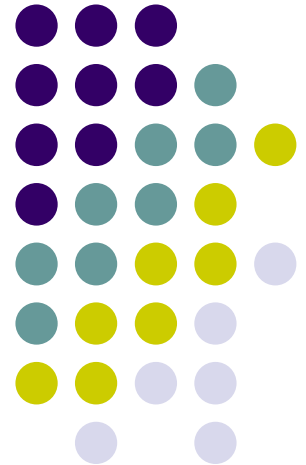


JavaScript



Asynchrone





Comprendre la notion de programmation asynchrone



Utiliser les promesses dans le cadre du développement d'une tâche asynchrone



2h

Asynchrone



➡ **Notion fondamentale en JavaScript**

➡ **Gérer l'exécution de tâches qui prennent "du temps"**

➡ **Appeler un script (ex: PHP)**

➡ **Consonner une API**



Code asynchrone



Un code *asynchrone* est un code qui commence maintenant et de terminera plus tard ...

Code synchrone



Javascript peut exécuter une instruction à la fois et de manière séquentielle (ordre dans lequel apparaissent les instructions)

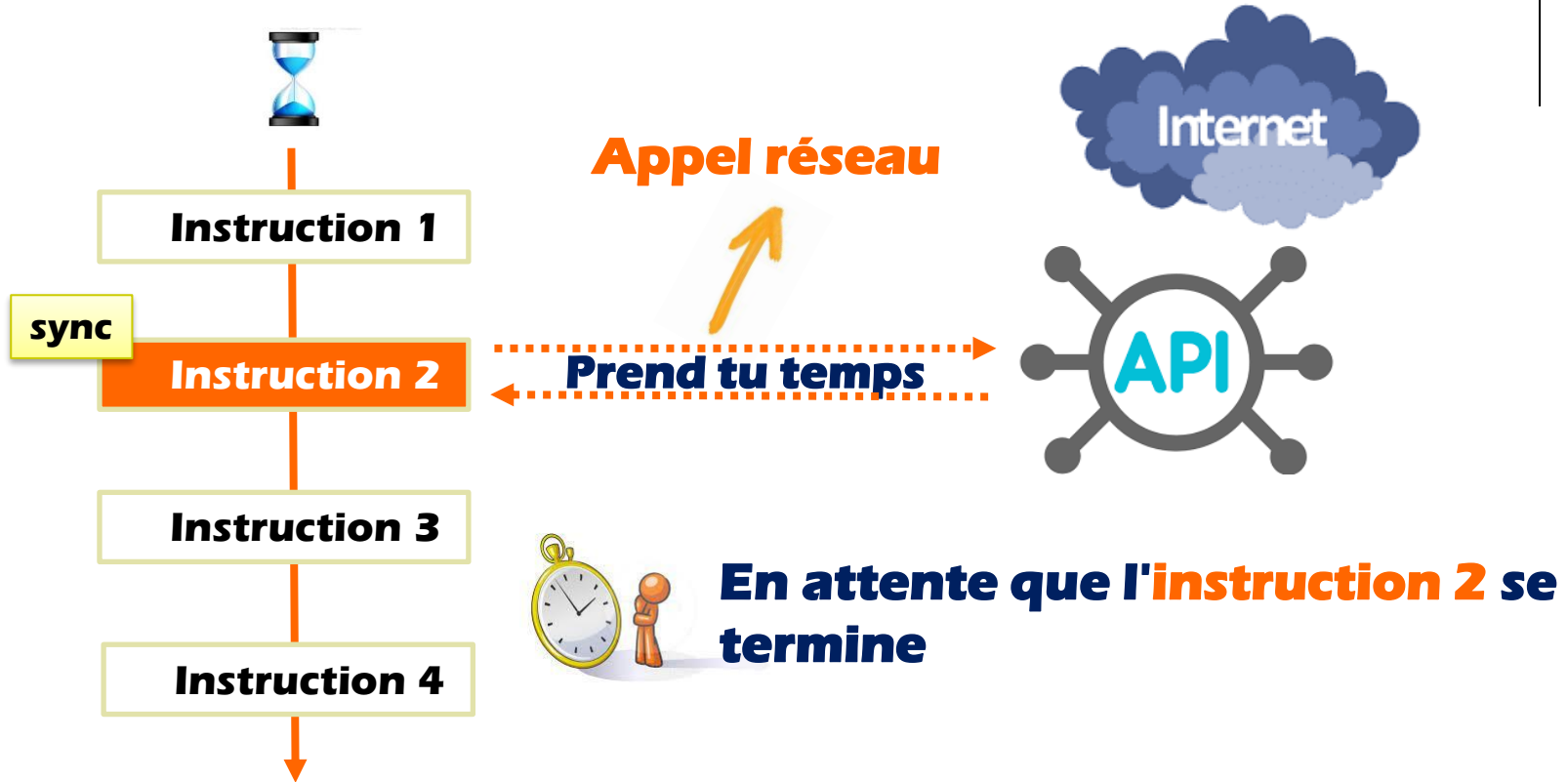


```
console.log('ligne 1');
```

```
console.log('ligne 2');
```

