Promises, Functions and Object Prototypes

Promises

Promises

Promise Features

Promises

```
function doAsync() {
    let p = new Promise(function (resolve, reject) {
        console.log('in promise code');
        setTimeout(function () {
            console.log('resolving...');
            resolve();
        }, 2000);
    });
    return p;
}

let promise = doAsync();
```

Q

What shows in the console?

A

in promise code
(2 second delay)
resolving...

```
function doAsync() {
    let p = new Promise(function (resolve, reject) {
        console.log('in promise code');
        setTimeout(function () {
            console.log('rejecting...');
            reject();
        }, 2000);
    });
    return p;
}

let promise = doAsync();
```

Q

What shows in the console?

A

in promise code
(2 second delay)
rejecting...

```
function doAsync() {
    // returns a Promise, will be rejected
}

doAsync().then(function () {
    console.log('Fulfilled!');
},
function () {
    console.log('Rejected!');
});
```

Q

What shows in the console?

A

in promise code
(wait for resolution)
Rejected!

```
function doAsync() {
    // returns a Promise, will be resolved
}

doAsync().then(function () {
        console.log('Fulfilled!');
    },
    function () {
        console.log('Rejected!');
    }
);
```

Q What shows in the console?

A in promise code
(wait for resolution)
Fulfilled!

```
function doAsync() {
    // returns a Promise, will be rejected using:
    // reject('Database Error');
}

doAsync().then(function (value) {
    console.log('Fulfilled! ' + value);
},
function (reason) {  console.log('Rejected! ' + reason);
});
```

Q

What shows in the console?

A

in promise code
(wait for resolution)
Rejected! Database Error

```
function doAsync() {
    // returns a Promise, will be resolved using:
    // resolve('OK');
}

doAsync().then(function (value) {
    console.log('Fulfilled! ' + value);
},
function (reason) {
    console.log('Rejected! ' + reason);
});
```

Q What shows in the console?

A in promise code
(wait for resolution)
Fulfilled! OK

```
function doAsync() {
    // returns a Promise, will be resolved using:
    // resolve('OK');
}

doAsync().then(function (value) {
    console.log('Fulfilled! ' + value);
    return 'For Sure';
}).then(function(value) {
    console.log('Really! ' + value);
});
```

Q

What shows in the console?

A

in promise code
(wait for resolution)
Fulfilled! OK
Really! For Sure

```
function doAsync() {
    // returns a Promise, will be rejected using:
    // reject('No Go');
}

doAsync().catch(function (reason) {
    console.log('Error: ' + reason);
});
```

Q

What shows in the console?

A

(wait for resolution)
Error: No Go

More Promise Features

```
function doAsync() {
    let p = new Promise(function (resolve, reject) {
        console.log('in promise code');
        setTimeout(function () {
            resolve( getAnotherPromise() );
        }, 2000);
    });
    return p;
}

doAsync().then(function () { console.log('Ok') },
               function () { console.log('Nope')});
```

Q

What shows in the console?

A

in promise code
Nope

```
function doAsync() {
    return Promise.resolve('Some String');
}

doAsync().then(
    function (value) { console.log('Ok: ' + value) },
    function (reason) { console.log('Nope: ' + reason)}
);
```

Q What shows in the console?

A Ok: Some String

```
function doAsync() {
    return Promise.reject('Some Error');
}

doAsync().then(
    function (value) { console.log('Ok: ' + value) },
    function (reason) { console.log('Nope: ' + reason)}
);
```

Q

What shows in the console?

A

Nope: Some Error

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.all([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);

// assume p1 resolves after 3 seconds,
// assume p2 resolves after 5 seconds
```

Q

What shows in the console?

A

(5 second delay)
Ok

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.all([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);

// assume p1 resolves after 1 second,
// assume p2 is rejected after 2 seconds
```

Q

What shows in the console?

