XDNATIS

Traitements asynchrones

Michael

XDNATIS

XWIKI NATIS

Compétence demandée :
Comprendre les tâches asynchrones

XEBIA NATIS

1. Les tâches asynchrones
2. Les callbacks et les promesses
3. Les REST API
4. L'event-driven programming

XWIKI SAS

XWIKI NATIS

LES TACHES ASYNCHRONES

XWIKI NATIS

Normalement, le code d'un programme donné se déroule sans interruption, une seule chose se produisant à la fois.

Si une fonction dépend du résultat d'une autre fonction, elle doit attendre que l'autre fonction se termine

XWIKI NATIS

Il est inutile de rester assis à attendre quelque chose alors que vous pouvez laisser une tâche se dérouler sur un autre cœur de processeur et être averti quand elle a terminé.

XWIKI NATIS

Il est inutile de rester assis à attendre quelque chose alors que vous pouvez laisser une tâche se dérouler sur un autre cœur de processeur et être averti quand elle a terminé. Cela vous permet d'effectuer d'autres travaux en même temps, ce qui est la base de la programmation asynchrone.

XEBIA NATIS

Il est inutile de rester assis à attendre quelque chose alors que vous pouvez laisser une tâche se dérouler sur un autre cœur de processeur et être averti quand elle a terminé. Cela vous permet d'effectuer d'autres travaux en même temps, ce qui est la base de la programmation asynchrone.

XWIKI NATIS

Il est inutile de rester assis à attendre quelque chose alors que vous pouvez laisser une tâche se dérouler sur un autre cœur de processeur et être averti quand elle a terminé.

Cela vous permet de lancer plusieurs travaux sans attendre la fin du précédent.

XWIKI NATIS

Dans sa forme la plus élémentaire, JavaScript est un langage synchrone, bloquant et à un seul processus, dans lequel une seule opération peut être en cours à la fois.

