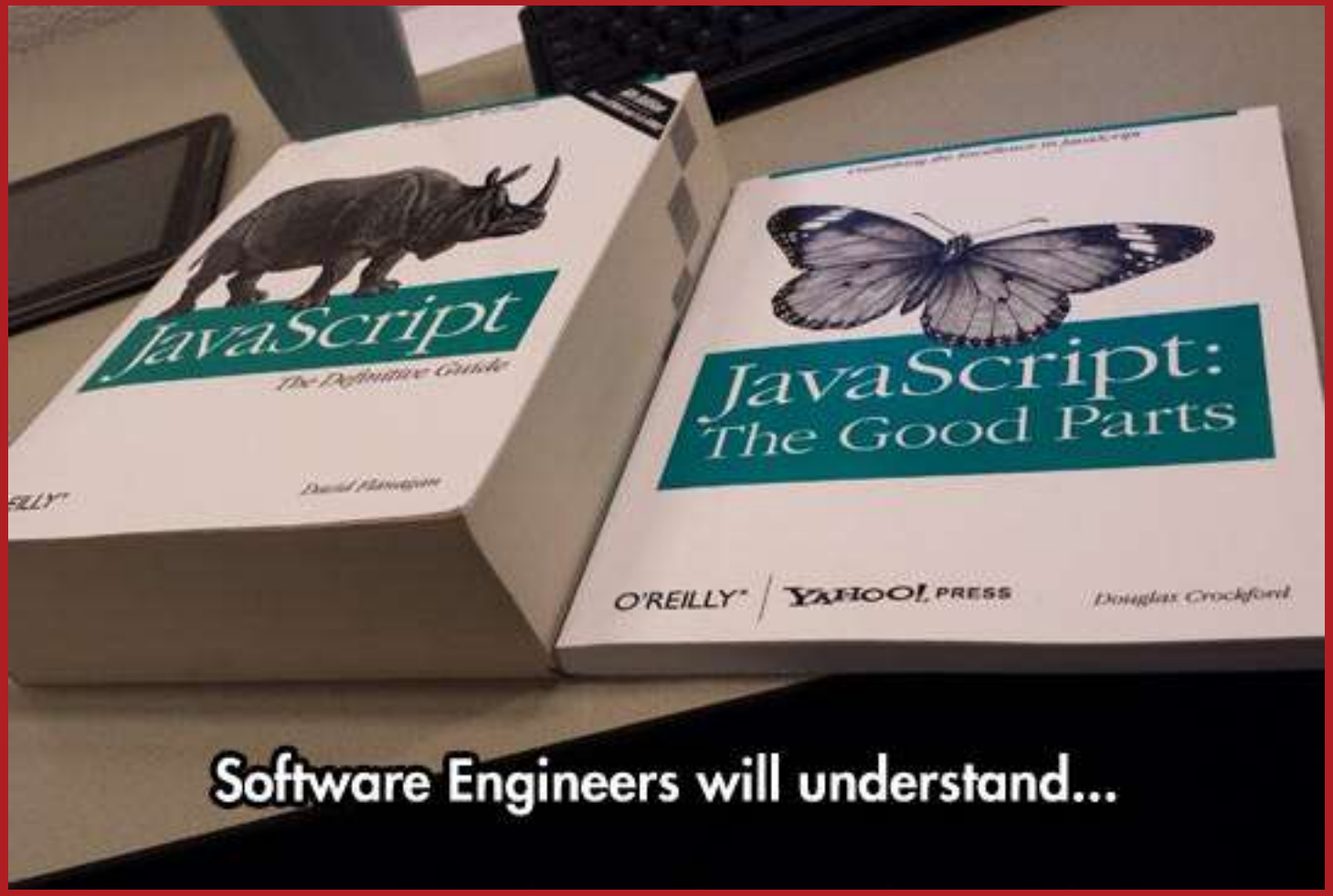
JAVASCRIPT PODSTAWY
PART I
10.06.2017

Hej Fronty!

Nazywam się Tomasz Nastały

Pracuję jako JS Developer w firmie Goyello. Rozwijam aplikację dla linii lotniczych w frameworku Angular v.4

Jeśli ktoś się interesuje Angularem, zapraszam na mój blog: www.angular.love


JavaScript
The Definitive Guide
JavaScript:
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford
Software Engineers will understand...

