Theory Class | JavaScript Interviews

🤐 Higher Order Functions | `map` , `filter` , `reduce`

- A higher order function is a function that takes a function as an argument or returns a function as a result.
- Few other functions : `find`, `findIndex`, `forEach`, `some` and `every`.
- `Arrow functions` are also higher-order functions.

Map function | `map()`

> Example to illustrate the concept of `higher-order function : map()`

```
let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

function squarer(x) {
  // callback function
  return x * x;
}

let squared = arr.map(squarer);
console.log(squared);
```

> Output :

```
$ node lecture-017/map.js
[
   1,  4,  9, 16,  25,
  36, 49, 64, 81, 100
]
```

- map() takes a callback function as an argument
- map() is an array function that expects a callback function as an argument
- map() will return a new array with the results of the callback function for each element in the original array

> Example :

```
let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let cubed = arr.map((x) => x * x * x);
console.log(cubed);
```

- operator `=>` is called the arrow operator

> Output:

```
$ node lecture-017/map.js
[
    1,    8,  27,  64,
  125,  216, 343, 512,
  729, 1000
]
```

- When we write a self defined higher-order function, we call it as `polyfill`.
- map() is a polyfill for `Array.prototype.map()`
- map() `does not tamper the original array`
- map() `returns a new array`

> 💥 Quick Task : Use `map() to create a new array of strings of the names`

```
let names = ["John", "Mary", "Mike", "Suzy"];
let nameStrings = names.map(function (x) {
  return x;
});
console.log(nameStrings);
```

> Output :

```
$ node lecture-017/map.js
[ 'John', 'Mary', 'Mike', 'Suzy' ]
```

Split function | `split()`

- `split()` is a string method that splits a string into an array of substrings based on the delimiter.
- split() splits a string into an array of substrings based on the separator string you provide as an argument.
- The separator string can be a character, a string, or a regular expression.
- If the separator is not specified, the string is split on every character.
- If the separator is an empty string (""), the string is split on every character.

> Examples :

```
let str = "pepcoder";
let parts = str.split("c");

let str2 = "Hello World";
let parts2 = str2.split(" ");
```

```
console.log(parts);
console.log(parts2);
```

> Output :

```
$ node lecture-017/splitJoin.js
[ 'pep', 'oder' ]
[ 'Hello', 'World' ]
```

Join function | `join()`

- join function reverses the process of split()
- join() takes an array and joins the elements into a string
- it takes an optional separator argument which is used to separate the elements of the array
- if the separator is not specified, the array elements are `separated by commas`

> Example :

```
let joinedStr = parts.join("c"); // "pepcoder"
let joinedStr2 = parts2.join(" "); // "Hello World"

console.log(joinedStr);
console.log(joinedStr2);
```

> Output :

```
$ node lecture-017/splitJoin.js
[ 'pep', 'oder' ]
[ 'Hello', 'World' ]
pepcoder
Hello World
```