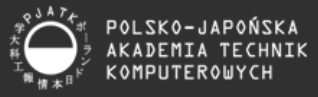
daftcode

FRONT END

Level UP



daftcode / spring **2018**

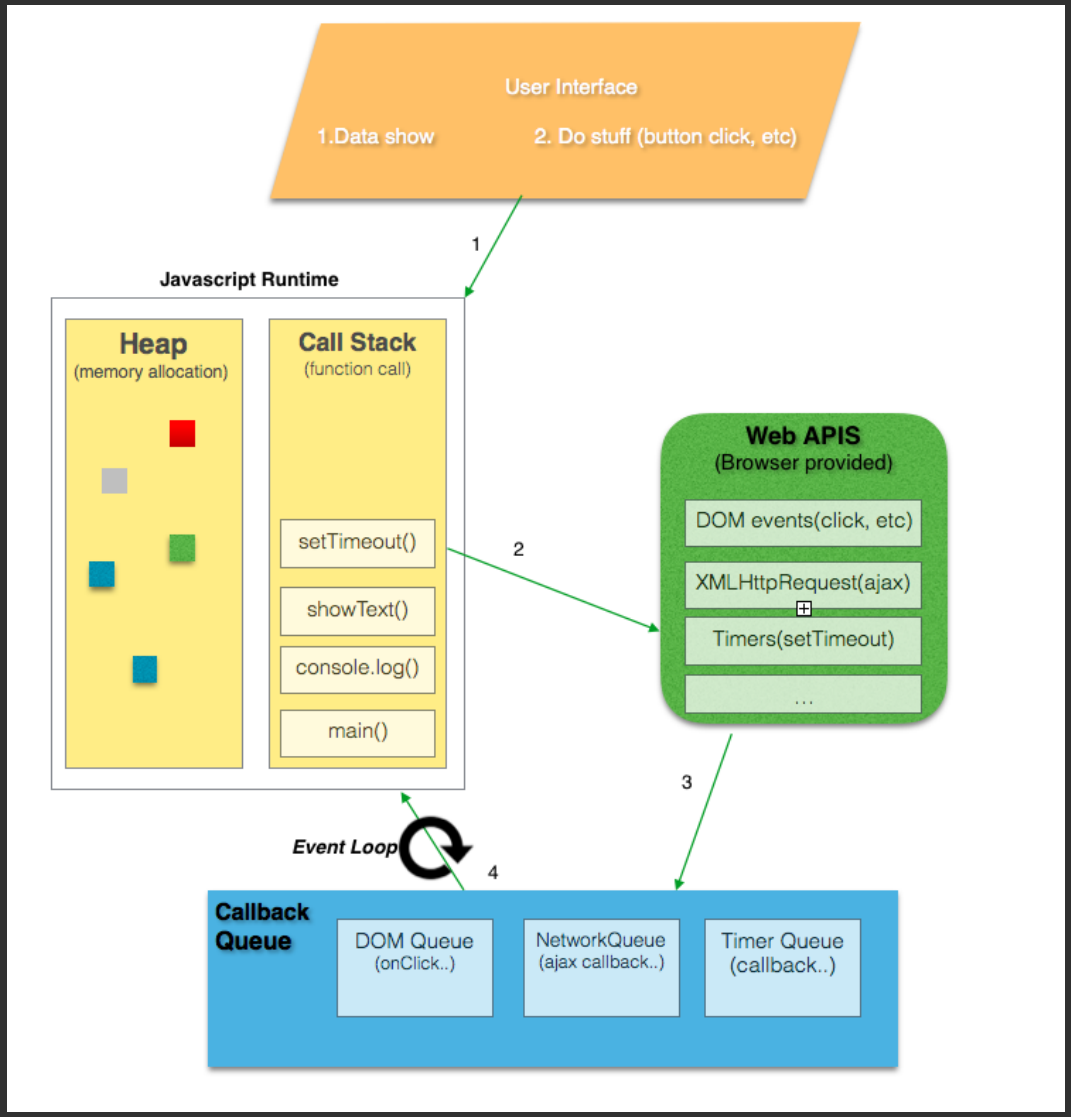
- Asynchronous JS
- & Fetch API
- Mariusz Kaczkowski

