JAVASCRIPT TRICKY OUTPUT-BASED QUESTIONS

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

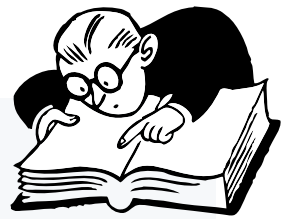
```
let randomValue = { name: "Dimple" }
randomValue = 23

if (!typeof randomValue === "string") {
        console.log("It's not a string!")
} else {
        console.log("Yay it's a string!")
}
```

- A: `It's not a string!`
- B: `Yay it's a string!`
- C: `TypeError`
- D: `undefined`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



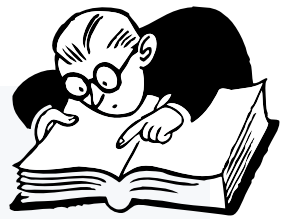
```
const user = {
        email: "my@email.com",
        updateEmail: email => {
                this.email = email
        }
}

user.updateEmail("new@email.com")
console.log(user.email)
```

- A: `my@email.com`
- B: `new@email.com`
- C: `undefined`
- D: `ReferenceError`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



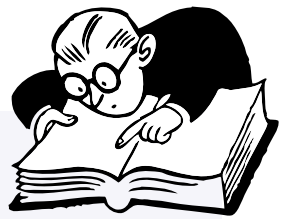
```
const fruit = ['🍌', '🍊', '🍎']

fruit.slice(0, 1)
fruit.splice(0, 1)
fruit.unshift('🍇')

console.log(fruit)
```

- A: ['🍌', '🍊', '🍎']
- B: ['🍊', '🍎']
- C: ['🍇', '🍊', '🍎']
- D: ['🍇', '🍌', '🍊', '🍎']

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



```
class Calc {
        constructor() {
                this.count = 0
        }

        increase() {
                this.count++
        }
}

const calc = new Calc()
new Calc().increase()

console.log(calc.count)
```

- A: `0`
- B: `1`
- C: `undefined`
- D: `ReferenceError`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

```
let count = 0;
const nums = [0, 1, 2, 3];

nums.forEach(num => {
        if (num) count += 1
})

console.log(count)
```

- A: 1
- B: 2
- C: 3
- D: 4

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

```
class Bird {
  constructor() {
    console.log("I'm a bird. 🦢");
  }
}

class Flamingo extends Bird {
  constructor() {
    console.log("I'm pink. 🌸");
    super();
  }
}

const pet = new Flamingo();
```

- A: `I'm pink. 🌸`
- B: `I'm pink. 🌸` `I'm a bird. 🦢`
- C: `I'm a bird. 🦢` `I'm pink. 🌸`
- D: Nothing, we didn't call any method

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

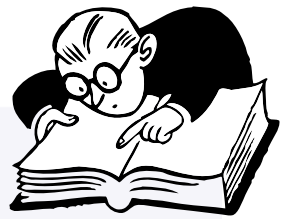
```
const person = {
  name: 'DimpleKumari',
  hobbies: ['coding'],
};

function addHobby(hobby, hobbies = person.hobbies) {
  hobbies.push(hobby);
  return hobbies;
}

addHobby('running', []);
addHobby('dancing');
addHobby('baking', person.hobbies);

console.log(person.hobbies);
```

- A: `["coding"]`
- B: `["coding", "dancing"]`
- C: `["coding", "dancing", "baking"]`
- D: `["coding", "running", "dancing", "baking"]`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



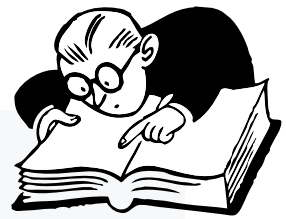
```
class Counter {
  #number = 10

  increment() {
    this.#number++
  }

  getNum() {
    return this.#number
  }
}

const counter = new Counter()
counter.increment()

console.log(counter.#number)
```

- A: `10`
- B: `11`
- C: `undefined`
- D: `SyntaxError`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



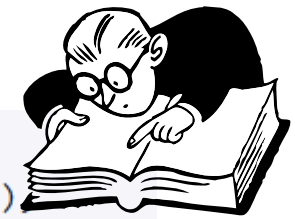
```
const add = x => x + x;

function myFunc(num = 2, value = add(num)) {
  console.log(num, value);
}

myFunc();
myFunc(3);
```

- A: `2` `4` and `3` `6`
- B: `2` `NaN` and `3` `NaN`
- C: `2` `Error` and `3` `6`
- D: `2` `4` and `3` `Error`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



```
const handler = {
  set: () => console.log('Added a new property!'),
  get: () => console.log('Accessed a property!'),
};

const person = new Proxy({}, handler);

person.name = 'Dimple';
person.name;
```

- A: `Added a new property!`
- B: `Accessed a property!`
- C: `Added a new property!` `Accessed a property!`
- D: Nothing gets logged

