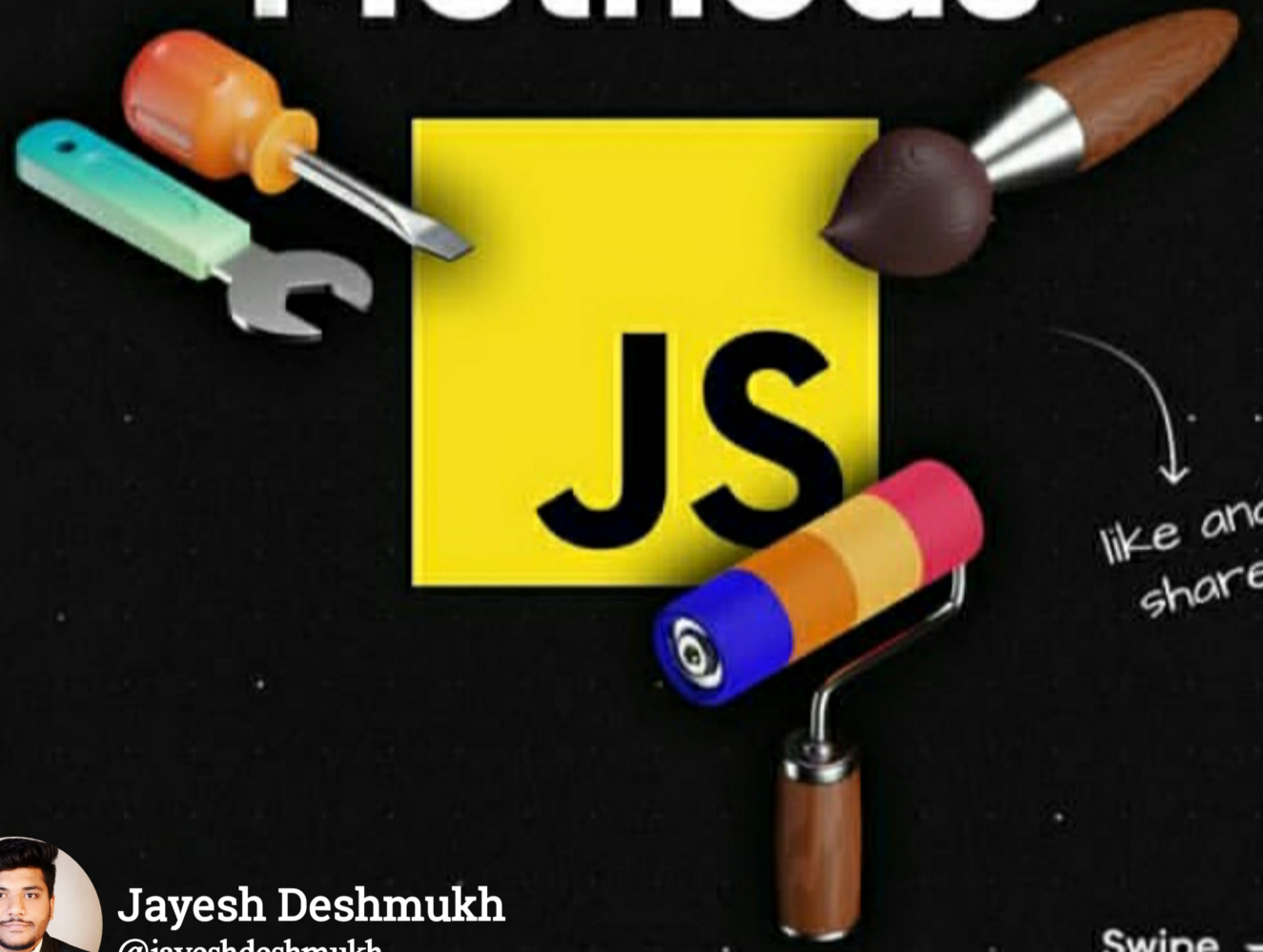
JS

Important Javascript

Object Methods



Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.keys()

A simple way to iterate over an object and return all of the object's keys

```
const employee = { name: 'Daniel',
age: 40, occupation: 'Engineer',
level: 4 };

console.log(Object.keys(employee));
```

output

```
// [object Array] (4)
["name","age","occupation","level"]
```

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.values()

Iterates over the object and returns the object's values!

```
const employee = { name: 'Daniel',
age: 40, occupation: 'Engineer',
level: 4 };

console.log(Object.values(employee));
```

output

```
// [object Array] (4)
["Daniel",40,"Engineer",4]
```

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.entries()

Takes an object and returns its own enumerable string-keyed property [key, value] pairs of the object.

```
const drinks = {
  maple: 'out of stock',
  orange: 3.5
};

for (const [name, cost] of
Object.entries(drinks)) {
  console.log(`${name}: ${cost}`);
}
```

output →

```
"maple: out of stock"
"orange: 3.5"
```

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.create()

Creates a new object, using an existing object as the prototype of the newly created object.

```
let Student = {
  name: "fuzzy",
  display() {
    console.log("Name:", this.name);
  }
};

let std1 = Object.create(Student);

std1.name = "wuzzy";
std1.display();
```

create object from Student prototype

output

```
"Name:" "wuzzy"
```

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.assign()

Copies all enumerable and own properties from the source objects to the target object. It returns the target object.
It is also called shallow copy.

```
const target = { a: 1, b: 2 };
const source = { b: 4, c: 5 };

const returnedTarget =
Object.assign(target, source);

console.log(target);
console.log(returnedTarget);
```

output →

```
// [object Object]
{
  "a": 1,
  "b": 4,
  "c": 5
}
```

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.seal()

Seals an object which prevents new properties from being added to it and marks all existing properties as non-configurable.

```
const car = {
  price: 15000
};

Object.seal(car);
car.price = 18000;
console.log(car.price);
// 18000

delete car.price;
console.log(car.price);
// 18000
```

value changed successfully

cannot delete when sealed

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

JS

Object.freeze()

Freezes an object. A frozen object can **no longer be changed**; this means:

1. New properties from being added to the object.
2. Existing properties to be removed from the object.
3. Changing the enumerability, configurability, or writability of existing properties.
4. Changing values of the existing object properties and prototype.

Swipe for example code

Jayesh Deshmukh
@jayeshdeshmukh

JS

```
const client = {
  budget: 3000
};

Object.freeze(client);

client.budget = 2500;
console.log(client.budget);
// 3000
```

Shows error in strict mode

unchanged value as output

Jayesh Deshmukh
@jayeshdeshmukh

Swipe ⟶

TURN ON POST NOTIFICATION

THANKS FOR READING

FOLLOW FOR MORE.

Jayesh Deshmukh