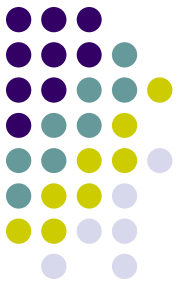
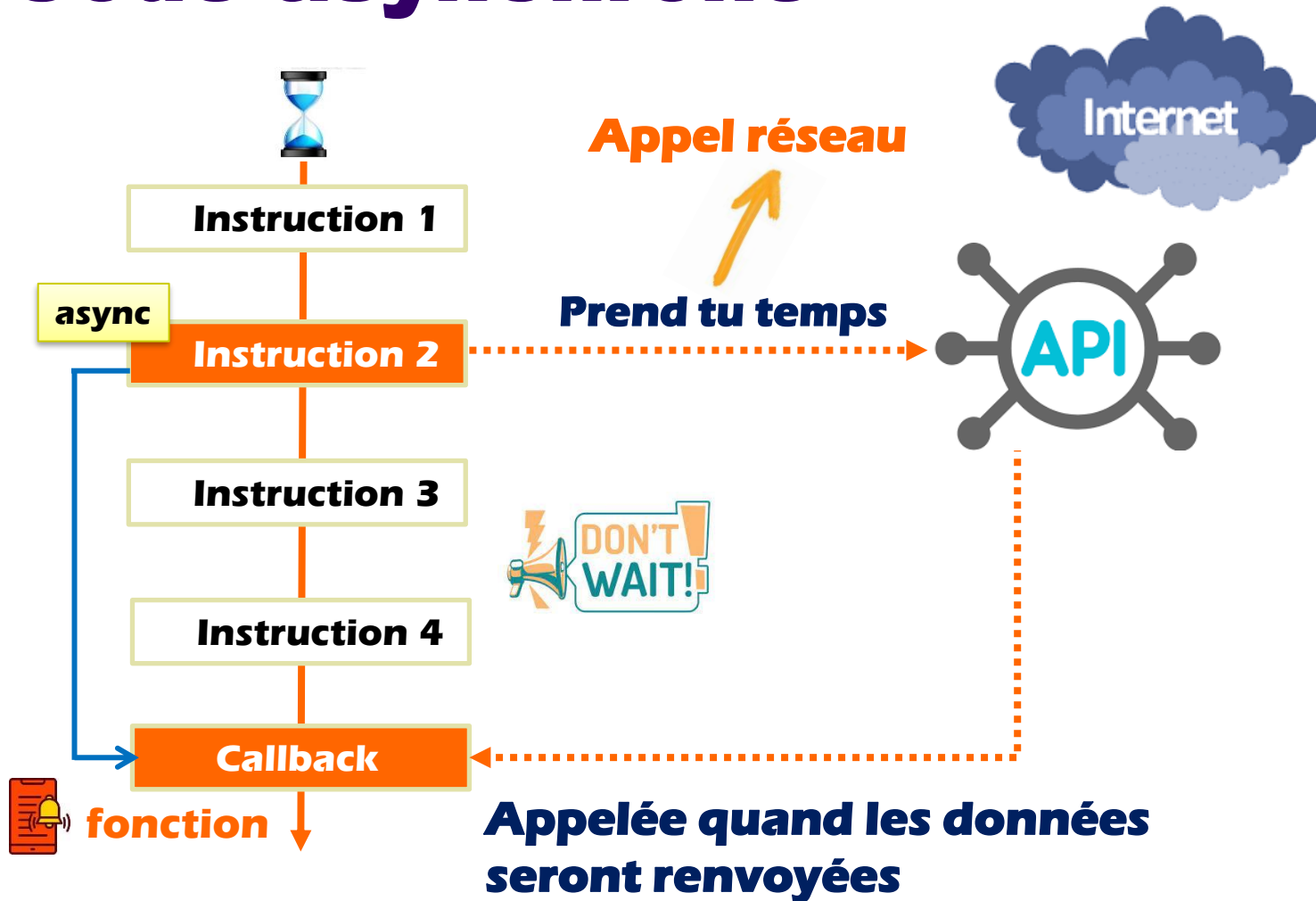
```
console.log('ligne 3');
```

Code synchrone !



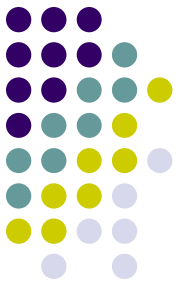
Code bloquant !