daftcode

Sync vs Async

daftcode

JS Engine



Async techniques

- Callbacks
- Promises
- Async / Await
- Generators*

daftcode

Callback ➡ call you back later

```
function handlePhoto (error, photo) {
    if (error) console.error('Download error!', error)
    else console.log('Download finished', photo)
}

downloadPhoto('http://coolcats.com/cat.gif', handlePhoto)
```

daftcode

Callback Hell

```
const verifyUser = function(username, password, callback){
  dataBase.verifyUser(username, password, (error, userInfo) ⇒ {
    if (error) {
      callback(error)
    }else{
      dataBase.getRoles(username, (error, roles) ⇒ {
        if (error){
          callback(error)
        }else {
          dataBase.logAccess(username, (error) ⇒ {
            if (error){
              callback(error);
            }else{
              callback(null, userInfo, roles);
            }
          })
        }
      })
    }
  })
};
```

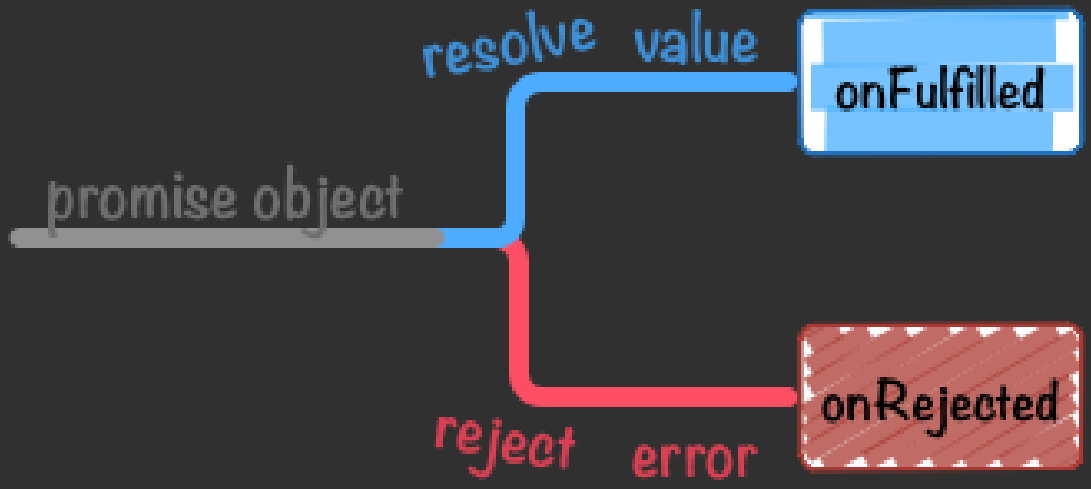
Callback Hell

"Write small modules that each do one thing, and assemble them into other modules that do a bigger thing. You can't get into callback hell if you don't go there."

Isaac Schlueter

daftcode

Promise



- pending
- fulfilled
- rejected

daftcode

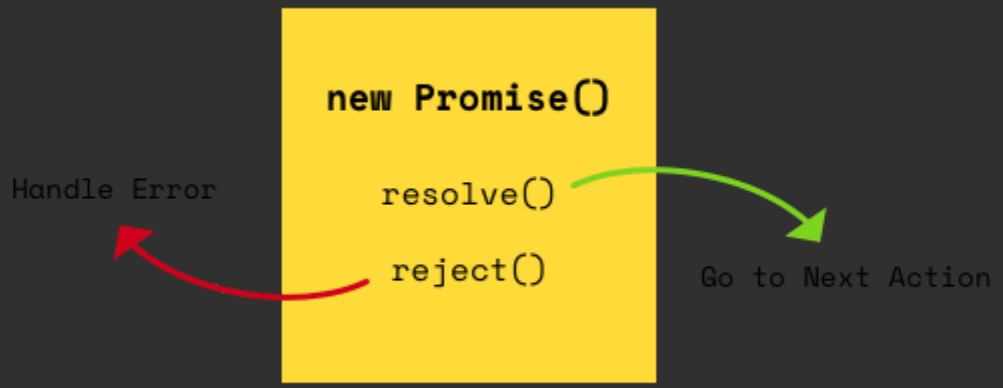
Promise - Sample

```
const promise = new Promise((resolve, reject) => {
  //asynchronous code goes here
  const request = new XMLHttpRequest();
  request.open('GET', 'https://api.spacex.com/launch/1');
  request.onload = () => {
    if (request.status === 200) {
      // we got data here, so resolve the Promise
      resolve(request.response);
    } else {
       // status is not 200 OK, so reject
      reject(new Error(request.statusText));
    }
  }
});
```


