Review

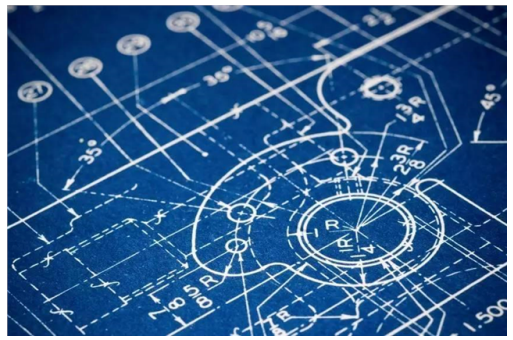
Software Studio

DataLab, CS, NTHU

2017 spring

What did we learn so far?

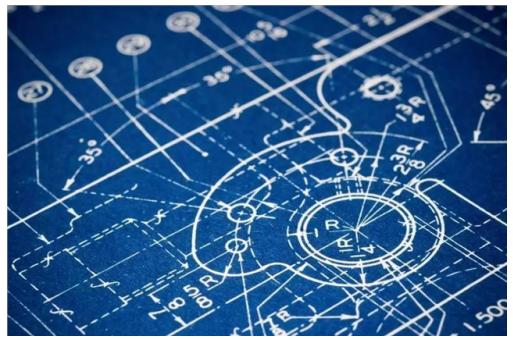
Object Oriented Programming



Class

```
class Employee {
  constructor(name) {
    this.name = name;
    this.salary = 1000;
  }
  addSalary(amt) {
    this.salary += amt;
  }
}
```

Object Oriented Programming



Class

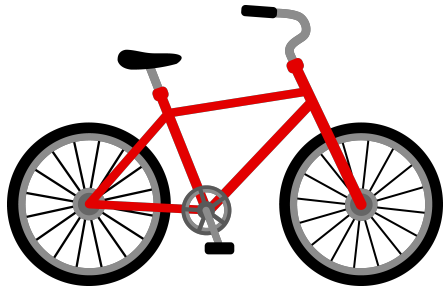
new

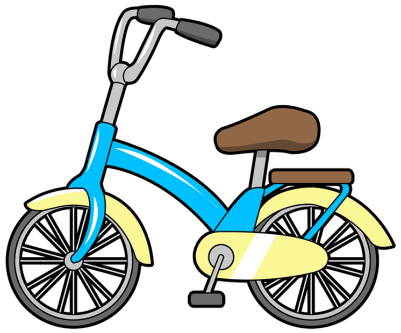
Object

Object.method()

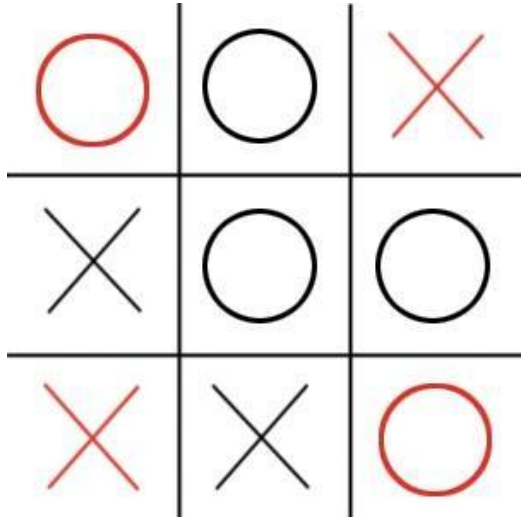
Object.property

Inheritance (繼承)

