Create, Resolve, Reject, and Respond to Promises

John Papa
DEVELOPER ADVOCATE

@john_papa www.johnpapa.net

Promises with TypeScript

- Learn the key terms
- How to think about promises
- Creating a promise
- Responding to a promise
- Chaining
- Error handling

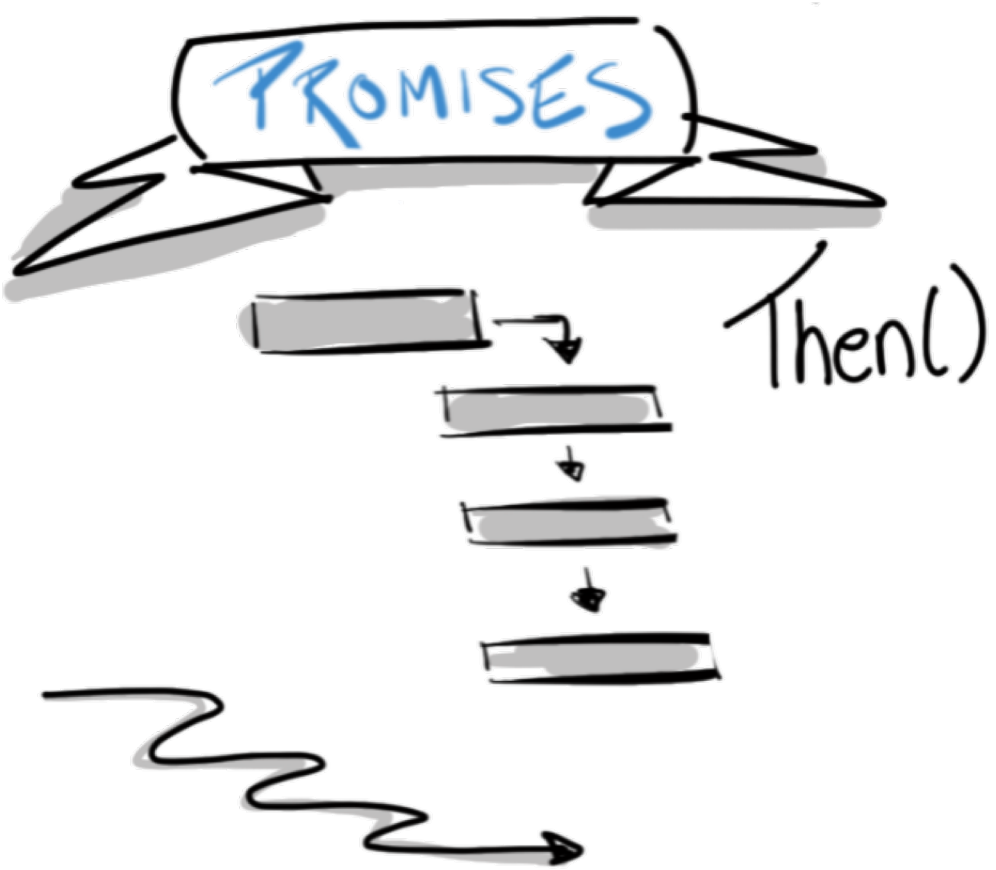
How to Think About Promises

The Promise represents the eventual completion (or failure) of an asynchronous operation, and its resulting value

Credit: MDN

https://jpapa.me/promise-def

Promises Promises



Promise Terminology

Fulfilled	The promise succeeded, often with resolve or Promise.resolve
Rejected	The promise failed, often with reject or Promise.reject
Pending	The promise is not fulfilled nor rejected yet
Settled	The promise has either been fulfilled or rejected

Prototype Methods

.then()

- Append a fulfillment handler
- Returns a new promise

```
function example() {
  showProgressbar(true);

  return getHeroes()
    .then((hero: Hero) => getOrders(hero))
    .then((hero: Hero) => showHero(hero))
    .catch((error: any) => {
      showMessage(error);
    })
    .finally(() => {
      showProgressbar(false);
    });
}
```

Prototype Methods

.then()

- Append a fulfillment handler
- Returns a new promise

.catch()

- Append a rejection handler
- Returns a new promise

```
function example() {
  showProgressbar(true);

  return getHeroes()
    .then((hero: Hero) => getOrders(hero))
    .then((hero: Hero) => showHero(hero))
    .catch((error: any) => {
      showMessage(error);
    })
    .finally(() => {
      showProgressbar(false);
    });
}
```

Prototype Methods

.then()

- Append a fulfillment handler
- Returns a new promise

.catch()

- Append a rejection handler
- Returns a new promise

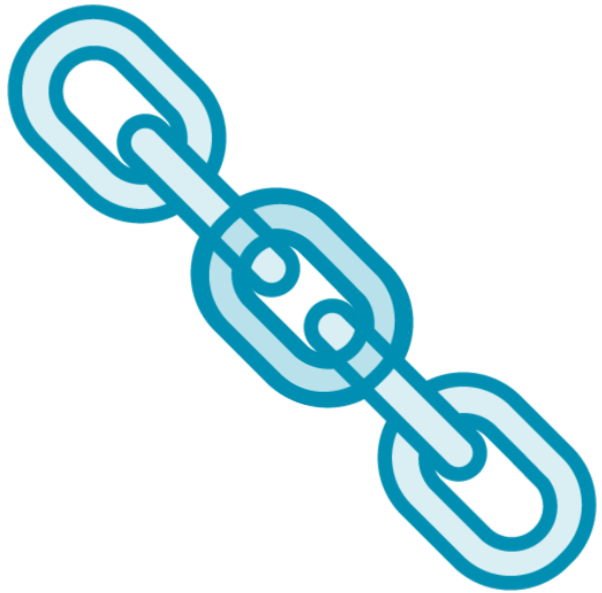
.finally()