1.
JavaScript - historia

JavaScript - historia

- Język stworzony w 1995 roku przez Brendana Eicha
- JS został napisany w 10dni
- Gdzieś do 2011 roku nie cieszył się dobrą opinią
- Przeszedł przez 20 lat ogromną ewolucję, obecnie możemy w nim również pisać backend za pomocą nodeJS
- Używany głównie do obsługi zachowania stron internetowych i aplikacji webowych
- Każda przeglądarka ma swój silnik potrafiący obsłużyć kod JS

JavaScript - historia

Year	Name	Description
1997	ECMAScript 1	First Edition.
1998	ECMAScript 2	Editorial changes only.
1999	ECMAScript 3	Added Regular Expressions. Added try/catch.
	ECMAScript 4	Was never released.
2009	ECMAScript 5	Added "strict mode". Added JSON support.
2011	ECMAScript 5.1	Editorial changes.
2015	ECMAScript 6	Added classes and modules.
2016	ECMAScript 7	Added exponential operator (**). Added Array.prototype.includes.

2.
JavaScript - zastosowanie

JavaScript - zastosowanie

- Interaktywne strony www
- Backend, tworzenie REST API (framework nodeJS)
- Skalowane aplikacje webowe (framework Angular / React)
- Aplikacje desktopowe (framework Electron, znane apki napisane w JS to np. Spotify, Slack)
- Gry (HTML5 + JS / jQuery)

3.
JavaScript - cechy

JavaScript - definicja

- Język programowania obsługiwany głównie przez przeglądarki
- Wspiera programowanie funkcyjne jak i obiektowe (i spaghetti :)
- Często używany w kombinacji z CSS i HTML
- Wysoko-poziomowy
- Dynamiczny
- Nietypowany
- Interpretowany podczas run-time'u (nie wymaga kompilacji)

4.
JavaScript – co mamy do dyspozycji

JavaScript - API

- JavaScript zawiera ogromny zbiór gotowych metod i obiektów

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

Najbardziej popularne:
- Metody do operacji na stringach, liczbach, obiektach, tablicach
- Metody do operacji na drzewie DOM
- Metody do obsługi eventów przeglądarki (click, mouseover, scroll)
- XHR (obsługa zapytań HTTP)

5.
JavaScript – zmienne

ZMIENNA

Zmienna – kontener do przetrzymywania danych, np. wartości liczbowych lub tekstowych

Deklarowanie zmiennych

```
var firstName = "John";
var lastName = 'Doe';
var age = 23;
var isAvailable = true;

var firstName = "John",
    lastName = 'Doe',
    age = 23,
    isAvailable = true;
```

Zmienne – typy prymitywne (primitive data)

```
var firstName = "John"; // string
var age = 23; // number
var isAvailable = true; // boolean
var salary = null; // null
var job; // undefined
```

Zmienne – typy złożone (complex data)

```
var address = {
  street: 'Jana Pawła II',
  city: 'Gdańsk'
}; // object

var skills = ['HTML5', 'JavaScript', 'CSS']; // object

var getFullName = function () {
  return firstName + surName;
}; // function
```

Zmienne – brak słówka var przy deklaracji

```
name = "John"; // propercja obiektu window, pseudo
globalna, możne zostać usunięta

var name = "John"; // zmienna lokalna lub globalna
```

WNIOSEK: Nigdy nie zapominaj o słówku var, deklarując zmienną

Zmienne - nazewnictwo

- Nazywy mogą posiadać cyfry, litery, podkreślniki lub znak dolara
- Nazwy muszą zaczynać się od litery lub podkreślnika
- Nazwy są caseSensitive
- Słowa zarezerwowane nie mogą być użyte jako nazwy
- Uzywamy camelCase
- Wyłącznie język angielski

„There are only two hard things in Computer Science: cache invalidation and naming things.”
-- Phil Karlton

Sprawdzenie typu zmiennej – metoda typeof()

```
var name = "Janusz";
typeof(name); // string
typeof name; // string
typeof(5); // number
typeof 4; // number
```

Zarezerwowane słowa kluczowe

Keywords

Reserved keywords as of ECMAScript 2015

- `break`
- `case`
- `catch`
- `class`
- `const`
- `continue`
- `debugger`
- `default`
- `delete`
- `do`
- `else`
- `export`
- `extends`
- `finally`
- `for`
- `function`
- `if`
- `import`
- `in`
- `instanceof`
- `new`
- `return`
- `super`
- `switch`
- `this`
- `throw`
- `try`
- `typeof`
- `var`
- `void`
- `while`
- `with`
- `yield`

