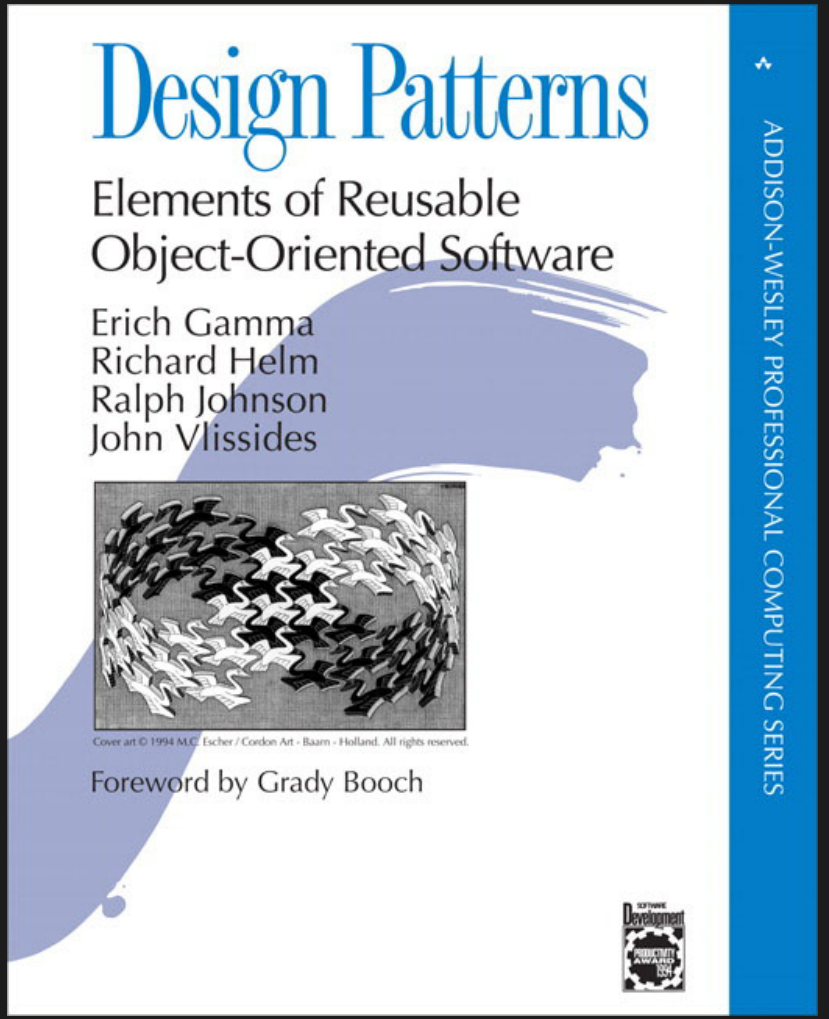
Contemporary Web Development
Lesson 3

https://www.youtube.com/channel/UCafxR2HWJRmMfSdyZXvZMTw

Missed from yesterday

Useful Design Patterns in Javascript ES6

GoF


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

But first : {Scoping}

Rules

- *var :* Function scope – Don't ever use it.
- *let:* Block scope ({}).
- *const:* Block scope and value cannot change (but inner properties or items can).

Example

Classes


Tamagotchi
Friends

When a new tamagotchi is born...

tamgotchi.js

```
export default class Tamagotchi {
        constructor(name) {
                this.name = name;
                this.food = 0;
                this.poop = 0;
        }
        eat() {
                if (this.food > 0) {
                        this.food -= 1;
                        this.poop += 1;
                }
        }
       clean() {
                this.poop = 0;
        }
}
```

index.js

```
import Tamagotchi from ‘./tamagotchi’

const yukino = new Tamagotchi("Yukino");
const arima = new Tamagotchi("Arima");

yukino.eat();
arima.eat();
```

A note about *this*

```
export default class Tamagotchi {
    constructor(name) {
    }
    clean() {
        // Cleaning takes 5 seconds
        setTimeout(function resetPoop() {
            // This will fail
            this.poop = 0;
        },5000);
    }
}
```

```
export default class Tamagotchi {
    constructor(name) {
    }
    clean() {
        // Cleaning takes 5 seconds
        setTimeout(() => {
            // This will work
            this.poop = 0;
        },5000);
    }
}
```

Observer
(publish/subscribe)

index.js

```
yukino.on(‘poop-alert’, (tamagotchi) => {
        console.log("Alert! There is too much poop (" + tamagotchi.poop + ").");
});
```

---

tamagotchi.js

```
if (this.poop >= 3) {
        this.trigger(‘poop-alert’, this);
}
```

Redundancy?

