Bsidesoft co.

TO BE **ACTUAL** PROGRAMMER

CODE SPITZ

기초편

- CSS Rendering - 최초 개발에는 시각적으로 이해하기 쉬운 컴퓨터 그리기의 원리와 이를 구현하는 시스템에 대한 공부를 먼저 진행합니다.

- **ES6+ 기초편** - 언어의 기본 구성과 문법을 배우며 인공언어의 특징과 프로그래밍 언어의 표현 방법에 대한 기초 공부를 진행합니다.

심화편

- **ES6+ 함수와 클래스** - 프로그래밍 시 주요한 도구인 함수와 클래스를 사용해보면서 각각의 실질적인 의미를 공부합니다.

- **ES6+ 디자인패턴과 뷰패턴** - 객체지향프로그래밍의 기본이 되는 디자인 패턴과 프레임웍 레벨에서 사용되는 뷰패턴을 공부합니다.

CODE SPITZ 75

ES6+ DESIGN PATTERN VIEW PATTERN

1 2 3 4 5 6

WARMING UP
ES2015+ & HTML5

WARMING UP

```
{
        "title":"TIOBE Index for June 2017",
        "header":["Jun-17","Jun-16","Change","Programming Language","Ratings","Change"],
        "items":[
                [1,1,"","Java","14.49%","-6.30%"],
                [2,2,"","C","6.85%","-5.53%"],
                [3,3,"","C++","5.72%","-0.48%"],
                [4,4,"","Python","4.33%","0.43%"],
                [5,5,"","C#","3.53%","-0.26%"],
                [6,9,"","change","Visual Basic .NET","3.11%","0.76%"],
                [7,7,"","JavaScript","3.03%","0.44%"],
                [8,6,"change","PHP","2.77%","-0.45%"],
                [9,8,"change","Perl","2.31%","-0.09%"],
                [10,12,"change","Assembly language","2.25%","0.13%"],
                [11,10,"change","Ruby","2.22%","-0.11%"],
                [12,14,"change","Swift","2.21%","0.38%"],
                [13,13,"","Delphi/Object Pascal","2.16%","0.22%"],
                [14,16,"change","R","2.15%","0.61%"],
                [15,48,"change","Go","2.04%","1.83%"],
                [16,11,"change","Visual Basic,2.01%","-0.24%"],
                [17,17,"","MATLAB","2.00%","0.55%"],
                [18,15,"change","Objective-C","1.96%","0.25%"],
                [19,22,"change","Scratch","1.71%","0.76%"],
                [20,18,"change","PL/SQL","1.57%","0.22%"]
        ]
}
```

https://goo.gl/XHpnMF

WARMING UP

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CodeSpitz75-1</title>
</head>
<body>
    <section id="data"></section>
<script>
const Table =(_=>{

    return class{







    };
})();
const table = new Table("#data");
table.load("75_1.json");
</script>
</body>
</html>
```

WARMING UP

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CodeSpitz75-1</title>
</head>
<body>
    <section id="data"></section>
<script>
const Table =(_=>{
            static private
    return class{

                constructor
              public methods
             private methods

    };
})();
const table = new Table("#data");
table.load("75_1.json");
</script>
</body>
</html>
```

WARMING UP

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CodeSpitz75-1</title>
</head>
<body>
    <section id="data"></section>
<script>
const Table =(_=>{

    return class{
        constructor(parent){
        }
        async load(url){
        }



