cphbusiness

JAVASCRIPT2

Install

- Visual Studio Code
  https://code.visualstudio.com/

- Node.js
  https://nodejs.org/en/

cphbusiness

JAVASCRIPT2

Subjects / Code examples

- ECMASCRIPT
- var / const / let
- Classes / Inheritance
- Arrow functions
- Modules
- Arrays
- Callback functions
- Promises
- Async / Await
- Closures
- Module pattern
- Scope
- This

cphbusiness

JAVASCRIPT2

JavaScript Projects

- SinglePageApplication

Frontend project

https://github.com - code_simple_SPA

- JSONServer

Backend REST project

https://github.com - code_jsonserver_with_errors

cphbusiness

JAVASCRIPT2

JavaScript Array Filter Callback Function

Create arrays

```
const names1 = ["Franco", "Alberto", "Miguel", "Jose", "Fernando", "Pepe", "Manuel"];
const names2 = ["Franco", "Alberto", "Miguel", "Jose", "Fernando", "Pepe"];
```

Implement myFilter(array, callback)

Implement filter callback function

Return element

If name is longer than 4 letters OR name position in array is even OR number of names in array is even

Test myFilter with array and filter callback function / Test array.filter(callback)

Correct output should be…

Names1 ---> ['Franco', 'Alberto', 'Miguel', 'Fernando', 'Manuel']

Names2 ---> ['Franco', 'Alberto', 'Miguel', 'Jose', 'Fernando', 'Pepe']

cphbusiness

JAVASCRIPT2

JavaScript Array Filter Callback Function

```
const names1 = ["Franco", "Alberto", "Miguel", "Jose", "Fernando", "Pepe", "Manuel"];
const names2 = ["Franco", "Alberto", "Miguel", "Jose", "Fernando", "Pepe"];
function myFilter(array, callback)
{
    let arrayFiltered = [];
    for(let index = 0; index < array.length; index++)
    {
        if(callback(array[index],index,array))
        {
            arrayFiltered.push(array[index]);
        }
    }
    return arrayFiltered;
}
const filter1 = (element, index, array) => {
    if (element.length > 4 || index % 2 == 0 || array.length % 2 == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
};
console.log(myFilter(names1, filter1));
console.log(names1.filter(filter1));
console.log(myFilter(names2, filter1));
console.log(names2.filter(filter1));
```

cphbusiness

JSON

JavaScript Object Notation

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand
- JSON evaluates to JavaScript Object

```
var text = '{"name":"John Johnson","street":"Oslo West 16","phone":"555 1234567"}';
var obj = JSON.parse(text);
document.getElementById(”Person").innerHTML = obj.name;
```

cphbusiness

JSON / XML

Differences / Similarities

Much Like XML Because

- Both JSON and XML is "self describing" (human readable)
- Both JSON and XML is hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

Much Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays
- The biggest difference is:
- XML has to be parsed with an XML parser, JSON can be parsed by a standard JavaScript function.

cphbusiness

JSON / XML

Advantages

Why JSON?
For AJAX applications, JSON is faster and easier than XML:

Using XML
- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

Using JSON
- Fetch a JSON string
- JSON.Parse the JSON string

cphbusiness

JSON Rules

Syntax rules

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

cphbusiness

JSON Conversion

String from Object

```
const employees = { employees: [
        {firstName:"John",lastName:"Doe"},
        {firstName:"Anna",lastName:"Smith"},
        {firstName:"Peter",lastName:"Jones"},
]};
const text = JSON.stringify(employees);
console.log(text);
```

Object from string

```
const text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
const obj = JSON.parse(text);
console.log(obj.employees[1].firstName + " " + obj.employees[1].lastName);
```

Older browsers / No JSON.parse() support

```
const text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
const obj = eval ("(" + text + ")");
console.log(obj.employees[1].firstName + " " + obj.employees[1].lastName);
```

cphbusiness

AJAX

XML

- AJAX = Asynchronous JavaScript and XML.
- AJAX is not a programming language, but a way to use existing standards.
- AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.
- AJAX is a technique for creating fast and dynamic web pages.
- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- The XMLHttpRequest object is used to exchange data with a server.

