JavaScript, JSON, AJAX

Grzegorz Ostrek
grzegorz.ostrek@pw.edu.pl
Wydział Matematyki i Nauk Informacyjnych
00-662 Warszawa
ul. Koszykowa 75

Functions - review

- Anonymous functions
- Callback functions
- IIFE (Immediately Invoked Function Expression).
- Constructors – new keyword
- Parameters
  - arguments – additional argument
  - arguments.length – number of actually passed
  - length property – number of arguments with name
- Function overloading
- Closures

Declarations, hoisting

- var
  - function scope
  - hoisted
- let
  - Block scope
  - Not hoisted – can be used following declaration
- const
  - Block scope
  - For object, allow change properties
  - Not hoisted – can be used following a declaration

Variables created without a declaration keyword (var, let, or const) are always global, even if they are created inside a function.

Declarations, hoisting, cont.

- **function declaration (function statement)**
  - function name([param[, param[, ... param]]]) { statements }

Must have a name

Starts with ‘function’

hoisted

- **function expression (function expression)**
  - function [name]([param[, param[, ... param]]]) { statements }

Not hoisted

    - var myFunction = function() { statements }
    - var myFunction = function namedFunction(){ statements }

Add debug trace

Anonymous / Callback function

name = ‘useless’

```
function useless(callback) { return callback(); }
```

```
var text = 'Domo arigato!';
```

anonymous – no name

```
useless(function(){ return text; }) === text;    //true
```

Immediately Invoked Function Expression

```
(function()
   { statements }
)();
```

- Call immediatly
- Separate scope

Data-* attributes

- Used to store custom data private to the page or application
  - The attribute name should not contain any uppercase letters, and must be at least one character long after the prefix "data-"
  - The attribute value can be any string
  - engaging user experience (without any Ajax calls or server-side database queries).
  - Custom attributes prefixed with "data-" will be completely ignored by the user agent.
  - Accessed : tag.dataset property

"use strict"

- First line (script/function)
- can not use undeclared variables
- eliminates some JavaScript silent errors by changing them to throw errors

https://www.w3schools.com/js/js_strict.asp
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

this

Context, not definition

It has different values depending on where it is used:

- In a method, this refers to the owner object.
  - In constructor refers to the new object
- Alone, this refers to the global object.
  - In a browser window the Global object is [object Window]
- In a function, this refers to the global object.
  - In a function, in strict mode, this is undefined.
- In an event, this refers to the element that received the event.
- Methods like call(), and apply() can refer ‘this’ to any object.

Arrow function

- Arrow function
  - Does not have its own bindings to this or super, and should not be used as methods.

```
hello = () => {
  return "Hello World!";
}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

Function cont.

Function **parameters** are the names listed in the function definition.

Function **arguments** are the real values passed to (and received by) the function

See: TypeScript

Parameter Rules

- JavaScript function definitions do not specify data types for parameters.
- JavaScript functions do not perform type checking on the passed arguments.
- JavaScript functions do not check the number of arguments received.
- If a function is called with missing arguments (less than declared), the missing values are set to undefined.
- If a function is called with too many arguments (more than declared), these arguments can be reached using the arguments object. JavaScript functions have a built-in object called the arguments object.

Overloading – one definition - switch-case by arguments.length

Closures

- A closure is the combination of a function and the scope object in which it was created.
- Closures let you save state — as such, they can often be used in place of objects.
- Allow simulate a private members in objects

- Memory reservation needed

Timing events

window object allows execution of code at specified time intervals. These time intervals are called timing events.

The two key methods to use with JavaScript (in browser) are:

- id = **setTimeout**(function, milliseconds)
  Executes a function, after waiting a specified number of milliseconds.

- id = **setInterval**(function, milliseconds)
  Same as setTimeout(), but repeats the execution of the function continuously.

The setTimeout() and setInterval() are both methods of the HTML DOM Window object.

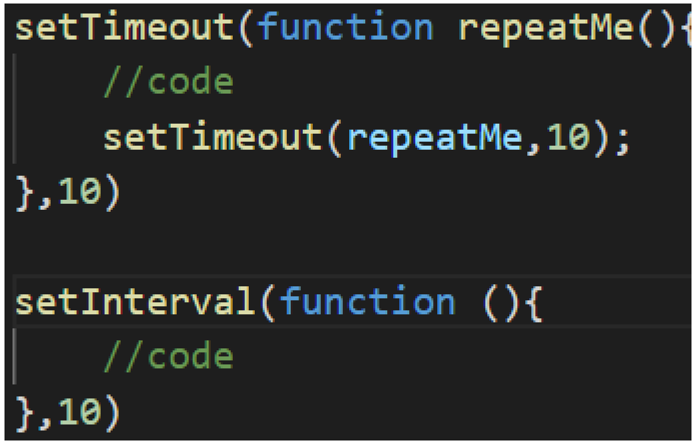
Guarantee minimum time, not exact time

clearTimeout(id), clearInterval(id) – delete a timer

Timing events cont.

- After finished function

- Queued functions (what if function takes more time than interval)



```
setTimeout(function repeatMe(){
    //code
    setTimeout(repeatMe,10);
},10)

setInterval(function (){
    //code
},10)
```

JavaScript Objects

- A JavaScript object is a <u>collection</u> of named values

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

- The named values, in JavaScript objects, are called properties

```
objectName.property
objectName["property"]
objectName[expression]
```

- Object properties can be both
  - primitive values,
  - other objects,
  - and functions

```
for (x in objectName) {  txt += objectName [x]; }
```

JavaScript Built-in Browser Objects: navigator, window, screen, history, location

https://web.archive.org/web/20100228114236/https://helephant.com/2008/08/functions-are-first-class-objects-in-javascript/

JSON: JavaScript Object Notation

- JSON is a subset of the object literal notation of JavaScript
- In fact, JSON is based on two structures:
  - A collection of name and value pairs, which can be transformed into an object,
  - A list of values (possibly objects), which can be transformed into an array or a list
- Thus, the text sent from the server-side script can be easily transformed into JavaScript object and used in the web interface layer
- JSON standard is widely supported in many programming languages incl. Java, C, C++, C#, PHP. As a consequence it can be easily used for structured data exchange.

www.json.org can be consulted for any JSON details going beyond the scope of this lecture. http://www.json.org/js.html is devoted to JSON in JavaScript. Also, see http://www.w3schools.com/json/default.asp

JSON example I – part I

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSON test</title>
<script language="JavaScript" src="json2.js" >
</script>
<script language="JavaScript">
var myJSONTestObject = {"productList": [
        {"book_id": "121", "title": "Java in 24 hours",
   "year_of_publication": "2002"},
        {"book_id": "187", "title": "UNIX Administration",
   "year_of_publication": "1997"},
        {"book_id": "203", "title": "Oracle DBA Handbook",
   "year_of_publication": "1999"}
    ]
};
```

json2.org comes from www.json.org and provides JSON object declaration. The JSON object is used to perform necessary conversions. myJSONTestObject is a JS object initialised using JSON notation.