ZMIENNE - DOBRE PRAKTYKI

NAZWA MA PRZEKAZYWAĆ INTENCJE

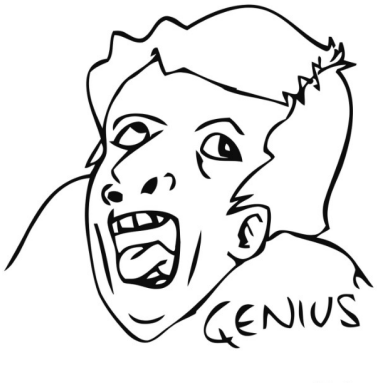
BAD CODE

```
var d;
var ds;
var dsm;
var faid;
```

CLEAN CODE

```
var elapsedTimeInDays;
var daysSinceCreation;
var daysSinceModification;
var fileAgeInDays;
```

TAK W PRACY WIDZĄ DEVELOPERA , KTÓRY TAK ROBI:



ODPOWIEDNIA DŁUGOŚĆ NAZW

BAD CODE

```
var theCustormsListWithAllCustomsIncludedWithoutFilter;
```

CLEAN CODE

```
var customers;
```

UNIKAJ DEZINFORMACJI

BAD CODE

```
var name = ["Tomek", "Janek"];
```

CLEAN CODE

```
var names = ["Tomek", "Janek"];
```

TRZYMAJ SIĘ JEDNEJ NOTACJI

BAD CODE

```
var namesLength;
var itemsLen;
var productsSize;
```

CLEAN CODE

```
var namesLength;
var itemsLength;
var productdLength;
```

DOMYKAJ KONTEKST

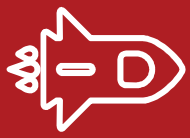
BAD CODE

```
var addressCity = "...";
var addressHomeNumber = "...";
var addressPostCode = "...";
```

CLEAN CODE

```
var address = {
    city: "...",
    homeNumber: "...",
    postCode: "..."
};
```

ZADANIA



6.
JavaScript – Funkcje

Funkcja

- Funkcja to blok kodu zaplanowany do wykonania konkretnego zadania, np. zsumowania dwóch liczb
- Funkcja zostanie wykonana jeśli ją wywołamy
- Funkcja może posiadać parametry

Funkcje - deklarowanie

Function declaration

```
function sum(a, b) {
    return a + b;
}
```

Function expression

```
var sum = function (a, b) {
    return a + b;
};
```

Invocation (wywołanie)

```
sum(3, 4); // OUTPUT: 7
```

Function Declaration vs Function Expression

Function declaration

```
sum(3, 4); // OUTPUT: 7

function sum(a, b) {
    return a + b;
}

sum(3, 4); // OUTPUT: 7
```

Function expression

```
sum(3, 4); // ERROR - sum is not
defined

var sum = function (a, b) {
    return a + b;
};

sum(3, 4); // OUTPUT: 7
```

Preferujemy używanie function declaration, bo na dowolnej wysokości kodu możemy sięgać po taką funkcję

Scope funkcji

Zmienne zadeklarowane w środku funkcji nie są widoczne poza funkcją, ale funkcja ma dostęp do zmiennych zadeklarowanych globalnie

```
console.log(VAT); // ERROR: VAT is not defined

function getGross(nettoPrice) {
    var VAT = 1.23;
    console.log(VAT); // 1.23
    return nettoPrice * VAT;
}
```

Domknięcia - closures

- Domknięcie to zasięg (obszar) tworzony przez funkcję, która ma dostęp do:
- Swoich zmiennych
- Zmiennych zadeklarowanych we funkcji, w której jest zagnieżdżona
- Do zmiennych globalnych
- Ma dostęp nie tylko do zmiennych funkcji nadrzędnej, ale również do jej parametrów

- Poprzez domknięcie funkcja tworzy „swoje środowisko”

Domknięcia - closures

```
function showName(firstName, lastName) {
    var nameIntro = "Your name is ";

    function makeFullName() {
        return nameIntro + firstName + " " + lastName;
    }

    return makeFullName();
}
showName("Michael", "Jackson"); // Your name is Michael Jackson
```

