

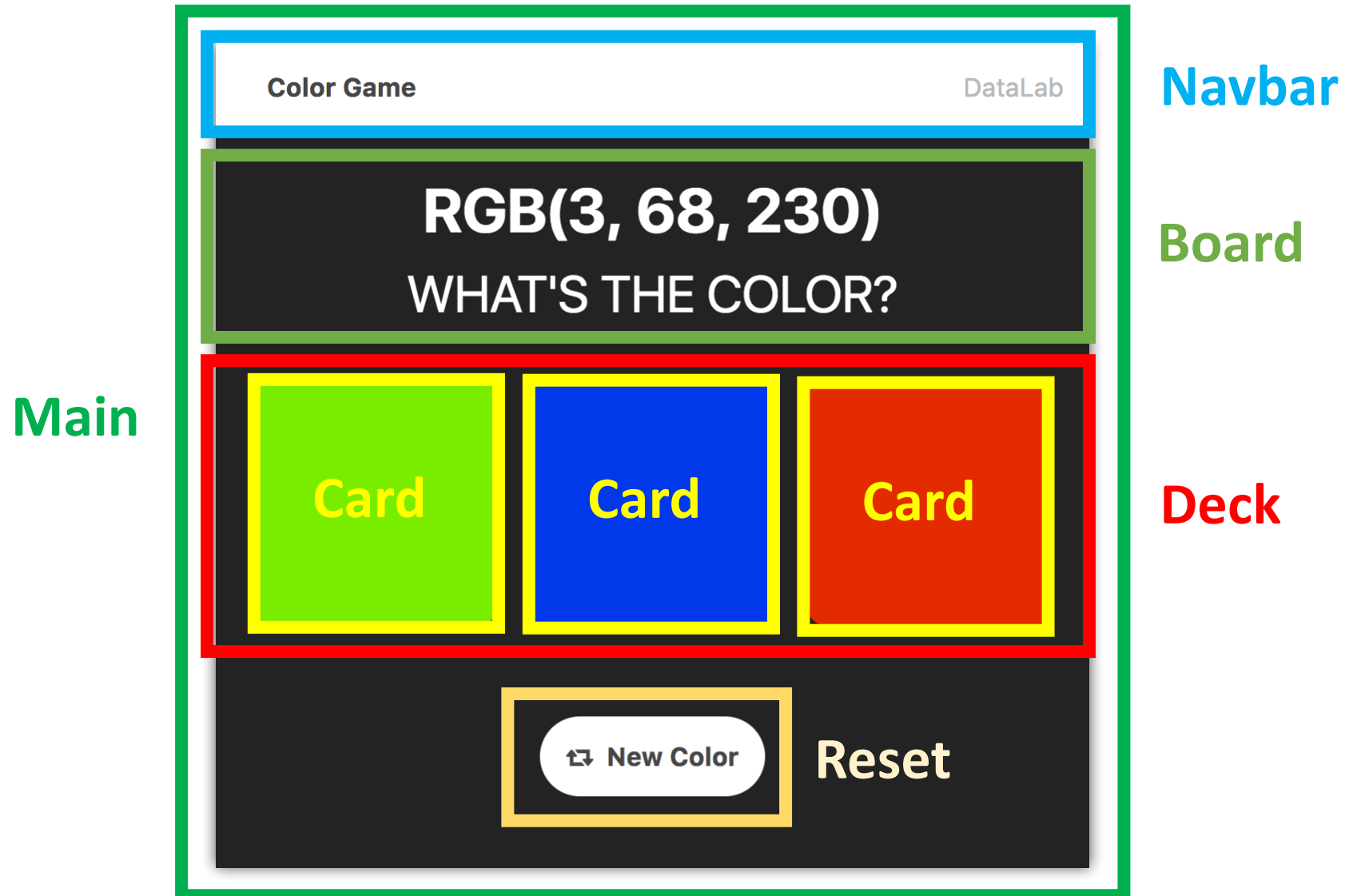
# Component based Game

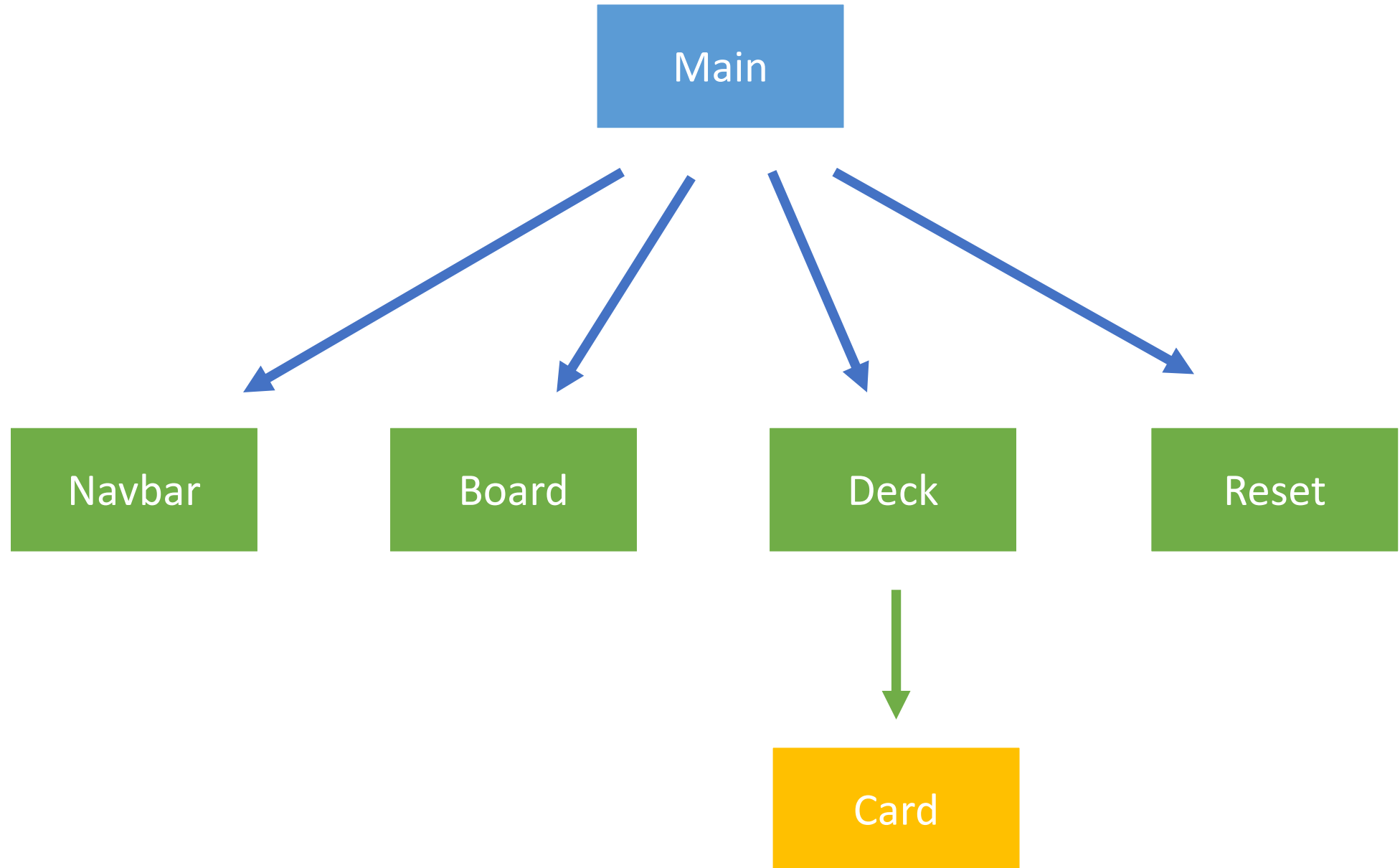
Software Studio

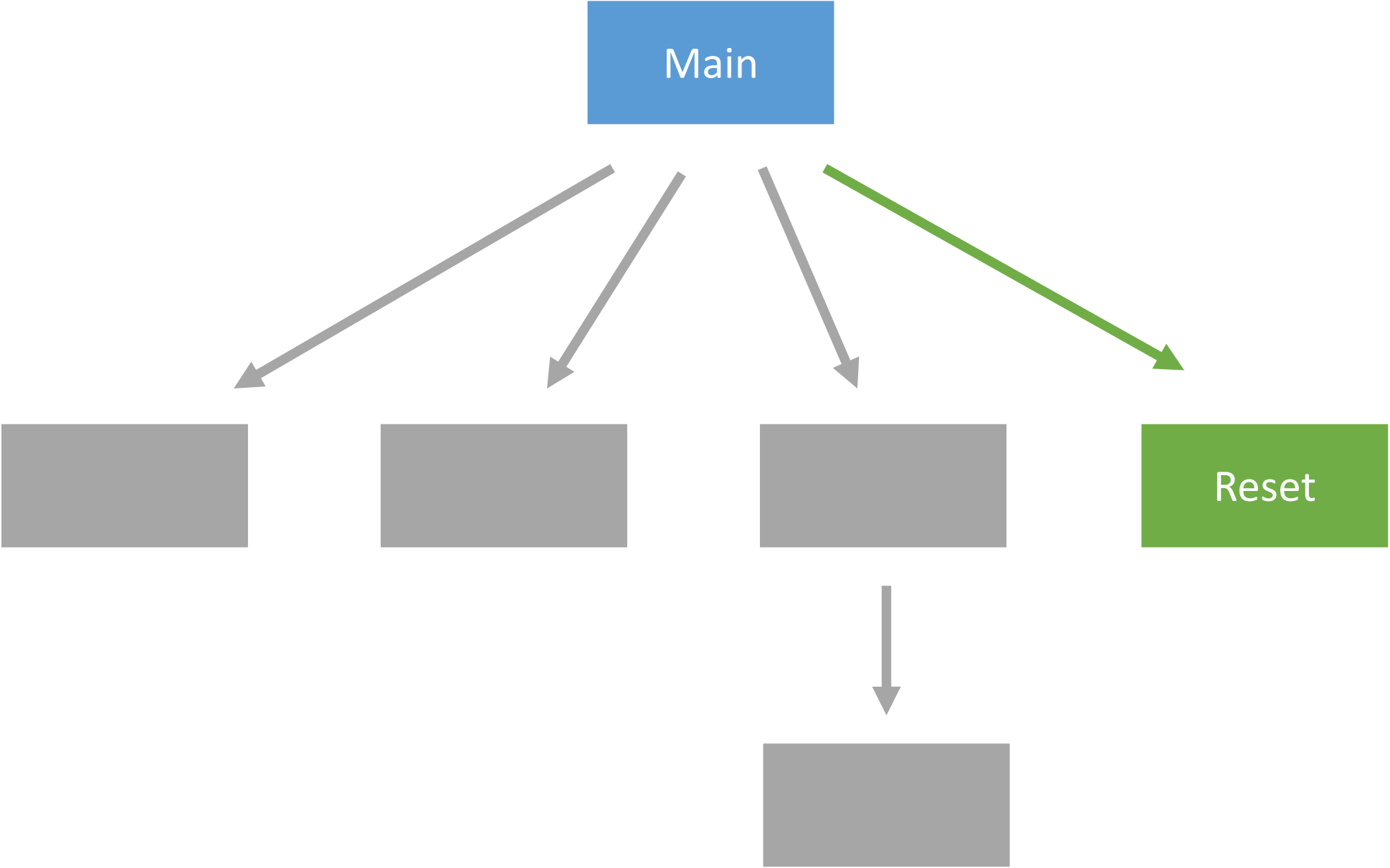
DataLab, CS, NTHU

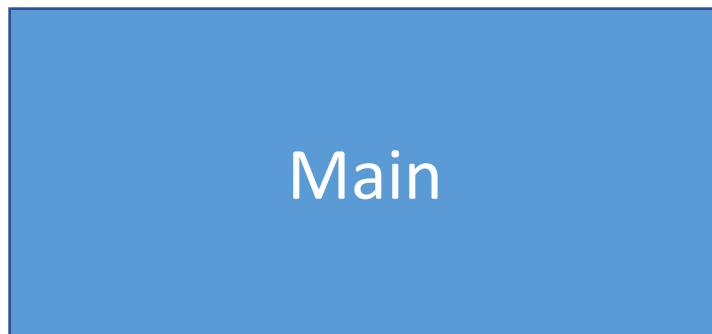