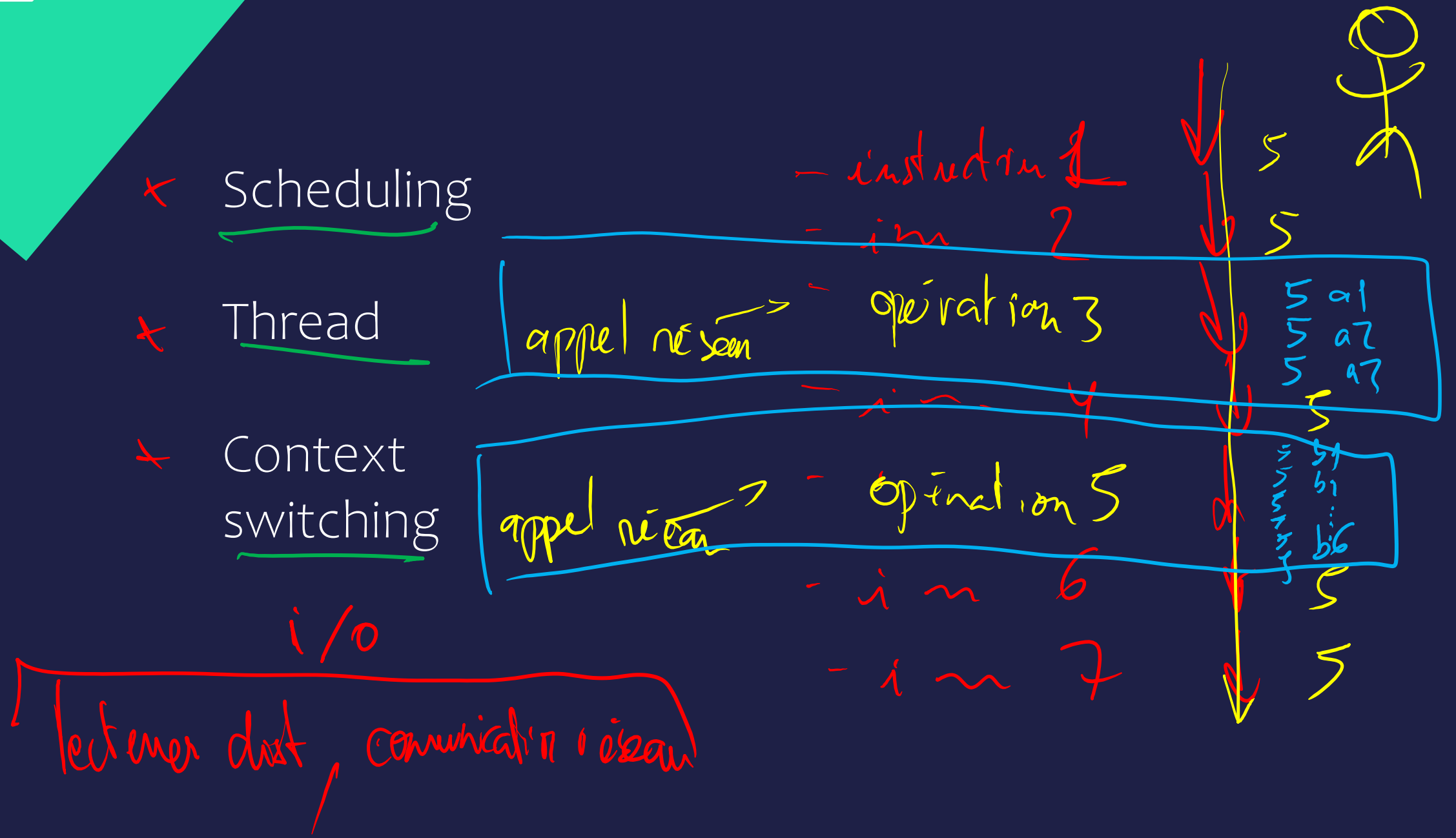
XEBIA NATIS

Attention, le asynchrone veut dire que l'on a pas besoin d'attendre la réponse d'un traitement avant d'en démarrer un autre, mais