ES .NEXT

MORDERN JAVASCRIPT

TODAY'S AGENDA

- DATA STRUCTURE, COLLECTIONS AND NEW CONTROL STATEMENT
  - SET OBJECT
  - MAP OBJECT
  - FOR ... OF
- ITERABLE
- GENERATORS
- NEW PRIMITIVE DATATYPE (SYMBOL)
- CLASSESS
- MODULARITY
- LAB

Set Object

- In mathematical sense, a set of group values that unique
- It's an iterable object
- We can pass an array when I'm creating a set and this will remove the duplicate items
- Method:
    - .has() / .add() / .delete() / .clear() / .entries() / .keys() / .values()

Map Object

- is an object of key/value pairs both key and value can be either privative or object
- Method:
    - .set(key,val) / .get(key) / .delete(key) / .clear() / .has(key) / .keys() / .values() / .entries()

For .... Of

- For .... Of statement iterates over property values
- It's better way to loop over iterable objects

```
var myStr = ""
var myArr = [,,]
var mySet = new Set([,,,,])
var myMap = new Map([[,],[,],[,]])
```

Iterable Objects

JS

- Must have @@iterator method
- The implementation [symbol.iterator]()
- Can use: for ... of , destructuring, ...spread operator

```
var arr = [1,2,3,4,5,6]
```

ITERABLE OBJECT → .NEXT() {VALUE,DONE }

```
var iter = arr[Symbol.iterator]()
```

ITERATOR OBJECT

Generator Function

- function* genfn() { yield 1 , yield 2, yield 3 }

```
function* gen() {

yield 10;
yield 15;
yield *[11,23,12]
yield *'ABC'

}
```

```
for( i of gen() ) {
    console.log(i)
}
```

Symbol

JS

- New primitive data type in JavaScript (NEW in ES6)
  - Unique
  -
  - Considered as UUID or GUID
    - Universally Unique Identifier or Globally Unique Identifier
  - Can be used as object key

```
SYMBOL('DESCRIPTION')

SYMBOL.FOR('DESCRIPTION') => REGISTRY
```

Symbol

- Static Properties
  - Symbol.match()
  - Symbol.replace()


```
[SYMBOL.REPLACE](STR,IDX){}
```

Symbol

JS

- Symbol as object property:
  - Non-enumerable
  - Anonymous
  - Can't convert to JSON object when we use JSON.stringfiy()

OBJECT.GETOWNPROPERTYSYMBOLS(OBJ)

EXPAMPLE: [SYMBOL.FOR](10): 123

Class


```
class className {

    constructor(p1,p2) {
     this._p1 = p1;
     this._p2 = p2;
 }

  get p1() { return this.p1; }

  set p1(val) { this.p1 = val; }

  static staticFn() { return ; }

  static get staticProp() { return ; }

}
```

Modules

NAMED EXPORT

```
<script type="module">

 import { ..... } from "moduleName"

 import * as someName from "moduleName"

</script>
```

