
const [name, setName] = useState(""); return (  type="text" value={name} onChange={(e) ⇒ setName(e.target.value)} /> );
```

REQUÊTES API AVEC FETCH

Cycle d'acquisition standard

Récupérer des données distantes depuis une API s'implémente généralement via un appel `fetch` dans un `useEffect` au montage du composant.

```
useEffect(() ⇒ { // Appel asynchrone sécurisé fetch('https://api.github.com/users/octocat') .then(res ⇒ res.json()) .then(data
```

🔄 Pensez à toujours afficher un indicateur de

ÉCOSYSTÈME STATE MANAGEMENT

Zustand

La solution moderne, ultra-légère et performante qui a largement détrôné Redux pour les applications standard.

React Query

Incontournable pour synchroniser les états serveurs, gérant nativement la mise en cache, la ré-acquisition et la pagination.

Redux Toolkit

Historique et standard pour les applications d'entreprise d'envergure ayant des flux de données extrêmement complexes.

06. Applications & Écosystème

Découvrir l'omniprésence de React dans l'ingénierie logicielle contemporaine.

REACT FACE AUX AUTRES SOLUTIONS

Technologie	Type d'outil	Courbe d'apprentissage	Point fort majeur
React	Bibliothèque UI	Modérée	Écosystème massif & flexibilité totale
Angular	Framework Complet	Élevée (TypeScript strict)	Structure robuste d'entreprise
Vue.js	Framework Progressif	Faible à Modérée	Simplicité d'écriture & intégration
Svelte	Compilateur UI	Faible	Pas de Virtual DOM, vitesse brute maximale

L'ÈRE DES SINGLE PAGE APPS

Avantages Majeurs

Fluidité de navigation comparable à une application de bureau. Pas de rechargement global de page.

✔ Expérience utilisateur immersive et chargement de composants à la demande.

Inconvénients Communs

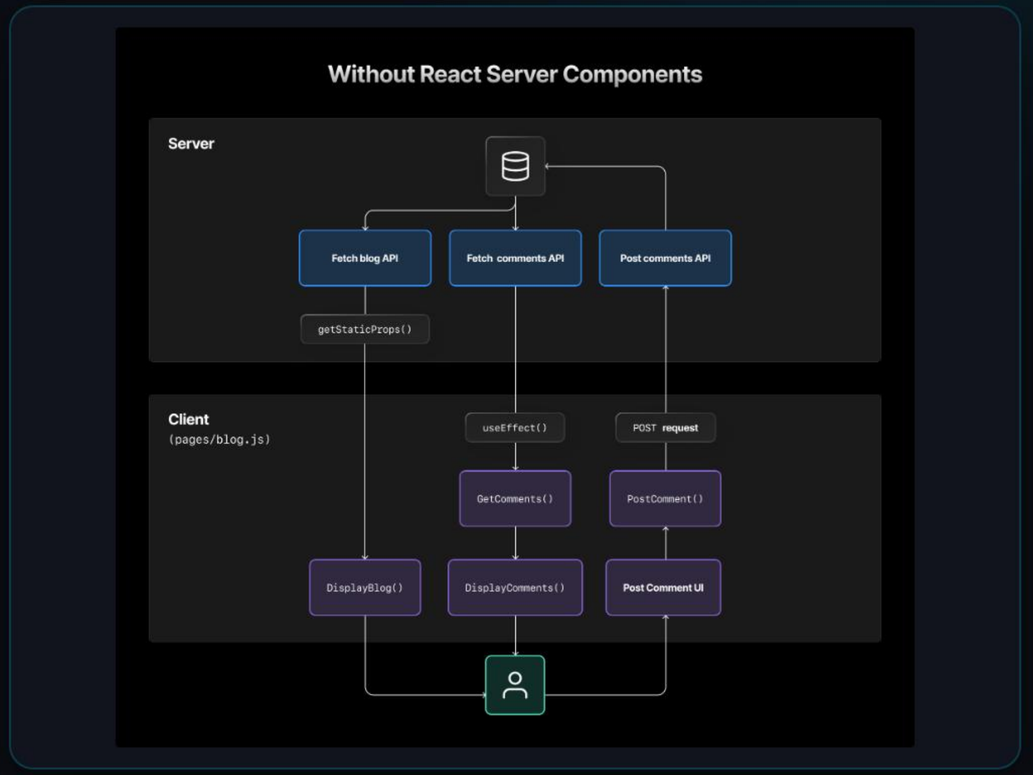
Indexation difficile pour les moteurs de recherche (SEO) et premier chargement (bundle initial) lourd.