daftcode

daftcode

Promise



daftcode

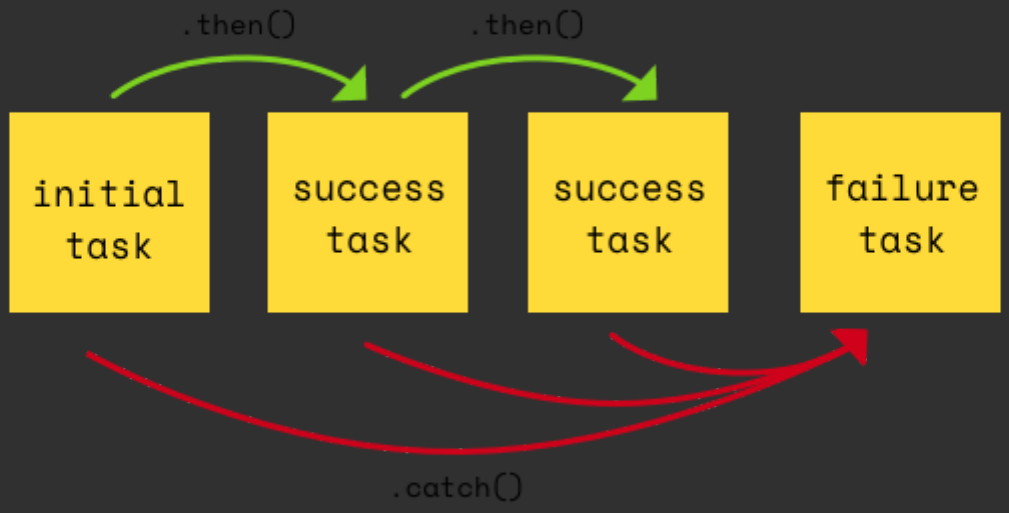
Promise - Handling error

```
promise.then(result ⇒ {
  console.log('Got data!', result);
}).catch(error ⇒ {
  console.log('Error occurred!', error);
});

// This is equivalent to:

promise.then(result ⇒ {
  console.log('Got data!', result);
}).then(undefined, error ⇒ {
  console.log('Error occurred!', error);
});
```


daftcode

daftcode

Promise - Flow



daftcode

Chaining Promises

```
promise.then(result ⇒ {
  return getPromise(result);
}).then(result ⇒ {
  //handle the final result
}).catch((error) ⇒ {
  console.log('Error occurred!', error);
});
```

daftcode

Promise utils

```
Promise.resolve('resolves').then(data ⇒
  alert(data); //this is called
});

Promise.reject('rejects').then(data ⇒
  alert(data); // this is never called
).catch(err ⇒ alert(err)) //this is called;
```

daftcode

Promise.all

```
const p1 = Promise.resolve('foo');
const p2 = new Promise((res, rej) ⇒ { setTimeout(res, 100, 'bar') });
const p3 = "baz";

Promise.all(p1, p2, p3)
  .then(([a1, a2, a3]) ⇒ console.log(a1, a2, a3));
  // "foo" "bar" "baz"

const p4 = Promise.reject('qux');
all(p1, p2, p3, p4)
  .then(([a1, a2, a3]) ⇒ console.log(a1, a2, a3))
  .catch(e ⇒ console.log(e));
  // "qux"
```

daftcode

Promise.race

```
const p1 = new Promise((res, rej) => { setTimeout(res, 200, 'foo') });
const p2 = new Promise((res, rej) => { setTimeout(res, 100, 'bar') });
race(p1, p2).then(a => console.log(a)); // "bar"

const p3 = Promise.reject('qux');
race(p1, p2, p3)
  .then(a => console.log(a))
  .catch(e => console.log(e)); // "baz"
```

daftcode

Promise - example

```
const verifyUser = function(username, password) {
  database.verifyUser(username, password)
    .then(userInfo ⇒ dataBase.getRoles(userInfo))
    .then(rolesInfo ⇒ dataBase.logAccess(rolesInfo))
    .then(finalResult ⇒ {
      //do whatever the 'callback' would do
    })
    .catch((err) ⇒ {
      //do whatever the error handler needs
    });
};
```

daftcode

async / await

```
// promise way
function read () {
  samplePromise().then(txt ⇒ console.log(txt))
}

//async/await way
async function read () {
  const txt = await samplePromise();
  console.log(txt);
}
```