Modules

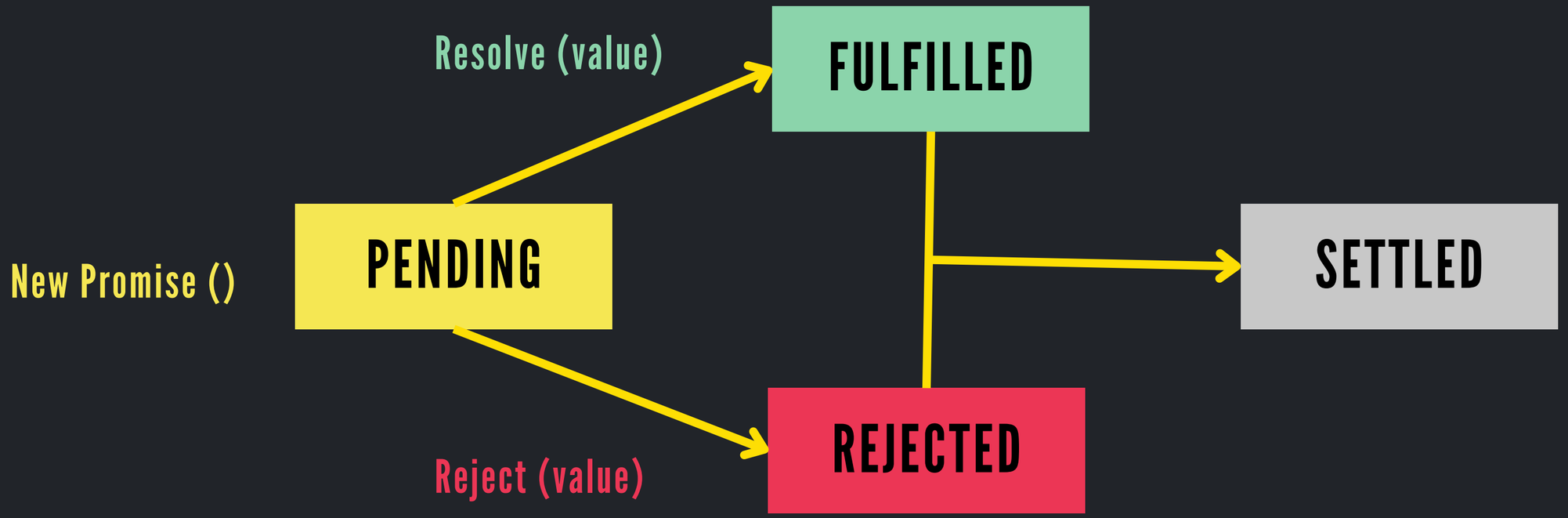
DEFAULT EXPORT

```
export default class className { }
```

```
import className from "moduleName"
```

Promise

Promise is an object representing the eventual completion or failure of an asynchronous operation



Promise

Consuming promise:

THEN()

Handle the success
of the promise

CATCH()

Handle the failure
of the proimse

FINALLY()

Handle after all
chain methods

Promise

Promise Static Methods

PROMISE.ALL()

Returns either resolved promise if all passed promises are resolved

Or rejected promise if as soon as of these promises is rejected

PROMISE .RACE()

Returns rejected or resolved promise as soon as one of the passed promises is settled

Promise

Promise Static Methods

PROMISE.REJECT(REASON)

Returns rejected promise object with the given reason

PROMISE.RESOLVE(REASON)

Returns resolved promise object with the given reason

Async & Await

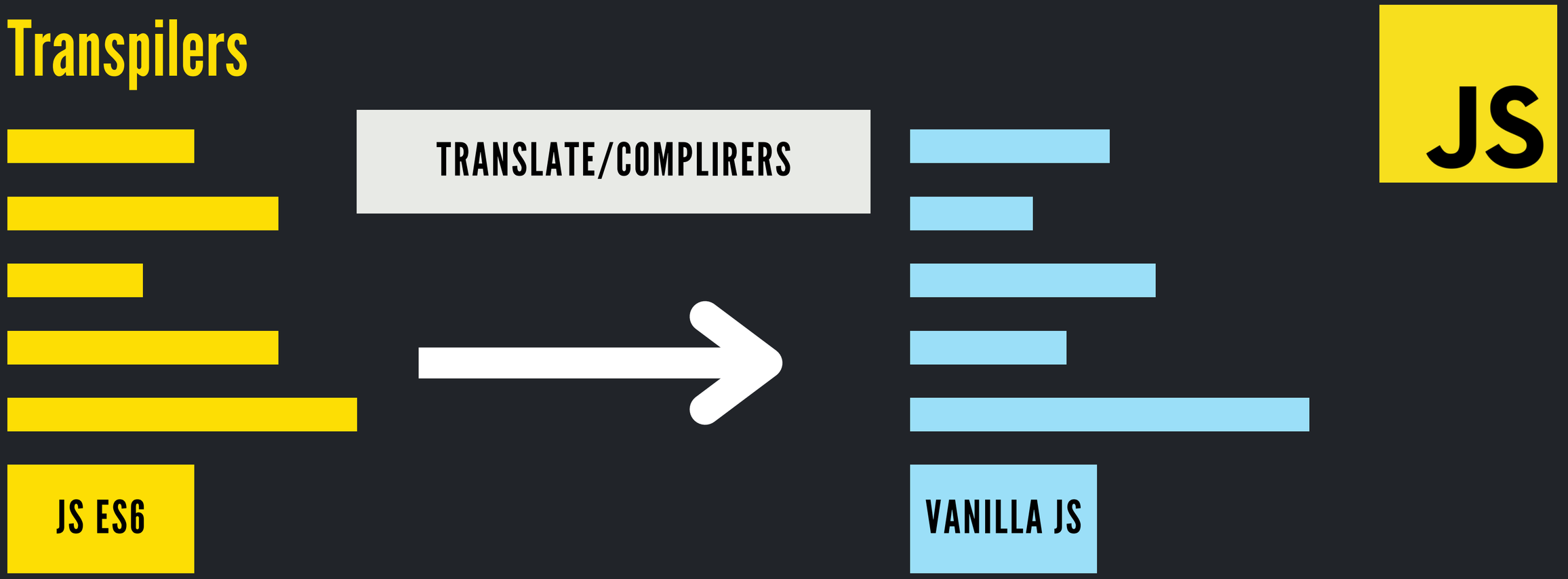
ASYNC FUNCTION WRITES PROMISE-BASED CODE AS BEHAVES IF IT WERE SYNCHOROUS CODE

```
async function funName {
        await new Promise().then().catch()
        await new Promise().then().catch()
        await new Promise().then().catch()
}
```