cphbusiness

XMLHttpRequest object

Asynchronous

TRUE

- Execute other scripts while waiting for server response
- Deal with the response when the response ready

FALSE

- Will NOT continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop.

cphbusiness

Server response

responseText

- get the response data as a string

responseXML

- get the response data as XML data

cphbusiness

Events

onreadystatechange

The onreadystatechange event is triggered every time the readyState changes.
The readyState property holds the status of the XMLHttpRequest.

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

cphbusiness

AJAX Basic

```
const ajaxBasic = () => {
    const request = new XMLHttpRequest();
    request.onreadystatechange = () => {
        console.log('State: ' + request.readyState + ' Status: ' + request.status);
        if (request.readyState === 4 && request.status === 200) {
            console.log(JSON.parse(request.responseText));
        }
    }
    request.open('GET', 'http://localhost:3333/api/users', true);
    request.send();
}
ajaxBasic();
```

cphbusiness

AJAX General

```
const ajaxGeneral = (url, cb) => {
    const request = new XMLHttpRequest();
    request.onreadystatechange = () => {
        if (request.readyState == 4 && request.status == 200) {
            cb(JSON.parse(request.responseText));
        }
    }
    request.open("GET", url, true);
    request.send();
}
ajaxGeneral('http://localhost:3333/api/users', (json) => {
    console.log(json);
});
```

cphbusiness

FETCH Basic

```
const fetchBasic = () => {
fetch('http://localhost:3333/api/users')
    .then(function (response) {
        return response.json();
    })
    .then(function (json) {
        console.log(json);
    });
}
fetchBasic();
```

cphbusiness

FETCH General

```
const fetchGeneral = (url, cb) => {
    fetch(url)
        .then(function (response) {
            return response.json();
        }).then(function (json) {
            cb(json);
        }).catch(function (error) {
            console.log('Error: ' + error);
        });
}
fetchGeneral('http://localhost:3333/api/users', json
=> {
    console.log(json);
});
```

cphbusiness

FETCH Options

```
function makeOptions(requestType, body) {
    return {
        method: requestType,
        headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(body)
    };
}
```

cphbusiness

FETCH Pattern

```
function handleHttpErrors(response) {
    if (!response.ok) {
        return Promise.reject({ status: response.status, error: response.json() })
    }
    return response.json();
}
const url1 = 'http://localhost:3333/api/users';
const data1 = { age: 34, name: "Anton Benson", gender: "female", email: "ab@ab.com" };
const options1 = makeOptions("POST", data1);
fetch(url1, options1)
    .then(handleHttpErrors)
    .then(json => console.log(json))
    .catch(error => {
        if (error.status)
        {
            error.error.then(e => console.log(e))
        }
        else
        {
            console.log("Network error");
        }
    });
```