Wniosek – zagnieżdżona funkcja makeFullName widzi zmienne wyżej zadeklarowane i parametry funkcji nadrzędnej

Parametry

- Funkcja może przyjmować dowolnie wiele parametrów, dowolnego typu (string, numer etc)
- Funkcja nie waliduje typów przyjętych parametrów (w przeciwieństwie do Javy np., gdzie deklaruje się typy parametrów)
- Funkcja nie sprawdza ilości przyjętych parametrów

```
function sum(a, b) {
    return a + b;
}

sum("Hey", 5); // OUTPUT:
"Hey5"
```

```
function sum(a, b) {
    return a + b;
}

sum(3, 4, "Hey!"); // OUTPUT: 7;
```

Return we funkcji

- Return zawsze kończy działanie funkcji
- Cokolwiek wrzucisz za returnem, się nie wykona
- Funkcja nie musi posiadać returna

```
function sum(a, b) {
    return a + b;
    console.log(„Hey!”); // UNREACHABLE CODE
}
```

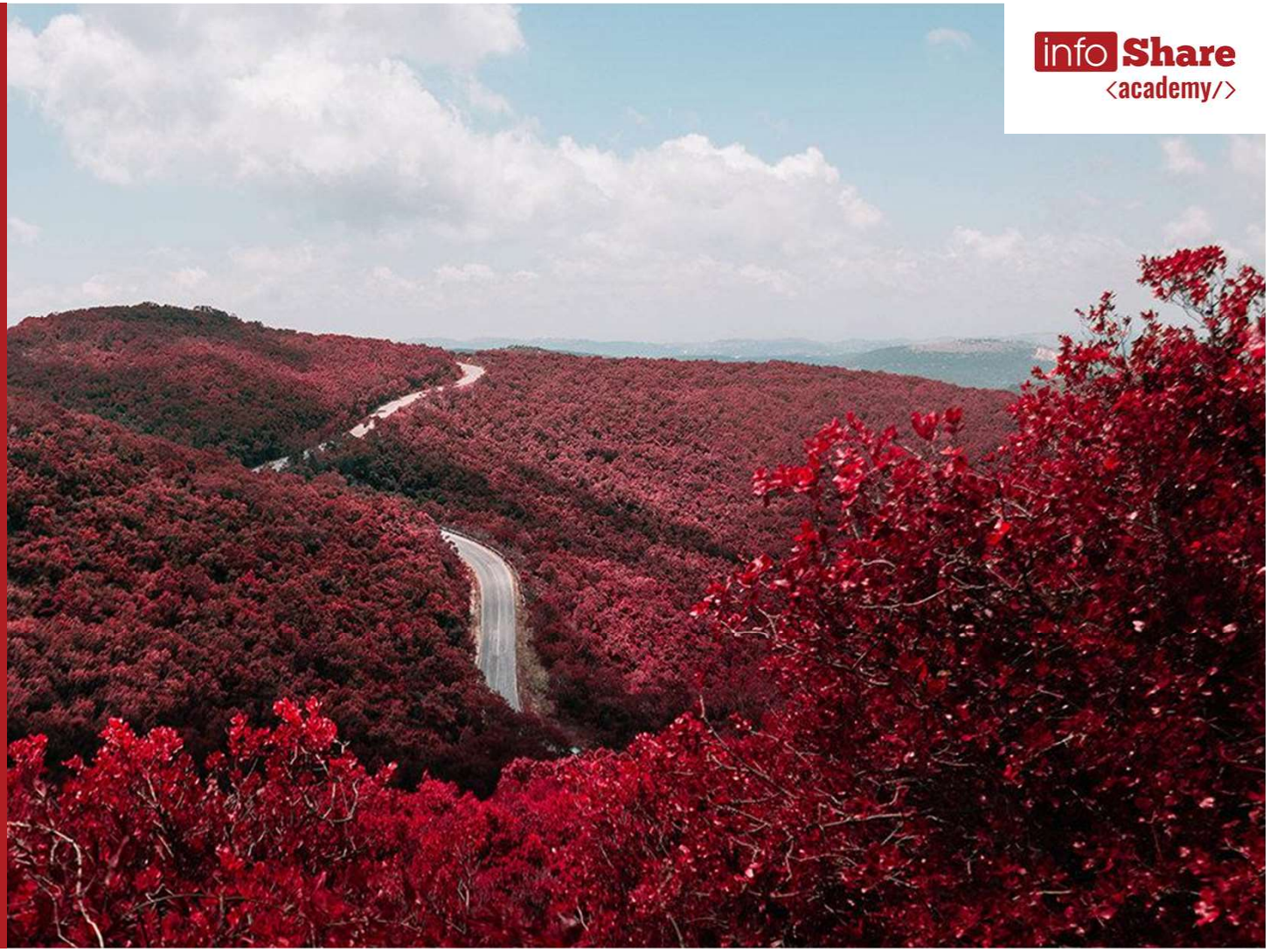
Immediately-Invoked Function Expression (IIFE)