~~Asynchronous~~ → Synchronous

WHEN USING AWAIT, THE FUCNTION IS PAUSED IN A NON-BLOCKING WAY UNTILL THE PROMISE SETTLES

Transpilers



EX: BABEL JS

https://es6console.com/

New Features (ES7, ES8, ES9)

ECMASCRIPT 2016 (ES7)

- ARRAY.INCLUDES()
- EXPONENTIAL OPERATOR **

New Features (ES7, ES8, ES9)

JS

ECMASCRIPT 2017 (ES8)

- STRING.PADSTART()
- STRING.PADEND()
- OBJECT.VALUES(OBJ)
- OBJECT.ENTRIES(OBJ)
- OBJECT.GETOWNPROPERYDESCRIPTOR(CONSTR)

New Features (ES7, ES8, ES9)

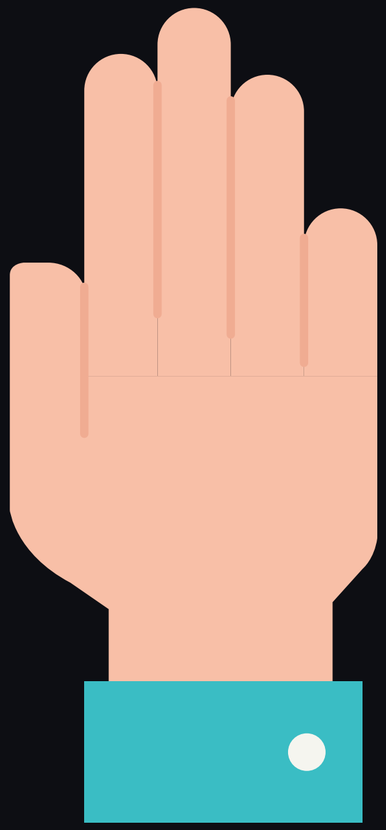
ECMASCRIPT 2018 (ES9)

- SYMBOL.DESCRPITION
- TRY {} CATCH {}
- PROMISE.FINALLY()
- ARRAY.FLAT() / ARRAY.FLATMAP()

New Features (ES7, ES8, ES9)

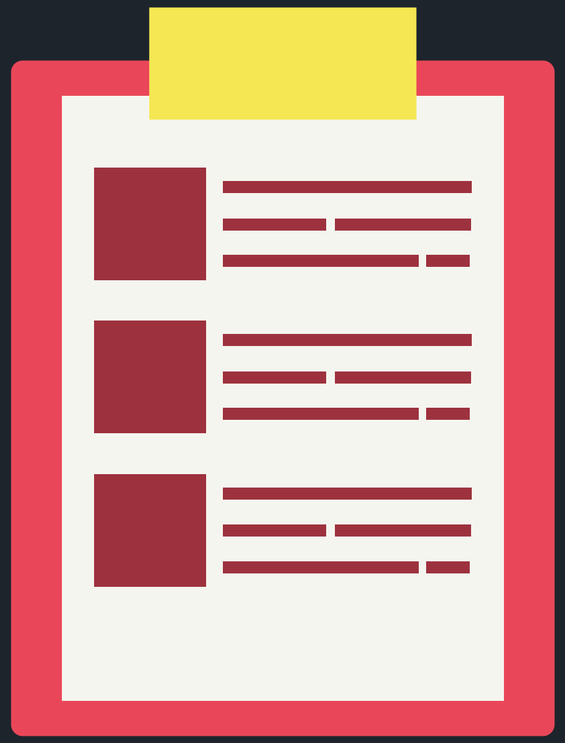
ECMASCRIPT 2020

- BIGINT (NUMBER N)
- CHAINING ?.
- NULLISH ??
- GLOBALTHIS



THANK YOU

ANY QUESTIONS ?