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

```
// sum.js
export default function sum(x) {
  return x + x;
}

// index.js
import * as sum from './sum';
```

- A: `sum(4)`
- B: `sum.sum(4)`
- C: `sum.default(4)`
- D: Default aren't imported with `*`, only named exports

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

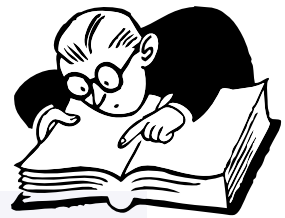
```
class Counter {
  constructor() {
    this.count = 0;
  }

  increment() {
    this.count++;
  }
}

const counterOne = new Counter();
counterOne.increment();
counterOne.increment();

const counterTwo = counterOne;
counterTwo.increment();

console.log(counterOne.count);
```

- A: `0`
- B: `1`
- C: `2`
- D: `3`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



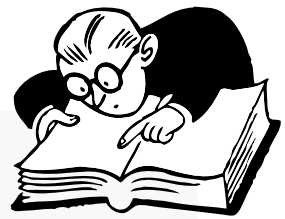
```
const myPromise = Promise.resolve('Woah some cool data');

(async () => {
  try {
    console.log(await myPromise);
  } catch {
    throw new Error(`Oops didn't work`);
  } finally {
    console.log('Oh finally!');
  }
})();
```

- A: `Woah some cool data`
- B: `Oh finally!`
- C: `Woah some cool data` `Oh finally!`
- D: `Oops didn't work` `Oh finally!`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



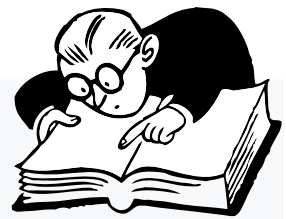
```
const add = x => y => z => {
  console.log(x, y, z);
  return x + y + z;
};

add(4)(5)(6);
```

- A: `4` `5` `6`
- B: `6` `5` `4`
- C: `4` `function` `function`
- D: `undefined` `undefined` `6`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



```
let config = {
  alert: setInterval(() => {
    console.log('Alert!');
  }, 1000),
};

config = null;
```

- A: The `setInterval` callback won't be invoked
- B: The `setInterval` callback gets invoked once
- C: The `setInterval` callback will still be called every second
- D: We never invoked `config.alert()`, config is `null`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?

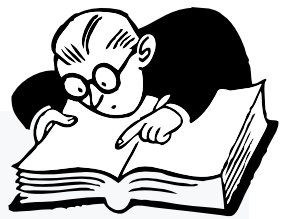
```
let name = 'Dimple';

function getName() {
  console.log(name);
  let name = 'Twinkle';
}

getName();
```

- A: Dimple
- B: Twinkle
- C: `undefined`
- D: `ReferenceError`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



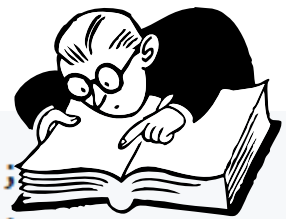
```
const colorConfig = {
  red: true,
  blue: false,
  green: true,
  black: true,
  yellow: false,
};

const colors = ['pink', 'red', 'blue'];

console.log(colorConfig.colors[1]);
```

- A: `true`
- B: `false`
- C: `undefined`
- D: `TypeError`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



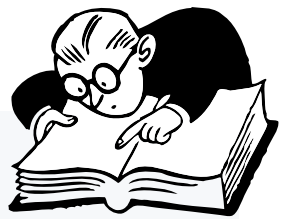
```
const one = false || {} || null;
const two = null || false || '';
const three = [] || 0 || true;

console.log(one, two, three);
```

- A: `false` `null` `[]`
- B: `null` `""` `true`
- C: `{}` `""` `[]`
- D: `null` `null` `true`

@DimpleKumari

Forming a network of fantastic coders.

What's the output?



```
const name = 'Dimple';

console.log(name());
```

- A: `SyntaxError`
- B: `ReferenceError`
- C: `TypeError`
- D: `undefined`

What's the output?

```
let newList = [1, 2, 3].push(4);

console.log(newList.push(5));
```

- A: `[1, 2, 3, 4, 5]`
- B: `[1, 2, 3, 5]`
- C: `[1, 2, 3, 4]`
- D: `Error`

@DimpleKumari

Forming a network of fantastic coders.

THANK YOU FOR READING!

If you found this informative and valuable, I'd love for you to connect with me. Follow me Medium, Codepen, and connect with me on LinkedIn to stay updated on the latest in web development, interviews, and more.

Let's connect!

💼 **LinkedIn** — https://www.linkedin.com/in/dimple-kumari/

🔗 **Medium** — https://medium.com/@dimplekumari0228

✍️ **Codepen** — https://codepen.io/DIMPLE2802

@DimpleKumari

Forming a network of fantastic coders.

DIMPLE KUMARI

Forming a network of fantastic coders.