cela veut pas dire concurrent (faire les choses en parallèle) !

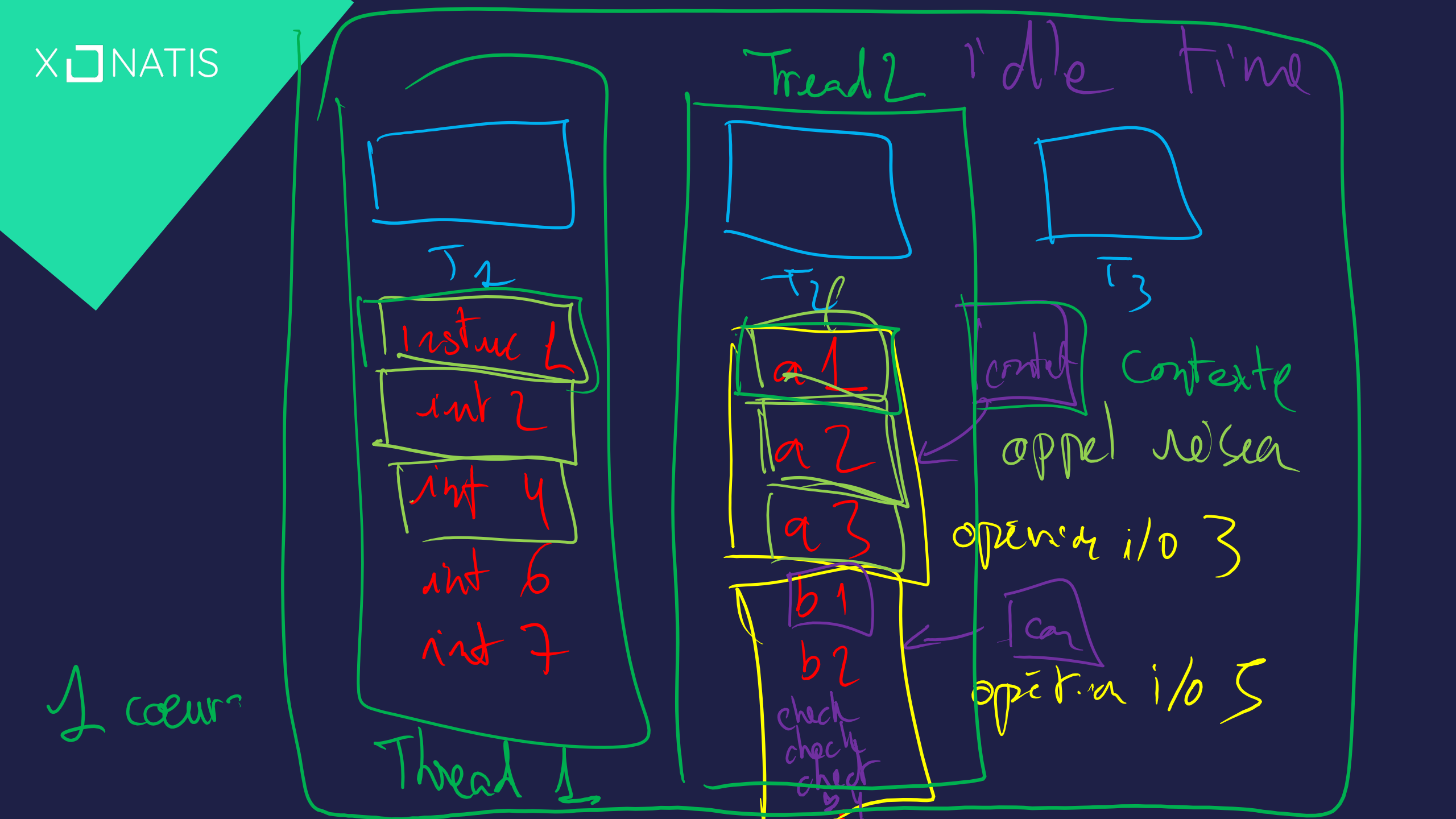
Le JavaScript reste mono-threadé (sauf exception avec les workers)