Code asynchrone



Code non bloquant !

Exemple



Code synchrone

```
console.log(1)
console.log(2)
console.log(3)
console.log(4)
```



Code asynchrone

```
console.log(1)
console.log(2)
```

Simule un appel réseau qui dure 3s

```
setTimeout( () => {
  console.log("Fonction de callback exécutée !")
},3000)
```

{ } -> callback

```
console.log(3)
console.log(4)
```

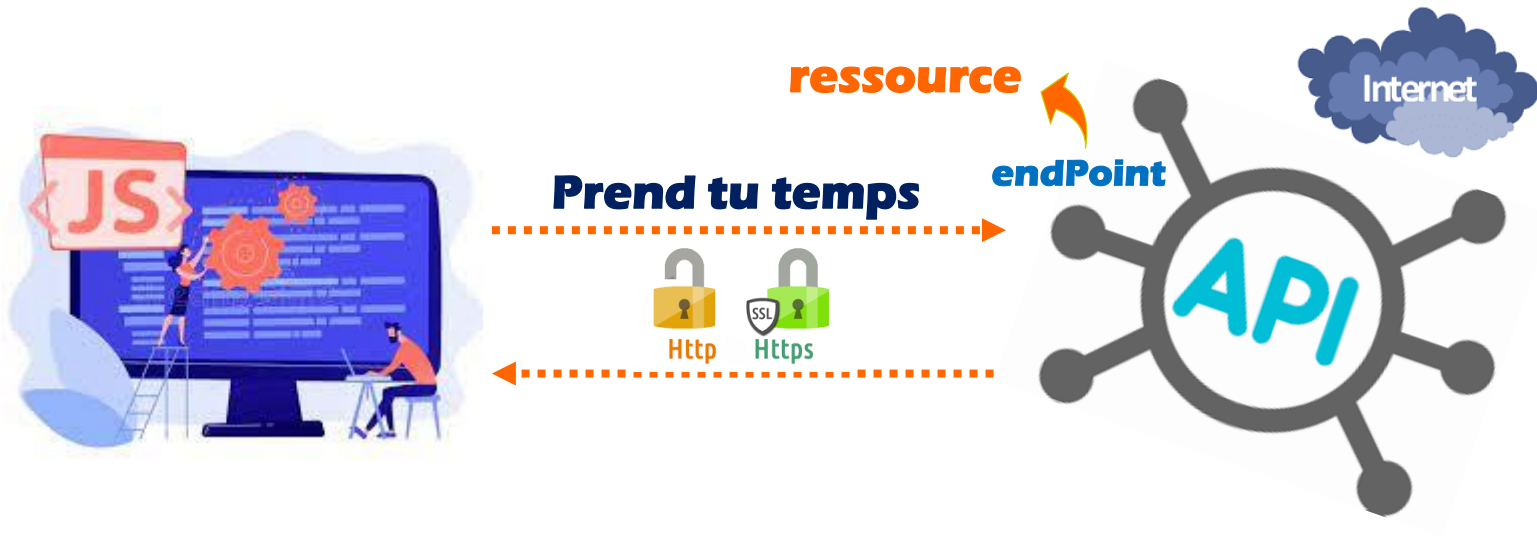
Prend tu temps



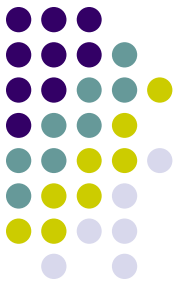
Requête HTTP



➔ En **javascript** (depuis le navigateur), on peut émettre une **requête HTTP** afin d'**accéder** à une **ressource** se trouvant sur un **serveur distant**



Promesses



Une **promesse** (**promise**) est un **objet** permettant de **réaliser une tâche** qui va **durer un certain temps** (ex : appel à un **endPoint**)



Le **résultat** d'une **promesse** est :



La **promesse** est **résolue** (elle a été **tenue**) : les données demandées sont disponibles



La **promesse** est **rejetée** (elle n'a pas été **tenue**) : il y a eu un problème quelque part !

Promesse tenue



Nous espérons que tu vas travailler dur afin de réussir ton BTS ?

DES PROMESSES,
TOUJOURS
DES PROMESSES.

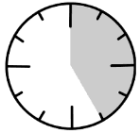


Oui je vous le promets !



La promesse est créée et en attente d'être tenue ou pas !

LA
VIE
CONTINUE



Quelque temps plus tard...

Le temps de passer son BTS et d'avoir le **résultat**

Le résultat est arrivé. La promesse est résolue et a été tenue

Je viens d'avoir mes résultats.
BTS obtenu !



Félicitations ! Tu as **tenu** ta promesse et **obtenu ton BTS** !

