

CIS 371 Web Application Programming

JS|TS Promise

Handling Asynchronous Results



GRAND VALLEY
STATE UNIVERSITY®

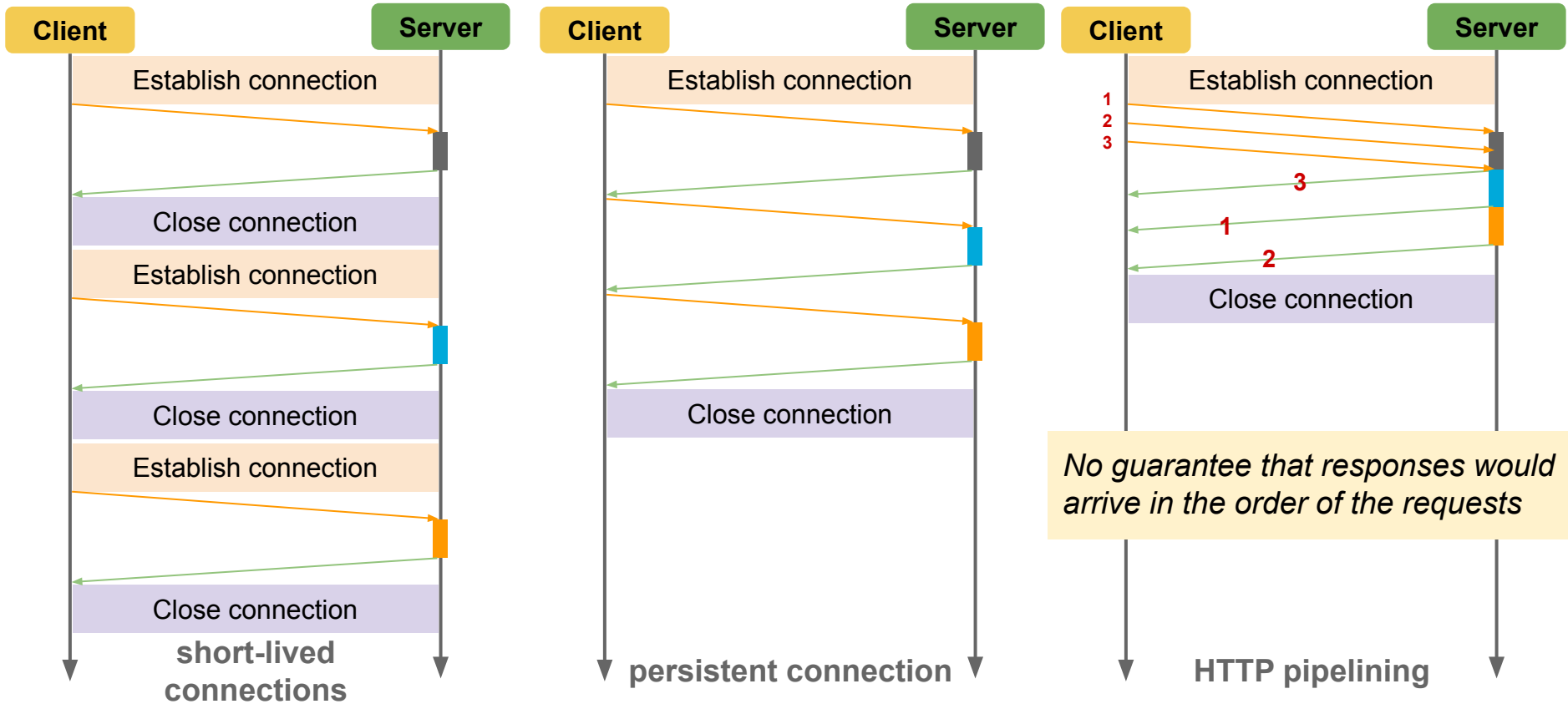
Lecturer: **Dr. Yong Zhuang**

Topics

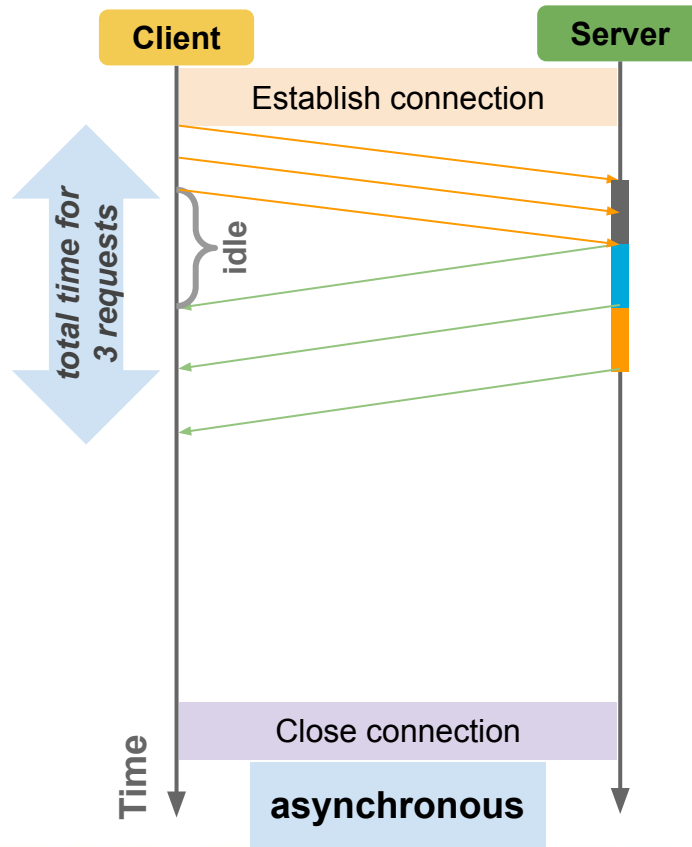