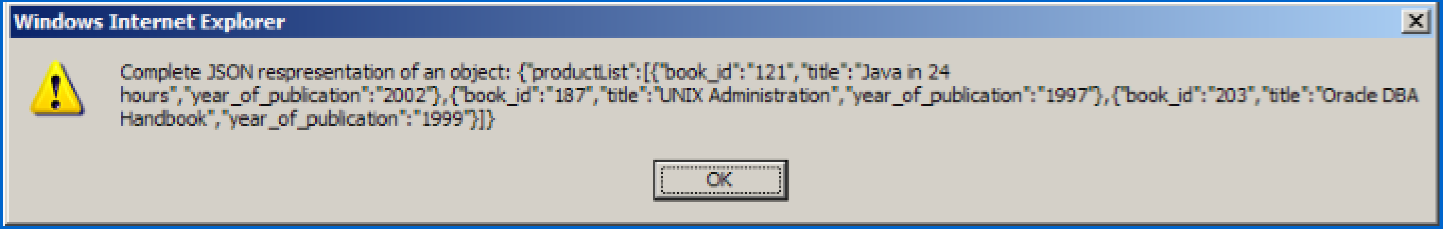
JSON Example I – part II

```
alert("The first item in an array:
   "+myJSONTestObject.productList[0].title+" published in
   "+myJSONTestObject.productList[0].year_of_publication);
```

```
var myJSONText = JSON.stringify(myJSONTestObject);
```

The JavaScript object is converted to its JSON text representation.

```
alert("Complete JSON representation of an object: "+myJSONText);
```



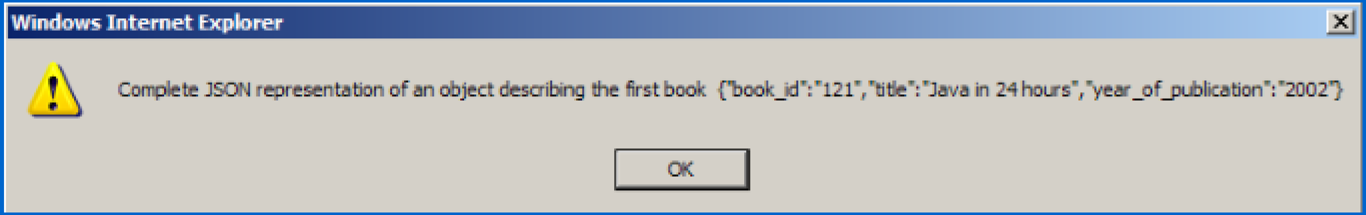
```
Complete JSON respresentation of an object: {"productList":[{"book_id":"121","title":"Java in 24
hours","year_of_publication":"2002"},{"book_id":"187","title":"UNIX Administration","year_of_publication":"1997"},{"book_id":"203","title":"Oracle DBA
Handbook","year_of_publication":"1999"}]}
```

HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

JSON Example I – part III

```
alert("Complete JSON representation of an object describing
   the first book
   "+JSON.stringify(myJSONTestObject.productList[0]));
 }
```



```
Complete JSON representation of an object describing the first book {"book_id":"121","title":"Java in 24 hours","year_of_publication":"2002"}
```

```
</script>
</head>
<body onLoad="uploadBooks()">
….
</body>
</html>
```

The attributes of any JS object can be easily converted into its JSON text representation. The text can be submitted to the server-side script to restore the object data in the form of the matching object e.g. Java object.

HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

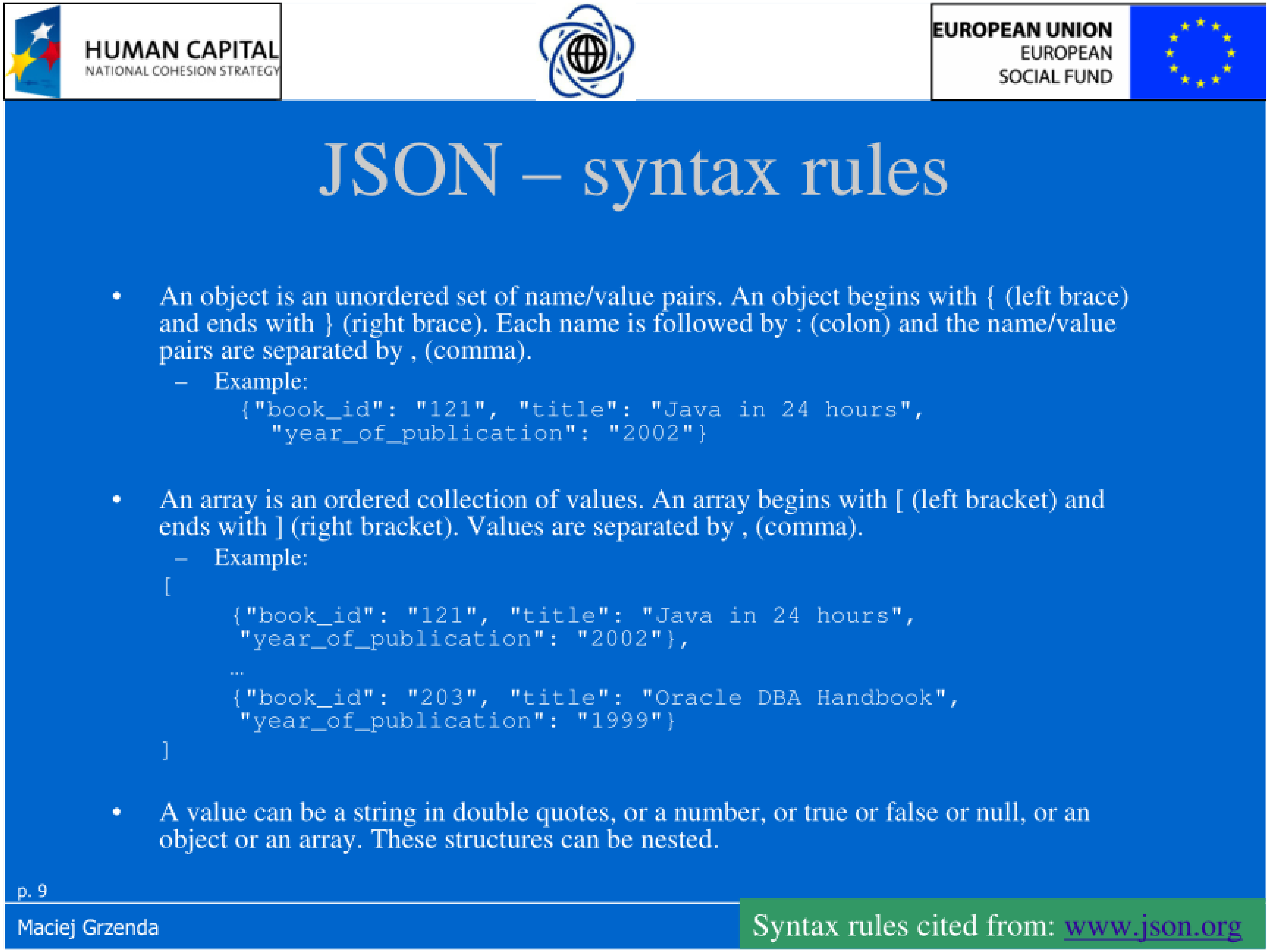
JSON – reverse operation

```
…
var myObject = eval('(' + myJSONText + ')');
alert("The title of the first book is:
   "+myObject.productList[0].title);
…
```

$\leftarrow$ Run as code

Var myObject = JSON.parse(myJSONText); $\leftarrow$ Check formatting

The JSON string contained in myJSONText variable is used to create the JavaScript object basing on its JSON text representation. The JSON text representation can be easily created by the server-side script and sent back to the browser in order to obtain a complex JavaScript object basing on it.
However, this may cause significant security problems. The eval() function may potentially call any code and inject risky code into the web site. Still JSON gives you a chance to obtain JavaScript objects without parsing the text received from the server at all.



HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

JSON – syntax rules

- An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).
  - Example:

```
{"book_id": "121", "title": "Java in 24 hours",
  "year_of_publication": "2002"}
```

- An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).
  - Example:

```
[
     {"book_id": "121", "title": "Java in 24 hours",
      "year_of_publication": "2002"},
     …
     {"book_id": "203", "title": "Oracle DBA Handbook",
      "year_of_publication": "1999"}
]
```

- A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.