LAB

Create any array of food called 'food':

['Burger', 'Pizza', 'Donuts', 'Pizza', 'Koshary', 'Donuts', 'Seafood','Burger']

01

Create a Set with values of this array

02

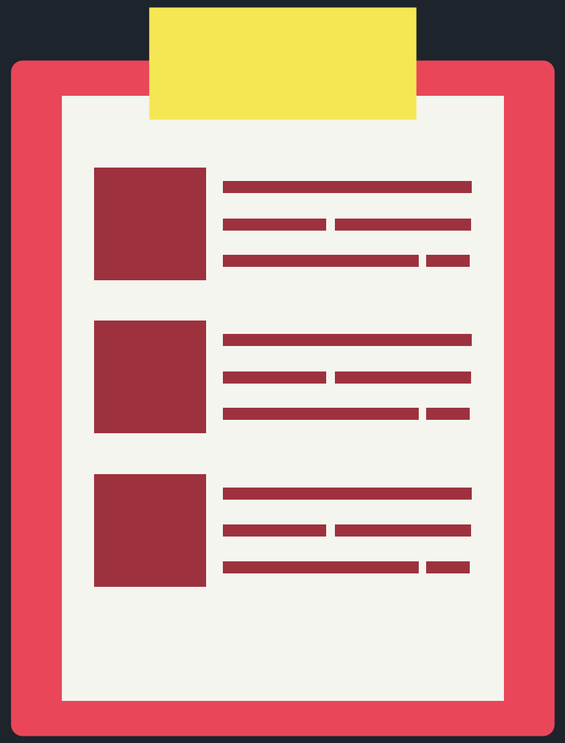
Add 'pasta' to the set and log the set to the console

03

Remove 'burger' from the set and log the set to the console

04

Write a function that takes the set as a parameter and clear the set if it has more than 2 items



LAB

Fetch the following API :

https://api.npoint.io/838397f84625a7abd979

01

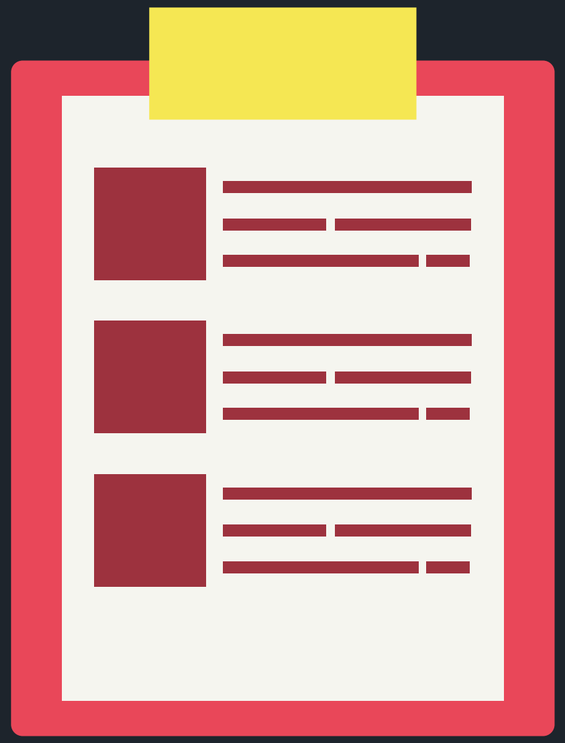
Format the array of users by adding an extra full_name attribute to each item

02

Get only males who are older than 30

03

Group the filtered users by nationality : { EG: [{ .... }, { .... }] };



LAB

Create a class called 'Vehicle'

01

The class has a constructor function that takes 2 parameters (wheels, speed)

02

Create a sub-class 'Bike' that inherits from vehicle and has different default values (wheels: 2 , spead: 'fast enough')

03

Add a static method to bike sub-class to count how many time it's called