EMPTY PROMISES

BY BRIAN SCHILLER

LOADING AN IMAGE IN JS

```
function imageSize(url) {
  return new Promise((resolve, reject) => {
    const img = new Image();

    img.onload = function() {
      resolve({ width: this.width, height: this.height });
    };
    img.onerror = reject;

    img.src = url;
  });
}
```

LOADING AN IMAGE IN JS

```
function imageSize(url) {
  const prom = emptyPromise();
  const img = new Image();

  img.onload = function() {
    prom.resolve({ width: this.width, height: this.height });
  };
  img.onerror = prom.reject;

  img.src = url;
  return prom;
}
```

WITHOUT emptyPromise

```
function fileContents(path) {
  return new Promise((resolve, reject) => {
    fs.readFile(path, (e, data) => {
      if (e) reject(e);
      else resolve(data);
    });
  });
}
```

WITH emptyPromise

```
function fileContents(path) {
  const p = emptyPromise();

  fs.readFile(path, (e, data) => {
    if (e) p.reject(e);
    else p.resolve(data);
  });

  return p;
}
```

DEFINITION

```
function emptyPromise() {
  let callbacks;
  const p = new Promise((resolve, reject) => {
    callbacks = { resolve, reject };
  });

  p.resolve = (val) => callbacks.resolve(val);
  p.reject = (val) => callbacks.reject(val);

  return p;
}
```

WHEN MIGHT YOU USE IT?

WHEN YOU NEED TO RESOLVE/REJECT A PROMISE OUTSIDE THE SCOPE WHERE THE PROMISE IS CREATED

EXAMPLES

- CONVERTING CALLBACK-BASED CODE
- WAITING FOR `isLoggedIn` TO SETTLE
- REQUEST CONSOLIDATION

WAIT FOR `isLoggedIn` TO SETTLE

(LIVECODING)

WAITING ON ACTIONS

```
hagridActions: ['fetchProjects'],
async mounted() {
  await this.hagridPromise('fetchProjects');
  // at this point, you can be confident that projects have been fetched.
  const toSelect = this.$route.query.projectId || this.projects[0].id;
  this.selectProject(this.projects.find(p => p.id === toSelect));
},
```

WAITING ON ACTIONS

```
hagridActions: ['fetchProjects'],
async mounted() {
  await this.hagridPromise('fetchProjects');
  // at this point, you can be confident that projects have been fetched.
  const toSelect = this.$route.query.projectId || this.projects[0].id;
  this.selectProject(this.projects.find(p => p.id === toSelect));
},
```

WHAT IF SOMEONE REQUESTS

`hagridPromise('notYetDispatched')`?

```
getPromise(actionName) {








}
setPromise(actionName, p) {





}
```

```
getPromise(actionName) {
  if (this.promises[actionName]) return this.promises[actionName];









}
setPromise(actionName, p) {







}
```

```
getPromise(actionName) {
  if (this.promises[actionName]) return this.promises[actionName];
  if (this.unknownPromises[actionName]) {
    return this.unknownPromises[actionName];
  }





}
setPromise(actionName, p) {






}
```

```
getPromise(actionName) {
  if (this.promises[actionName]) return this.promises[actionName];
  if (this.unknownPromises[actionName]) {
    return this.unknownPromises[actionName];
  }

  this.unknownPromises[actionName] = emptyPromise();
  //                                 ☝

}
setPromise(actionName, p) {




}
```

```
getPromise(actionName) {
  if (this.promises[actionName]) return this.promises[actionName];
  if (this.unknownPromises[actionName]) {
    return this.unknownPromises[actionName];
  }

  this.unknownPromises[actionName] = emptyPromise();
  //                                 ☝️
  return this.unknownPromises[actionName];
}
setPromise(actionName, p) {





}
```

```
getPromise(actionName) {
  if (this.promises[actionName]) return this.promises[actionName];
  if (this.unknownPromises[actionName]) {
    return this.unknownPromises[actionName];
  }

  this.unknownPromises[actionName] = emptyPromise();
  //                                 ☝️
  return this.unknownPromises[actionName];
}
setPromise(actionName, p) {
  if (this.unknownPromises[actionName]) {
    this.unknownPromises[actionName].resolve(p);
    delete this.unknownPromises[actionName];
  }

}
```

```
getPromise(actionName) {
  if (this.promises[actionName]) return this.promises[actionName];
  if (this.unknownPromises[actionName]) {
    return this.unknownPromises[actionName];
  }

  this.unknownPromises[actionName] = emptyPromise();
  //                                 ☝️
  return this.unknownPromises[actionName];
}
setPromise(actionName, p) {
  if (this.unknownPromises[actionName]) {
    this.unknownPromises[actionName].resolve(p);
    delete this.unknownPromises[actionName];
  }
  this.promises[actionName] = p;
}
```