Design Patterns: Factory

Mitch Keenan

Design Patterns: Factory Method

1. What are design patterns?
2. What is the Factory Method pattern?
3. Examples
4. When should it be used?

Design Patterns

> A **design pattern** is a general repeatable solution to a commonly occurring problem in software design.

– Gang of Four

Factory Method

The intent of the Factory Method pattern is to...

> Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses

Which is a fancy way of saying, you can tell a factory which subclass you want constructed, and it will make it for you.

Factory Method in JavaScript - 1

```
class Bird {
  constructor(name) {
    this.name = name
  }

  fly() {
    console.log(`${this.name} is flying`);
  }
}
```

Factory Method in JavaScript - 2

```
class BirdOfPrey extends Bird {
  hunt() {
    console.log(`${this.name} is hunting`)
  }
}

class BirdOfParadise extends Bird {
  tap() {
    console.log(`${this.name} is making mana`)
  }
}
```

Factory Method in JavaScript - 3

```
class BigBird extends Bird {
  constructor(name, size) {
    super(name)
    this.size = size
  }

  teach() {
    console.log(`${this.name} is teaching`)
  }

  /* override */ fly() {
    if(this.size > 5) {
      throw new Error(`${this.name} is too big to fly :(`
    } else {
      super.fly()
    }
  }
}
```

Factory Method in JavaScript - 4

```
class BirdFactory {
  createBird(type, name) {
    let bird;
    if(type === "ofPrey") {
      bird = new BirdOfPrey(name);
    } else if (type === "ofParadise") {
      bird = new BirdOfParadise(name);
    } else if (type === "big") {
      bird = new BigBird(name, 6);
    }

    return bird;
  }
}
```

Factory Method in JavaScript - 5

```
const factory = new BirdFactory();

const predator = factory.createBird("ofPrey", "Owl")
predator.hunt() // Owl is hunting
predator.fly() // Owl is flying

const creature = factory.createBird("ofParadise", "Mana D
creature.tap() // Mana Dork is making mana
creature.fly() // Mana Dork is flying

const character = factory.createBird("big", "Big Bird")
character.teach() // Big Bird is teaching
character.fly() // Error: Big Bird is too big to fly :(
```

live demo

When should it be used?

When you...

- Want to abstract the construction of objects away from the caller
- Need the type of object created to be determined at run-time
- Want a clean and consistent interface for construction of many subclasses

Resources

Demos

- demo

Reading

- Aligator.io: Factory
- dofactory: Factory Method
- Design Patterns - Gang of Four (*warning*: **slow** pdf link)