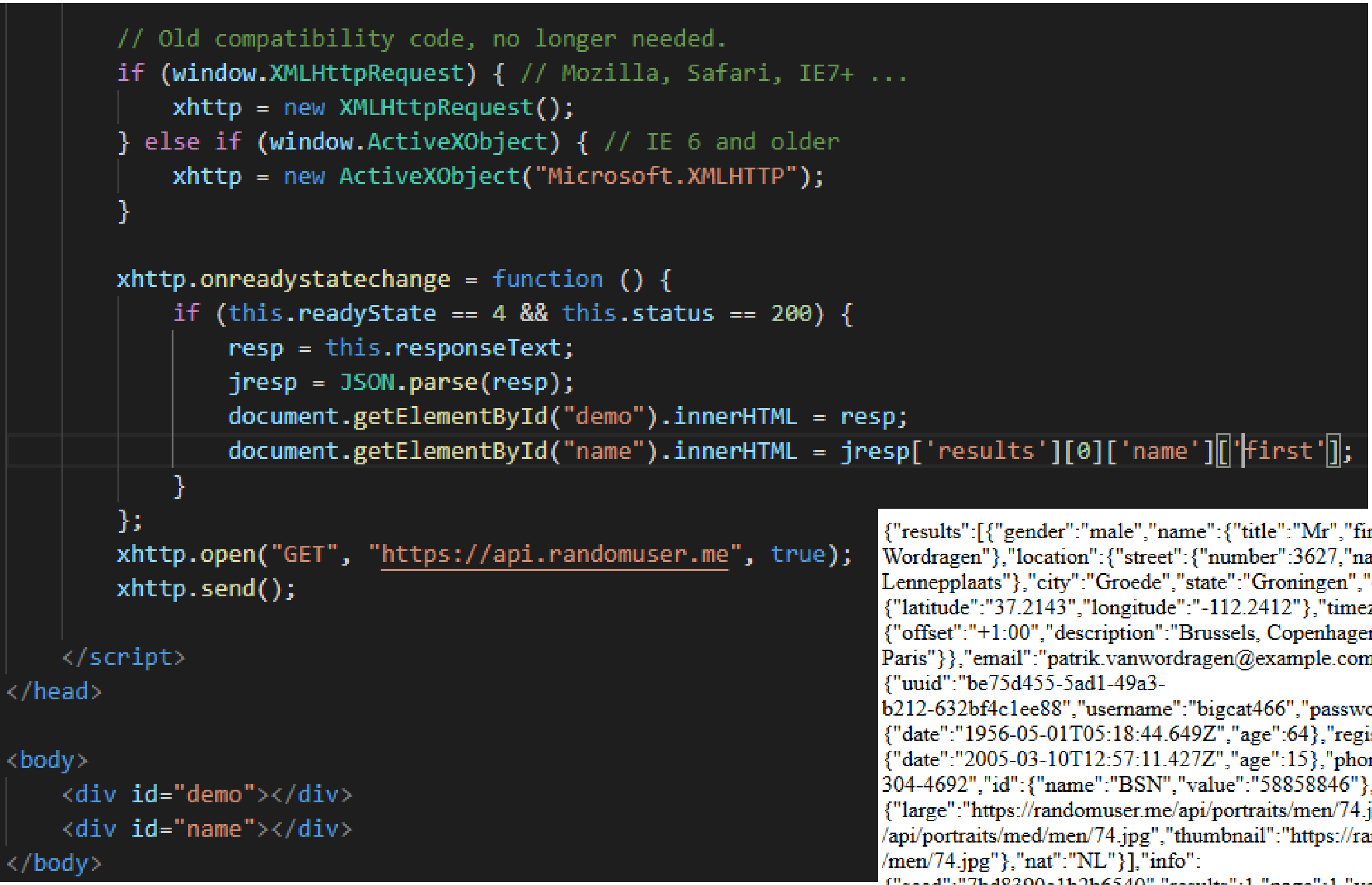
Syntax rules cited from: www.json.org

Trailing comma!!

```
JSON.parse('[1, 2, 3, 4, ]'); // SyntaxError JSON.parse: unexpected character
JSON.parse('{"foo" : 1, }');  // at line 1 column 14 of the JSON data
```

AJAX- JSON

- AJAX = Asynchronous JavaScript And XML.

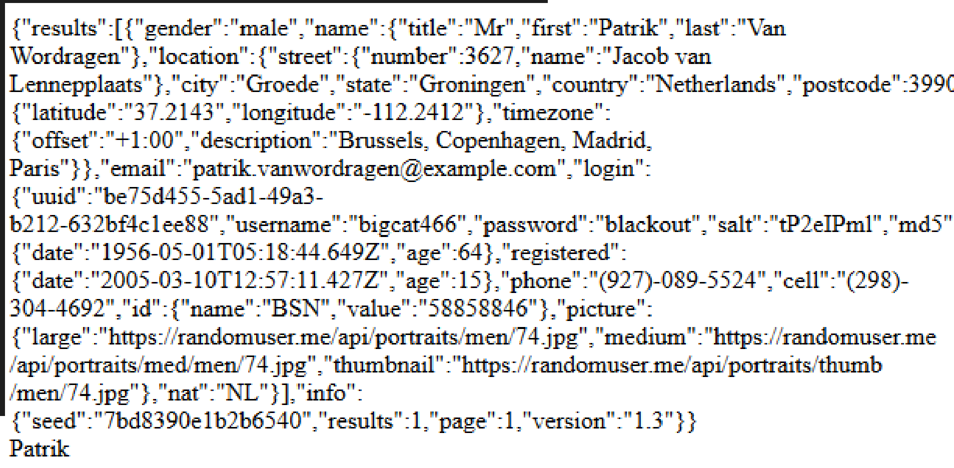


```
    // Old compatibility code, no longer needed.
    if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
        xhttp = new XMLHttpRequest();
    } else if (window.ActiveXObject) { // IE 6 and older
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }

    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            resp = this.responseText;
            jresp = JSON.parse(resp);
            document.getElementById("demo").innerHTML = resp;
            document.getElementById("name").innerHTML = jresp['results'][0]['name']['first'];
        }
    };
    xhttp.open("GET", "https://api.randomuser.me", true);
    xhttp.send();

  </script>
</head>

<body>
  <div id="demo"></div>
  <div id="name"></div>
</body>
```



