建立通用開發模式，一個打十個！

王証頡 @sorosora

- Fontech 網頁前端
- React with Next.js and styled-components

大綱

- 為何選擇框架
  - 需求
- 開發問題與解法
  - 切版
  - 資料注入
  - 檔案架構
  - 模組共用
  - Template

為何選擇框架

需求

- 前後端分離
- 客製化網站
  - SEO
  - 靜態網站
  - 動態網站
  - 無障礙網站

開發問題與解法

切版

- 避免巢狀地獄
- 定義相對關係

巣状地獄

```
.city {
  .road {
    .wrapper {
      .container {
        .cat {
          p {
            color: cornflowerblue;
          }
        }
      }
    }
  }
}
```

```
<div className='city'>
  <div className='road'>
    <div className='wrapper'>
      <div className='container'>
        <div className='cat'>
          <p>市井小猫</p>
        </div>
      </div>
    </div>
  </div>
</div>
```

避免巢狀地獄

```
.cat {
  .name {
    color: cornflowerblue;
  }
}

/* .city, .road, ...其他style */
```

```
<div className='city'>
  <div className='road'>
    <div className='wrapper'>
      <div className='container'>
        <div className='cat'>
          <p className='name'>市井小貓</p>
        </div>
      </div>
    </div>
  </div>
</div>
```

避免巢狀地獄

```
const Name = styled.div`
  color: cornflowerblue;
`;

// 宣告 City, Road, ...
```

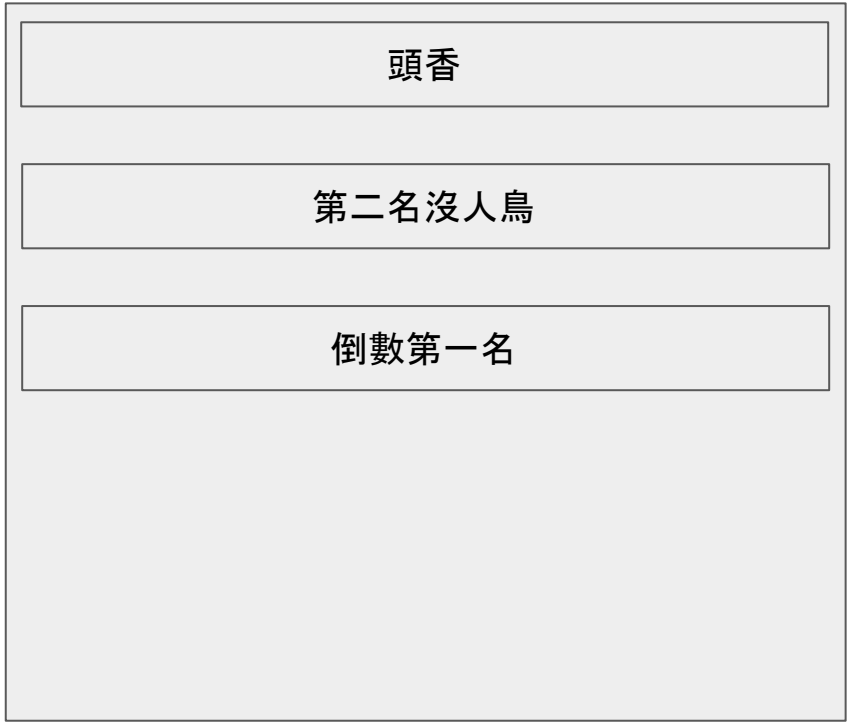
```
<City>
  <Road>
    <Wrapper>
      <Container>
        <Cat>
          <Name>市井小貓</Name>
        </div>
      </div>
    </div>
  </div>
</div>
```

相對關係

```
const Item = styled.div`
  color: black;
  margin-bottom: 20px;
`;

const Container = styled.div`
  display: flex;
  flex-direction: column;
`;

const Line = () => (
  <Container>
    <Item>頭香</Item>
    <Item>第二名沒人鳥</Item>
    <Item>倒數第一名</Item>
  </Container>
);
```