2017 spring

# Color Game component

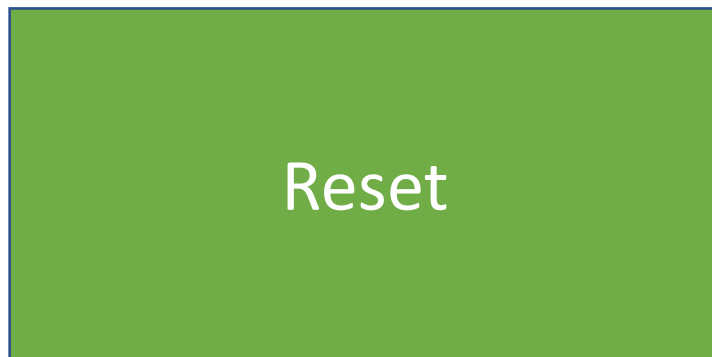








**Fire event**



```
export default class Reset extends Component {  
  constructor(root) {  
    super(root);  
    root.addEventListener("click", this.handleClick.bind(this));  
  }  
  handleClick(e) {  
    this.fire('resetClick');  
  }  
}
```

```
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.handleResetClick.bind(this));  
  }  
  
  handleResetClick() {  
    this.root.style.backgroundColor = "#232323";  
  
    this.deck.reset();  
    this.board.reset(this.deck.getPickedColor());  
    this.reset.reset();  
  }  
}
```