- Dzięki użyciu IIFE nie zaśmiecamy globalnej przestrzeni nazw

```
(function() {
…
})();
```

```
(function() {
    var a = b = 5;    // NIGDY TAK NIE RÓB!
})();
console.log(a);
console.log(b);
```

JAVASCRIPT BUILT IN FUNCTIONS



infoShare <academy/>

Built-in Functions

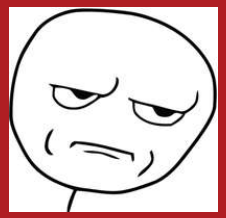
Oprócz funkcji, które możemy tworzyć sami, Javascript udostępnia nam ogromny zestaw wbudowanych funkcji

Najbardziej popularne:

- **Math**: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects/Math
- **Number**: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects/Number
- **String**: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects/String
- **Array**: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects/Array

FUNKCJE - DOBRE PRAKTYKI

SINGLE RESPONSIBILITY
JEDNA FUNKCJA WYKONUJE JEDNA RZECZ
(ŁATWE TESTOWANIE KODU, CZYTELNOŚĆ)



BAD CODE

```
var name = "Janusz";
var city;

function getUpperCasedName() {
    if (name === "Janusz") {
        city = "Sosnowiec"
    }

    return name.toUpperCase();
}
```

infoShare academy

CLEAN CODE

```
var name = "Janusz";
var city;

function getUpperCasedName() {
    return name.toUpperCase();
}

function setCity() {
    if (name === "Janusz") {
        city = "Sosnowiec"
    }
}

getUpperCasedName();
setCity();
```

NAZWY FUNKCJI NAJCZĘŚCIEJ POŁĄCZONE Z CZASOWNIKIEM

BAD CODE

```
function data() {}
function name() {}
function disabled() {}
function nameFromFullName() {}
```

CLEAN CODE

```
function loadData() {}
function fetchData() {}
function getName() {}
function setName() {}
function isDisabled() {}
function extractNameFromFullName() {}
```

NIE RÓB FUNKCJI Z DUŻA ILOŚCIA PARAMETRÓW – ZAMIAST TEGO PRZEKAŻ JEDEN JAKO OBIEKT

infoShare academy

BAD CODE

```
function getTotalSpeed(powerHorses, avgSpeed, maxSpeed, minSpeed, weight, wheels) {
    var factorOne = powerHorses * avgSpeed;
    var factorTwo = maxSpeed / minSpeed;
    var factorThree = weight + wheels;

    return factorOne + factorTwo + factorThree;
}
```

CLEAN CODE

```
var carConfig = {
    powerHorses: "...",
    avgSpeed: "...",
    maxSpeed: "...",
    minSpeed: "...",
    weight: "...",
    wheels: "..."
};
```

```
function getTotalSpeed(carConfig) {
    var factorOne = carConfig.powerHorses * carConfig.avgSpeed;
    var factorTwo = carConfig.maxSpeed / carConfig.minSpeed;
    var factorThree = carConfig.weight + carConfig.wheels;

    return factorOne + factorTwo + factorThree;
}
```

NIE DUPLIKUJ FUNKCJI O TEJ SAMEJ IMPLEMENTACJI – SPARAMETRYZUJ

BAD CODE

```
var city = "Sosnowiec";
var name = "Janusz";
var surName = "Kowalski";

function getLowerCasedCity() {
    return city.toLowerCase();
}

function getLowerCasedName() {
    return name.toLowerCase();
}

function getLowerCasedSurName() {
    return surName.toLowerCase();
}
```

CLEAN CODE

```
var city = "Sosnowiec";
var name = "Janusz";
var surName = "Kowalski";

function getLowerCased(str) {
    return str.toLowerCase();
}
```