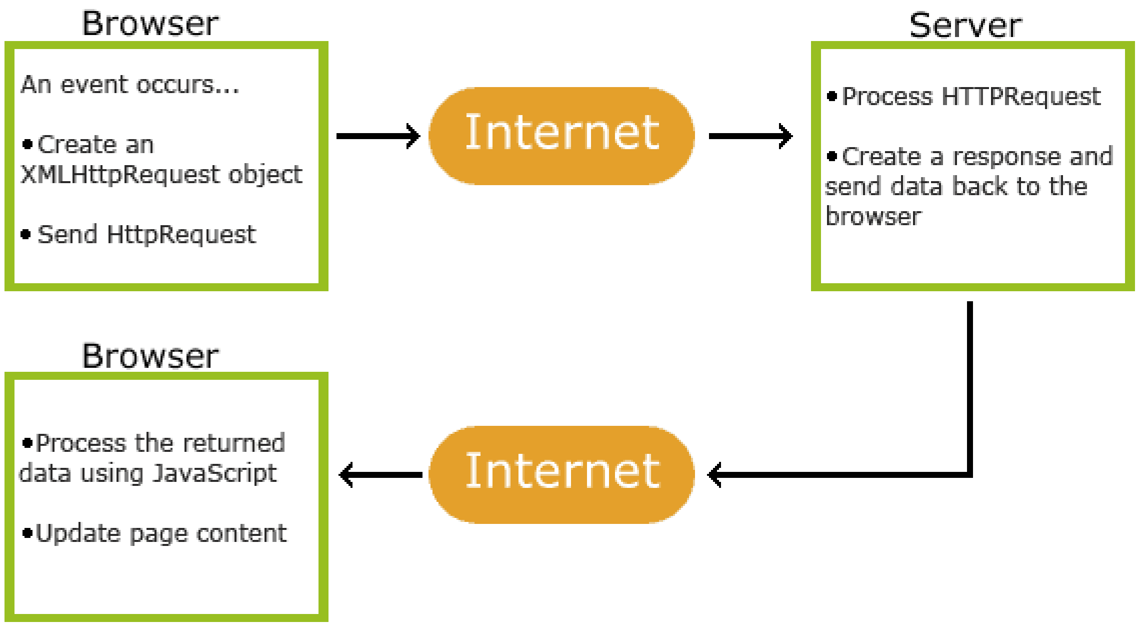
{"results":[{"gender":"male","name":{"title":"Mr","first":"Patrik","last":"Van
Wordragen"},"location":{"street":{"number":3627,"name":"Jacob van
Lennepplaats"},"city":"Groede","state":"Groningen","country":"Netherlands","postcode":3990
{"latitude":"37.2143","longitude":"-112.2412"},"timezone":
{"offset":"+1:00","description":"Brussels, Copenhagen, Madrid,
Paris"}},"email":"patrik.vanwordragen@example.com","login":
{"uuid":"be75d455-5ad1-49a3-
b212-632bf4c1ee88","username":"bigcat466","password":"blackout","salt":"tP2eIPml","md5"
{"date":"1956-05-01T05:18:44.649Z","age":64},"registered":
{"date":"2005-03-10T12:57:11.427Z","age":15},"phone":"(927)-089-5524","cell":"(298)-
304-4692","id":{"name":"BSN","value":"58858846"},"picture":
{"large":"https://randomuser.me/api/portraits/men/74.jpg","medium":"https://randomuser.me
/api/portraits/med/men/74.jpg","thumbnail":"https://randomuser.me/api/portraits/thumb
/men/74.jpg"},"nat":"NL"}],"info":
{"seed":"7bd8390e1b2b6540","results":1,"page":1,"version":"1.3"}}

Patrik

Asynchronous **J**avaScript **A**nd **X**ML

- AJAX is a technique for accessing web servers from a web page.
- approach to using a number of existing technologies together, including HTML or XHTML, CSS, JavaScript, DOM, XML, XSLT, and most importantly the XMLHttpRequest object.
- was originally created by the developers of Outlook Web Access (by Microsoft) for Microsoft Exchange Server 2000

AJAX cont.


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started#Step_1_%E2%80%93_How_to_make_an_HTTP_request
https://www.w3schools.com/js/js_ajax_intro.asp

AJAX cont.

```
var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   document.getElementById("demo").innerHTML = this.responseText;
  }
 };

 xhttp.open("GET", "ajax_info.txt", true);
 xhttp.send();          // GET
  xhttp.send(String); //POST
setRequestHeader();
```

open(method, url, async, user, psw)

Specifies the request
method: the request type GET or POST
url: the file location
async: true (asynchronous) or false synchronous)
user: optional user name
psw: optional password

readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready

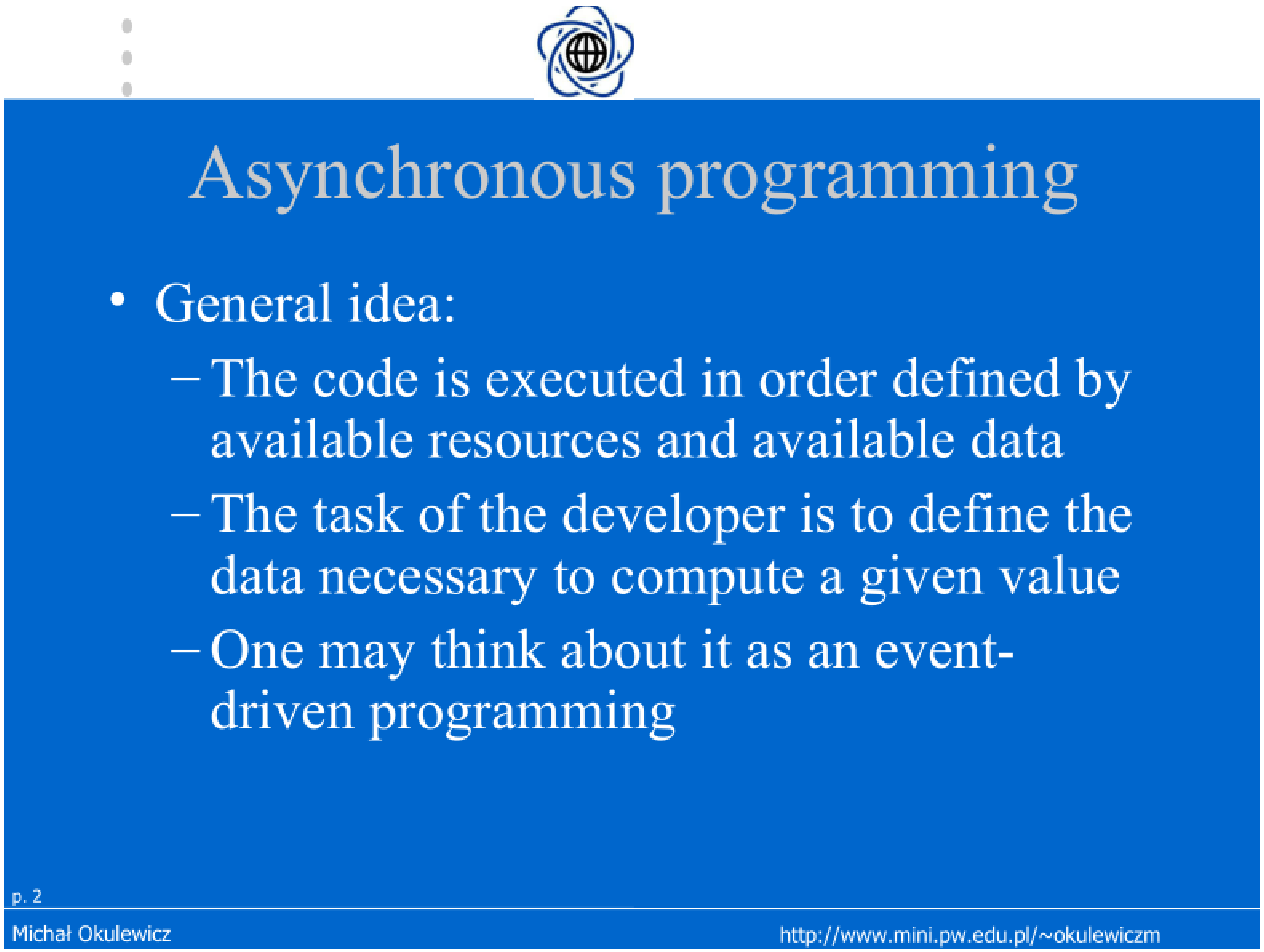
AJAX-ready server scripts

- In order to generate proper responses to AJAX calls server script should generate only a part of the website
- It may also generate a more general XML or JSON content instead of an HTML matching only one website
- While XML has been initially used as a response for an AJAX call it is better to generate a part of HTML or a JSON object

Fetch API

- Fetch provides a generic definition of Request and Response objects (and other things involved with network requests)
- XMLHttpRequest alternative that can be easily used by other technologies such as Service Workers
- provides a single logical place to define other HTTP-related concepts such as CORS and extensions to HTTP
- **returns a <u>promise</u> containing the response** (a Response object).

