JAVASCRIPT

2 programming paradigms for javascript app developers:

Prototypal inheritance (prototypes)

Functional programming (closures, first class functions, lambdas)

What is functional programming?

Pure functions, simple function composition avoiding side-effects

Class Inheritance vs Prototypal Inheritance:

Prototypal is simpler and more flexible than class.

typeof null = "object"

var a = b = 3 --> b=3; var a=b; (typeof a === undefined pq s'ha declarat dins del enclosing scope i la b la pren com a global en canvi)

Reason for wrapping the entire content of a JS source file in a function block:

Creates a private namespace and thereby helps avoid potential name clashhes between different JS modules and libraries.

Significance and benefits of including 'use strict' at the beggining of a JS source file:

Is a way to voluntarily enforce stricter parsing and error handling at runtime.

- Makes debugging easier, prevents accidental globals variable assigning values to undeclared variables, eliminates 'this' coercion, disallow duplicate parameter values, ...

```
return
{
	bar:"hello"
};
```

will return 'undefined' because after return they expect something at the same line, to solve it the '{' should go at the same line.

NaN:

Not a Number - value results from an operation that can't be performed because some operand or the result is non-numeric

Numbers in JavaScript are all treated with floating point precision, and as such, may not always yield the expected results (0.1 + 0.2 isn't necessary 0.3)

Possible way to write a function 'isInteger(x)':

{ return Math.round(x) === x; }

* 0 || 1 = 1

1 || 2 = 1

0 && 1 = 0

1 && 2 = 2

How do you clone an object?

var objclone = Object.assign({},obj);

Inside a for loop, if there is "setTimeout(function() { console.log(i);}, i*1000);"…

If "i" is declared using "var" it will print the last value of "i" all the times

If "i" is declared using "let" it will print each number correctly

* 1 < 2 < 3 = true → 1<2 = true → true(1) < 3 true

3 > 2 > 1 = false → 3>2 = true → true(1) > 1 false

Add element beginning and end of array:

myArray.push('end');

myArray.unshift('start');

* typeof typeof 1 = string ! (typeof 1 = "number" → typeof "number" = string)

How to empty an array in JS? arrayVar = []

* Number + Number -> Addition

Boolean + Number -> Addition

Boolean + Number -> Addition

Number + String -> Concatenation

String + Boolean -> Concatenation

String + String -> Concatenation

== / === -> compare object / compare object and type

Prototypes → avoid duplicated methods or variables in objects with multiple instances (can add properties or methods to constructors)

Debouncing is one way to solve this issue by limiting the time that needs to pass by until a function is called again (scroll event)

==	binary operator(compares the reference)
equals	compares the values, hashcode - compares the integer hash code

Angular 2

Changes in angular 2:	Typescript, component-based, change some directives…
Languages to build angular 2 application :	ECMAScript(ES) Typescript
Why typescript?	Can type the variables and it offers more possibilities
ng-show and ng-hide alternative:	[hidden]
Inter-component communications:	components send and receive information through @input and @output and is none of their business to know who is getting or sending the info. To receive a specific component info, create a listener to that component.
Components to configure routing:	Routes, RouterOutlet and RouterLink
Decorators:	allow developers to configure classes as particular elements by setting metadata on them (@component, @injectable, …)
Pipes:	transform and format data right from your template
Internationalization using angular 2:	using the directive i18n

AngularJS

Data binding:	automatic synchronization of data between model and view	
Scope:	acts like a bridge between view and model	
Controllers:	JavaScript functions that are bound to a particular scope	
Services:	singleton objects which are instantiated only once in app for a specific task	
Provider:	special factory method with a method get() which is used to return the value/service/factory	
Dependency injection:	dependent objects are injected rather than being created by the consumer (better testing)	
Filters:	select a subset of items from an array based on criteria and return a new array	
Directives:	markers on DOM elements that tell the compiler to attach a specific behavior	
Types of directives:	- Comment directives	- CSS class directives (C)
	- Attribute directives (A)	- Element directives (E)
Templates:	rendered view with information from the controller and model (can be single file or multiple views)	
Routing:	Concept of switching views. (i.e. Used to create Single Page Applications)	
Digest cycle:	When the scope model values change and there is an update	
Decrease digest cycle:	decrease the number of watchers…	
Deep linking:	allows you to encode the state of application in the URL so that it can be bookmarked	
Interceptor:	middleware code where all the $http requests go through	
Advantages:	- Capability to create Single Page Application clean and maintainable	
	- Provides data binding	- Unit testable
	- Uses dependency injection	- Reusable components
	- Can run on all major browsers	
Disadvantages:	- Not secure	- Not degradable (if user disables js, u're done)
Core directives:	- ng-app - ng-model	- ng-bind
Internationalization:	way to show locale specific information on a website	
How do you share data between controllers?	Services, events, $rootscope	
ng-show/ng-hide vs ng-if:	ng-if will not insert the DOM element if the condition isn't true	
Specify variable should have onw-time binding only:	using "::" in front of it	
Link function:	Combines directives with a scope to produce a live view. Instances DOM manipulation and registers DOM listeners.	

Promises: (inject $q dependency)

```
getName: function() {

        var deferred = $q.defer();

        //API call here that returns data
        testAPI.getName().then(function(name) {
              deferred.resolve(name); //Can be reject
        });

        return deferred.promise;
}
```

Reset a timeout and/or $interval → .cancel()

To disable $watch() → call deregistration function

How to validate text input for a twitter username (with @) → using ngPattern

JAVA

MAVEN - Build tool (compile files,...)

SPRING - Java framework with diferent modules [MVC, DAO, Web Module,...] (initially designed to controll injection dependency)

JENKINS - Continuous integration server

HIBERNATE - object-relational mapping. DB Framework (criteria API)

STATIC - same for all the instances

FINAL - non modifying

RUN vs START – start creates a new thread and executes run while run execute “run()” in that thread

THREAD-SAFE – property which guarantees that if executed by multiple threads it will behave as expected (it can be done by synchronizing…)

SPEECH

My name is Jordi Vila and I graduated in Computer Engineering at Autonomous University of Barcelona in 2016. Since almost 7 month, I work for Deloitte in IT consulting service and I have been pre-selected to work as an Angular Senior Developer in the Work Package 1, in Bux.

During the last two years, I have worked for several clients from the private sector. I had the chance to use a wide range of technologies ranging from web technologies to backend technologies like java using frameworks such as spring or hibernate.

Even though I focused mostly on the frontend side, I worked on the backend side too.

For example, in the last project, which was a hiring tool for a mortuary, we used angular for the frontend and micro services on the backend side, using both an SQL and a NoSQL databases to deal with the different parts of the app and having a CRM to manage some of the data used.

Before joining Deloitte I worked for another company whose main area was the restoration and I used to do websites for different clients and a web portal to provide our clients an environment to manage everything related with us. Working there I used mostly web programming languages as HTML5, PHP, JQuery among others, having the roles of frontend developer and database administrator.

Apart from the projects I have done in the companies that I have worked in, I have also worked as a freelance doing different projects using some of the programming languages that I've already said and some others like python.

Besides the projects, I prepare myself for the Oracle Java Programmer Certification and I do some trainings in web technologies like HTML5, PHP and Angular.

(To finish I would like to mention that…)In all projects I have taken part in, I have always worked agile. So from my point of view due to my recent experience and my knowledge in programming Angular I think am a good selection for this role and I would be happy to be part for the Bux Team.

(Topics to discuss if asking)

- Circular Injection Dependency - When injecting service in logs (or provider?)

- BD Local – No SQL – IndexDB(Application) – TreoDatabase(Library)