========== Javascript ES6+ ==========

\* 13 ES6+ popular:

- let/const

- Template Literals

- Arrow Function

- Enhanced Object Literals

- Default Parameters

- Rest/Spread (...)

- Destructuring Assignment

- Promises

- Modules (import/export)

- Classes (class, extends, this, supper, static, private field - #, method override)

- Numeric separator (\_)

- Nullish Coalescing (??)

- Optional Chaining (?.)

========== Javascript Basic ==========

\* Giá trị falsy vs truthy:

- Falsy: 0, 0n, NaN, '', "", null, undefined, false

- Truthuy: Tất cả các giá trị không phải giá trị falsy

\* Data type:

- Dữ liệu nguyên thuỷ - Primitive type (number, string, boolean, undefined, null)

- Dữ liệu phức tạp - Complex type (object, function)

- Kiểm tra kiểu dữ liệu bằng toán tử typeof (đặc biệt typeof null là object)

- Kiểm tra biến có thuộc 1 lớp cho trước hay không dùng toán tử instanceof

\* Toán tử trong Javascript

- Toán tử số học - Arthmetic (+, -, \*, /, \*\*, %, ++, --)

- Toán tử gán - Assignment (=, +=, -=, \*=, /=, \*\*=, %=)

- Toán tử so sánh - Comparison (<, <=, >, >=, ==, ===, !=, !==)

- Toán tử logic - Logical (&&, ||, !)

\* Toán tử cộng/trừ trong Javascript

// Đối với phép cộng:

- Number + Number = Number

- String + String = String

- Number + String = String + String = String

// Đối với phép trừ:

- Number - Number = Number

- String - String = NaN - NaN = NaN

- Number - String = Number - NaN = NaN

- String - Number = NaN - Number = NaN

- NumberString - Number = Number - Number = Number

- Number - NumberString = Number - Number = Number

=> Đối với phép cộng thì ưu tiên chuỗi, đồi với phép trừ thì ưu tiên số.

\* String Javascript:

- 2 cách tạo:

- Cách 1: const fullName = 'Minh Tân';

- Cách 2: const fullName = new String('Minh Tân');

- Nên dùng cách 1 vì:

- Cách 1 ngắn hơn

- Cách 1 xử lý nhanh hơn

- Cách 2 cho kiểu dữ liệu không như mong muốn (dùng typeof sẽ cho giá trị 'object')

- Làm việc với String:

- Get length (length)

- Find index (indexOf, lastIndexOf, search)

- Check key word includes string (includes)

- Cut string (slice)

- Replace (replace)

- Convert to upper case (toUpperCase)

- Convert to lower case (toLowerCase)

- Trim (trim, trimStart, trimEnd)

- Split (split)

- Get a character by index (charAt, [])

- Check string start with value (startsWith)

\* Number Javascript:

- 2 cách tạo:

- Cách 1: const age = 24;

- Cách 2: const age = new String(24);

- Nên dùng cách 1 vì:

- Cách 1 ngắn hơn

- Cách 1 xử lý nhanh hơn

- Cách 2 cho kiểu dữ liệu không như mong muốn (dùng typeof sẽ cho giá trị 'object')

- Làm việc với Number:

- To string

- To fixed

\* Math Javascript:

- Làm việc với Math:

- Math.PI

- Math.abs()

- Math.random()

- Math.round()

- Math.ceil()

- Math.floor()

- Math.min()

- Math.max()

\* Date Javascript:

- Cách tạo:

const date = new Date();

- Làm việc với Date:

- Get year (getFullYear)

- Get month (getMonth => 0 - 11)

- Get date (getDate => 1 - 31)

- Get day (getDate => 0 - 6)

- Get hours (getHours => 0 - 23)

- Get minutes (getMinutes => 0 - 59)

- Get seconds (getSeconds => 0 - 59)

- Get mili seconds (getMilisSeconds => 0 - 999)

- Get now (Date.now())

- Get timestamp (milliseconds since 1/1/1970)

\* JSON Javascript:

- Data type:

- Number

- String

- Boolean

- Null

- Array

- Object

- Làm việc với JSON:

- JSON.stringify

- JSON.parse

\* Array Javascript:

- Tạo mảng:

- Cách 1: const pets = ['Chó', 'Mèo', 'Gà', 'Vịt'];

- Cách 2: const pets = new Array('Chó', 'Mèo', 'Gà', 'Vịt');

- Nên dùng cách 1 vì:

- Cách 1 ngắn hơn

- Cách 1 xử lý nhanh hơn

- Làm việc với Array:

- Dùng typeof của Array sẽ cho giá trị 'object', để kiểm tra có phải Array hay không ta dùng Array.isArray()

- Get length (length)

- Get a element by index ([])

- To string (toString)

- Join (join)

- Split (split)

- Access element (unshift, shift, push, pop)

- Splicing (splice)

- Concat (concat)

- Slicing (slice)

- Check element includes array (includes)

- Array callback (forEach, every, some, find, filter, map, reduce):

- forEach(callback(currentValue?, index?, array?): undefined, thisArg?): undefined => Duyệt qua tất cả phần tử của mảng (mutable element).

- every((callback(element, index?, array?): boolean), thisArg?): boolean => Trả về true nếu tất cả callback của mảng trả về true và ngược lại.

- some((callback(element, index?, array?): boolean), thisArg?): boolean => Trả về true nếu có 1 callback của mảng trả về true và ngược lại.

- find((callback(element, index?, array?): boolean), thisArg?): element || undefined => Trả về element đầu tiên tìm được nếu callback tại vị trí đó trả về true, ngược lại trả về undefined.