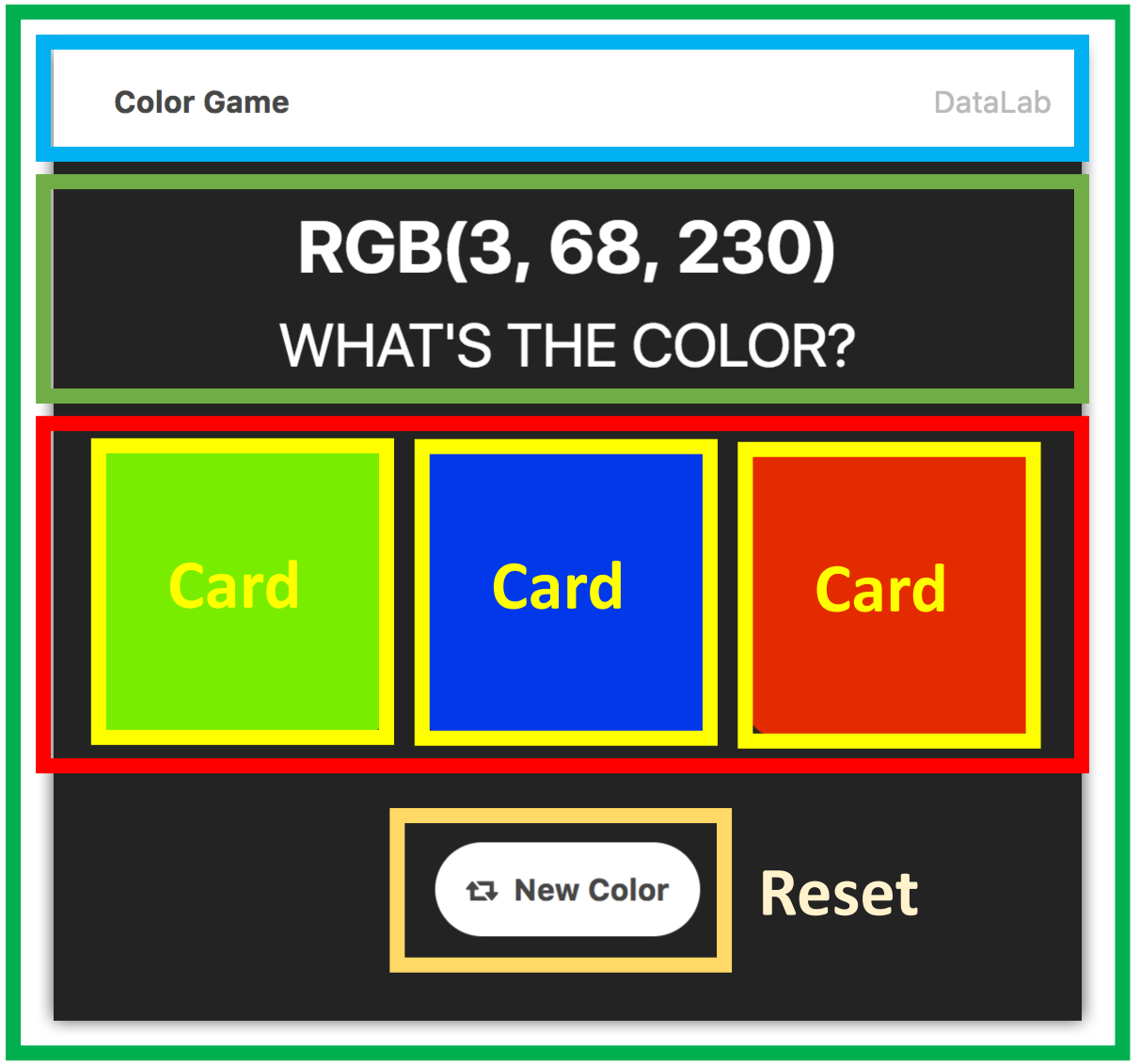


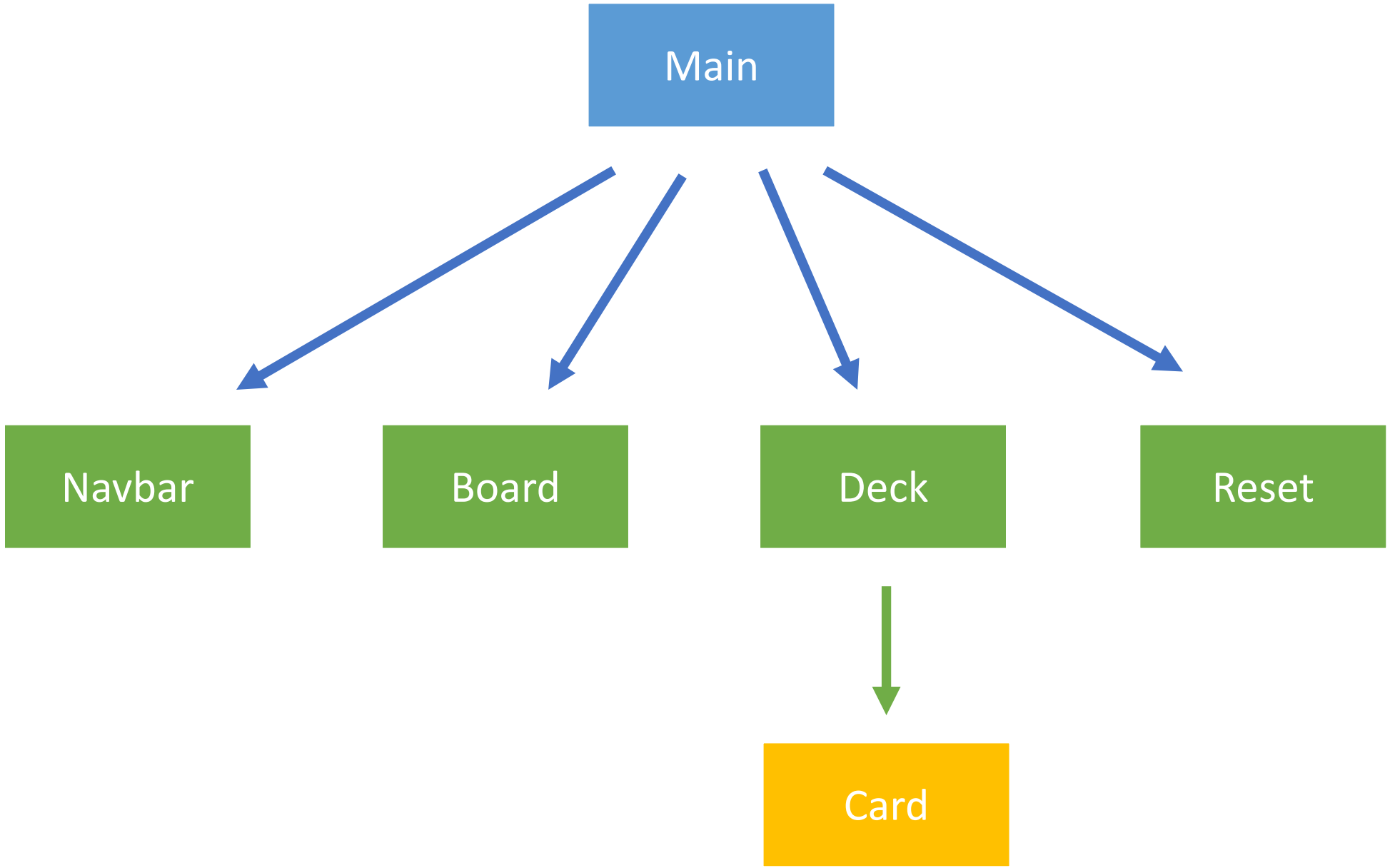


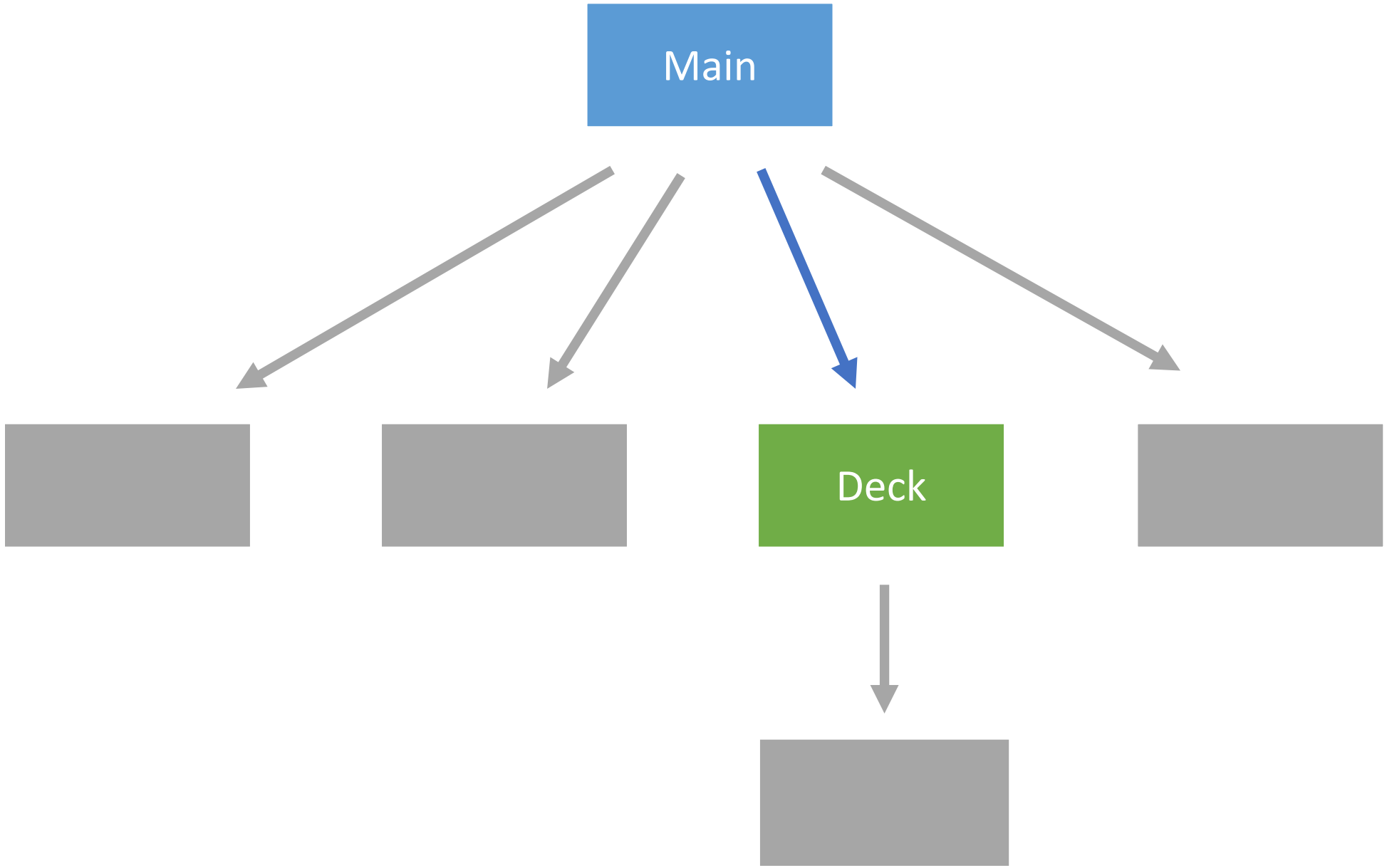
Case study : Tic-Tac-Toe



Color Game component

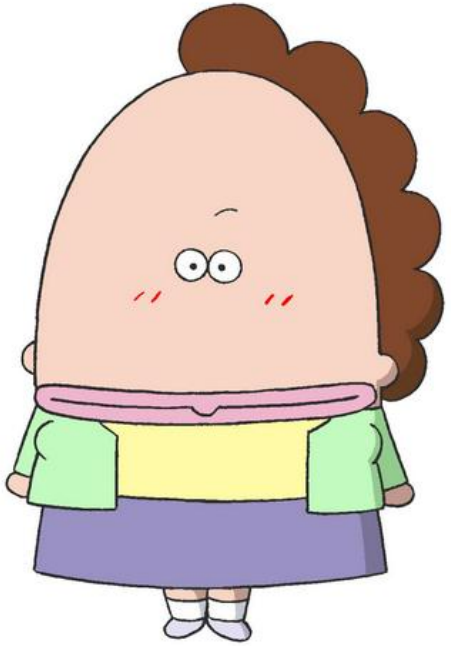



Main
Navbar
Board
Deck
Reset
Card


Main
Deck

rightClick

Deck



wrongClick

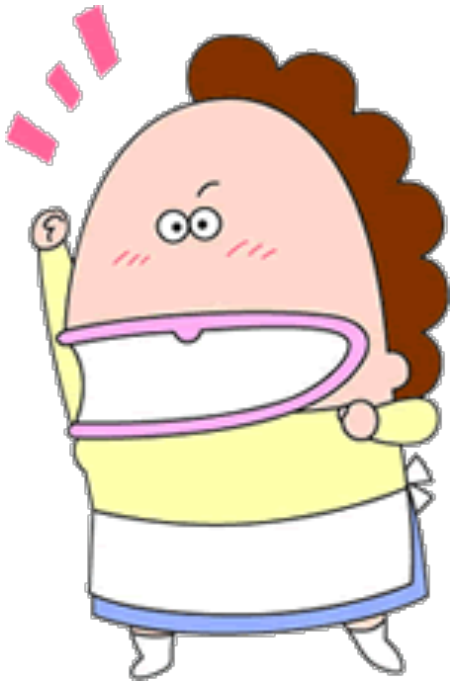
rightClick

Main

rightClick

Deck

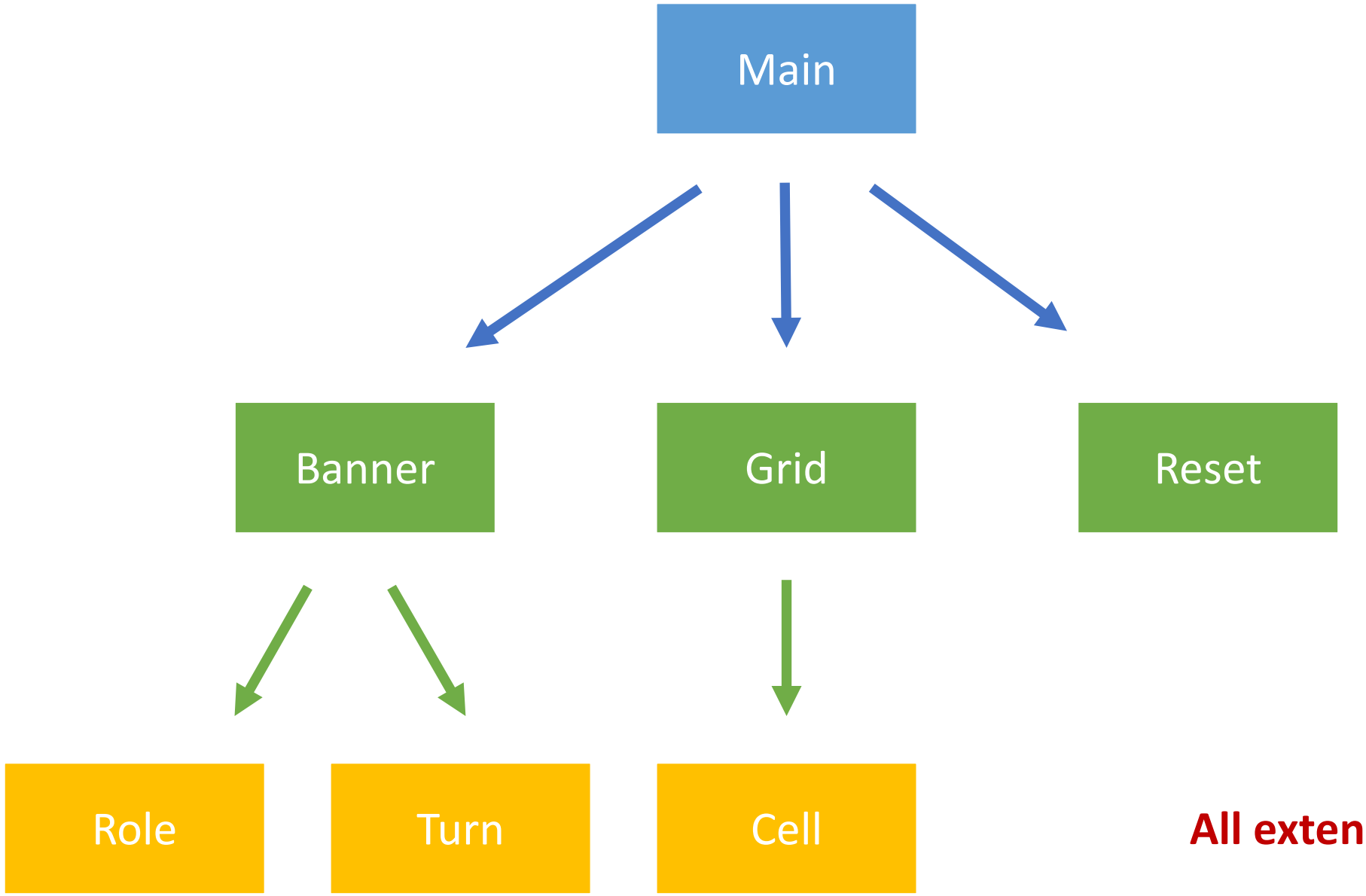