daftcode

async / await - errors

```
async function read () {
  try{
    var txt = await samplePromise();
    console.log(txt);
  } catch(e) {
    // finished with failure, reject the promise
  }
}
```

daftcode

async / await - sample

```
const verifyUser = async function(username, password){
  try {
    const userInfo = await dataBase.verifyUser(username, password);
    const rolesInfo = await dataBase.getRoles(userInfo);
    const logStatus = await dataBase.logAccess(userInfo, userInfo);
    return userInfo;
  }catch (e){
    //handle errors as needed
  }
};
```

daftcode

async / await bonus 🚀

```
//Promise.all
const all = (...params) => Promise.all(...params);
const [r1, r2] = await all(p1, p2);

//Spread for array of promises
const [...results] = await all(...requests);
```

daftcode

Client ↔ Server

daftcode

Before Fetch API

XMLHttpRequest

```
const reqError = (err) => alert(err);
const reqListener = () => {
  const data = JSON.parse(this.responseText);
  console.log(data);
};

const oReq = new XMLHttpRequest();
oReq.onload = reqListener;
oReq.onerror = reqError;
oReq.open('get', './api/some.json', true);
oReq.send();
```

daftcode

Before Fetch API

JQuery

```
$.ajax('some-url', {
  success: (data) => { /* OK */ },
  error: (err) => { /* Error */}
});
```

daftcode

Fetch API

Fetch

```
fetch('./api/some.json')
  .then(response ⇒ response.json())
  .then(data ⇒ alert('success'))
  .catch(error ⇒ alert('error'));
```

daftcode

Fetch - Response

```
fetch('users.json', options).then(response => /* ... */)

//response
{
  body: ReadableStream
  bodyUsed: false
  headers: Headers
  ok : true
  redirected : false
  status : 200 // 400 Bad Request, 404 Not Found, 401 Unauthorized,
  statusText : "OK"
  type : "cors"
  url : "http://some-website.com/some-url"
  __proto__ : Response
}
```

daftcode

Fetch - options

```
fetch(url, {
  method: 'POST', // *GET, POST, PUT, DELETE, etc.
  headers: {"Content-type": "application/x-www-form-urlencoded"},
  credentials: 'include', // include, same-origin, *omit
  cache: 'no-cache', // *default, no-cache, reload, force-cache
  mode: 'cors', // no-cors, cors, *same-origin
  body: 'foo=bar&lorem=ipsum', // string, FormData
  redirect: 'follow', // manual, *follow, error
  referrer: 'no-referrer', // *client, no-referrer
})
  .then(response => {
    if (response.ok) {
      return response.json() //arrayBuffer,blob,json,text,formData
    } else {
      /* not ok :( */
    }
  }.catch(function (error) {
    throw new Error('something went wrong!')
   // OR - (but not both)
    return Promise.reject('something went wrong!')
})
```

- method
- headers
- credetnails
- cache
- mode
- body
- redirect
- referrer

daftcode

Fetch compatibility

Desktop

Chrome	Opera	Firefox	IE	Edge	Safari
42	29	39	No	14	10.1

Mobile / Tablet

iOS Safari	Opera Mobile	Opera Mini	Android	Android Chrome	Android Firefox
10.3	37	No	62	66	57

Polyfills:

- whatwg-fetch [IE10+]
- wisomorphic-fetch
- fetch-ie8

daftcode

Cookies 🍪 (IE 2 - 1995)

Set-Cookie header (server):

- Set-Cookie: yummy_cookie=choco
- Domain=mydomain.com; Secure; HttpOnly; Expires=Wed, 21 Oct 2015 07:28:00 GMT

Cookie HTTP header (client):

- Cookie:yummy_cookie=choco

JS

- document.cookie = "tasty_cookie=strawberry";
- console.log(document.cookie);

daftcode

Fetch - async / await

```
async function fetchTopFive(sub) {
  try {
    const URL = `https://www.reddit.com/r/${sub}/.json?limit=5`;
    const fetchResult = fetch(URL)
    const response = await fetchResult;
    const jsonData = await response.json();
    console.log(jsonData);
  } catch (e) {
    throw Error(e);
  }
}

fetchTopFive('javascript');
```

daftcode

Fetch - Axios

```
// POST
axios.post('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });

// GET
const response = await axios.get('/user?ID=12345');
```