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

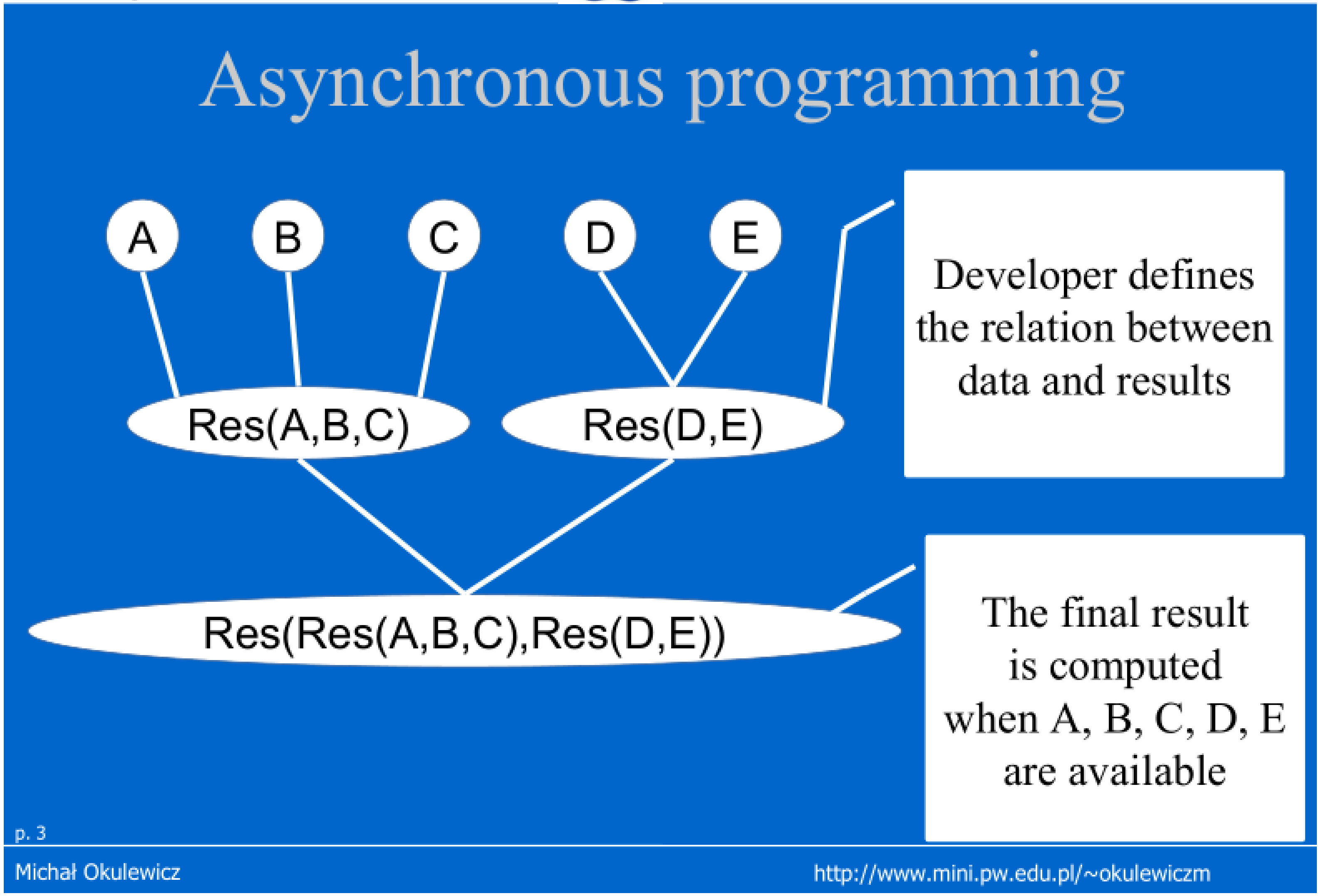


Asynchronous programming

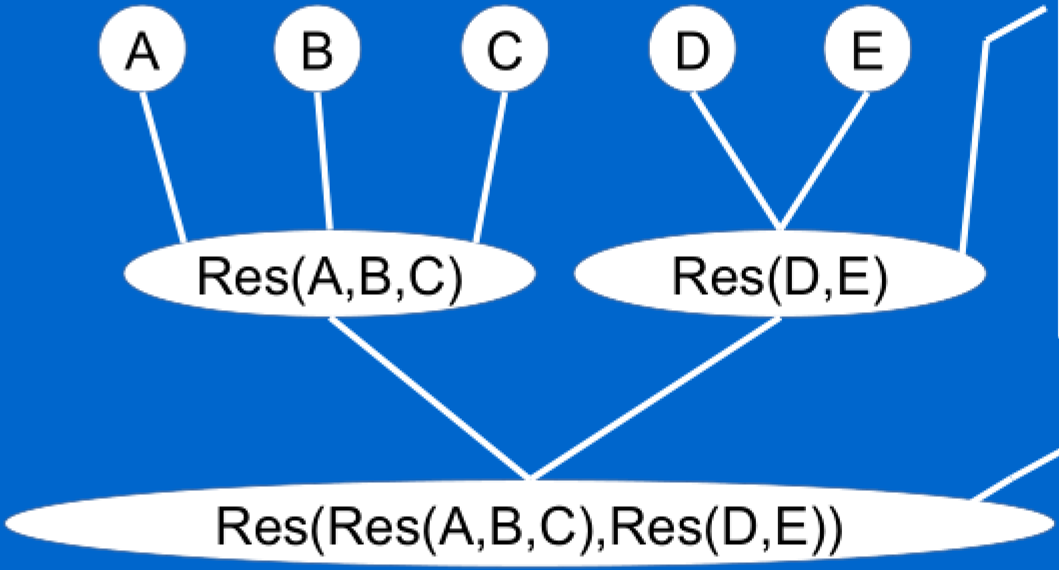
- General idea:
  - The code is executed in order defined by available resources and available data
  - The task of the developer is to define the data necessary to compute a given value
  - One may think about it as an event-driven programming