- Append a handler
- Always called
- Returns a new promise

```
function example() {
  showProgressbar(true);

  return getHeroes()
    .then((hero: Hero) => getOrders(hero))
    .then((hero: Hero) => showHero(hero))
    .catch((error: any) => {
      showMessage(error);
    })
    .finally(() => {
      showProgressbar(false);
    });
}
```

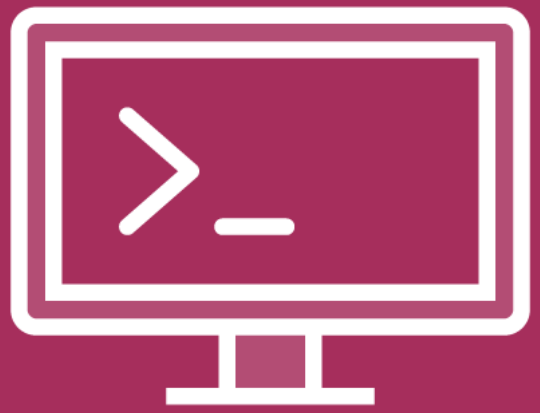
Chaining

- Promises can be chained
- Pass the previous promise's resolved return value (with "then")



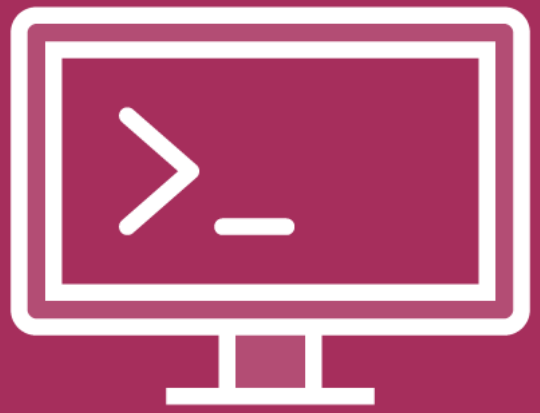
```
function example() {
  showProgressbar(true);

  return getHeroes()
    .then((hero: Hero) => getOrders(hero))
    .then((hero: Hero) => showHero(hero))
    .catch((error: any) => {
      showMessage(error);
    })
    .finally(() => {
      showProgressbar(false);
    });
}
```

Demo



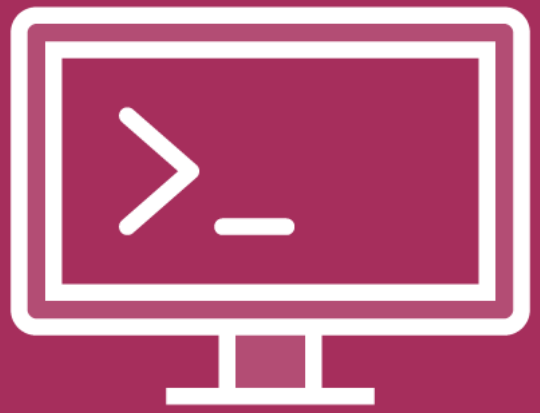
Creating Promises

Demo



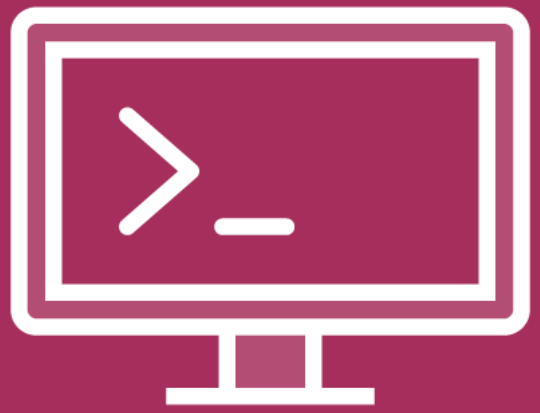
Resolve or Reject a Promise

Demo



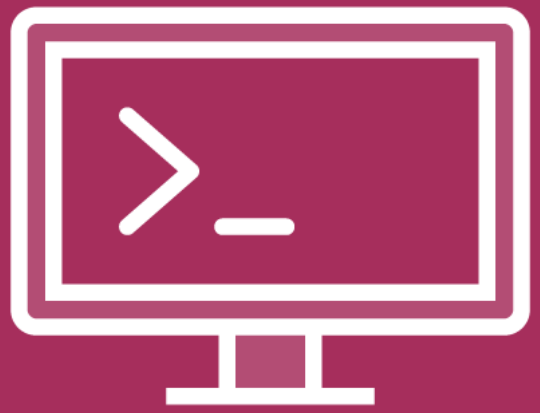
Rejecting and Promise Methods

Demo



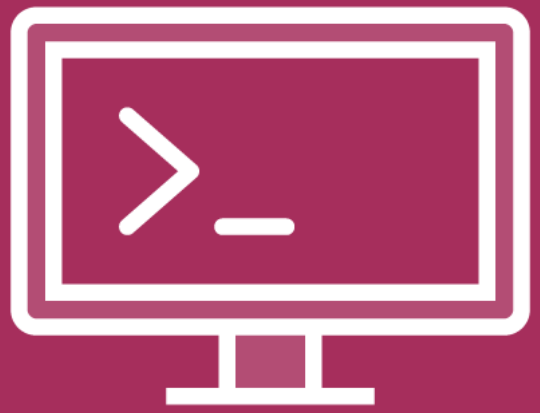
Getting Heroes with Promises

Demo



Promise Chaining and Promise.all

Demo



Turning Callbacks into Promises

Summary

- Promise terminology
- Creating a promise and typing
- Executor function
- Resolve or reject a promise
- Chaining promises
- Error handling
- Execute multiple promises
- Manage callbacks as promises