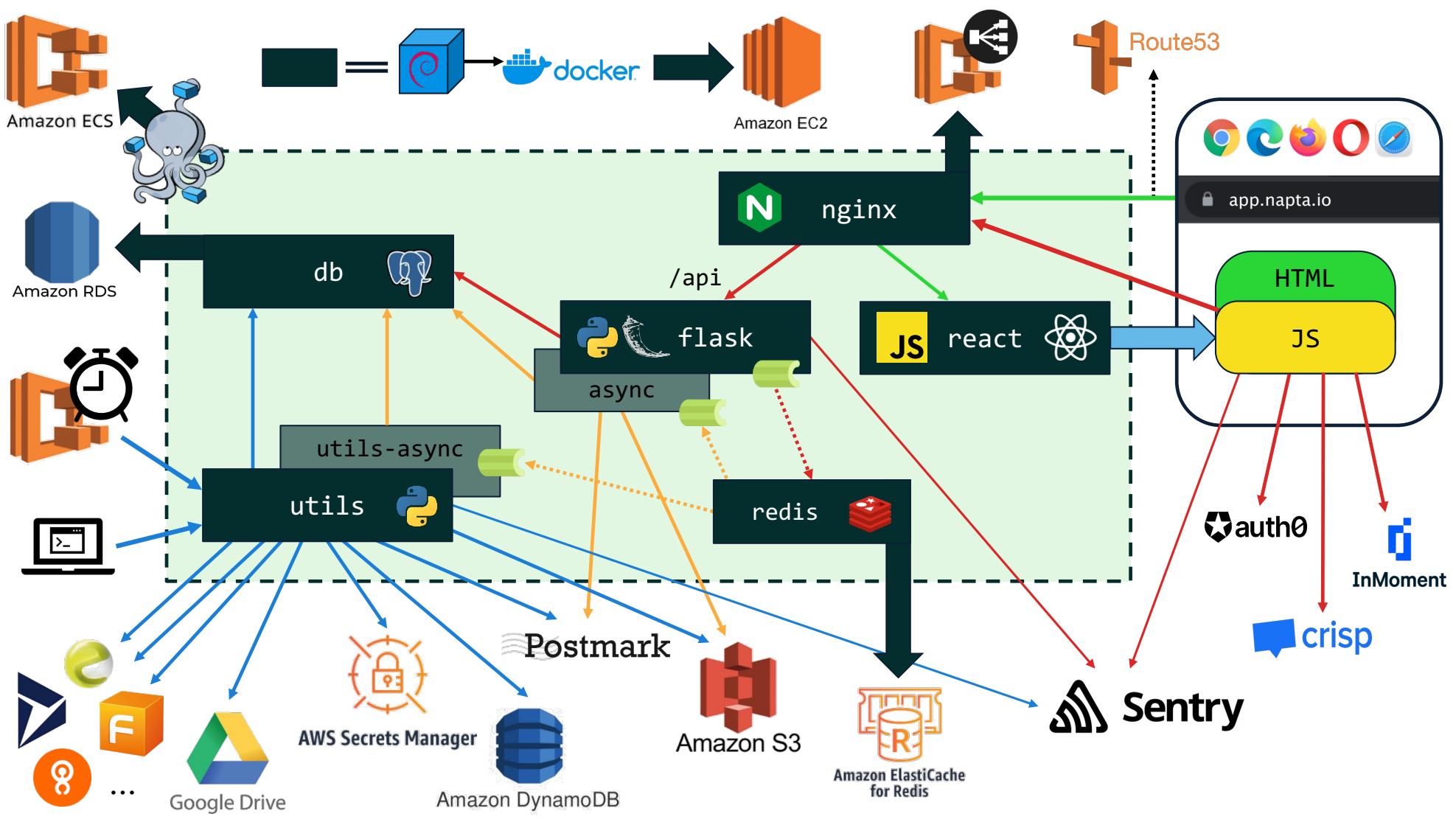
MobApp – Introduction à React Native

Laurent Daverio, Olivier Hermant

Centre de recherche en informatique
MINES Paris, Université PSL

Plan

1. Programmation Asynchrone en JS
2. React Native
3. JSX
4. Git
5. Hooks
6. Travaux pratiques
7. Requêtes
8. Année prochaine