p. 2

Michał Okulewicz http://www.mini.pw.edu.pl/~okulewiczm

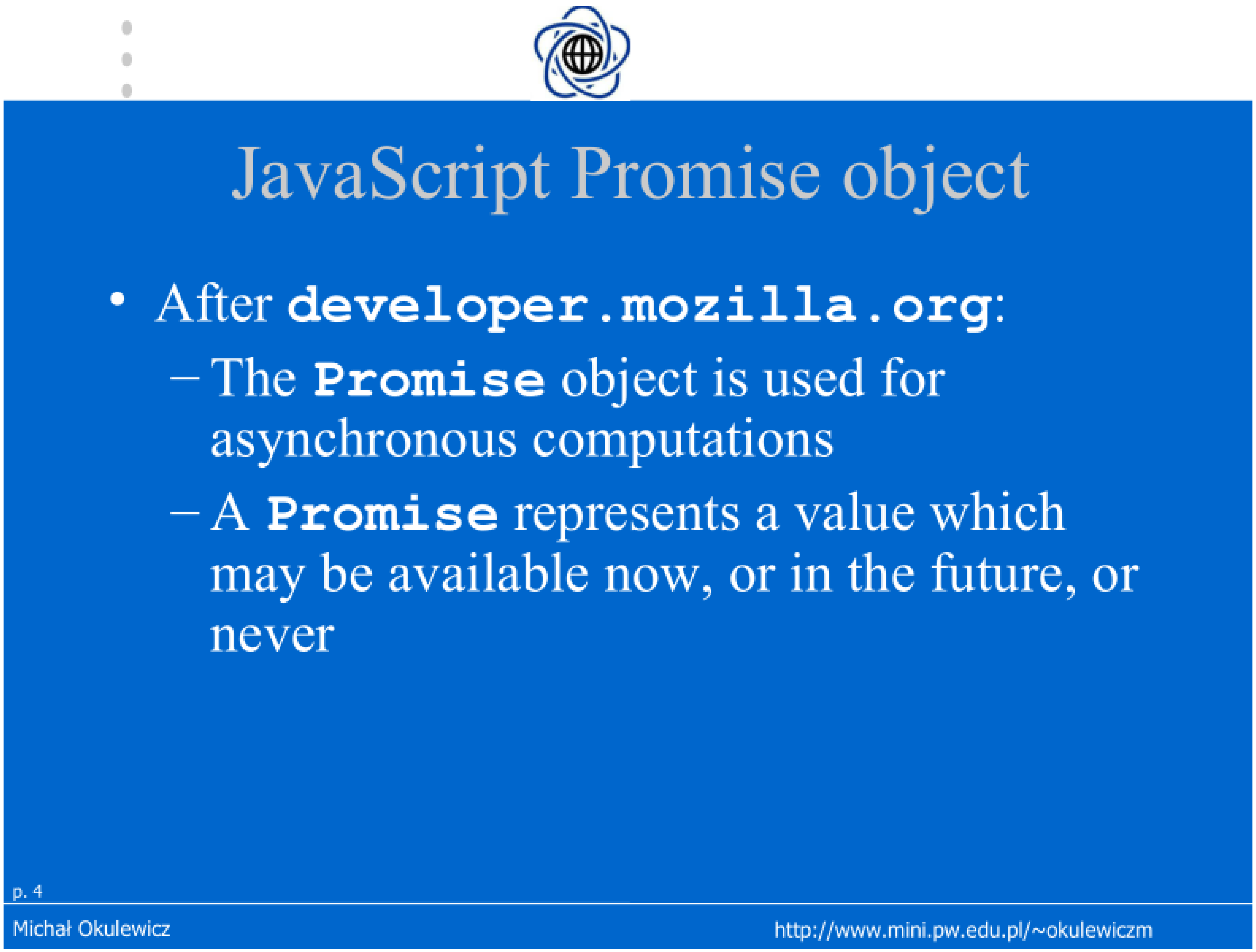


Asynchronous programming



Developer defines the relation between data and results

The final result is computed when A, B, C, D, E are available



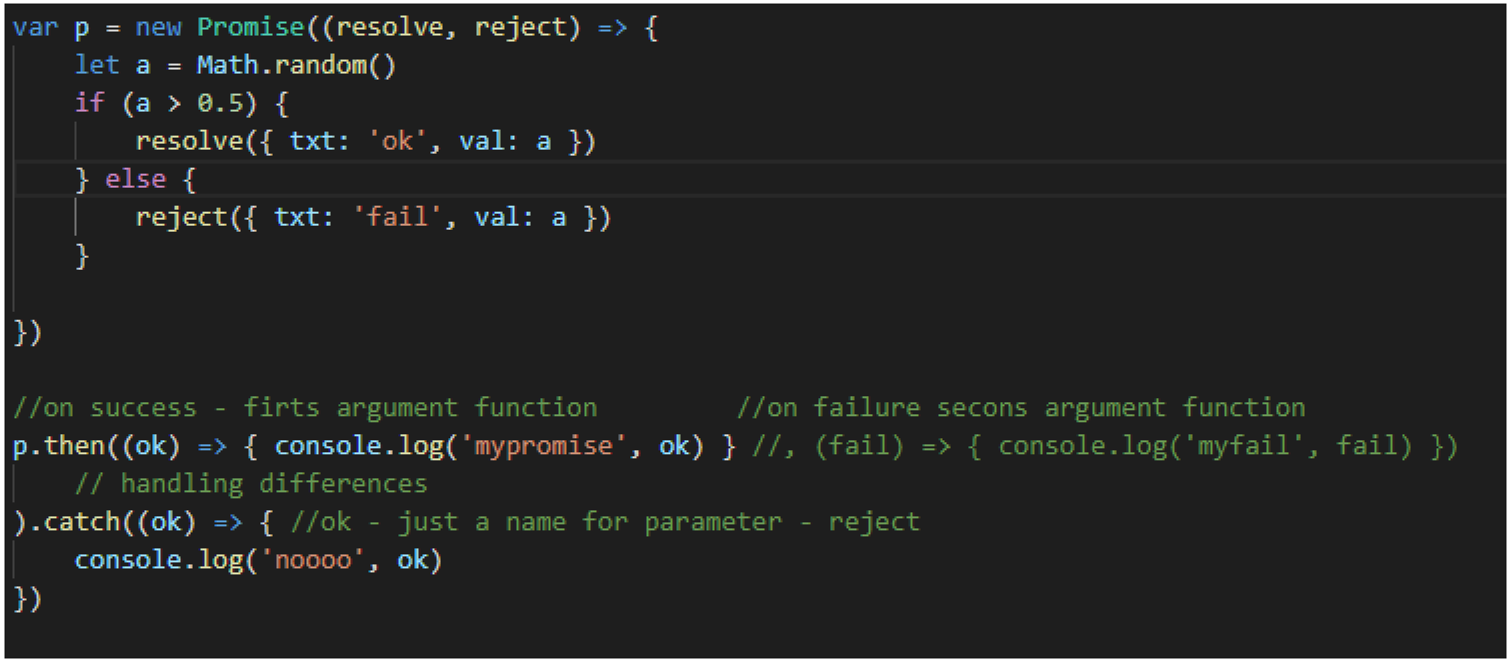
JavaScript Promise object

- After **`developer.mozilla.org`**:
  - The **`Promise`** object is used for asynchronous computations
  - A **`Promise`** represents a value which may be available now, or in the future, or never

p. 4

Michał Okulewicz http://www.mini.pw.edu.pl/~okulewiczm

Promise example



```javascript
var p = new Promise((resolve, reject) => {
    let a = Math.random()
    if (a > 0.5) {
        resolve({ txt: 'ok', val: a })
    } else {
        reject({ txt: 'fail', val: a })
    }

})

//on success - firts argument function         //on failure secons argument function
p.then((ok) => { console.log('mypromise', ok) } //, (fail) => { console.log('myfail', fail) })
    // handling differences
).catch((ok) => { //ok - just a name for parameter - reject
    console.log('noooo', ok)
})
```

Fetch API

global fetch() method

The fetch() method can optionally accept a second parameter, an init object that allows you to control a number of different settings

```
fetch('http://example.com/movies.json')
.then(response => response.json())
  .then(data => console.log(data));
```

Return promise, transform body with JSON content to object

When sucess

Promise – transforming method

A Promise is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with an asynchronous action's eventual success value or failure reason.
This lets asynchronous methods return values like synchronous methods: instead of immediately returning the final value, the asynchronous method returns a promise to supply the value at some point in the future.
A Promise is in one of these states:

pending:	initial state, neither fulfilled nor rejected.
fulfilled:	meaning that the operation was completed successfully.
rejected:	meaning that the operation failed.

```
async function postData(url = '', data = {}) {
  // Default options are marked with *
  const response = await fetch(url, {                        //init object
    method: 'POST', // *GET, POST, PUT, DELETE, etc.
    mode: 'cors', // no-cors, *cors, same-origin
    cache: 'no-cache', // *default, no-cache, reload, force-cache, only-if-cached
    credentials: 'same-origin', // include, *same-origin, omit
    headers: {
      'Content-Type': 'application/json'   // 'Content-Type': 'application/x-www-form-urlencoded',
    },
    redirect: 'follow', // manual, *follow, error
    referrerPolicy: 'no-referrer',
// no-referrer, *no-referrer-when-downgrade, origin, origin-when-cross-origin, same-origin, strict-
origin, strict-origin-when-cross-origin, unsafe-url
    body: JSON.stringify(data) // body data type must match "Content-Type" header
  });
  return response.json(); // parses JSON response into native JavaScript objects
}

postData('https://example.com/answer', { answer: 42 })
  .then(data => {
    console.log(data); // JSON data parsed by `data.json()` call
  });
```