cphbusiness

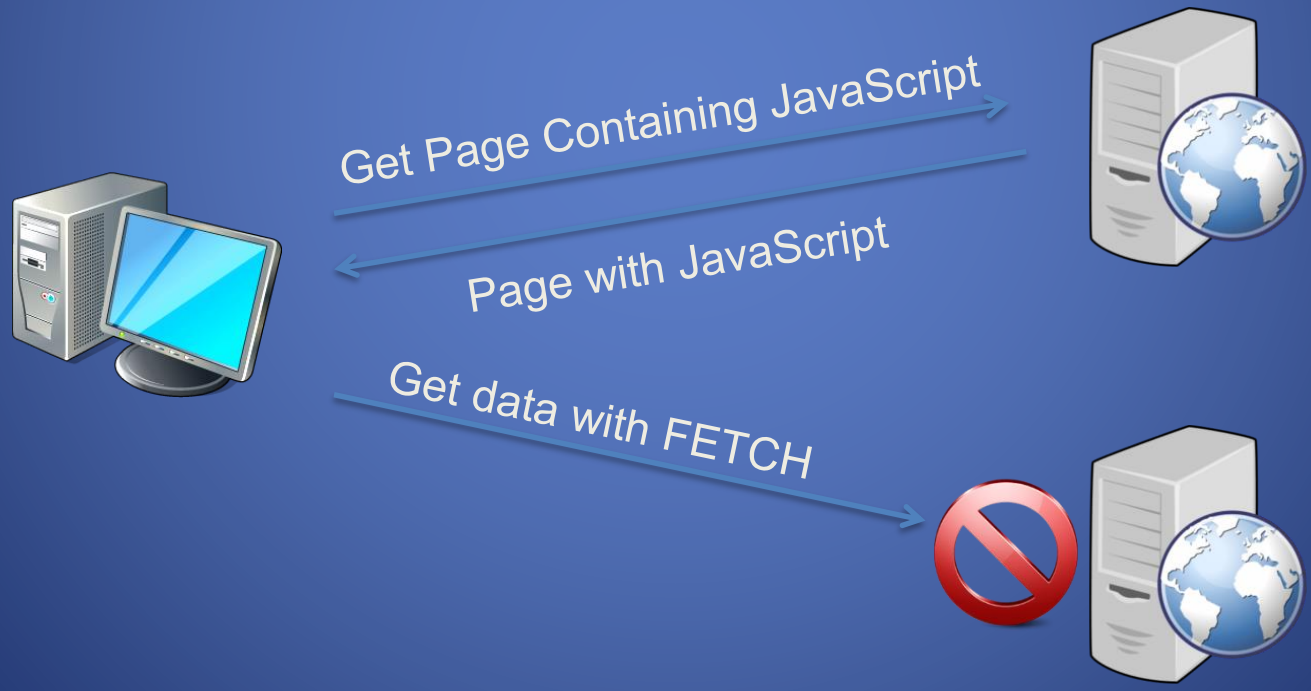
FETCH Final

```
const fetchJson = async (url, options) => {
    try {
        const response = await fetch(url, options);
        const json = await response.json();
        if (!response.ok) {
            return Promise.reject({ status: response.status, json: json });
        }
        return Promise.resolve({ json: json });
    }
    catch (error) {
        return Promise.reject({ error: 'Network error' });
    }
}
const url2 = 'http://localhost:3333/api/users';
const data2 = { age: 34, name: "Anton Benson", gender: "male", email: "ab@ab.com" };
const options2 = makeOptions("POST", data2);
fetchJson(url2, options2)
    .then((resolve) => console.log(resolve.json))
    .catch((reject) => console.log(reject.json));
```