Main

`reset.method()`

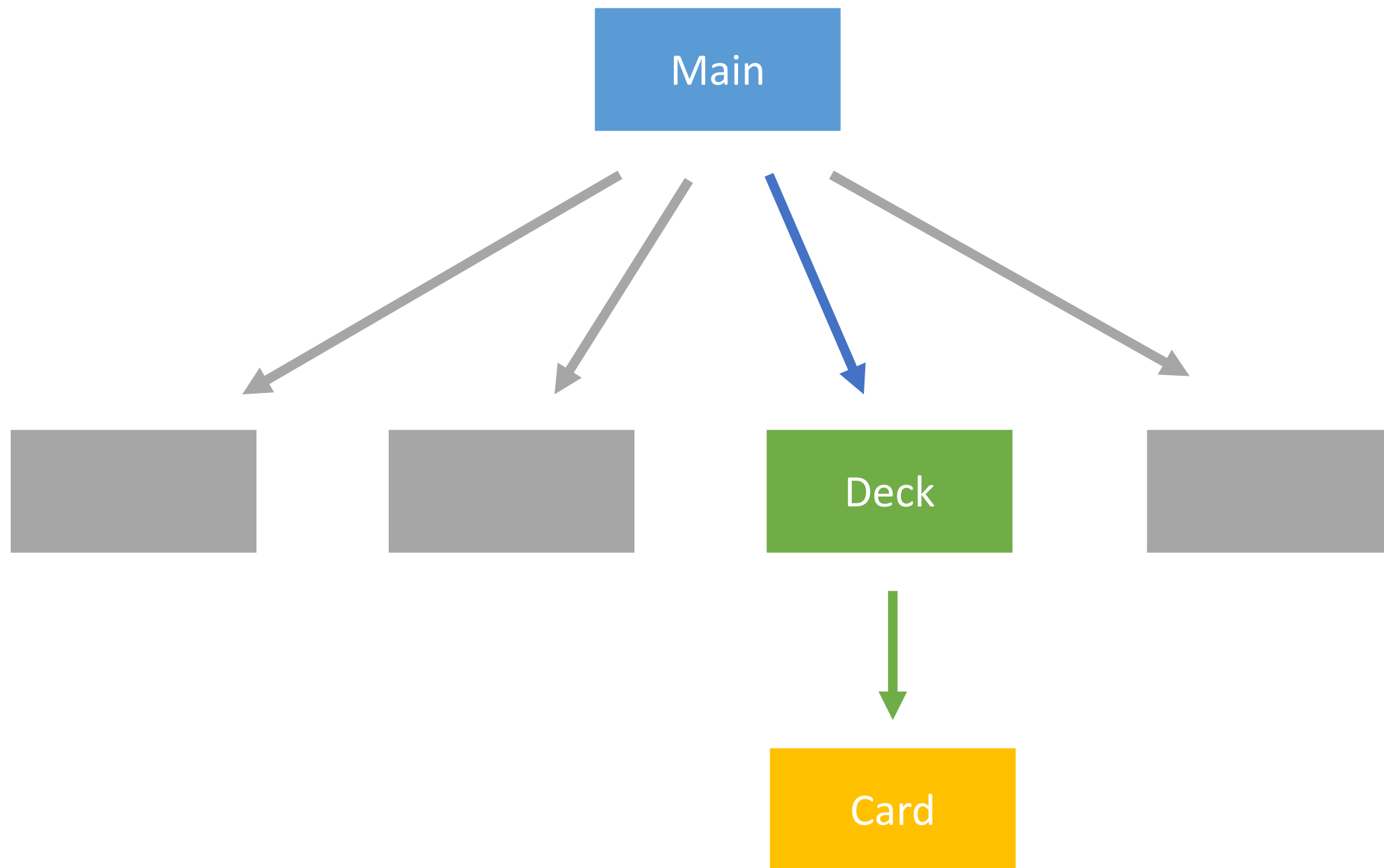


Reset



```
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.handleResetClick.bind(this));  
  }  
  
  handleResetClick() {  
    this.root.style.backgroundColor = "#232323";  
  
    this.deck.reset();  
    this.board.reset(this.deck.getPickedColor());  
    this.reset.reset();  
  }  
}
```

How about click card?



```
export default class Card extends Component {  
  constructor(root) {  
    super(root);  
    root.addEventListener("click", this.handleClick.bind(this));  
  }  
  handleClick(e) {  
    this.fire('cardClick', this.color);  
  }  
}
```

```
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.handleClick.bind(this));  
      this.cards.push(card);  
    }  
  }  
  
  handleClick(firer, color) {  
    if (this.gameOver)  
      return;  
  
    if (color === this.pickedColor) {  
      // do something  
      this.fire('rightClick', this.pickedColor);  
    } else {  
      // do something  
      this.fire('wrongClick');  
    }  
  }  
}
```