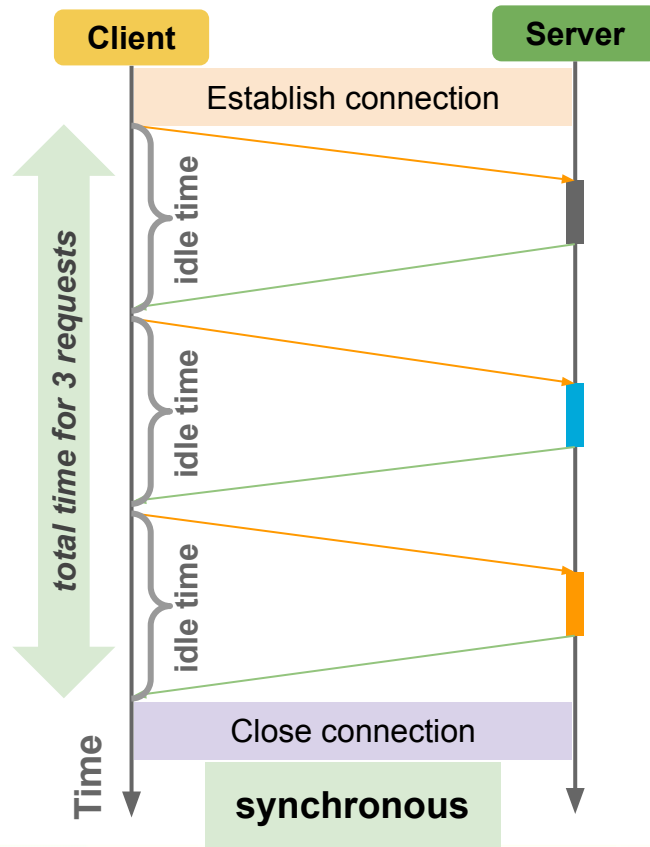
- Client/Server Communication
 - Synchronous
 - Asynchronous
- Callback functions (for handling asynchronous events)
- Promise

Reference: Promise Documentation (@ MDN)

