Javascript Promises

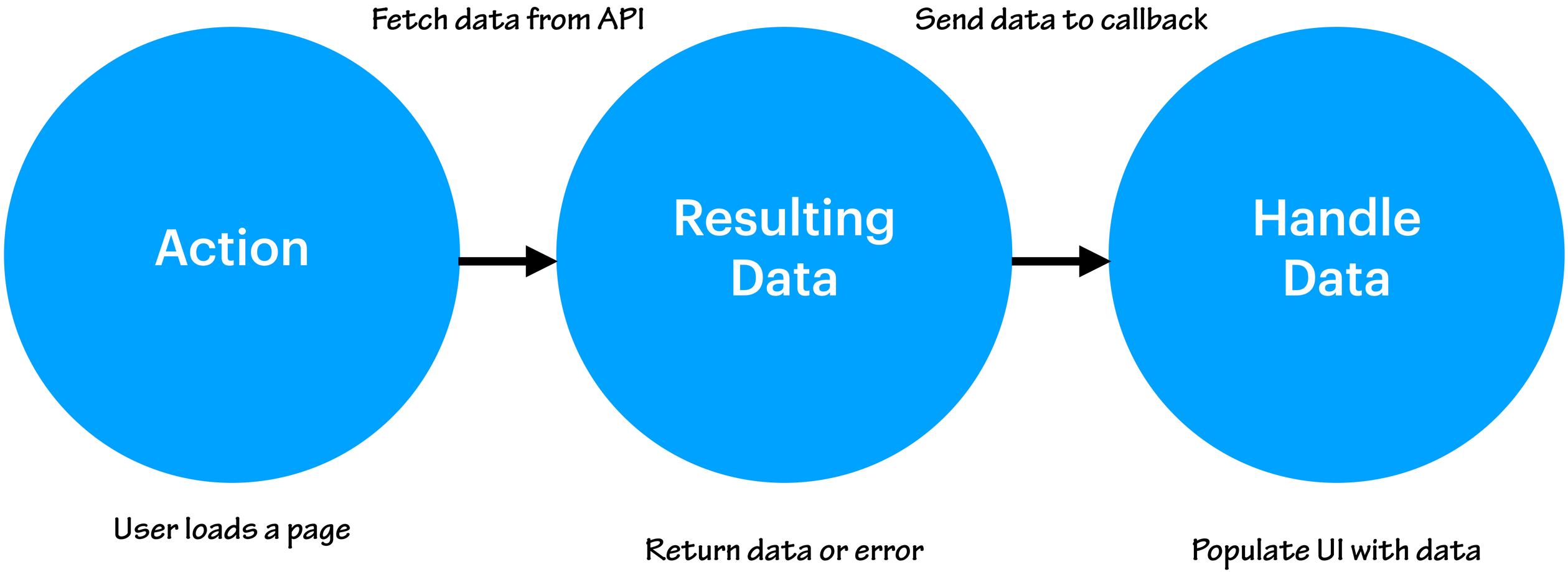
Elijah Wilson

```
1 function greeter(name) {
2   console.log('hello, ' + name)
3 }
```

```
1 const greeter = name ⇒ {
2   console.log('hello, ' + name)
3 }
```

```
1  const findBobForUser = userId ⇒ {
2    const user = api.getUser(userId)
3
4    if (!user.isStaff) {
5      return new Error('unauthorized')
6    }
7
8    const userFriends = api.getUserFriends(userId)
9    return userFriends.filter(f ⇒ /^bob.*/i.test(f.name))
10 }
11
12 console.log(findBobForUser(123))
```