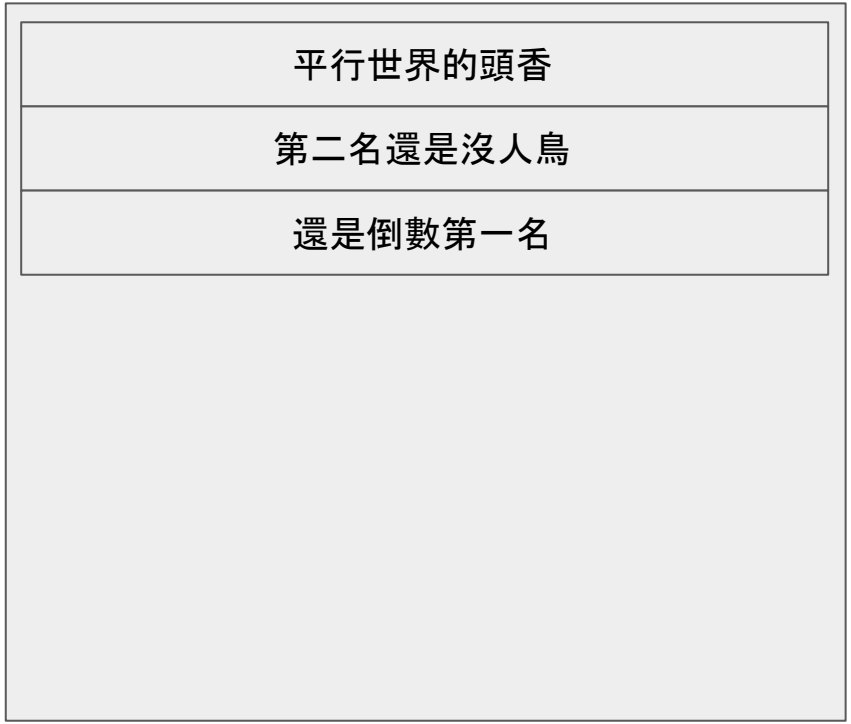
相對關係

```
import Item from 'components/Item';

const Container = styled.div`
  display: flex;
  flex-direction: column;

  ${Item} {
    margin-bottom: 20px;
  }
`;

const AnotherLine = () => (
  <Container>
    <Item>頭平行世界的頭香</Item>
    <Item>第二名還是沒人鳥</Item>
    <Item>還是倒數第一名</Item>
  </Container>
);
```



資料注入

- 資料從元件抽離
- 多國語言
- Fetch data

資料從元件抽離

```
// 空的啦
```

```
const CatWorld = () => (
  <Container>
    <Cat>
      <Name>貓1</Name>
      <Color>橘子</Color>
    </Cat>
    <Cat>
      <Name>貓2</Name>
      <Color>牛奶</Color>
    </Cat>
    <Cat>
      <Name>阿花</Name>
      <Color>玳瑁</Color>
    </Cat>
  </Container>
);
```

資料從元件抽離

```
const cats = [
 {
   name: '貓1',
   color: '橘子',
 },
 {
   name: '貓2',
   color: '牛奶',
 },
 {
   name: '阿花',
   color: '玳瑁',
 },
];

// render return
<CatWorld cats={cats} />
```

```
const CatWorld = (props) => {
  const { cats } = props;
  return (
    <Container>
      {
        cats.map(cat => (
          <Cat>
            <Name>{cat.name}</Name>
            <Color>{cat.color}</Color>
          </Cat>
        ))
      }
    </Container>
  );
};
```

多國語言

```
// t is a function for translation
const cats = [...Array(3)].map((value, index) =>
({
 name: t(`cat.${index}.name`),
 color: t(`cat.${index}.color`),
}));

// render return
<CatWorld cats={cats} />
```

```
const CatWorld = (props) => {
  const { cats } = props;
  return (
    <Container>
      {
        cats.map(cat => (
          <Cat>
            <Name>{cat.name}</Name>
            <Color>{cat.color}</Color>
          </Cat>
        ))
      }
    </Container>
  );
};
```

Fetch data

```
const useCats = () => {
  const [cats, setCats] = useState([]);
  useEffect(() => {
    // 如同在componentDidMount fetch 資料
    const getCatsPromise = fetch('https://ha.ha.ha/cats');
    getCatsPromise.then(response => setCats(response));
  }, []);
  return {
    cats,
  };
};


// render return
<CatWorld />
```

```
const CatWorld = () => {
  const { cats } = useCats();
  return (
    <Container>
      {
        cats.map(cat => (
          <Cat>
            <Name>{cat.name}</Name>
            <Color>{cat.color}</Color>
          </Cat>
        ))
      }
    </Container>
  );
};
```