Client/Server: HTTP Requests & Responses

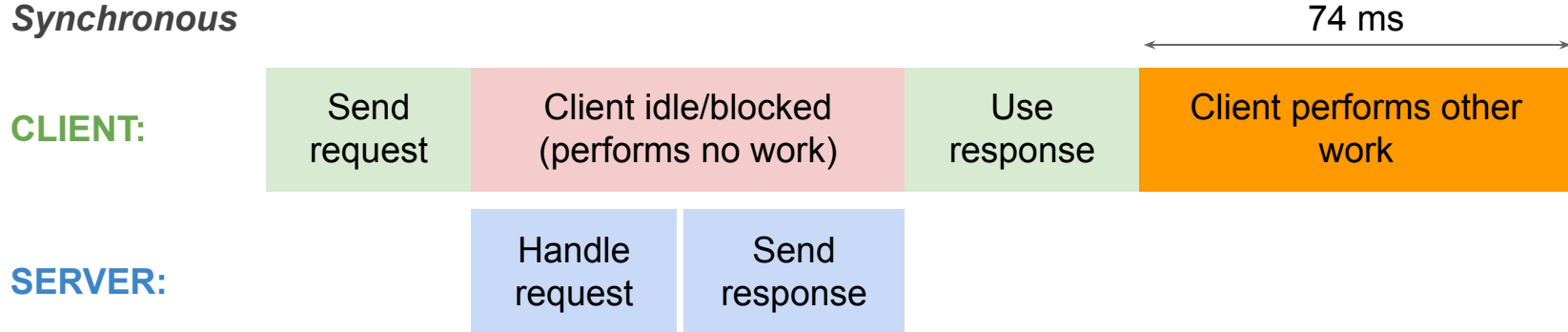


Client/Server: HTTP Requests & Responses

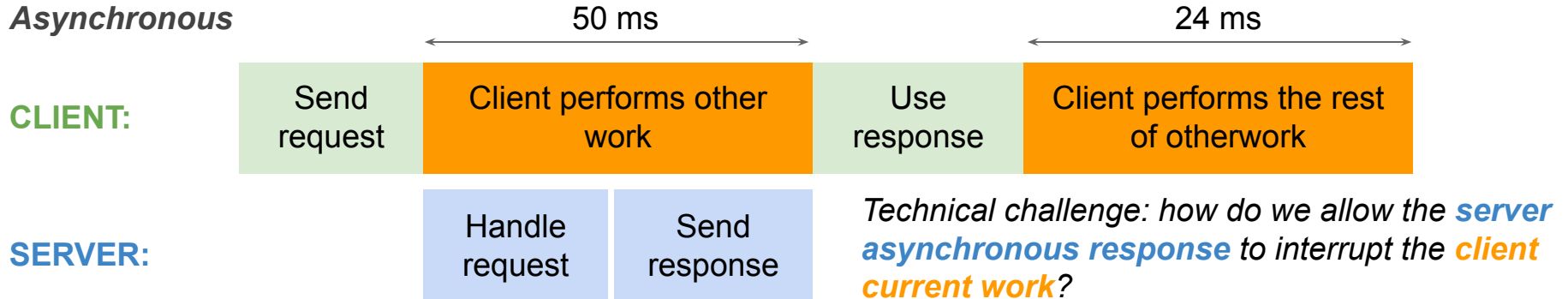


Synchronous vs. Asynchronous Requests

Synchronous



Asynchronous



Sending Requests: easy
Receiving Async Responses: requires extra setup

Callback Actions (JS Callback Functions)

You are number 17 in line.....



Would you like us to call you back?

1-888-I-CAN-HELP

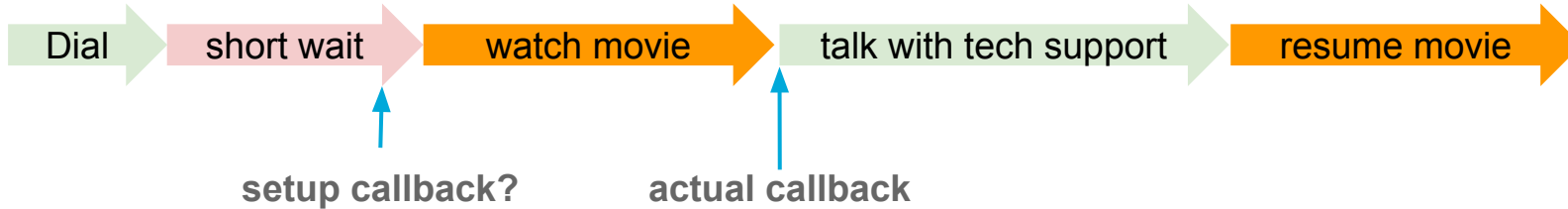




Option #1: without callback



Option #2: with callback



Synchronous Call (in code)

555-4321



1-888-I-CAN-HELP



```
dial("888-I-CAN-HELP");  
connect_and_long_wait();  
talk_with_tech();  
watch_movie();
```