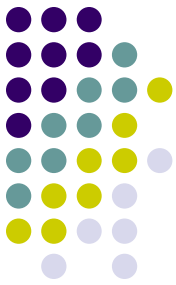
L'action réalisée suite à la résolution de la promesse (les félicitations)

Promesse non tenue



Nous espérons que tu vas travailler dur afin de réussir ton BTS ?

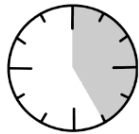
DES PROMESSES,
TOUJOURS
DES PROMESSES.



Oui je vous le promets !

La promesse est créée et en attente d'être tenue ou pas !

LA
VIE
CONTINUE



Quelque temps plus tard...

Le temps de passer son BTS et d'avoir le **résultat**



Le résultat est arrivé. La promesse est rejetée et n'a pas été tenue

Je viens d'avoir mes résultats.
BTS non obtenu !



Tu n'**as pas tenu** ta promesse !

L'action réalisée suite au rejet de la promesse (la déception)

Promesses



Déclaration d'une promesse

```
let obtenirBTS = new Promise(executeur);
```



La promesse



Fonction exécutée à la création de la promesse afin de déclencher les actions qui permettront de résoudre ou rejetée la promesse

Promesses



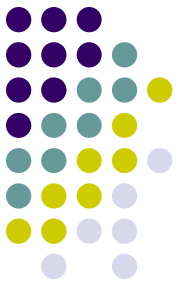
Déclaration d'une promesse

```
let obtenirBTS = new Promise() => {  
    // Passer le BTS  
    // Obtenir le résultat  
});
```



Exécuteur

Promesses



Déclaration d'une promesse

```
let obtenirBTS = new Promise( (resolve, reject) => {  
    // Passer le BTS  
    // Obtenir le résultat  
});
```



resolve et **reject** sont 2 fonctions :



resolve est appelée en cas de **promesse tenue**




reject est appelée en cas de **promesse non tenue**

Promesses



Déclaration d'une promesse

```
let obtenirBTS = new Promise( (resolve, reject) => {  
  // Passer le BTS et obtenir le résultat   
  let BTS = true; // le résultat est arrivé!  
  if (BTS) {  
    // résultat  
    resolve("BTS obtenu !"); // Promesse tenue  
  } else {  
    reject("BTS non obtenu !"); // Promesse non tenue  
  }  
} );
```


Promesses



Récupérer le **résultat** d'une **promesse**



La **promesse** est **résolue** et à été **tenue**



Si la **promesse** est **résolue** :
resolve a été **appelée**

Le **résultat**

```
obtenirBTS.then( (resultat) => {  
  } );
```

Action(s) à réaliser en cas
de **promesse résolue**

Promesses



Récupérer le résultat d'une promesse



La promesse est résolue et à été tenue



```
obtenirBTS.then( (resultat) => {  
  console.log(resultat);  
  console.log("Félicitations ! Tu as tenu ta promesse !")  
} );
```

Promesses



Récupérer le résultat d'une promesse



La promesse est **rejetée** et n'a pas été tenue



Si la promesse est **rejetée** :
reject a été appelée

Le résultat

```
obtenirBTS.catch( resultat ) => {  
  } );
```

Action(s) à réaliser en cas
de promesse **rejetée**

Promesses



Récupérer le résultat d'une promesse



La promesse est **rejetée** et n'a pas été tenue



```
obtenirBTS.catch( resultat) => {  
  console.log(resultat);  
  console.log("Tu n'as pas tenu ta promesse !")  
} );
```

Promesses



Récupérer le résultat d'une promesse

obtenirBTS

```
.then( (resultat) => {  
    console.log(resultat);  
    console.log("Félicitations ! Tu as tenu ta promesse !")  
})  
.catch( (resultat) => {  
    console.log(resultat);  
    console.log(" Tu n'as pas tenu ta promesse!")  
});
```

Promesses

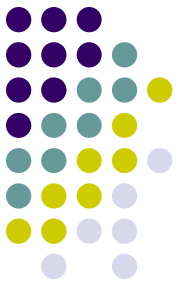


Simuler le temps



```
let obtenirBTS = new Promise( (resolve, reject) => {  
  setTimeout( () => {  
    let BTS = true;  
    if (BTS) {  
      resolve("BTS obtenu !");  
    } else {  
      reject("BTS non obtenu !");  
    }  
  }, 5000);  
} );
```

Promesses



Fonction asynchrone qui retourne une promesse

```
const reussirBTS = () => {  
  return new Promise( (resolve, reject) => {  
    setTimeout( () => {  
      let BTS = false;  
      if (BTS) {  
        resolve("BTS obtenu !");  
      } else {  
        reject("BTS non obtenu !");  
      }  
    }, 5000);  
  } );  
}
```



DEMO

**Fonction asynchrone qui
récupère les notes d'un
étudiant**