Composition / Mixin

./observable-mixin.js

```
export default function(target) {
  Object.defineProperties(target, {
    subscribers: {
      value : {}
    },
    on: {
      value: function(event,callback) {
            ...
      }
    },
    off: {
      value: function(event,callback) {
            ...
      }
    },
    trigger: {
      ....
```

./tamgotchi.js

```
export default class Tamagotchi {
  constructor(name) {
    observable(this);
    }
```

Dependency Injection

Sound of poop – class or module?

./sound-manager.js

```
const audioCotext = window.webkitAudioContext();

export function initGain(volume) {
        let gainNode. ...
}
export function makeNoiseFor(seconds) {

 }
```

./sound-manager.js

```
export default class SoundManager {
   constructor() {
   }
   init(volume) {
        this.audioContext = new window.webkitAudioContext();
        let gainNode. ...
   }
   makeNoiseFor(seconds) {
   }
}
```

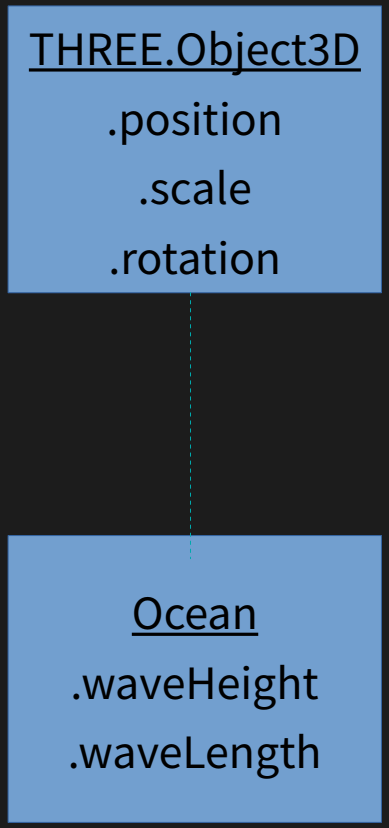
./tamagotchi.js

```
import {makeNoiseFor, initGain} from ./sound-manager
initGain(20);
makeNoiseFor(2);
```

./index.js

```
import SoundManager from ‘./sound-manager’
const soundManager = new SoundManager();
soundManager.init(20);
const yukino = new Tamagotchi(soundManager);
```

Inheritance




```
export default class Ocean extends THREE.Object3D  {
  constructor(config) {
    super();

    this.waveFrequencey = 0.07;
    this.waveHeight = 0.5;
    this.waveLength = 0.3;
}
```

Index.js

```
const ocean = new Ocean();
ocean.waveHeight = 1.0;
camera.focusOn(ocean).
```

camera.js

```
function focusOn(object3d) {
      … // Expects THREE.Object3D
      ….object3d.position
}
```

Exercise
Stalker app
in Pseudo code

1) You are given the module Instacrawl
    a) Use is by writing import Instacrawl from 'instacrawl'
2) It's an observable that triggers 'new-post' every time there is a new post on your instagram feed.
3) The argument it sends in the trigger is a "post" object containing the fields:
    a) "author" - name string
    b) "likes" – a list of likes that you can iterate using "for each like in likes"
4) Each like object in the list has "name" – the name of the user who liked the post.

1) Implement the class "InstaStalk" in pseudo code.
2) In its constructor, you will provide :
    a) The instaCrawl module.
    b) The user name of the stalkee.
3) It has a run() method that starts the stalking.
4) The instaStalk class is also an observable (has 'on', 'trigger' methods).
5) It uses InstaCrawl so that every time that a post was liked by the stalkee, it triggers the "stalk-like" message, as an argument it provides the name of the original author of the post.

1) Implement the module "stalker".
2) It imports Instacrawl and constructs Instastalk.
3) It listens to "stalk-like" events, and sums up the number of times an author was liked by the stalkee using a hash.
    a) For example hash [key] = 0;
4) If any author is liked more than 3 times, it prints to the console.