rightClick



Main

```
export default class Main extends Component {

    constructor(root) {
        super(root);

        this.navbar = new Navbar(root.querySelector('.navbar'));

        this.deck = new Deck(root.querySelector('.deck'));
        this.deck.on('wrongClick', this.handleDeckWrongClick.bind(this));
        this.deck.on('rightClick', this.handleDeckRightClick.bind(this));

        this.board = new Board(root.querySelector('.board'), this.deck.getPickedColor());

        this.reset = new Reset(root.querySelector('.reset'));
        this.reset.on('resetClick', this.handleRestClick.bind(this));
    }

    handleDeckWrongClick(firer) {
        this.board.showWrongMessage();
    }

    handleDeckRightClick(firer, pickedColor) {
        this.root.style.backgroundColor = pickedColor;
        this.board.showCorrectMessage();
        this.reset.showPlayAgain();
    }
```

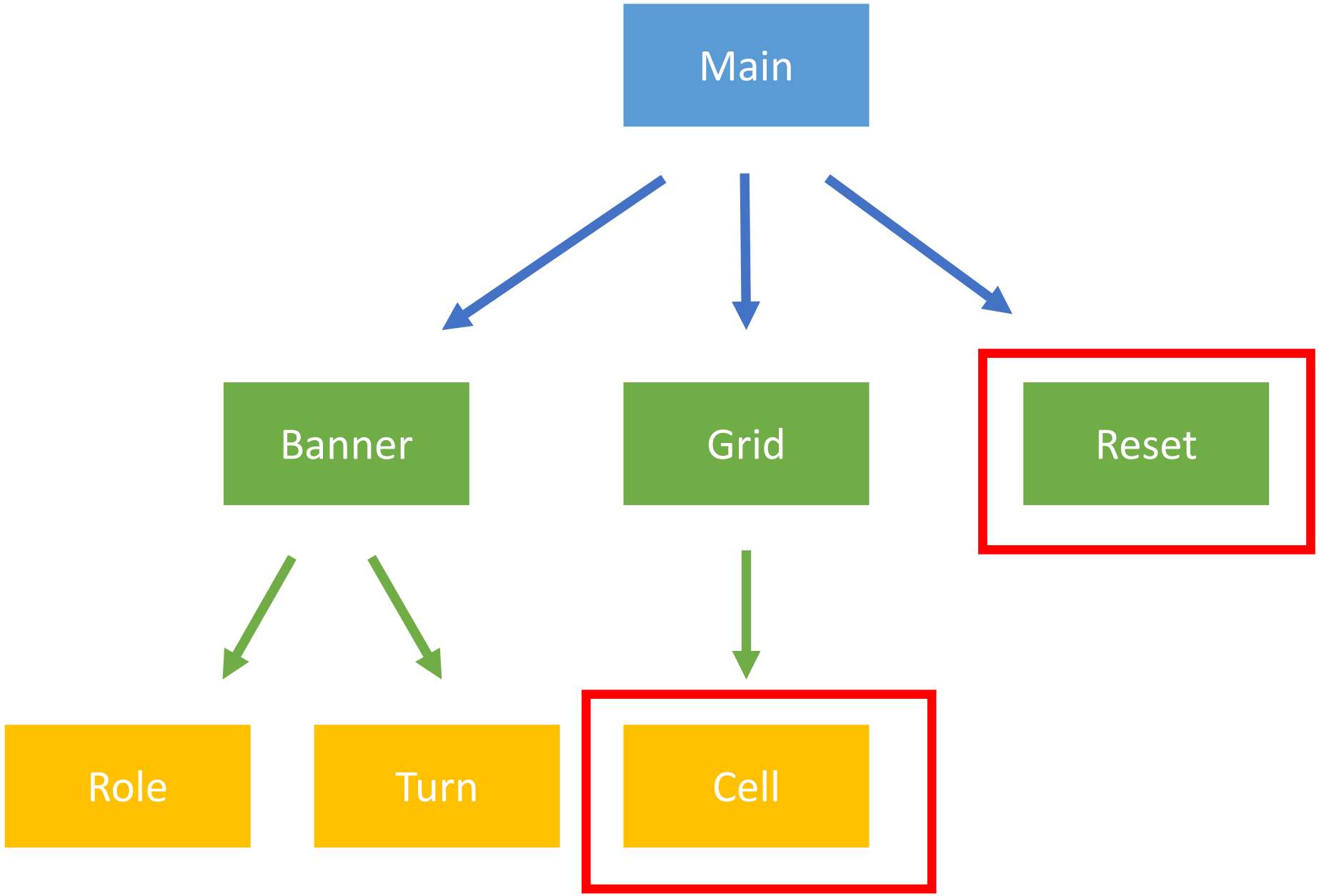
```
export default class Deck extends Component {

    constructor(root) {
        super(root);

        this.cards = [];
        const els = root.querySelectorAll(Card.getRootClass());

        for (let el of els) {

            const card = new Card(el);
            card.on('cardClick', this.handleCardClick.bind(this));

            this.cards.push(card);
        }
    }

    handleCardClick(firer, color) {
        if (this.gameOver)
            return;

        if (color === this.pickedColor) {
            // do something
            this.fire('rightClick', this.pickedColor);
        } else {
            // do something
            this.fire('wrongClick');
        }
    }
```