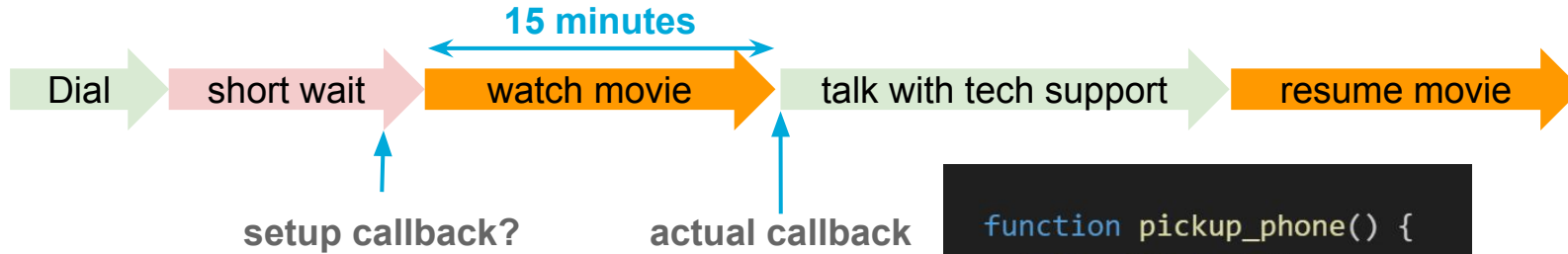
**Async: “out-of-order” execution
(Order of execution \neq order of line of code)**

Async Phone Calls with Callback (in code)

555-4321



1-888-I-CAN-HELP



```
dial("888-I-CAN-HELP");  
setup_cb("555-4321", pickup_phone);  
watch_movie();
```

```
function pickup_phone() {  
  talk_with_tech();  
}
```

15 mins later

Asynchronous (incoming call) while you're watching movie

Callback fns (Fat Arrow)

```
function pickup_phone() {  
  talk_with_tech();  
}  
dial("888-I-CAN-HELP");  
setup_cb("555-4321", pickup_phone);  
watch_movie();
```

named function

```
dial("888-I-CAN-HELP"); 1  
setup_cb("555-4321", () => { 2  
  // 15 min later  
  talk_with_tech(); 4  
});  
watch_movie(); 3 5
```

fat arrow

Async: order of execution ≠ order of line of code

555-4321

1-888-I-CAN-HELP



15 minutes

Dial

short wait

watch movie

talk with tech support

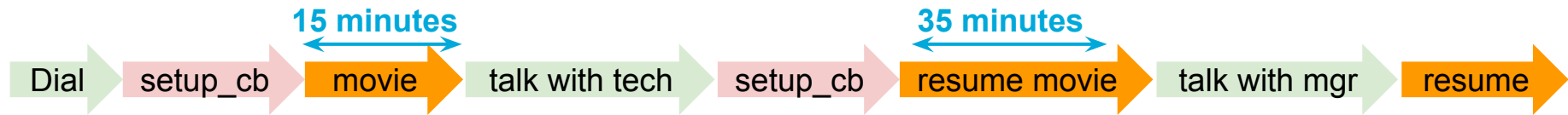
resume movie

setup callback?

actual callback

```
// start dialing ...  
dial("888-I-CAN-HELP"); 1  
// call me back @ 555-4321  
// then hangup to watch movie  
setup_cb("555-4321", () => { 2  
    // 15-min later  
    talk_with_tech(); 4  
});  
// watch it NOW!!!  
watch_movie(); 3 5
```

**Tech: “But, you have to talk with my manager”
(Nested Callback)**



```
1 dial("888-I-CAN-HELP");
2 setup_cb("555-4321", () => {
3   // 15-min later
4   // Talk with tech
5   setup_cb("555-4321", () => {
6     // 37-min later
7     // Talk with manager
8   });
9 });
10 watch_movie();
```