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

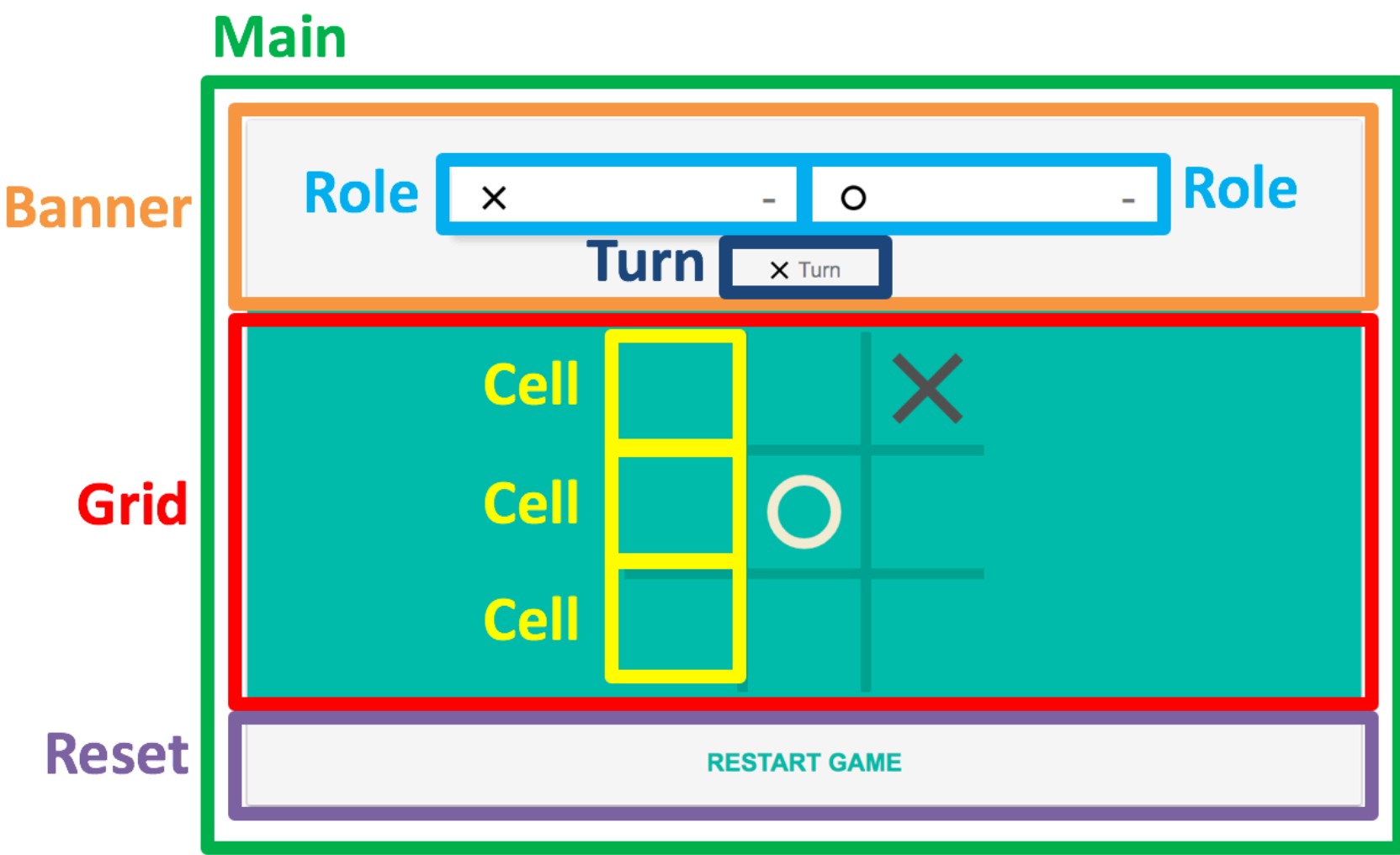
```
export default class Main extends Component {  
  constructor(root) {  
    super(root);  
  
    this.navbar = new Navbar(root.querySelector('.navbar'));  
  
    this.deck = new Deck(root.querySelector('.deck'));  
    this.deck.on('wrongClick', this.handleDeckWrongClick.bind(this));  
    this.deck.on('rightClick', 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.handleResetClick.bind(this));  
  }  
}
```