Promise example

```javascript
var promise = new Promise(function (resolve, reject) {
    var result = Math.random();
    setTimeout(function () {
        if (result < 0.8) {
            resolve(result);
        } else {
            reject("No luck!");
        }
    }, 1000);
});
promise
    .then(function (val) {
        console.log('Promise fulfilled with ' + val);
    })
    .catch(function (msg) {
        console.error('Promise failed because: ' + msg);
    });
```

TypeScript

- can compile pure JavaScript Rename .js to .ts – use tsc
- static type checking vs dynamic type checking
- Structural Typing Nominal Typing vs Structural Typing vs Duck Typing
- TypeScript is OOP-first, and mostly is C# for the browser (C# author involved)
- Transpiler (source-to-source translator)
- Server side (node.js) advantages

- Some limitations present

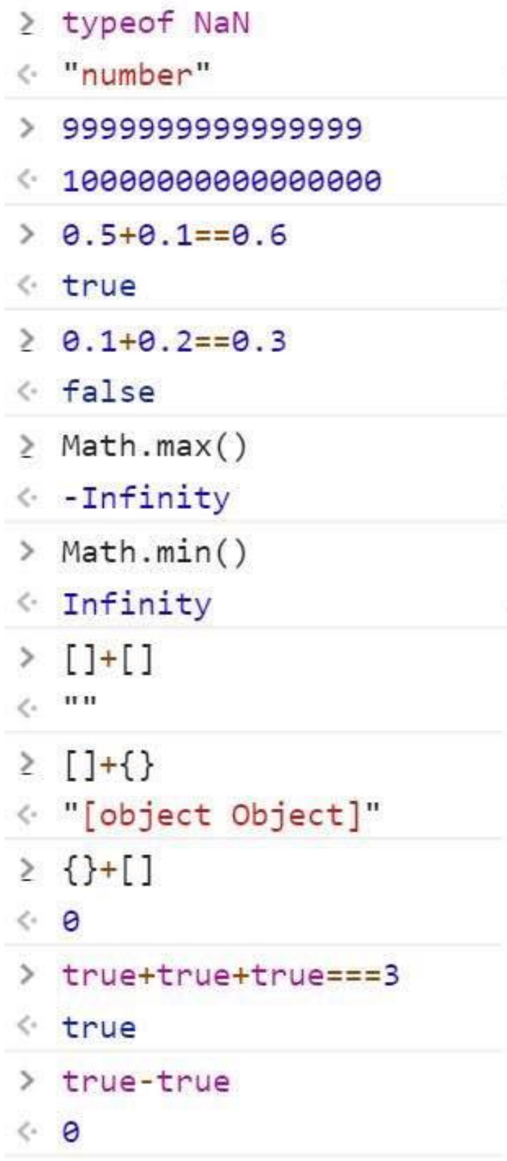
TypeScript cont.

- let myName: **string** = "Alice";

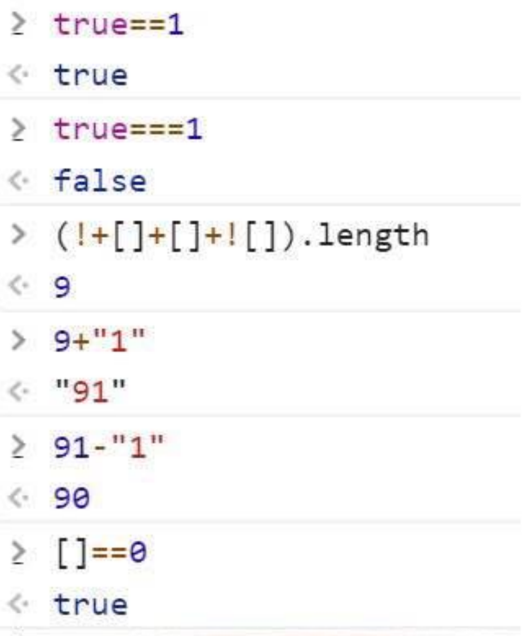
- function printCoord(pt: **{ x: number; y: number }**) {

```
console.log("The coordinate's x value is " + pt.x);
console.log("The coordinate's y value is " + pt.y);
}
printCoord({ x: 3, y: 7 });
```

https://www.typescriptlang.org/

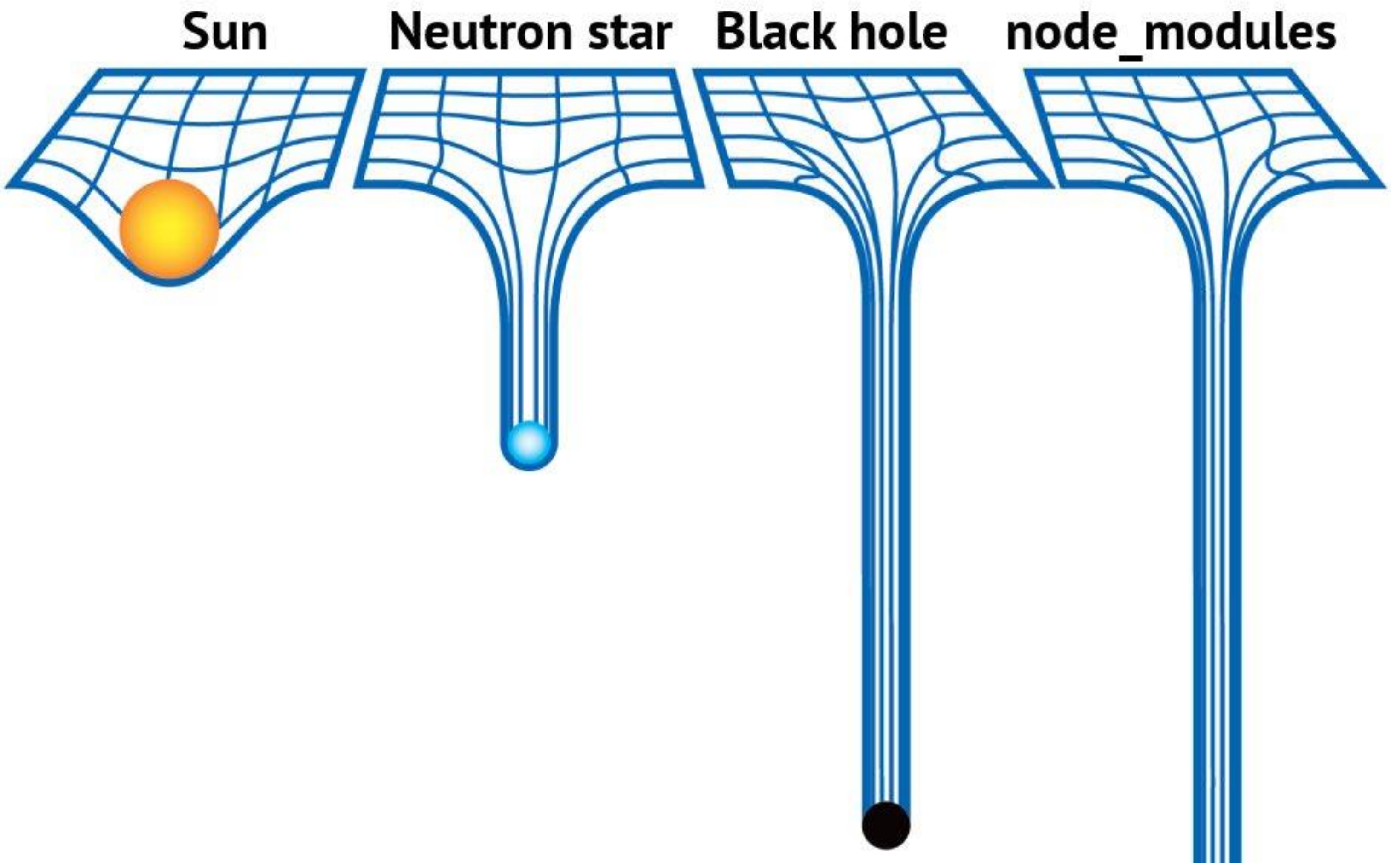


```
> typeof NaN
<· "number"
> 9999999999999999
<· 10000000000000000
> 0.5+0.1==0.6
<· true
> 0.1+0.2==0.3
<· false
> Math.max()
<· -Infinity
> Math.min()
<· Infinity
> []+[]
<· ""
> []+{}
<· "[object Object]"
> {}+[]
<· 0
> true+true+true===3
<· true
> true-true
<· 0
```