XEBIA

- Scheduling
- Thread
- Context switching



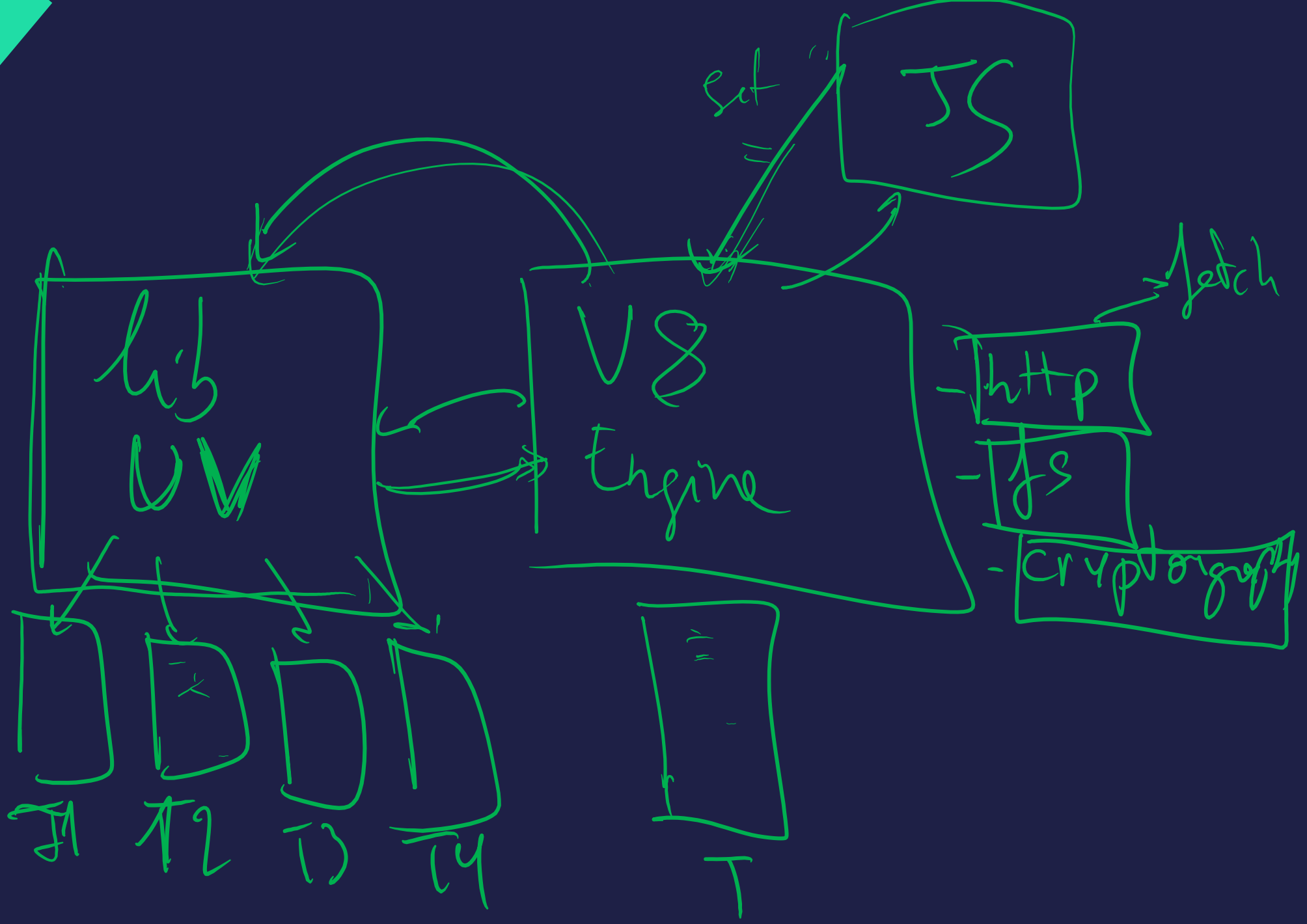
XDNATIS


Thread 2
idle time
T1
T2
T3
instruc 1
int 2
int 4
int 6
int 7
a 1
a 2
a 3
b 1
b 2
check
check
check
contxt
Contexte
appel réseau
opération i/o 3
can
opération i/o 5
1 coeur
Thread 1

XWIKI NATIS

Eclaircissement : le JavaScript est mono-threadé et multi-threadé ?

XONATIS


JS
set
libuv
V8
Engine
-http
-fs
-cryptography
fetch
T1
T2
T3
T4
T

XWIKI SAS

XEBIA NATIS

LES CALLBACKS ET PROMESSES

XNATIS

Cela signifie que vous pouvez laisser votre code faire plusieurs choses en même temps sans arrêter ou bloquer votre processus principal.

Il existe deux principaux types de code asynchrone que vous rencontrerez dans le code JavaScript : les anciens callbacks et le code plus récent de type promesse

XWIKI NATIS

Les callbacks asynchrones ou fonctions de rappels asynchrones sont des fonctions qui sont passées comme arguments lors de l'appel d'une fonction qui commencera à exécuter une fois un traitement terminé.

XDNATIS

```
function afficherBonjour() {
    console.log('Afficher bonjour');
}

setTimeout(afficherBonjour, 5000);
```

XWIKI NATIS

L'utilisation des callbacks est un peu démodée aujourd'hui, mais vous les verrez encore dans un certain nombre d'API plus anciennes encore couramment utilisées.

XWIKI NATIS

Les promesses sont le nouveau style de code asynchrone que vous verrez utilisé dans les API Web modernes.
Les promesses présentent certaines similitudes avec les anciennes fonctions de rappel.

XEBIA NATIS

Promise

```
fetch('**URL DE L'API**', {                    // Endpoint
    method: 'POST',                            // HTTP method
    headers: {
        'Accept': 'application/json',          // Server's response format
        'Content-Type': 'application/json', // Body's format
    },
    body: JSON.stringify(data),                // Payload
})
.then(function(response) {
    return response.json();
})
.then(function(data) {
    // Handle data here...
})
.catch(function(error) {
    // Handle error here...
});
```

comme nos callbacks

XWIKI NATIS

Il s'agit essentiellement d'un objet retourné auquel vous attachez des fonctions de rappel, plutôt que de devoir passer des callbacks dans une fonction.

XWIKI NATIS

Vous pouvez enchaîner plusieurs opérations asynchrones en utilisant plusieurs opérations .then(), en passant le résultat de l'une dans la suivante comme entrée ! ☺

XWIKI NATIS

Les fonctions de rappel passées aux promesses sont toujours appelées dans l'ordre strict où ils sont placés dans la file d'attente des événements.

XKNATIS

La gestion des erreurs est bien meilleure - toutes les erreurs sont traitées par un seul bloc .catch() à la fin du bloc, plutôt que d'être traitées individuellement à chaque niveau de la « pyramide ».

