

02

337-029 | Week 49

DECEMBER
2020

Wednesday

Uses operator → ?

1) Js Optional Chaining

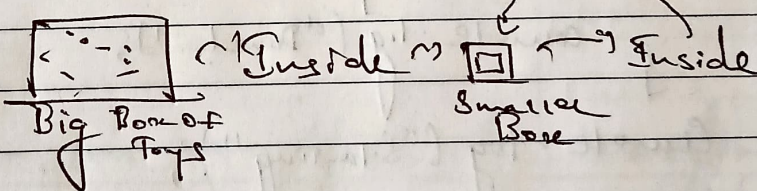
10) Having → Backup plan → When trying access

11) object you ← inside ← deep ← something
Array

(2) Helps → avoid errors → When something doesn't exist when you expect it to be.

1) Main Purpose → Does not throw any error if any specific property or item } not

3) Allows program to run even though even though we're not able to access that item. } prevented or can't be accessed.

5) Imagine →  → Inside7) Without optional chaining

```
<script>
```

```
let p = doc.querySelector("p");
```

```
const adventure = {
```

```
  name: "Alice",
```

```
  weapon: {
```

```
    type: "Sword",
```

```
    damage: 20
```

```
  },
```

```
  const totalGold = adventure.treasureChest.gold
```


DECEMBER
2020

Week 49 | 338-028

03

Thursday

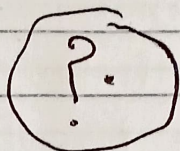
9 `p.innerHTML = "Total Gold: " + totalGold;
</script>`

10 %p : Error : Cannot read properties of undefined
Type ("reading gold")

11 `const totalGold = adventure.treasurechest.gold;`

12 Add this property in adventure

object → `{ treasurechest: {
gold: 1000,
jewels: 5
}}`



← Change to

2 `TotalGold : undefined`

The optional chaining uses `'?.'` operator

In the following example, `adventurer.treasureChest?.gold` is the optional chaining part. The `?.` allows JavaScript to safely check if `treasureChest` exists before trying to access `gold`. If `treasureChest` doesn't exist, it just returns `undefined` instead of throwing an error.

```
const adventurer = {  
  name: "Alice",  
  weapon: {  
    type: "sword",  
    damage: 20,  
  },  
  treasureChest: {  
    gold: 1000,  
    jewels: 5,  
  },  
};
```

Using optional chaining to
safely access the property

```
const totalGold = adventurer.treasureChest?.gold;  
console.log("Total Gold:", totalGold);  
// Output: Total Gold: 1000
```

