

Alternatives to if...else statements in JavaScript

Dynamic dispatch

Executes a version of a method based on the object's type.



```
const handleGreet = (animal) => { animal.greet() }
```

```
class Creature {  
  constructor (health = 100) {  
    this.health = health  
  }  
  
  greet() {  
    console.log('Hi there')  
  }  
}
```

```
class Dog extends Creature {  
  greet() {  
    console.log('Woof!')  
  }  
}
```

```
class Cat extends Creature {  
  greet() {  
    console.log('Meow!')  
  }  
}
```

```
const winston = new Dog()  
handleGreet(winston) // "Woof!"
```

executes the
method for the
object type

overrides
inherited method

Ternary operator

Checks a condition and executes the first expression if the condition is true, otherwise it runs the second expression.

condition



```
const isLoggedIn = false
```

```
const buttonText = isLoggedIn ? "Log Out" : "Log In"
```

```
console.log(buttonText) // "Log In"
```

returns this if the
condition is true

returns this if the
condition is false

Switch statement

Finds the matching value of an expression and executes the code block associated to the matching value.

The diagram illustrates the execution of a switch statement. It shows a code snippet with several annotations:

- variable/expression**: Points to the `day` variable in the `switch(day)` statement.
- checks if the variable/ expression matches this value...**: Points to the opening curly brace of the switch statement.
- ...runs this code block if it does**: Points to the code block for the `case "Friday":`.
- runs this if none of the values above match**: Points to the `default:` case.

```
const day = "Monday"
let greeting;

switch(day) {
  case "Saturday":
    greeting = "Enjoy your weekend!";
    break;
  case "Sunday":
    greeting = "Sunday Funday!";
    break;
  case "Friday":
    greeting = "Happy FriYAY!";
    break;
  default:
    greeting = "Let's smash those goals today!";
}

console.log(greeting) // "Let's smash those goals today!"
```


Jump/dispatch table

Stores value-function pairs in an object to quickly fetch and run a function based on a value (which is treated as an object key).

```
const run = () => { console.log("Run away!") }
const attack = () => { console.log("Go get 'em!") }
const specialAttack = () => { console.log("Finish them!") }
```

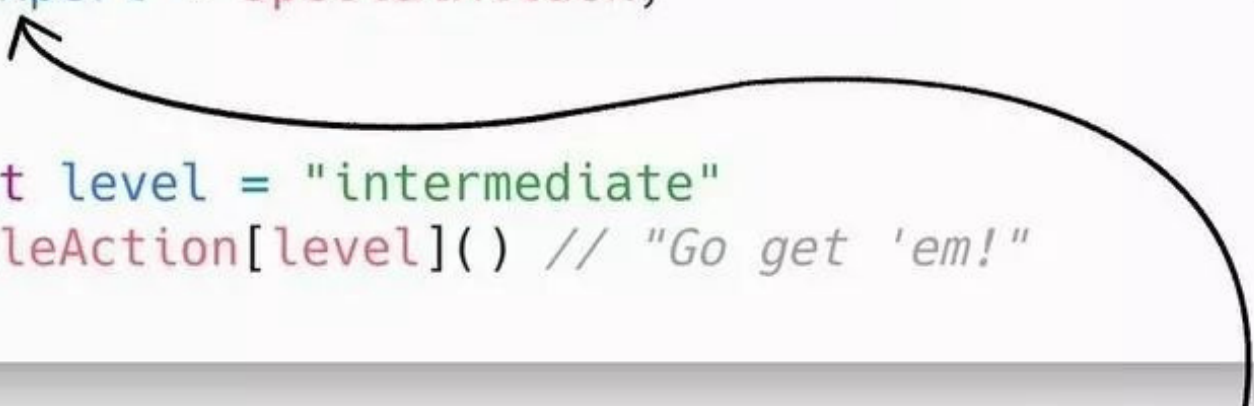
```
const handleAction = {
  "newbie": run,
  "intermediate": attack,
  "advanced": attack,
  "expert": specialAttack,
}
```

associates functions
to possible expression
values



```
const level = "intermediate"
handleAction[level]() // "Go get 'em!"
```

treat variable values
as object keys to
retrieve functions





Abhishek Mankuskar

Frontend Web & App Developer || Social Winter Of Code (SWOC) Founding
Organiser || DevRel 🥑 || Open Source Contributor & Community Builder |

Follow for more such content.

**Please Let me know what do
you think about this post.**



Next →