A

(2 second delay)
Nope

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.all([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);

// assume p1 is rejected after 3 second,
// assume p2 is rejected after 5 seconds
```

Q

What shows in the console?

A

(3 second delay)
Nope

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.race([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);

// assume p1 resolves after 3 second,
// assume p2 resolves after 5 seconds
```

Q

What shows in the console?

A

(3 second delay)
Ok

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.race([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);

// assume p1 is rejected after 3 second,
// assume p2 resolves after 5 seconds
```

Q

What shows in the console?

A

(3 second delay)
Nope

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.race([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);

// assume p1 resolves after 4 second,
// assume p2 is rejected after 5 seconds
```

Q

What shows in the console?

A

(4 second delay)
Ok

Function call()

```
const person = {
  fullName: function () {
    return this.firstName + " " + this.lastName;
  },
};
const person1 = {
  firstName: "John",
  lastName: "Doe",
};
console.log(person.fullName.call(person1));
```

Q What shows in the console?

A John Doe

```
const person = {
  fullName: function () {
    return this.firstName + " " + this.lastName;
  },
};
const person1 = {
  firstName: "John",
  lastName: "Doe",
};
console.log(person.fullName.call(person1));
```

Q What shows in the console?

A John Doe,Oslo,Norway

Function apply()

```
const person = {
  fullName: function () {
    return this.firstName + " " + this.lastName;
  },
};

const person1 = {
  firstName: "Mary",
  lastName: "Doe",
};

console.log(person.fullName.apply(person1));
```

Q What shows in the console?

A Mary Doe

```
const person = {
  fullName: function (city, country) {
    return this.firstName + " " + this.lastName + "," + city + "," + country;
  },
};

const person1 = {
  firstName: "John",
  lastName: "Doe",
};

console.log(person.fullName.apply(person1, ["Oslo", "Norway"]));
```

Q What shows in the console?

A John Doe,Oslo,Norway

Function bind()

```
const person = {
  firstName: "John",
  lastName: "Doe",
  fullName: function () {
    return this.firstName + " " + this.lastName;
  },
};
const member = {
  firstName: "Hege",
  lastName: "Nilsen",
};
let fullName = person.fullName.bind(member);

console.log(fullName());
```

Q What shows in the console?

A Hege Nilsen

```
const person = {
  firstName: "John",
  lastName: "Doe",
  display: function () {
    console.log(this.firstName + " " + this.lastName);
  },
};

person.display();
```

Q What shows in the console?

A John Doe

```
const person = {
  firstName: "John",
  lastName: "Doe",
  display: function () {
    console.log(this.firstName + " " + this.lastName);
  },
};

setTimeout(person.display, 3000);
```

Q What shows in the console?

A undefined undefined

```
const person = {
  firstName: "John",
  lastName: "Doe",
  display: function () {
    console.log(this.firstName + " " + this.lastName);
  },
};

let display = person.display.bind(person);

setTimeout(person.display, 3000);
```

Q What shows in the console?

A John Doe

Object Prototypes

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}

Person.prototype.nationality = "English";

const myFather = new Person("John", "Doe", 50, "blue");

console.log(myFather.nationality);
```

Q What shows in the console?

A English

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}

Person.prototype.name = function () {
  return this.firstName + " " + this.lastName;
};

const myFather = new Person("John", "Doe", 50, "blue");
console.log(myFather.name());
```

Q What shows in the console?

A John Doe

Summary

Promises

```
function doAsync() {
    let p = new Promise(function (resolve, reject) {
        console.log('in promise code');
        setTimeout(function () {
            console.log('resolving...');
            resolve();
        }, 2000);
    });
    return p;
}

let promise = doAsync();
```

Summary

Promise.all() and Promise.race()

```
let p1 = new Promise(...);
let p2 = new Promise(...);

Promise.all([p1, p2]).then(
    function (value) { console.log('Ok') },
    function (reason) { console.log('Nope') }
);
```