- filter((callback(element, index?, array?): boolean), thisArg?): Array => Trả về 1 mảng các element nếu callback tại vị trí đó trả về true, ngược lại trả về mảng rỗng.

- map((callback(currentValue, index?, array?): element), thisArg?): Array => Trả về 1 mảng các element với các giá trị trả về của callback.

- reduce((callback(accumulator, currentValue, currentIndex?, array?): element), initialValue?): value => Trả về 1 giá trị tích luỹ cuối cùng khi mỗi callback trả về.

\* Object Javascript:

- Làm việc với Object:

- Object.create()

- Object.keys()

- Object.values()

- Object.entries()

- Object.assign()

- Object.freeze()

\* Loop Javascript:

- Các loại vòng lặp:

- for: Lặp với điều kiện đúng

- for/in: Lặp qua key của đối tượng (array, object)

- for/of: Lặp qua value của đối tượng (array)

- while: Lặp khi điều kiện đúng

- do/while: Lặp ít nhất 1 lần rồi sau đó lặp khi điều kiện đúng

- Làm việc với Loop:

- Loop by keys (Object.keys)

- Loop by values (Object.values)

- Key word (break, continute)

\* Promise Javascript:

- Lý do Promise được sinh ra:

- Khái niệm sync/async

- Phải sử dụng callback để xử lý async

- Giải quyết "nỗi đau" của callback (callback hell, pyramid of doom)

- Tư tưởng của Promise (Promise concept):

- Pennding (chưa call 1 trong 2 function resolve hoặc reject)

- Fulfilled (đã call function resolve)

- Rejected (đã call function reject)

- Tạo Promise:

- const promise = new Promise((resolve, reject) => {

// Logic

// Thành công: resolve()

// Thất bại: reject()

});

- promise.then(function() {

// Khi resolve được gọi, callback then sẽ được chạy

})

- promise.catch(function() {

// Khi reject được gọi, callback catch sẽ được chạy

})

- promise.finally(function() {

// Khi resolve hoặc reject được gọi, callback fiinally sẽ được chạy

})

- Làm việc với Promise:

- Sử dụng tính chất chuỗi (chain) của Promise (Promise có thể có nhiều then nhưng chỉ có 1 catch và 1 finally)

- Promise.resolve()

- Promise.reject()

- Promise.all()

- Promise.race()

========== Javascript Advance ==========

\* Arrow function: Là cách viết hàm ngắn gọn hơn, nhưng có 1 đặc điểm khác biệt về context (this)

\* Higher-order function: MDN định nghĩa 1 hàm nhận vào 1 hàm làm tham số hoặc trả về 1 hàm thì gọi là higher-order function.

\* Closure: Là khả năng truy xuất đến biến bên ngoài scoped của 1 hàm.

\* Function declarations vs function expressions: khác nhau về context this và khả năng hoisting

\* Immediately Invoked Function Expression - IIFE: Là cách tạo ra 1 hàm và thực thi chúng luôn.

\* Hoisting: Là năng đưa khai báo lên lên trước khi sử dụng.

\* Value type vs references type: Chú ý đến vấn đề cùng 1 vùng nhớ.

\* Falsy value vs truthy value: 0, NaN, undefined, null, '', false

\* Function context: bind, call, apply – this.

\* Template string: Chuỗi đc đặt trong cặp `` và đối số được truyền dưới dạng ${variable}.

\* Array-like object và arguments: 1 object có dạng giống array nhưng không có những method của Array, arguments (array-like object) là 1 biến tồn tại bên trong scoped của 1 function (không tồn tại trong arrow function).

\* Default parameters: Tham số mặc định khi hàm không được truyền đối số.

\* This keyword and context(bind, call, apply): 3 phương thức cho phép truyền context vào bên trong hàm nhưng mỗi phương thức có 1 cách sử dụng khác nhau.

\* Object literal vs object constructor: Các cách tạo ra 1 đối tượng.

\* Enhanced object literal: Cách tạo ra 1 object literal bằng cú pháp ngắn(shorthand syntax).

\* Classes: class, extends, this, super, static, method override.

\* Rest: Rest là kỹ thuật tham số được truyền dưới dạng ...rest và biến rest chính là 1 mảng chứa phần còn lại của các đối số khi hàm được gọi khi các đối số đầu tiên đã thoả mãn.

\* Spread: Spread là kỹ thuật clone array hoặc clone object(shallow-clone).

\* Destructuring: Destructuring là kỹ thuật tạo nhanh các biến thông qua array và object ban đầu.

========== Javascript HTML DOM ==========

\* DOM:

- Element Node (Get Element methods)

- Attribute Node (Access to Attribute)

- Text Node (Access to Text)

\* Get Element methods (id, class, tag, CSS selector, HTML collection):

- getElementById

- getElementsByClassName

- getElementsByTagName

- querySelector

- querySelectorAll

- forms

- Element Node:

- innerHTML

- outerHTML

- attributes

- childElementCount

- childNodes

- children

- classList (add, contains, remove, toggle, value)

- className

- firstChild

- lastChild

- firstElementChild

- lastElementChild

- nextSibling

- nextElementSibling

- nodeName

- nodeType (element - 1, attribute - 2, text - 3)

- nodeValue

- ownerDocument

- parentElement

- parentNode

- spellCheck

- style

- tabIndex

- tagName

- title

- \_\_proto\_\_

\* Access to Attribute:

- getter/setter (chỉ có tác dụng với các attribute hợp lệ)

- getAttribute

- setAttribute

\* Access to Text:

- innerText

- textContent

\* DOM CSS (style):

- width

- height

- backgroundColor

- ... (thuộc tính CSS hợp lệ)

\* DOM Event:

- Keyboard events

- Mouse events

- ...

- Event method support:

- preventDefault

- stopPropagation