# CoreJS Interview #2

## JavaScript

### Objects Built-in methods

#### Know static Object methods

[1](matherials/JavaScript/Objects%20Built-in%20methods/Object%20Static%20Methods%20—%20Javascript%20_%20by%20Massoud%20Sharifi%20_%20Medium.mhtml)

[2](matherials/JavaScript/Objects%20Built-in%20methods/Методы%20объектов%20в%20JavaScript%20_%208HOST.COM.mhtml)

#### Property flags & descriptors (student is able to set property via Object. defineProperty)

#### Know how to create iterable objects, Symbol.iterator usage (optional)

### ECMAScript Data Types & Expressions

#### Object computed props

#### Be able to loop through Object keys

### Functional Scope

#### Know global scope and functional scope

#### Know variables visibility areas

#### Understand nested scopes and able work with them

### Functions Parameters / Arguments

#### Know how to define Function parameters

#### Know difference between parameters passing by value and by reference

#### Know how to handle dynamic amount of Function parameters

### Closures Advanced

#### Context (lexical environment)

#### Understand function creation context (lexical environment)

#### Be able to explain difference between scope and context

#### Inner/outer lexical environment

#### Understand lexical environment traversing mechanism

#### Understand connection between function and lexical environment

### ECMAScript Intermediate

#### Function default parameters

#### Know how to use spread operator for Function arguments

#### Be able to compare arguments and rest parameters

#### Spread operator for Array

#### Understand and able to use spread operator for Array concatenation

#### Destructuring assignment

#### Be able to discover destructuring assignment concept

#### Understand variables and Function arguments destructuring assignment

#### Know how for..of loop works (optional)

### Modules in JavaScript

#### What is module / module pattern? For what purposes they were created?

#### Modules types (AMD, ES6, CommonJS, UMD).

#### Modules syntax.

#### Common modules features (export default, named exports, exports as, etc).

#### Dynamic imports.

### Advanced Functions

#### this in functions

#### Reference Type & losing this

#### Understand difference between function and method

#### Understand how this works, realize this possible issues

#### Manage this

#### Be able to replace this value

#### Be able to use call and apply Function built-in methods

#### Know how to bind this scope to function

#### Binding, binding one function twice

### Functional Patterns

#### Callback (Function as argument)

#### Know callback pattern

#### Know IIFE pattern (optional)

#### Understand callback limitations (callback hell) (optional)

#### Carrying and partial functions

### Object Oriented Programming

#### new keyword

#### Understand how new keyword works

#### Function constructor

#### Know function constructor concept

#### Able to create constructor functions

#### Public, private, static members

#### Know how to create public/static/private members

#### Understand OOP emulation patterns and conventions (optional)

### ECMAScript Classes

#### Class declaration

#### Know class declaration syntax

#### Understand difference between class and constructor function

#### Getter/setter

#### What does super() do and where we have to use it?

### Prototypal Inheritance Basics

#### \_\_proto\_\_ property

#### Understand \_\_proto\_\_ object property

#### Able to use [Object.create] and define \_\_proto\_\_ explicitly

#### prototype property

#### Know function prototype property

#### Understand dependency between function constructor prototype and instance \_\_proto\_\_

#### Able to create 'class' methods using function prototype property

#### Able to set / get object prototype (optional)

#### ECMAScript Advanced Data Types & Expressions

#### Set/Map data types

#### WeakSet/WeakMap data types

### JavaScript Errors

#### JavaScript Errors (throw, Error class)

#### try..catch statement

#### Error handling

#### Error class

#### error logging

#### async error events

#### Custom errors (optional)

### ECMAScript Advanced

#### Promises

#### Promise states

#### Promise chaining

#### Promise static methods

#### Be able to compare promise and callback patterns (optional)

#### Be able to handle errors in promises

#### async/await

#### event loop

#### Garbage collector (concept) (optional)

## JavaScript in Browser:

### Global object window

#### Location

#### Know browser location structure

#### History API (Global object window)

#### Know browser History APIconcept

#### Be able to navigate within browser history

#### Be able to use history state (optional)

#### Navigator (optional)

#### Know how to parse user agent (optional)

#### Know how to discover client platform, browser

#### Cookies

### Page Lifecycle

#### Parsing

#### Reflow

#### Repaint

#### Critical rendering path (CRP) (optional)

#### Events Basics (optional)

#### Custom events (optional)

### Web components (optional)

#### Web components, shadow DOM (concept) (optional)

### Network requests

#### Fetch (with usage)

#### XMLHTTPRequest (concept) (optional)

#### WebSocket (concept) (optional)

### Timers (optional)

#### requestAnimationFrame (optional)

#### Be able to explain difference between setTimeout and requestAnimationFrame (optional)

### Web Storage API & cookies

#### Cookies

#### Difference between localStorage, sessionStorage and cookies

## Typescript:

### Ability to write concise TypeScript code using its constructs

#### basic types

#### enums

#### type / interface, differences between them

#### using interfaces with optional properties, read-only properties, etc...

#### function types

#### utitily types (optional)

#### typeguards (optional)

#### creating custom types

#### generic types (concept)

#### understanding TS (ES6) module system

## Design patterns:

### Creational Design Patterns

### Structural Design Patterns

### Behavioral Design Patterns

### MVC (optional)

### Intermediate knowledge of patterns and best practices:

#### SOLID principles

#### design patterns used on a student's project, and able to compare these patterns (optional)

### Software Development Methodologies (optional)

#### Agile

#### Scrum / Kanban / Waterfall

#### Estimation

### Testing (optional)

#### Testing Types

##### Integration Testing

##### E2E

##### Security Testing

##### Perforamance Testing

#### Test Pyramid

#### Testing approaches (optional)

#### FIRST

#### TDD и BDD

#### Frameworks (optional)

## Web Communication Protocols: (optional)

### HTTP vs HTTPS

### HTTP 1.x, 2.x, 3.x

### HTTP methods, headers, responses, body

### HTTP status codes groups (1xx, 2xx, 3xx, 4xx, 5xx)

### RESTful API

## Common web-security knowledge (optional)

### Basic understanding of most common security terms (CORS, XSS) (optional)

#### XSS

#### CORS

#### OWASP Top 10

#### Auth (JWT, OAuth, Basic, etc.)

## Coding tasks:

#### Function.prototype.bind implement polyfill

#### Object.create implement polyfill

#### Array.flat implement polyfill

#### Array.reduce implement polyfill

#### 'hello world'.repeating(3) -> 'hello world hello world hello world'. How to implement?

#### myFunc('!', 4, -10, 34, 0) -> '4!-10!34!0`. How to implement?

#### five(plus(seven(minus(three())))) -> 9. How to implement?

#### add(5)(9)(-4)(1) -> 11. How to implement?

#### periodOutput(period) method should output in the console once per every period how mach time has passed since the first function call. Example: periodOutput(100) -> 100(after 100 ms), 200(after 100 ms), 300(after 100 ms), ...

#### extendedPeriodOutput(period) method should output in the console once per period how mach time has passed since the first function call and then increase the period. Example: // extendedPeriodOutput(100) -> 100(after 100 ms), 200(after 200 ms), 300(after 300 ms)