JavaScript Promises - What You Need To Know

The four functions you need to know

1. **`new Promise(fn)`**
   - `fn` takes two arguments: `resolve` and `reject`
   - `resolve` and `reject` are both functions which can be called with one argument
   - Returned promise will be rejected if an exception is thrown in the passed in function
2. **`promise.then(onResolve, onReject)`**
   - Returns a promise
   - Returned promise resolves to value returned from handler
   - Chain by returning a promise from `onResolve` or `onReject`
   - Returned promise will be rejected if an exception is thrown in a handler
   - Use `Promise.reject` to return a rejected promise from `onReject`
   - Make sure to follow by `promise.catch`
3. **`promise.catch(onReject)`**
   - Returns a promise
   - Equivalent to `promise.then(null, onReject)`
4. **`Promise.all([promise1, promise2, ...])`**
   - Returns a promise
   - When all arguments resolve, returned promise resolves to an array of all results
   - When any arguments are rejected, returned promise is immediately rejected with the same value
   - Useful for managing doing multiple things concurrently

Packages

- es6-promise - Polyfill older browsers
- bluebird - Get extra promise methods
- promisify-node - Promisify callback-accepting functions (npm)

Extra Reading

- Are JavaScript Promises swallowing your errors?
- Promises at MDN
- Promise browser support at Can I Use

The two functions you should know

- **`Promise.resolve(value)`**
  - Returns a promise which resolves to `value`
  - If `value` is a promise, just returns `value`
- **`Promise.reject(value)`**
  - Returns a rejected promise with the value `value`
  - Useful while processing errors with `promise.catch`

Patterns

- **Promisify a callback-accepting function `fn`**

  Assume callback passed to `fn` takes two arguments: `callback(error, data)`, where `error` is `null` if successful, and `data` is null if unsuccessful.

```
new Promise(function(resolve, reject) {
  fn(function(error, data) {
    if (error) {
      reject(error);
    }
    else {
      resolve(data);
    }
  });
});
```

- **Catch exceptions thrown in `then` handlers**

```
promise
  .then(function() { ... })
  .catch(function(err) {
    console.log(err.stack);
  });
```