UNIKAJ MAGIC NUMBERS

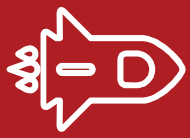
BAD CODE

```
function getGross(nettoPrice) {
    return nettoPrice * 1.23;
}
```

CLEAN CODE

```
function getGross(nettoPrice) {
    var VAT = 1.23;
    return nettoPrice * VAT;
}
```

ZADANIA



7.
JavaScript – Operatory matematyczne

Operatory matematyczne

Podstawowe operatory matematyczne to:

+ (plus służy również do konkatenacji stringów)

- (odejmowanie)

* (mnożenie)

/ (dzielenie)

% (reszta z dzielenia, pomocna przy sprawdzaniu parzystości liczb)

++ (inkrementacja, najczęściej używana w pętlach)

-- (dekrementacja, najczęściej używana w pętlach)

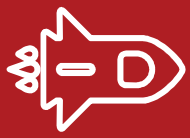
Inkrementacja: ++x vs x++

x++ najpierw egzekucja potem inkrementacja

++x najpierw inkrementacja potem egzekucja

```
var x = 1;
var y = x++; // y = 1, x = 2
var z = ++x; // z = 3, x = 3
```

ZADANIA



8.
JavaScript – Obiekt Date

Obiekt Date

Nowy obiekt Date tworzymy poprzez wywołanie new Date():

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global Objects/Date

```
var now = new Date();
console.log(now.getDay())
;
```

```
new Date("1995,11,17");
```

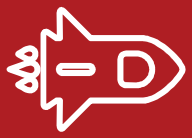
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Biblioteka MomentJS

Najczęściej w aplikacjach używa się biblioteki momentJS do obsługi dat

https://momentjs.com/

ZADANIA



9.
Odczytywanie wartości obiektów i tablic

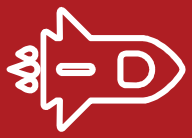
Odczytywanie wartości w obiektach i tablicach

```
var names = ['Janusz', 'Brajan', 'Seba', 'Mati'];
names[0] // 'Janusz';
names[3] // 'Mati';

var person = {
    name: 'Mirek',
    surName: 'Nowak',
    profession: 'Handlarz samochodów'
};

person.name // 'Mirek'
person.surName // 'Nowak'
```

ZADANIA



10.
Hoisting

HOISTING

Hoisting – przenoszenie deklaracji zmiennych na początek kodu bez ich wartości.

Hoisting

```
console.log(a); // "undefined" a nie "NOT DEFINED"
var a = 3;

// JAK KOD ZOSTANIE ZINTERPRETOWANY:
var a;
console.log(a);
a = 3;
```

Hoisting - również we funkcjach

```
function getGross(nettoPrice) {
    console.log(VAT); // undefined
    var VAT = 1.23;
    console.log(VAT);  // 1.23
    return nettoPrice * VAT;
}

// HOISTING
function getGross(nettoPrice) {
    var VAT;
    console.log(VAT); // undefined
    VAT = 1.23;
    console.log(VAT);  // 1.23
    return nettoPrice * VAT;
}
```

```
var one;
function getOne() {};
var two;
function getTwo() {};
var three;
getTwo();
getThree();
function getThree() {};
getOne();
```

```
(function() {
    var one;
    var two;
    var three;

    function getOne() {};
    function getTwo() {};
    function getThree() {};

    getOne();
    getTwo();
    getThree();
})();
```

Podsumowanie dnia I – podstawy JS

Co zapamiętać pod rozmowy kwalifikacyjne?

- Hoisting
- Closures
- Zmienne globalne vs lokalne
- Scope (zakres widoczności zmiennych)
- Co to jest IIFE i po co się używa
- Dlaczego JS'y podłącza się na koniec body w pliku HTML
- Co znaczy dynamiczność w kontekście języka JavaScript

Książki & Platformy & Blogi do nauki

JavaScript The Good Parts - O'Reilly
Eloquent JavaScript

https://www.codecademy.com/
https://www.codewars.com/

https://davidwalsh.name/
http://dailyjs.com/
https://www.sitepoint.com/javascript/
https://www.javascript.com/
https://www.polskifrontend.pl/ (agregator polskich blogów)

THE END

Pytania?

nastalytomasz@gmail.com
tomasz.nastaly @slack