marp: true

Design Patterns: Builder

Mitch Keenan

Design Patterns: Builder

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

Design Patterns

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

– Gang of Four

Examples:

- Singleton
- Observer
- **Builder**

Builder

The intent of the Builder is to...

> Separate the construction of a complex object from its representation so that the same construction process can create different representations.

Which is a fancy way of saying, a builder is a class which lets you use chained methods to build your target object. This also allows you to easily make a small change and build another.

When should Builder be used? (in JS)

When you...

- Have an object which could take many parameters in it's constructor
- Need multiple instances of said object, with some or all being slightly different

Builders in JavaScript - 1

```
class Bird {
        constructor(builder) {
        this.wingspan = builder.wingspan;
        this.habitat = builder.habitat;
        this.nestType = builder.nestType;
    }
}
```

Builders in JavaScript - 2

```
static class BirdBuilder {
        static withWingspan(wingspan) {
        this.wingspan = wignspan;
    }

    static withHabitat(habitat) {
        this.habitat = habitat;
    }

    static withNestType(nestType) {
        this.nestType = nestType;
    }

    static build() {
        return new Bird(this);
    }
}
```

Builders in JavaScript - 3

```
const builder = new BirdBuilder();

const woodpecker = builder
  .withWingspan(10)
  .withHabitat('forest')
  .withNestType('hole')
  .build()

console.log(woodpecker)
// { wingspan: 10, habitat: 'forest', nestType: 'hole' }

const parrot = builder
  .withWingspan(40)
  .withHabitat('ship')
  .build()

console.log(parrot)
// { wingspan: 40, habitat: 'ship', nestType: 'hole' }
```

demo

Builder Example

Rosie JS

jQuery

Builder Pros

1. Simplify complex construction
2. Reduces repetition in certain cases

Builder Cons

1. Extra overhead for every property
2. Not *as* useful in a highly mutable language with restricted access to `private`
3. Harder to ensure a fully initialized object

Resources

Demos

- demo
- Rosie JS

Reading

- Source Making: Builder
- Addy Osmani: Builder
- Design Patterns - Gang of Four (*warning*: **slow** pdf link)