LESSON 20

PROMISE

ASYNC APPROACHES

- callback functions -> callback hell (http://callbackhell.com/)
- ➡ promise
- async/await

PROMISE (ОБЕЩАНИЕ)

- Interface
- The `Promise` object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
- Create promise

  new Promise(function(resolve, reject) {…})
- States
  - pending
  - fulfilled
  - rejected

EXAMPLE

```
asyncAction(function (result) {
  // ...
  innerFunction(result, function (data) {
    // ...
    anotherAction(data, function (processedData) {
      // ...
      console.log(processedData);
    });
  });
});
```

```
promise
  .then(function (result) {
    // ...
    return asyncAction(result);
  })
  .then(function (data) {
    // ...
    return innerFunction(data);
  })
  .then(function (processedData) {
    // ...
    console.log(processedData);
    return anotherAction(processedData);
  });
```

EXAMPLE

```
let combineStr = new Promise((resolve, reject) => {
  resolve("Data");
});

combineStr
  .then((str) => {
    return `${str} shipped`;
  })
  .then((str) => {
    return `${str} successfully`;
  })
  .then((str) => {
    console.log(str);
  });
```

PROMISIFICATION FETCH

```
function httpGet(url) {
  return new Promise(function (resolve, reject) {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", url, true);

    xhr.onload = function () {
      if (this.status == 200) {
        resolve(this.response);
      } else {
        var error = new Error(this.statusText);
        error.code = this.status;
        reject(error);
      }
    };

    xhr.onerror = function () {
      reject(new Error("Network Error"));
    };

    xhr.send();
  });
}
```

THROW

- throw new Error(error)

STATIC METHODS

- Promise.all(promises)
- Promise.allSettled(promises)
- Promise.race(promises)
- Promise.resolve(value)
- Promise.reject(error)

EXAMPLES

```
let promise1 = new Promise((resolve, reject) => {
  setTimeout(resolve, 2000, "promise1 finished");
});

let promise2 = new Promise((resolve, reject) => {
  setTimeout(resolve, 4000, "promise1 finished");
});

Promise.all([promise1, promise2]).then((values) => {
  console.log(values);
});
```

LINKS

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
- https://habr.com/ru/company/zerotech/blog/317256/
- https://medium.com/@stasonmars/%D0%BF%D1%80%D0%BE%D0%BC%D0%B8%D1%81%D1%8B-%D0%B2-javascript-%D0%B4%D0%BB%D1%8F-%D1%87%D0%B0%D0%B8%CC%86%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2-60bbef963541
- https://learn.javascript.ru/promise
- https://learn.javascript.ru/promise-api