檔案架構

- 元件
- 頁面

元件

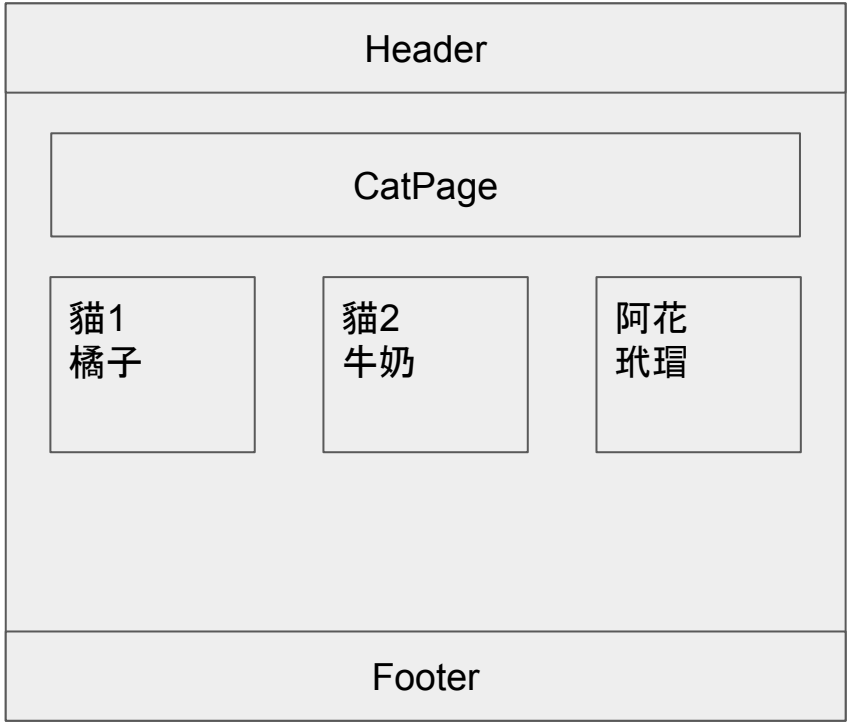
```
└ src
   ├ components
   │  ├ Header
   │  │  └ index.js
   │  └ Footer
   │     └ index.js
   │
   └ containers
      └ IndexPage
         └ index.js
```

元件

```
└ src
   ├ components
   │  ├ Header
   │  │  └ index.js
   │  └ Footer
   │     └ index.js
   │
   └ containers
      └ IndexPage
         └ CatWorld.js
         └ index.js
```

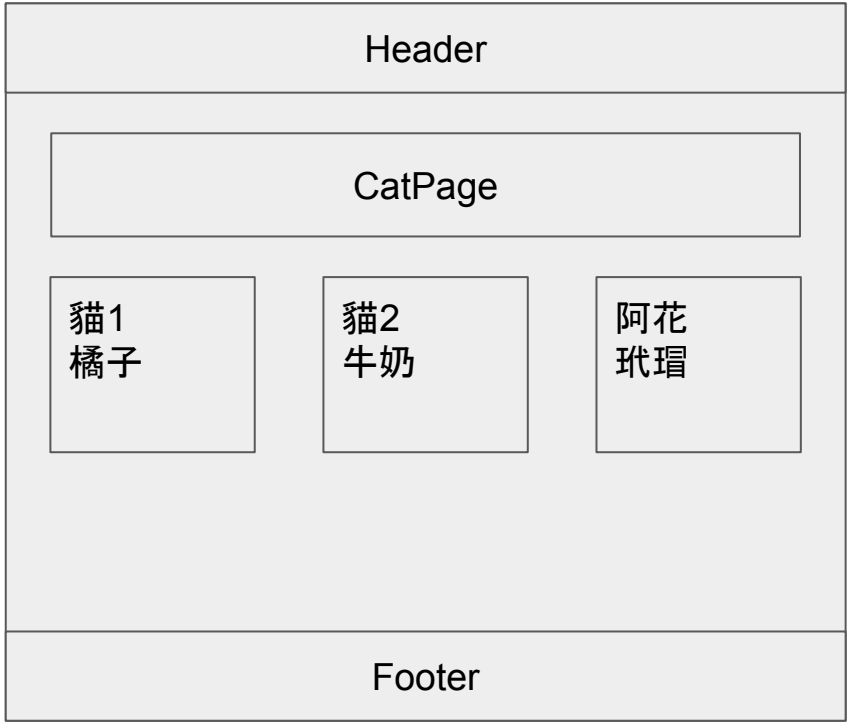
元件

```
└ src
   ├ components
   │  ├ Header
   │  │  └ index.js
   │  └ Footer
   │     └ index.js
   │
   └ containers
      ├ CatPage
      │  ├ CatWorld.js
      │  └ index.js
      └ IndexPage
         ├ CatWorld.js
         └ index.js
```