Amazon ECS
docker
Amazon EC2
Route53
app.napta.io
nginx
db
Amazon RDS
/api
flask
async
react
HTML
JS
utils-async
utils
redis
auth0
InMoment
crisp
Postmark
AWS Secrets Manager
Amazon S3
Amazon ElastiCache for Redis
Sentry
Google Drive
Amazon DynamoDB

1. Programmation Asynchrone en JS

Manipuler les fonctions

- fonction = citoyen de 1^{e} classe = variable comme une autre
  - comme un entier, un tableau, une chaîne de caractères, etc

Manipuler les fonctions

- fonction = citoyen de 1^{e} classe = variable comme une autre
  - comme un entier, un tableau, une chaîne de caractères, etc
- **Question**. Qu'est-ce qui est une fonction ?

1. `alert`
2. `alert('Hello, World !')`
3. $\sin(\pi)$
4. sin
5. $x \mapsto \sin(x)$
6. `(x) => { alert(x) }`

Manipuler les fonctions

- fonction = citoyen de 1$^{\text{e}}$ classe = variable comme une autre
  - comme un entier, un tableau, une chaîne de caractères, etc
- **Question**. Qu'est-ce qui est une fonction ?

1.	`alert`	2.	
3.		4.	sin
5.	$x \mapsto \sin(x)$	6.	`(x) => { alert(x) }`

Manipuler les fonctions

- fonction = citoyen de 1^{e} classe = variable comme une autre
  - comme un entier, un tableau, une chaîne de caractères, etc
- **Question**. Qu'est-ce qui est une fonction ?

  1. `alert`
  2.
  3.
  4. sin
  5. $x \mapsto \sin(x)$
  6. `(x) => { alert(x) }`
- **fonction anonyme**
  - syntaxes

```
function (x,y) { /* du code */ }
   (x,y) => { /* du code */ }
```

  - et, sans les accolades

```
(x,y) => 3
```

    équivaut à `(x,y) => {return 3}`
- exemples :

```
const f = function (x,y) { /* du code */ }
const f = (x,y,z) => { /* de l'autre code */ }
```

JS et asynchrone

- `window.onload = f` : exécutera (futur) `f` à la fin du chargement de la page
  - crucial si `f` doit accéder à des éléments de la page
  - `f` est une fonction acceptant **1** argument
- "évidemment" `window.onlad = f(arg)` est incorrect (pourquoi ?)
- DEMO
- [page_dyn.html] du cours html/css/js

JS et asynchrone

- `window.onload = f` : exécutera (futur) `f` à la fin du chargement de la page
  - crucial si `f` doit accéder à des éléments de la page
  - `f` est une fonction acceptant **1** argument
- "évidemment" `window.onlad = f(arg)` est incorrect (pourquoi ?)
- DEMO
- [page_dyn.html] du cours html/css/js
- questions ?

2. React Native

Pourquoi React Native ?

- Cahier des charges : dev mobile, front-end 2023++

Pourquoi React Native ?

- Cahier des charges : dev mobile, front-end 2023++
- choix "naturels" : iOS $\Rightarrow$ Swift, Android $\Rightarrow$ Kotlin, Java.
  - [-] spécifique à **une** plateforme
  - [-] développement plus complexe
  - [+] contrôle total

Pourquoi React Native ?