    };
})();
const table = new Table("#data");
table.load("75_1.json");
</script>
</body>
</html>
```

WARMING UP

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CodeSpitz75-1</title>
</head>
<body>
    <section id="data"></section>
<script>
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){
        }
        async load(url){
        }
        render(){
        }
    };
})();
const table = new Table("#data");
table.load("75_1.json");
</script>
</body>
</html>
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){
            if(typeof parent != 'string' || !parent) throw "invalid param";
            this[Private] = {parent};
        }
        load(url){
        }
        _render(){
        }
    };
})();
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){
            if(typeof parent != 'string' || !parent) throw "invalid param";
            this[Private] = {parent};
        }
        load(url){
            fetch(url).then(response=>{
                return response.json();
            }).then(json=>{
                this._render();
            });
        }
        _render(){
        }
    };
})();
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){
            if(typeof parent != 'string' || !parent) throw "invalid param";
            this[Private] = {parent};
        }
        load(url){
            fetch(url).then(response=>{
                return response.json();
            }).then(json=>{
                this._render();
            });
        }
        _render(){
        }
    };
})();
```

```
async load(url){
    const response = await fetch(url);
    const json = await response.json();
    this._render();
}
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){
            if(typeof parent != 'string' || !parent) throw "invalid param";
            this[Private] = {parent};
        }
        async load(url){
            const response = await fetch(url);
            const json = await response.json();
            this._render();
        }
        _render(){
        }
    };
})();
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){
            if(typeof parent != 'string' || !parent) throw "invalid param";
            this[Private] = {parent};
        }
        async load(url){
            const response = await fetch(url);
            if(!response.ok) throw "invaild response";
            const json = await response.json();
            const {title, header, items} = json;
            if(!items.length) throw "no items";
            Object.assign(this[Private], {title, header, items});
            this._render();
        }
        _render(){
        }
    };
})();
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){...}
        async load(url){...}
        _render(){

        }
    };
})();
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){...}
        async load(url){...}
        _render(){
            //부모, 데이터 체크
            //table생성
            //title을 caption으로
            //header를 thead로
            //items를 tr로
            //부모에 table삽입
        }
    };
})();
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){...}
        async load(url){...}
        _render(){
            //부모, 데이터 체크
            //table생성
            //title을 caption으로
            //header를 thead로
            //items를 tr로
            //부모에 table삽입
        }
    };
})();
```

```
const fields = this[Private], parent = document.querySelector(fields.parent);
if(!parent) throw "invaild parent";
if(!fields.items || !fields.items.length){
    parent.innerHTML = "no data";
    return;
} else parent.innerHTML = "";
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){...}
        async load(url){...}
        _render(){
            //부모, 데이터 체크
            const fields = this[Private], parent = document.querySelector(fields.parent);
            if(!parent) throw "invaild parent";
            if(!fields.items || !fields.items.length){
                parent.innerHTML = "no data";
                return;
            }else parent.innerHTML = "";
            //table생성
            //title을 caption으로
            //header를 thead로
            //items를 tr로
            //부모에 table삽입
        }
    };
})();
```

```
const table = document.createElement("table");
const caption = document.createElement("caption");
caption.innerHTML = fields.title;
table.appendChild(caption);
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){...}
        async load(url){...}
        _render(){
            //부모, 데이터 체크
            const fields = this[Private], parent = document.querySelector(fields.parent);
            if(!parent) throw "invaild parent";
            if(!fields.items || !fields.items.length){
                parent.innerHTML = "no data";
                return;
            }else parent.innerHTML = "";
            //table생성
            //title을 caption으로
            const table = document.createElement("table");
            const caption = document.createElement("caption");
            caption.innerHTML = fields.title;
            table.appendChild(caption);
            //header를 thead로
            //items를 tr로
            //부모에 table삽입
        }
    };
})();
```