cphbusiness

Same Origin Policy

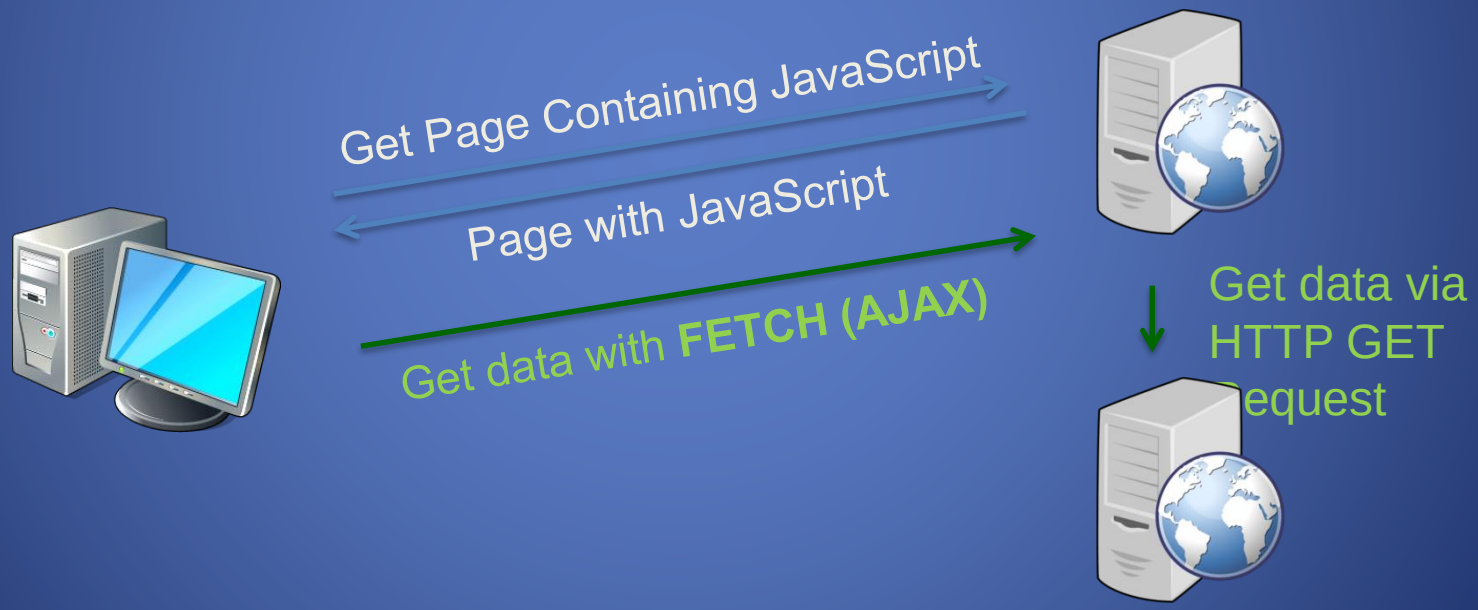
Problem: External servers disallowed



cphbusiness

Same Origin Policy

Solution. Proxy server



cphbusiness

CORS

Cross Origin Ressource Sharing

Solution 1 (If not own server) - Technique for proxying requests

```
URL url = new URL("http://ip.jsontest.com/?alloworigin=false");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("User-Agent", "");

StringBuilder result = new StringBuilder();
BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
String line = "";

while ((line = br.readLine()) != null)
{
   result.append(line);
}

br.close();

return Response.ok(result.toString()).build();
```

cphbusiness

CORS

Cross Origin Ressource Sharing

Solution 2 (If own server) - Technique for relaxing the same-origin policy

```
return Response
        .ok("{\"success\":true}")
        .header("Access-Control-Allow-Origin", "*")
        .header("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE")
        .build();
```

cphbusiness

CORS

Cross Origin Ressource Sharing

Solution 3 (If own server) – Set up response filters

```
@Provider
@PreMatching
public class CORSResponseFilter implements ContainerResponseFilter
{
   @Override
   public void filter(ContainerRequestContext requestCtx, ContainerResponseContext res) throws IOException
   {
      System.out.println("RESPONSE HEADER ADDED...");

      res.getHeaders().add("Access-Control-Allow-Origin", "*");
      res.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");
      res.getHeaders().add("Access-Control-Allow-Credentials", "true");
      res.getHeaders().add("Access-Control-Allow-Headers", "Origin, Accept, Content-Type");
   }
}
```

cphbusiness

CORS

Cross Origin Ressource Sharing

Set up request filters

```
@Provider
@PreMatching
public class CORSRequestFilter implements ContainerRequestFilter
{
   @Override
   public void filter(ContainerRequestContext requestCtx) throws IOException
   {
      if (requestCtx.getRequest().getMethod().equals("OPTIONS"))
      {
         System.out.println("REQUEST WITH METHOD OPTIONS DETECTED...");

         requestCtx.abortWith(Response.status(Response.Status.METHOD_NOT_ALLOWED).build());
      }
   }
}
```

cphbusiness

DEPLOYMENT

Single Page Application

Deploy backend to Digital Ocean
Deploy frontend to Surge.sh