- Cahier des charges : dev mobile, front-end 2023++
- choix "naturels" : iOS ⇒ Swift, Android ⇒ Kotlin, Java.
  - [−] spécifique à **une** plateforme
  - [−] développement plus complexe
  - [+] contrôle total
- React Native
  - [+−] modèle de programmation = logique & style
  - [+] plus facile d'accès
  - [+] programmation Web en "react" similaire
  - [−] liberté, contrôle
  - [−] couches empilées ⇒ +ressources (cf. VM, vrai autres choix)
  - [+] bien sur un CV (césure)
  - [+−] *à la mode*
  - [+] support, documentation, etc.

Pourquoi React Native ?

- Cahier des charges : dev mobile, front-end 2023++
- choix "naturels" : iOS ⇒ Swift, Android ⇒ Kotlin, Java.
  - [−] spécifique à **une** plateforme
  - [−] développement plus complexe
  - [+] contrôle total
- React Native
  - [+−] modèle de programmation = logique & style
  - [+] plus facile d'accès
  - [+] programmation Web en "react" similaire
  - [−] liberté, contrôle
  - [−] couches empilées ⇒ +ressources (cf. VM, vrai autres choix)
  - [+] bien sur un CV (césure)
  - [+−] *à la mode*
  - [+] support, documentation, etc.
- le front-end évolue **très** rapidement. Concurrence ?
  - Flutter, Ionic, votre techno préférée
  - tout ce qui est déjà passé de mode (ou que je ne connais pas)

3. JSX

JSX : JavaScript and XML

- tout est dans le titre : javascript, balises XML au milieu
- JS : pour la logique
- balises XML : pas ~~du HTML~~ (mais y ressemble)
  - composants React Native
  - servent au rendu
  - liées aux composants graphiques (et composants *natifs*) de l'appareil mobile
- possibilité de créer ses **propres balises**
  - en développant ses **propres fonctions**
  - qui retournent un rendu

Premier projet en React-Native

- installation de la stack technique (pour Android) : VM
  - autres architectures (M1) : voir Laurent et/ou ordinateurs de prêt (audiovisuel).

Premier projet en React-Native

- installation de la stack technique (pour Android) : VM
  - autres architectures (M1) : voir Laurent et/ou ordinateurs de prêt (audiovisuel).
- créer son projet, git clone proprement : cf. TP
- Deux styles de programmation en React Native
  - ~~avec des classes~~ ($<$ 2019)
  - avec des fonctions (notre choix)

Premier projet en React-Native

- installation de la stack technique (pour Android) : VM
  - autres architectures (M1) : voir Laurent et/ou ordinateurs de prêt (audiovisuel).
- créer son projet, git clone proprement : cf. TP
- Deux styles de programmation en React Native
  - ~~avec des classes~~ ($<$ 2019)
  - avec des fonctions (notre choix)
- demo time !

  `git clone https://gitlab.cri.ensmp.fr/hermant/kivappa`

npm et npx

- `Node.js` : plateforme logicielle en JS
- `npm` : gestionnaire de paquets (node package manager)
  - installation de librairies/paquets développés pour `Node.js`
  - et les paquets requis par ces paquets, etc (les *dépendances*).
  - ces dépendances sont (automatiquement) listées dans les fichiers `package.json` et `package-lock.json`
    - ne pas éditer ces fichiers à la main
- `npx` : exécute un paquet Node.js (node package execute)
  - `react-native` est l'un de ces paquets
  - il peut recevoir différentes *commandes* (`start`, `run-android`)

Hello World

Mode d'emploi :

1. fork, cloner et installer son projet (ou le créer)
2. connecter la tablette
3. démarrer le packager de React Native (`npx react-native start`)
4. construire le projet et le lancer (`npx react-native run-android`)

Tout se passe dans le fichier `App.js`:

- `imports`
- `export` du composant principal de la page,
- `KivAppA` : la fonction de rendu
  - retourne les composantes RN qui doivent être affichés
  - balises XML, avec propriétés
- [tag `01-hello`]

Programmer son propre composant

Fonction sans arguments :

- écrire une fonction qui retourne le composant `<Text>`
- appeler cette fonction “par le composant homonyme”
- [tag `02-hello`]

Fonction avec arguments :

- donnés dans les attributs de la balise XML
- **un seul** argument, `props`
- contiendra tous les attributs
- [tag `03-hello`]

NB :

- entre { `accolades` } , le code JS,
- entre { `accolades` } , dans du code JS, un objet en ligne,
- conséquence : parfois deux paires d’accolades

4. Git

Git et GitLab

User Story

En tant que développeur, je programme un composant RN, et je le partage avec mon groupe (et le prof).