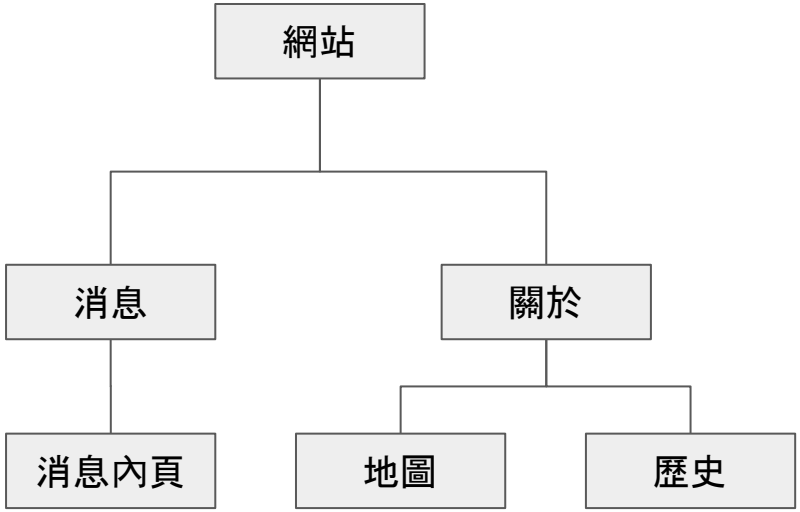
元件

```
└ src
   ├ components
   │  ├ Header
   │  │  └ index.js
   │  ├ Footer
   │  │  └ index.js
   │  └ CatWorld.js
   │
   └ containers
      ├ CatPage
      │  └ index.js
      └ IndexPage
         └ index.js
```



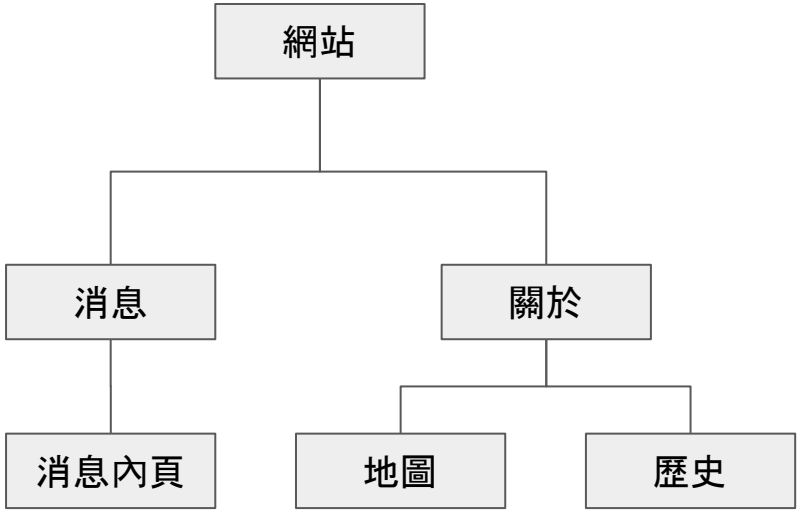
頁面

```
└ src
   └ containers
      ├ AboutPage
      │  └ index.js
      ├ HistoryPage
      │  └ index.js
      ├ LocationPage
      │  └ index.js
      ├ NewsContentPage
      │  └ index.js
      ├ NewsPage
      │  └ index.js
      └ SearchPage
         └ index.js
```