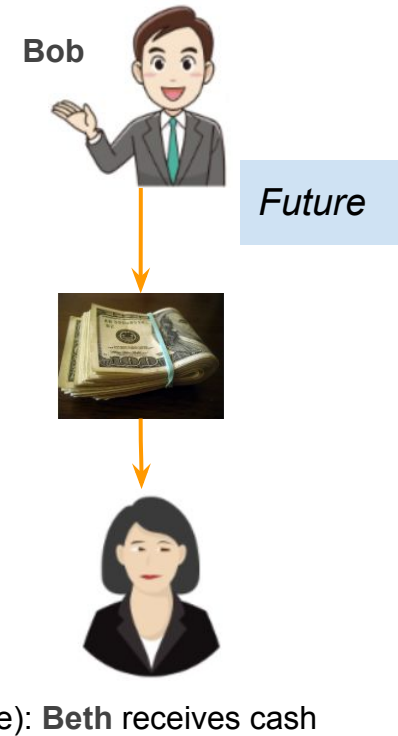
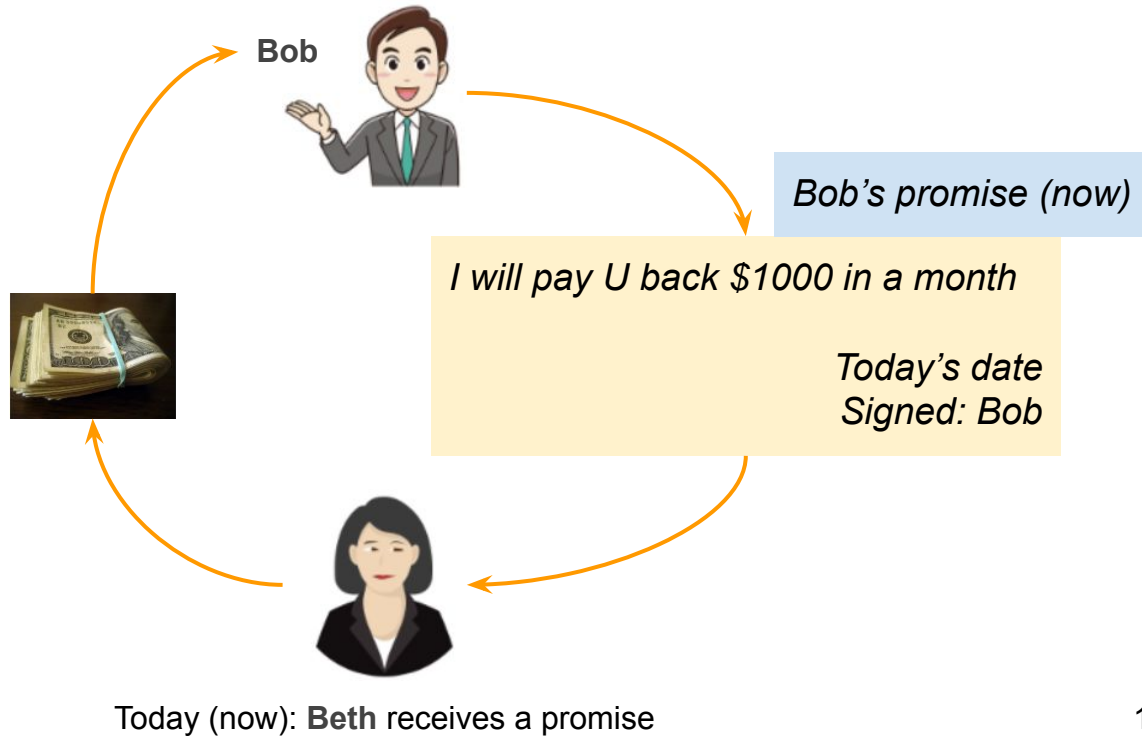
Nested callbacks

How to Initiate Async HTTP Requests?

- `fetch()` function
 - Native in browser
 - NPM `node-fetch`
- Axios library
- Both `fetch()` and `axios()` use JS Promise

IOU = I owe you note
Promise to pay debt/loan

Borrowing Money: Promise Now, Pay Later



A promise = now confirmation of future action(s)
**A JS promise = a “now” object representing data
which will become available in the future**

Promise Example

```
function nthPrime(nth: number): Promise<number> {  
  // work takes 10 seconds  
  return Promise.resolve(____);  
}
```

```
console.log("Start");  
const prom = nthPrime(500);  
prom.then((pr: number) => {  
  console.log("The 500th prime is", pr);  
});  
doMoreWork();
```

Start
Partial output of doMoreWork()
After 10 seconds
The 500th prime is 3571
More output from doMoreWork()

```
function nthPrimeNow(nth: number): number {  
  // work takes 10 seconds  
  return ____;  
}
```