All extends component.js

Some hint

- Build the architecture first.
- Start from small components which are independent and can be interacted with user.


Main
Banner
Grid
Reset
Role
Turn
Cell

Excellent work

FAQ

- bind(this)
- How to debug?
- What happened about On() / Fire() ?

.bind(this)

```
export default class Main extends Component {
    static getRootClass() {
        return '.main';
    }

    constructor(root) {
        super(root);

        this.navbar = new Navbar(root.querySelector('.navbar'));

        this.deck = new Deck(root.querySelector('.deck'));
        this.deck.on('wrongClick', this.handleDeckWrongClick);
        this.deck.on('rightClick', this.handleDeckRightClick);

        this.board = new Board(root.querySelector('.board'), this.deck.getPickedColor());

        this.reset = new Reset(root.querySelector('.reset'));
        this.reset.on('click', this.handleRestClick.bind(this));
    }
```

Who call the function ? window!

.bind(this)

- “this” always binds to “**current owner**”.
- So remember to use .bind(this) in class.

.bind(this)

```
export default class Main extends Component {

    constructor(root) {
        super(root);

        this.navbar = new Navbar(root.querySelector('.navbar'));

        this.deck = new Deck(root.querySelector('.deck'));
        this.deck.on('wrongClick', this.handleDeckWrongClick.bind(this));
        this.deck.on('rightClick', this.handleDeckRightClick.bind(this));

        this.board = new Board(root.querySelector('.board'), this.deck.getPickedColor());

        this.reset = new Reset(root.querySelector('.reset'));
        this.reset.on('resetClick', this.handleRestClick.bind(this));
    }
```

Who call the function ? Main!

Console.log 大法

What happened when I call on() / fire()

```
export default class Component {
    /*
     * Override this method
     */
    static getRootClass() {
        return '.component';
    }

    constructor(root) {
        this.root = root;
        this.handlers  = {};
    }

    on(event, handler) {
        this.handlers[event] = handler;
    }

    fire(event, ...args) {
        this.handlers[event](this, ...args);
    }
}
```