XWIKI SAS

XWIKI NATIS

LES REST API

XWIKI NATIS

Une API (Application Programming Interface) est très utile dans le cas d'intégration de systèmes hétérogènes.
Une API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

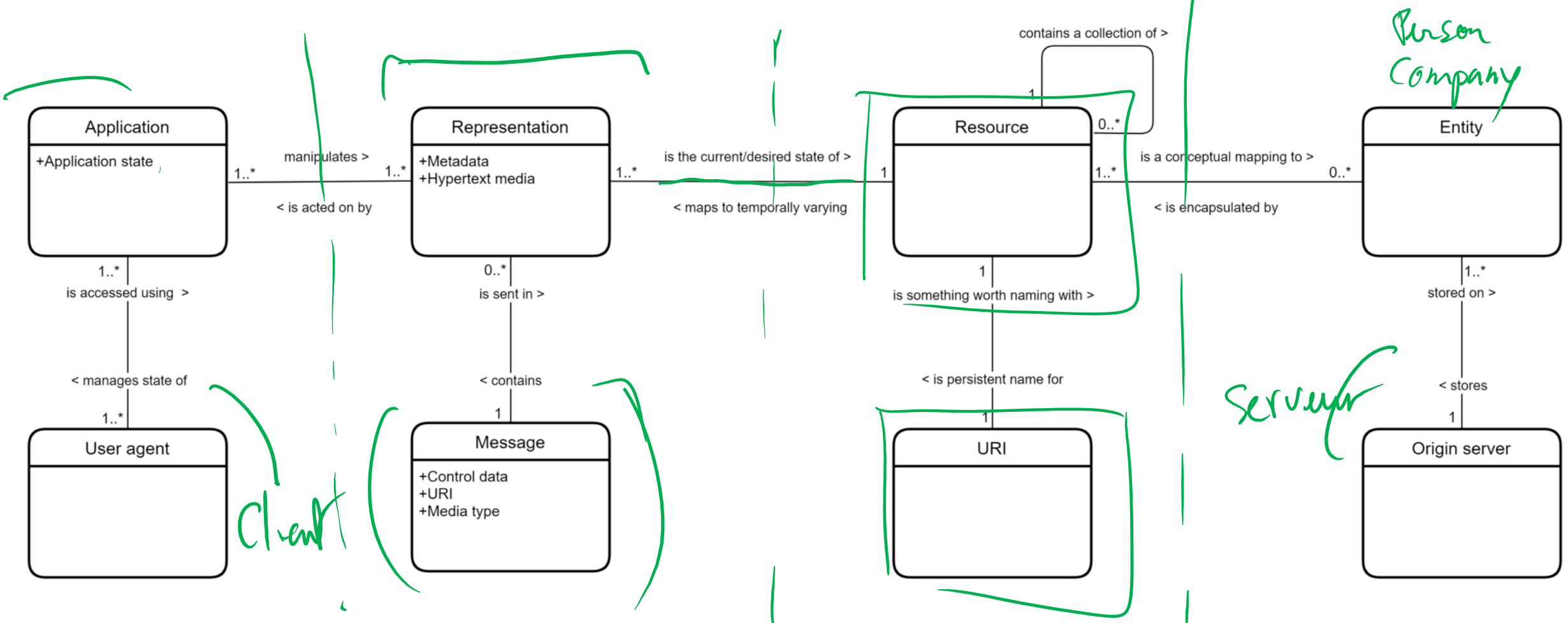
XWIKI NATIS

Lorsque vous souhaitez interagir avec un ordinateur ou un système pour récupérer des informations ou exécuter une fonction, une API est un ensemble de fonctionnalités de ce système utilisables par votre programme.

XWIKI SAS

Les REST API sont un ensemble de fonctionnalités qui suivent les contraintes d'architecture web REST.

XONATIS


Application
+Application state
manipulates >
1..*
1..*
< is acted on by
Representation
+Metadata
+Hypertext media
is the current/desired state of >
1..*
1..*
< maps to temporally varying
Resource
1
contains a collection of >
1
0..*
1..*
is a conceptual mapping to >
0..*
< is encapsulated by
Entity
1..*
is accessed using >
1..*
< manages state of
1..*
User agent
0..*
is sent in >
< contains
1
Message
+Control data
+URI
+Media type
1
is something worth naming with >
< is persistent name for
1
URI
1..*
stored on >
< stores
1
Origin server

XWIKI NATIS

L’architecture REST :

1. Client–serveur : architecture client - serveur
2. Sans état : aucun conservation de la requête
3. Mise en cache : mise en cache des réponses possibles
4. Couches : le client peut passer par un intermédiaire
5. Interface uniforme : JSON par exemple

XWIKI SAS

XWIKI NATIS

EVENT-DRIVEN PROGRAMMING

XWIKI NATIS

La programmation événementielle est un paradigme de programmation dans lequel le flux du programme est déterminé par des événements tels que les actions de l'utilisateur (clics de souris, pressions sur les touches), les sorties de capteurs ou les messages transmis par d'autres programmes

XONATIS