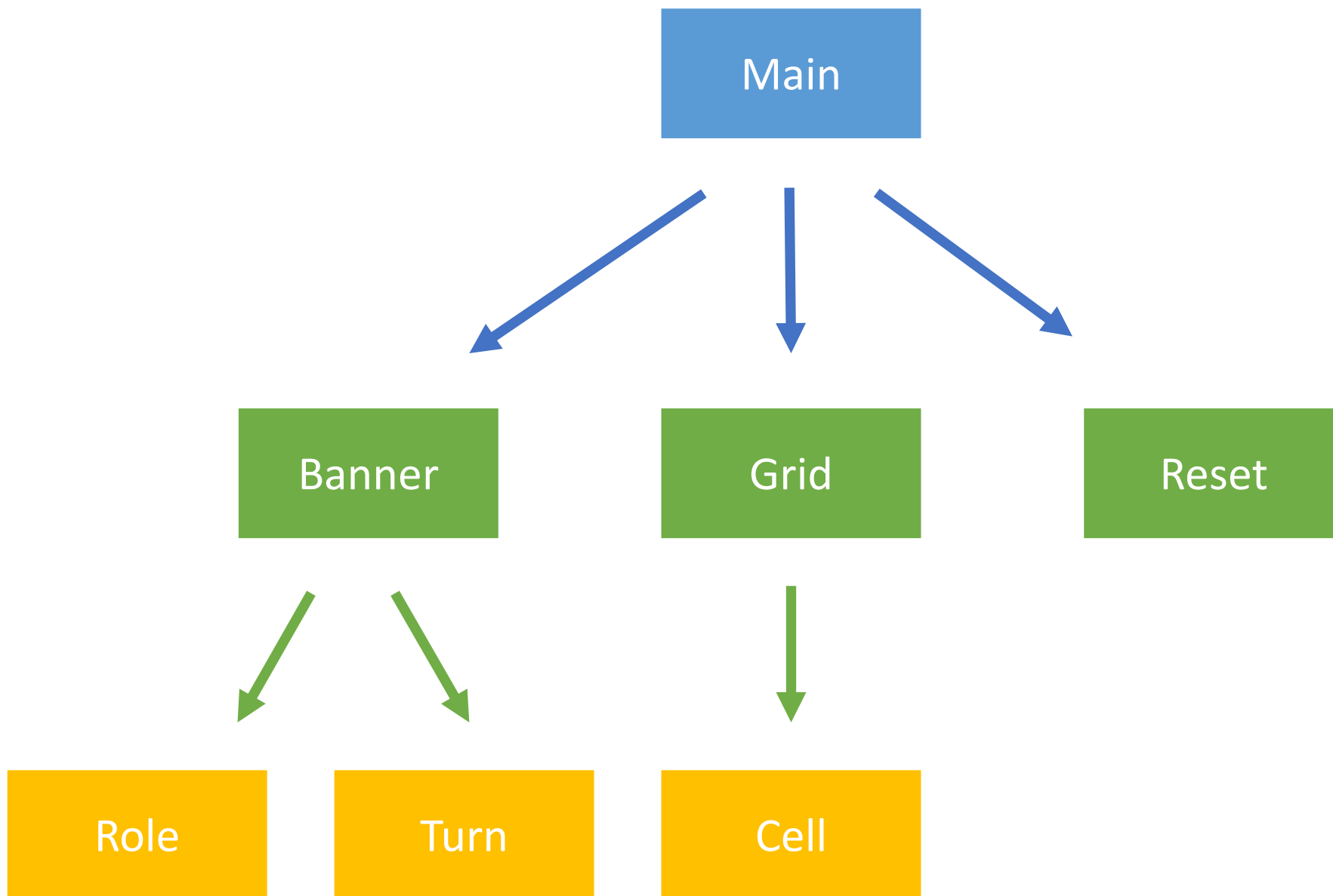
```
handleDeckWrongClick(firer) {  
  this.board.showWrongMessage();  
}  
  
handleDeckRightClick(firer, pickedColor) {  
  this.root.style.backgroundColor = pickedColor;  
  this.board.showCorrectMessage();  
  this.reset.showPlayAgain();  
}
```

```
export default class Board extends Component {  
  static getRootClass() {  
    return '.board';  
  }  
  
  constructor(root, color) {  
    super(root);  
  
    this.colorDisplay = root.querySelector('.color-picked');  
    this.messageDisplay = root.querySelector('.message');  
    this.reset(color);  
  }  
  
  showWrongMessage() {  
    this.messageDisplay.textContent = "Try Again";  
  }  
}
```



# Tic-Tac-Toe





# Hint (1/3)

- In the Cell component you may have following functions :
  - *constructor*
    - maybe you need to add an event listener here
  - *reset*
    - when finish game or reset button clicked you need to reset cell
  - *handleDomClick*
    - when cell is clicked maybe you need to fire up
  - *occupyCell*
    - maybe you need to change the cell content when the cell is clicked
  - *isOccupied*
    - maybe you need to judge whether the cell is occupied
  - *isMatch*
    - maybe you need to judge whether the cell is occupied by O or X

# Hint (2/3)

- In the Grid Component maybe you need to do following tasks :
  - Reset all cell
  - Handle cell click
  - Check whether the game is over (someone win / draw)

## Hint (3/3)

- You don't need to follow our template.
- You don't need any css decoration.
- You can use `<table></table>` to present container of cell.
- Be sure you have all component .js file.
- **You can reuse the code of component based Color Game.**
- If you want to use `Object.on()` / `Object.fire()`, make sure you extend *Component*.
- Good Luck!

離開前請簽名！！！！