頁面

```
└ src
   └ containers
      ├ about
      │  ├ HistoryPage
      │  │  └ index.js
      │  ├ LocationPage
      │  │  └ index.js
      │  └ Page
      │     └ index.js
      ├ news
      │  ├ ContentPage
      │  │  └ index.js
      │  └ Page
      │     └ index.js
      └ SearchPage
         └ index.js
```



模組共用

- 手動加版本號
- git submodule
- npm package

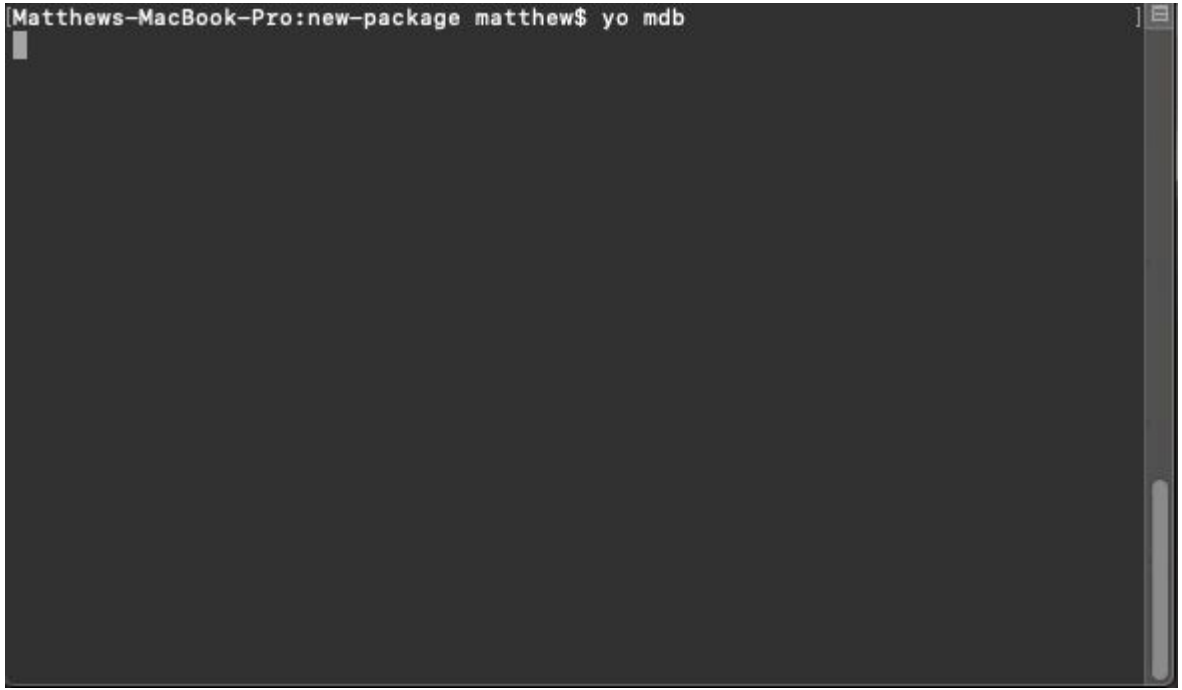
npm package

```
import {
  FullWidthWrapper,
  GridWrapper,
  Grid,
  Col,
} from '@sorosora/grid';
import {
  createBreakpoints,
} from '@sorosora/styled-breakpoints';
import {
  ConditionalWrap,
  RatioBox,
  Collapse,
  Img,
  InlineCenter
} from 'shared-components';
```

```
{
  "dependencies": {
    "@sorosora/grid": "^1.1.3",
    "@sorosora/styled-breakpoints": "^5.0.1",
    "next": "^8.1.0",
    "react": "^16.8.6",
    "react-dom": "^16.8.6",
    "shared-components":
"github:sorosora/shared-components#v0.4.0",
    "styled-components": "^4.3.1",
  },
}
```

Template

- yeoman



Medium: Generating code with Yeoman js

通用開發模式

- 選對適合工具
- 統一規則
- 易於團隊使用
- 可擴充