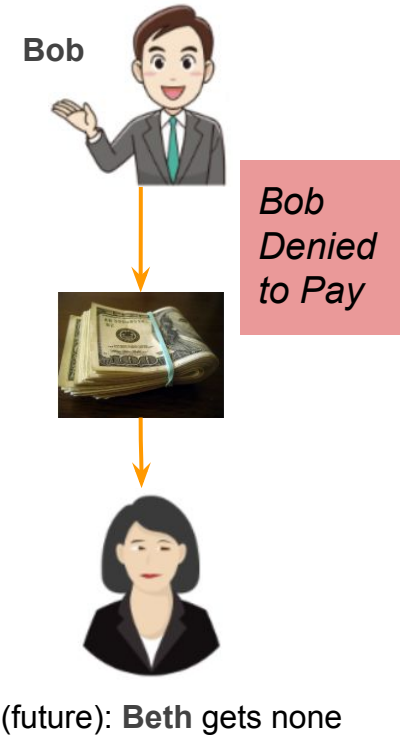
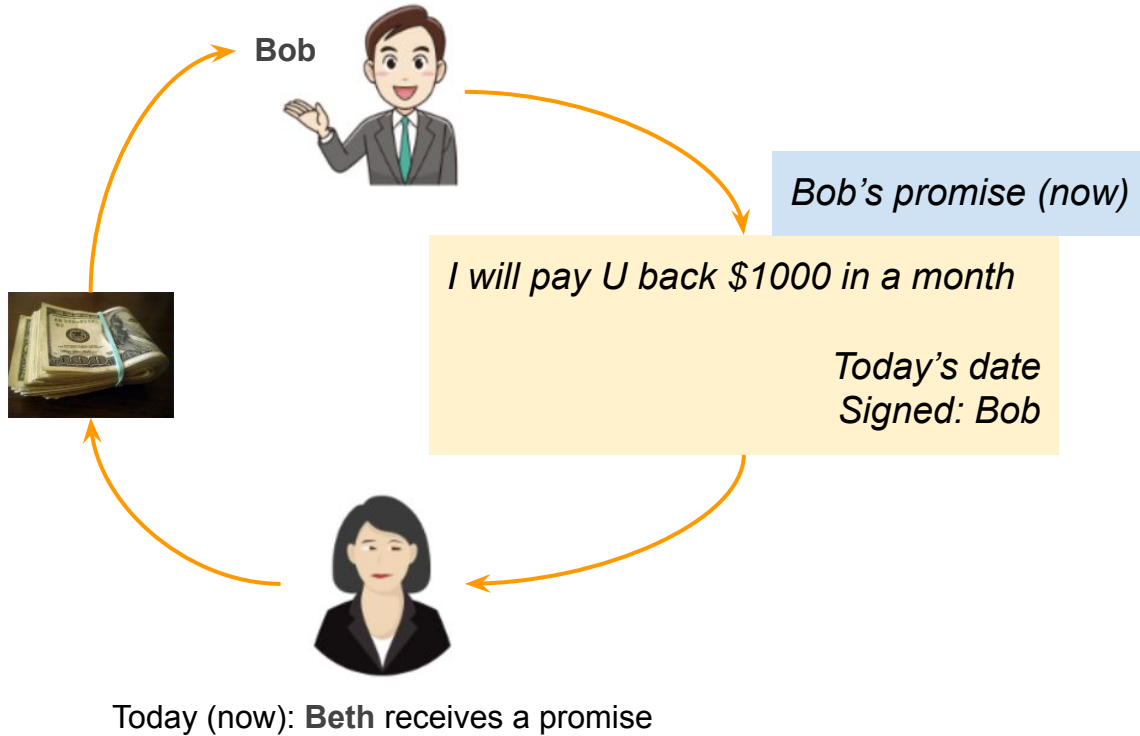
```
console.log("Start");  
const pr = nthPrimeNow(500);  
console.log("The 500th prime is", pr);  
doMoreWork();
```

Start
After 10 seconds
The 500th prime is 3571
Output of doMoreWork()

Compare the order of execution

Loan is either paid-off or defaulted
Promise is either resolved or rejected

Borrowing Money: Promise Now, **Never** Pay



Promise settlement: resolve() or reject()

```
function nthPrime(nth: number): Promise<number> {  
  if (nth < 100_000) {  
    // assume prime calculation takes 10 seconds  
    return Promise.resolve(a_prime_number_here);  
  } else return Promise.reject("Can't compute prime");  
}
```

```
console.log("Start");  
nthPrime(500).then((pr: number) => {  
  console.log("Prime is", pr);  
});  
.catch((err:any) => {  
  console.log("Rejected", err);  
});  
console.log("Here");
```

```
# Watch for order of execution  
Start  
Here  
# if the promise is resolved  
# After 10 seconds ...  
Prime is 3571  
# if the promise is rejected  
Rejected Can't compute prime
```