JavaScript Jeopardy!

A Deep Dive into Angular & JavaScript Quirks

Presented by Martin Ragan

Q1 - Angular challenge lifecycle hooks with a twist

What is the output of the console.log after user enters page?

```
export class Component implements OnInit, OnChanges, AfterViewInit {
  ngOnInit(): void {
    console.log(1);

    setTimeout(() => {
      console.log(2);
    })
  }

  ngOnChanges(): void {
    console.log(3);
  }

  ngAfterViewInit(): void {
    console.log(4);

    Promise.resolve().then(() => {
      console.log(5);
    })
  }
}
```

Q2 - Angular challenge data flow

Q: What is the output of the console.log after 4 seconds app component renders?

```ts
// app.component.ts
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [ChildComponent],
  template: `
    <app-child [data]="data"></app-child>
  `,
  styleUrl: './app.component.css',
})
export class AppComponent implements OnInit {
  protected data: Data = {payload: {value: 'test'}}

  ngOnInit(): void {
    setTimeout(() => {
      console.log('data', this.data.payload.value);
    }, 3000);
  }
}
```

```ts
// child.component.ts
@Component({
  selector: 'app-child',
  standalone: true,
  templateUrl: './child.component.html',
  styleUrl: './child.component.css',
})
export class ChildComponent implements OnInit {
  @Input() data!: Data;

  ngOnInit(): void {
    setTimeout(() => {
      this.data = {
        ...this.data,
        payload:{
          ...this.data.payload,
          value: 'test2'
        }
      };
    }, 1000);
  }
}
```

Q3 - Angular challenge dependency injection

Q: What is the output of the console.log after app component renders?

```
// app.component.ts
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [ChildComponent],
  template: `
    <app-child></app-child>
  `,
  providers: [ServiceAService]
})
export class AppComponent {
  private readonly serviceAService = inject(ServiceAService);
}
```

```
// service-a.service.ts
@Injectable({
  providedIn: 'root'
})
export class ServiceAService {
  private value = 0;

  constructor() {
    console.log(++this.value);
  }
}
```

```
// child.component.ts
@Component({
  selector: 'app-child',
  standalone: true,
  providers: [ServiceAService]
})
export class AppComponent {
```

Q4 - Javascript challenge - type coercion

Q: After we run the following code, what is printed to the console?

```
const foo = (a: string) => +(+(+a + a) + +a);

console.log(foo('2'))
```

Q5 - Javascript challenge - closures

Q: After we run the following code, what is printed to the console?

```
const getCounter = (init: number) => {
  let count = init;
  count += 1;
  return () => (count += 1);
};

const incrementBy = getCounter(2);
console.log(incrementBy())
console.log(incrementBy())
```

Q6 - Javascript challenge - garbage collection

Q: Select true statement:

```
let obj = { a: { b: 1 } };
let ref = obj.a;
obj = null;
```

Q7 - Javascript challenge - event propagation

Q: What is the output of the console.log after clicking the button?

```
<body>
    <div id="wrapper">
        <button>Click me!</button>
    </div>

    <script>

        const wrapper = document.getElementById('wrapper');
        const button = document.getElementsByTagName('button')[0];

        wrapper.addEventListener("click", () => console.log("A"), true);
        wrapper.addEventListener("click", () => console.log("B"));
        button.addEventListener("click", () => console.log("C"), true);
        button.addEventListener("click", (e) => {
            console.log("D");
            e.stopPropagation();
            console.log("E");
        });
    </script>
</body>
```

Q8 - Javascript challenge - generator functions

Q: When does we get 6 printed to the console ?

```
function *myGenerator() {
    const x = yield;
    const y = yield 2;
    console.log(x + y);
}

const caller = myGenerator();
```

Q9 - Javascript challenge - Well, that escalated quickly

Q: What is the output of the console.log after we run the following code?

```
const arr = [1, 2, 3];
arr.length = 1;
console.log(arr);
```

Q10 - Last but not least - TypeScript challenge

Q: What is the output of the console.log after we run the following code?

```
const wrapper = {
    name: 'martin',
    foo: function() {
        return this.name;
    }
}

const wrapperClone = structuredClone(wrapper);

wrapperClone.name = 'peter';

console.log(wrapperClone.foo())
```

Thank you! 🎯

- Questions?
- Feedback?
- Let’s connect!