⚠ Nécessite l'adoption de stratégies SSR ou SSG.

SERVER COMPONENTS (RSC)

Introduits pour optimiser les performances, les **React Server Components** s'exécutent côté serveur :

- **Côté Serveur :** Zéro impact sur le poids du bundle JavaScript client.
- **Accès direct :** Peuvent requêter directement la base de données de manière asynchrone.
- **Client Components :** Toujours utilisés pour l'interactivité (boutons, formulaires).



REACT NATIVE : APPLICATIONS MOBILES

Code unique, plateformes multiples

React Native permet d'écrire du JavaScript tout en générant de réels éléments d'interface utilisateur natifs pour **iOS** et **Android**.

🚀 Accès complet aux capteurs de l'appareil (caméra, GPS, Bluetooth).

Économie d'échelle majeure

Jusqu'à 90% du code métier partagé entre les versions mobiles, réduisant drastiquement les coûts de développement et d'ingénierie.

👥 Choisi par Facebook, Discord, Tesla, et Skype.

07. Avancé & Futur (React 19)

Découvrir les innovations majeures de la dernière version majeure.

LA RÉVOLUTION REACT 19

React Compiler

Automatisation complète de la mémoïsation. Fini l'usage verbeux et manuel de `useMemo` et `useCallback`.

Actions natives

Support complet pour gérer les états d'attente (pending), d'erreur et de succès des formulaires asynchrones.

useHook API

Nouvel opérateur unifié capable de consommer des promesses ou du contexte de manière conditionnelle.

NOUVEAUTÉS DE SYNTAXE (USE)

Le Hook de Demain

La nouvelle fonction `use()` permet de charger une promesse directement au sein de votre boucle de rendu.

Elle remplace de manière élégante les multiples états d'attente codés manuellement, et s'intègre

```
import { use } from 'react'; const PostList = ({ dataPromise }) => { // Résolution directe de la promesse const posts = use(dat
```

BONNES PRATIQUES DE PRODUCTION

- **Keep State Local :** Évitez de globaliser des variables d'état qui n'impactent qu'un seul composant (limite les re-rendus).
- **Error Boundaries :** Sécurisez l'application en interceptant proprement les erreurs de rendu pour éviter de casser l'intégralité du DOM.
- **Lazy Loading :** Divisez le bundle final en utilisant `React.lazy()` et `Suspense` pour charger les routes lourdes à la demande.

FEUILLE DE ROUTE & DIFFICULTÉ

Concept Écosystème	Niveau de difficulté	Temps d'acquisition recommandé
Composants, Props & JSX	Facile	Quelques heures
Gestion de l'État & Events	Modérée	1 à 2 jours
Hooks Complexes (useEffect, custom)	Intermédiaire	1 semaine
Server Components & Architecture SSR	Avancé	Plusieurs semaines

RESSOURCES & DOCUMENTATION

La Documentation Officielle

Toujours privilégier la documentation officielle qui a été entièrement réécrite pour se concentrer sur les hooks et les bonnes pratiques modernes.

site : react.dev

GitHub : github.com/facebook/react

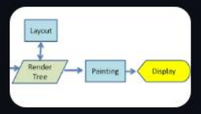
SESSION Q & A

react.dev/blog

architecture@expert-react.fr

> **La simplicité est la sophistication suprême. React nous a appris à ne plus manipuler les éléments un par un, mais à décrire simplement notre vision de l'interface graphique.**
>
> — Pensée collective de l'ingénierie logicielle moderne

IMAGE SOURCES



https://media2.dev.to/dynamic/image/width=1000,height=420,fit=cover,gravity=auto,format=auto/https%3A%2F%2Fdev-to-uploads.s3.amazonaws.com%2Fuploads%2Farticles%2Fx8katugqbphwou3vxz1x.png

Source: dev.to

https://repository-images.githubusercontent.com/196048036/5fae96d6-a1e5-43bc-8556-4ab9d83d4ff2

Source: github.com

https://images.ctfassets.net/e5382hct74si/kOA2be85nVBQhL0LA9zmO/71f1f53fbf206a18f791b38f8cb2fd6b/Without_React-1.png

Source: vercel.com

https://itforgehub.online/images/hero_main.jpg

Source: itforgehub.online