Action
Resulting Data
Handle Data


Fetch data from API
Send data to callback
Action
Resulting Data
Handle Data
User loads a page
Return data or error
Populate UI with data

	t0	t1	t2	t3	t4	t5
UI Interaction	Page is loaded					See data from API, and interact with it
UI		Loading spinner is shown			Loading spinner hidden, populate UI with data	
Foreground	HTTP request is sent to load data, callback is created			Callback is called		
Background			Data is received			

```
1 const findBobForUser = (userId, cb) ⇒ {
2   api.getUser(userId, user ⇒ {
3     if (!user.isStaff) {
4       throw new Error("unauthorized")
5     }
6
7     api.getUserFriends(userId, userFriends ⇒ {
8       cb(userFriends.filter(f ⇒ /^bob.*/i.test(f.name)))
9     })
10  })
11 }
12
13 findBobForUser(123, friends ⇒ console.log(friends))
```

```
1 const getUser = userId ⇒ {
2   return fetch('/api/users' + userId).then(resp ⇒ resp.json())
3 }
4
5 getUser(123)
6   .catch(err ⇒ console.error(err))
7   .then(user ⇒ console.log(user))
8
```

```
1 const findBobForUser = userId ⇒ {
2   return api.getUser(userId).then(user ⇒ {
3     return api.getUserFriends(user.id).then(userFriends ⇒ {
4       return userFriends.filter(f ⇒ /^bob.*/i.test(f.name))
5     })
6   })
7 }
8
9 findBobForUser(123)
10   .catch(err ⇒ console.error(err))
11   .then(friends ⇒ console.log(friends))
12
```

```
const findBobForUser = userId => {
  return new Promise((resolve, reject) => {
    api
      .getUser(userId)
      .catch(err => reject(err))
      .then(user => {
        if (!user.isStaff) {
          reject(new Error('unauthorized'))
          return
        }

        api
          .getUserFriends(userId)
          .catch(err => reject(err))
          .then(userFriends => {
            resolve(userFriends.filter(f => /^bob.*/i.test(f.name)))
          })
      })
  })
}

findBobForUser(123)
  .catch(err => console.error(err))
  .then(friends => console.log(friends))
```

resolve = succeeded

reject = failed

```
1  const { Octokit } = require('@octokit/rest')
2  const octokit = new Octokit()
3
4  octokit.repos
5    .listForOrg({
6      org: 'octokit',
7      type: 'public'
8    })
9    .then(({ data }) => {
10     // handle data
11   })
```

```
async function findBobForUser(userId) {
  const user = await api.getUser(userId)
  if (!user.isStaff) {
    throw new Error('unauthorized')
  }

  const userFriends = await api.getUserFriends(userId)
  return userFriends.filter(f => /^bob.*/i.test(f.name))
}

try {
  const friends = await findBobForUser(123)
} catch (e) {
  console.error(e)
}
```

```
const findBobForUser = userId ⇒ {
  const user = api.getUser(userId)

  if (!user.isStaff) {
    return new Error('unauthorized')
  }

  const userFriends = api.getUserFriends(userId)
  return userFriends.filter(f ⇒ /^bob.*/i.test(f.name))
}

console.log(findBobForUser(123))
```

```
async function findBobForUser(userId) {
  const user = await api.getUser(userId)
  if (!user.isStaff) {
    throw new Error('unauthorized')
  }

  const userFriends = await api.getUserFriends(userId)
  return userFriends.filter(f ⇒ /^bob.*/i.test(f.name))
}

try {
  const friends = await findBobForUser(123)
  console.log(friends)
} catch (e) {
  console.error(e)
}
```

With async/await, why have Promises?

```
async function getUser(userId) {
  return await fetch('/api/users/' + userId)
}

getUser(123).then((data) ⇒ console.log(data))
```

```
async function getUser(userId) {
  return await fetch('/api/users/' + userId)
}

const data = await getUser(123)
console.log(data)
```

```
function getUser(userId) {
  return fetch('/api/users/' + userId)
}

const data = await getUser(123)
console.log(data)
```

```
function getUser(userId) {
  return fetch('/api/users/' + userId)
}

getUser(123).then((data) ⇒ console.log(data))
```

```
1 const getUsers = userIds ⇒ {
2   return Promise.all([ ... userIds.map(userId ⇒ api.getUser(userId))])
3 }
4
5 const users = await getUsers([123, 42])
6 console.log(users) // [{id: 123, name: "john"}, {id: 42, name: "bob"}]
```

https://git.io/JvKKZ