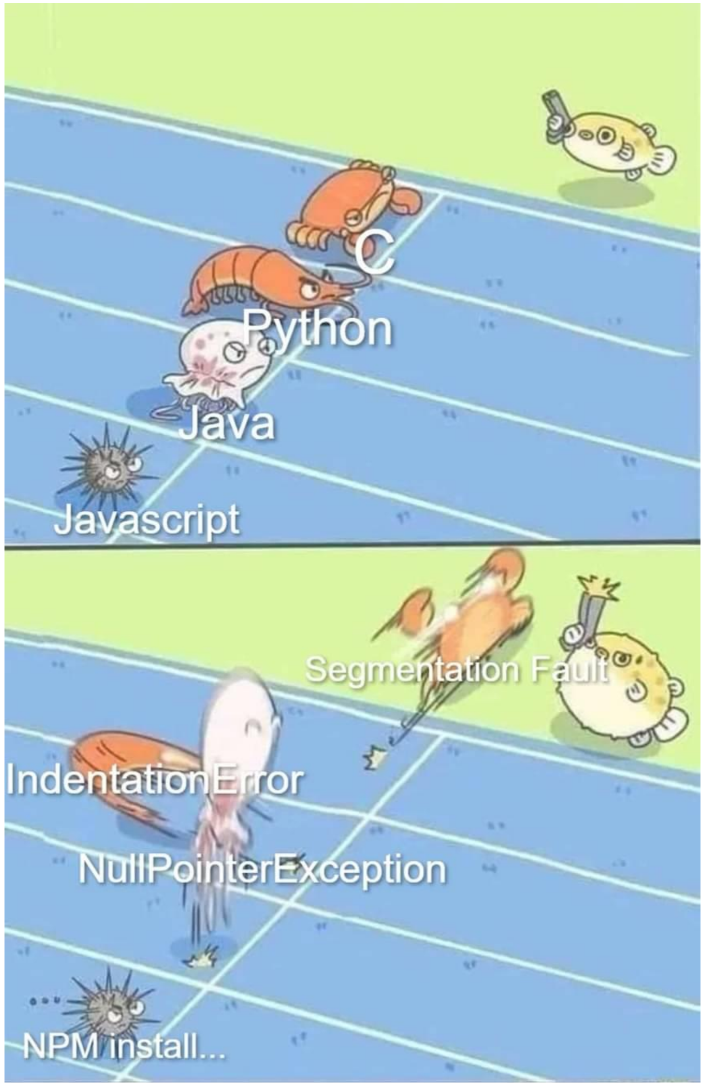
```
> true==1
<· true
> true===1
<· false
> (!+[]+[]+![]).length
<· 9
> 9+"1"
<· "91"
> 91-"1"
<· 90
> []==0
<· true
```

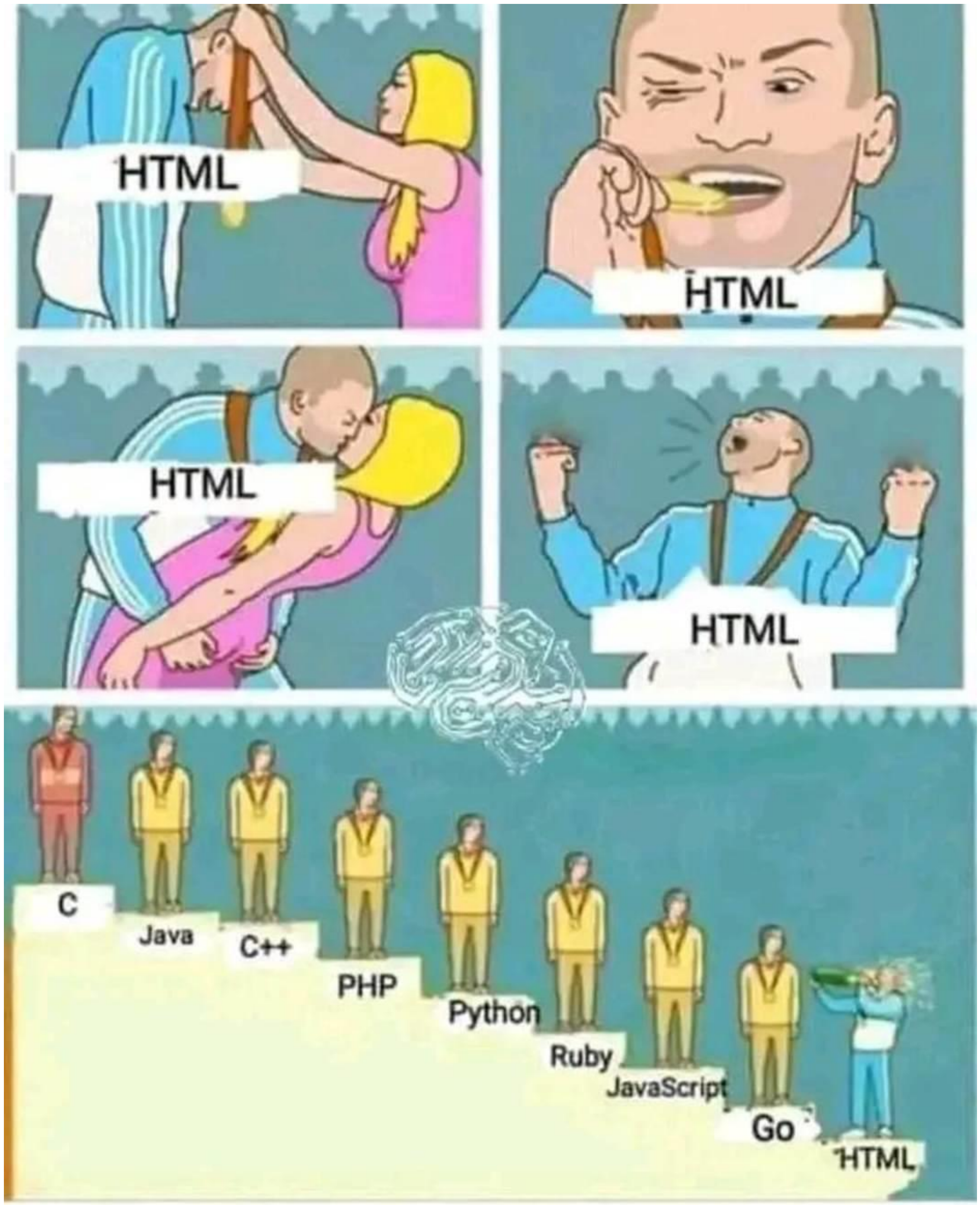
YOU COME TO ME AT RUNTIME
TO TELL ME THE CODE YOU ARE EXECUTING DOES NOT COMPILE
memegenerator.net


Sun
Neutron star
Black hole
node_modules

Full Stack Developer
Let's see who this really is
yuva.krishna.memes
Backend developer, who also can develop an avg front end with help of google..

FULL STACK DEVELOPER
FRONTEND DEVELOPER WHO CAN WRITE SQL WITH HELP FROM GOOGLE
imgflip.com


C
Python
Java
Javascript
Segmentation Fault
IndentationError
NullPointerException
NPM install...


HTML
HTML
HTML
HTML
C
Java
C++
PHP
Python
Ruby
JavaScript
Go
HTML