- gitlab.cri.minesparis.psl.eu
- contiendra votre projet
- utilise le système de versionnement git
  - principe local–distant
- illustration :

  https://illustrated-git.readthedocs.io/en/latest/#working-with-remote-repositories

Git aujourd'hui

- un "repository" pour votre TP front-end, cloné d'un point de départ fourni par nous
- vous êtes seul contributeur (plus simple)
- processus :
  1. développer une fonctionnalité
  2. `git add`, puis `git commit` (local), puis `git push` ($\rightarrow$ distant)
- possibilité de collaborer avec soi-même

User Story 2

En tant que développeur, je développe sur 3 ordinateurs différents.

- `git pull` pour màj du repository local par rapport au distant.

5. Hooks

Ajouter un bouton : le hook d'état

User Story 3

En tant que'utilisateur, je clique sur le bouton, une variable s'incrémente et s'affiche.

- ajouter le composant `<Button>`, attribut `onPress`
- solution "naïve" ne fonctionne pas
- RN demande un `callback` (fonction asynchrone) pour `onPress`
  - ne marche toujours pas : modifications non prises en compte par le modèle RN

Ajouter un bouton : le hook d'état

User Story 3

En tant que'utilisateur, je clique sur le bouton, une variable s'incrémente et s'affiche.

- ajouter le composant `<Button>`, attribut `onPress`
- solution "naïve" ne fonctionne pas
- RN demande un `callback` (fonction asynchrone) pour `onPress`
  - ne marche toujours pas : modifications non prises en compte par le modèle RN
- utiliser le hook ($\approx$ feature RN) d'état
- changer la valeur avec `setCompteur` le re-rendu est automatisé
  - merci le framework RN (cf. semaine prochaine) et les hook d'état
  - [tag `04-button-incr`]

Ajouter un bouton : le hook d'état

User Story 3

En tant que'utilisateur, je clique sur le bouton, une variable s'incrémente et s'affiche.

- ajouter le composant `<Button>`, attribut `onPress`
- solution "naïve" ne fonctionne pas
- RN demande un `callback` (fonction asynchrone) pour `onPress`
  - ne marche toujours pas : modifications non prises en compte par le modèle RN
- utiliser le hook ($\approx$ feature RN) d'état
- changer la valeur avec `setCompteur` le re-rendu est automatisé
  - merci le framework RN (cf. semaine prochaine) et les hook d'état
  - [tag `04-button-incr`]
- on peut passer les fonctions de modification en tant

6. Travaux pratiques

7. Requêtes

Requêtes en React-Native

- utiliser `fetch`
- états pour passer les informations pertinentes
- callbacks pour la mise à jour
- et des jolis composants RN
- DEMO ([branche `devel`])
- revue de code

8. Année prochaine

Cours Front-End en 2023

- le 07/01/2023 : récapitulatif RN (états & fetch), formulaires propres, effets
- le 13/01/2023
  - **le modèle RN expliqué**,
  - authentification/login.
- le 23/01/2023 : à la demande.

Cours Front-End en 2023

- le 07/01/2023 : récapitulatif RN (états & fetch), formulaires propres, effets
- le 13/01/2023
  - **le modèle RN expliqué**,
  - authentification/login.
- le 23/01/2023 : à la demande.
- conséquence:

  NE PAS DÉVELOPPER CES FONCTIONALITÉS
  (authentification, login, droits) dans votre front-end

- autre conséquence : le TP et RN seront légèrement "magiques" aujourd'hui