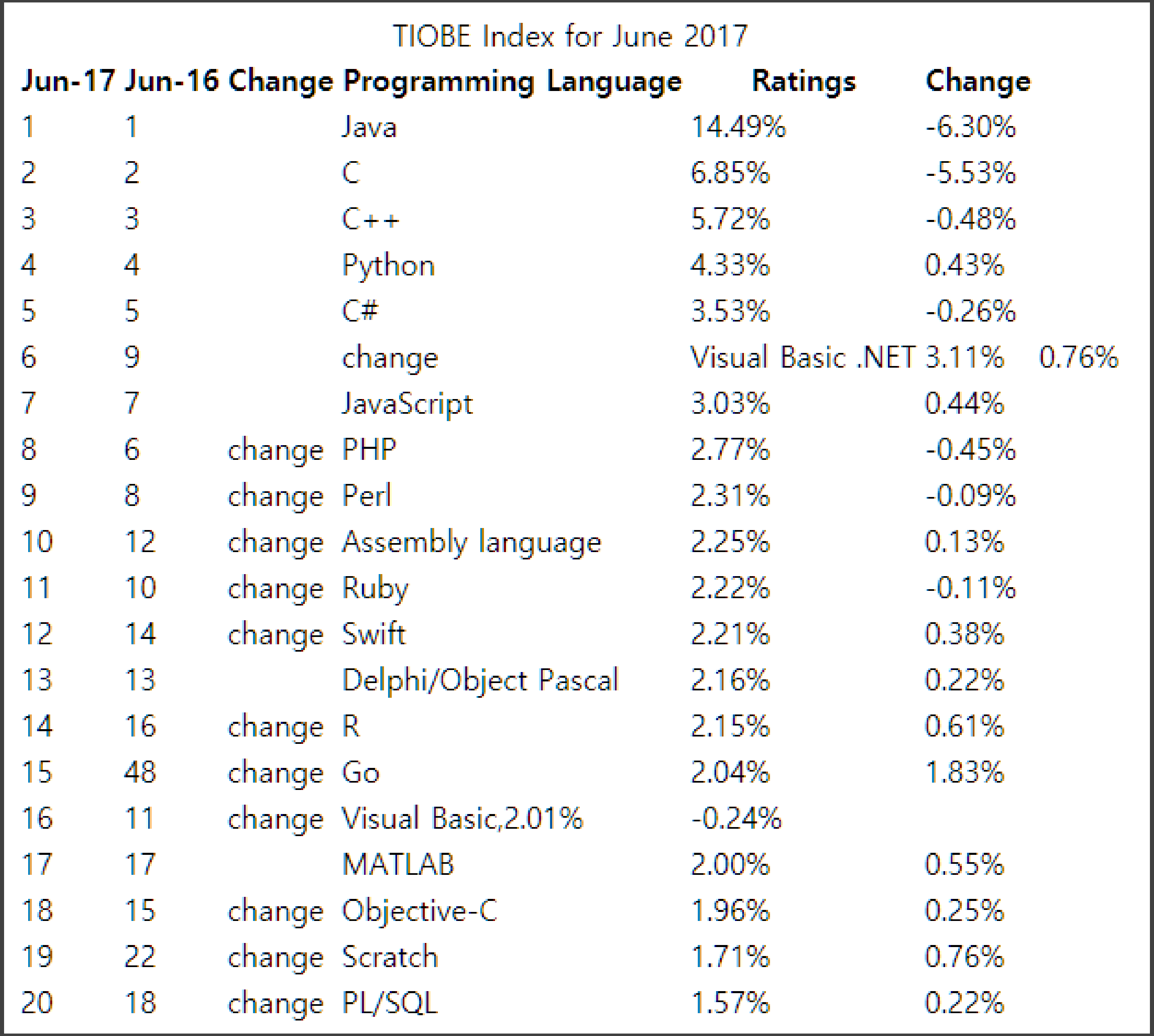
```
table.appendChild(
    fields.header.reduce((thead, data)=>{
        const th = document.createElement("th");
        th.innerHTML = data;
        thead.appendChild(th);
        return thead;
    }, document.createElement("thead"))
);
```

```
const table = new Table("#data");
table.load("75_1.json");
```

WARMING UP

```
const Table =(_=>{
    const Private = Symbol();
    return class{
        constructor(parent){...}
        async load(url){...}
        _render(){
            //부모, 데이터 체크
            const fields = this[Private], parent = document.querySelector(fields.parent);
            if(!parent) throw "invaild parent";
            if(!fields.items || !fields.items.length){
                parent.innerHTML = "no data";
                return;
            }else parent.innerHTML = "";
            //table생성
            //title을 caption으로
            const table = document.cre
            const caption = document.c
            caption.innerHTML = fields
            table.appendChild(caption)
            //header를 thead로
            table.appendChild(
                fields.header.reduce(
                    const th = docum
                    th.innerHTML = d
                    thead.appendChil
                    return thead;
                }, document.createEle
            );
            //items를 tr로
            //부모에 table삽입
        }
    };
})();
```

```
parent.appendChild(
     fields.items.reduce((table, row)=>{
        table.appendChild(
            row.reduce((tr, data)=>{
                const td = document.createElement("td");
                td.innerHTML = data;
                tr.appendChild(td);
                return tr;
            }, document.createElement("tr"))
        );
        return table;
    }, table)
);
```



TIOBE Index for June 2017

Jun-17	Jun-16	Change	Programming Language	Ratings	Change	
1	1		Java	14.49%	-6.30%	
2	2		C	6.85%	-5.53%	
3	3		C++	5.72%	-0.48%	
4	4		Python	4.33%	0.43%	
5	5		C#	3.53%	-0.26%	
6	9		change	Visual Basic .NET	3.11%	0.76%
7	7		JavaScript	3.03%	0.44%	
8	6	change	PHP	2.77%	-0.45%	
9	8	change	Perl	2.31%	-0.09%	
10	12	change	Assembly language	2.25%	0.13%	
11	10	change	Ruby	2.22%	-0.11%	
12	14	change	Swift	2.21%	0.38%	
13	13		Delphi/Object Pascal	2.16%	0.22%	
14	16	change	R	2.15%	0.61%	
15	48	change	Go	2.04%	1.83%	
16	11	change	Visual Basic,2.01%	-0.24%		
17	17		MATLAB	2.00%	0.55%	
18	15	change	Objective-C	1.96%	0.25%	
19	22	change	Scratch	1.71%	0.76%	
20	18	change	PL/SQL	1.57%	0.22%	

https://goo.gl/tXfseq

INTRODUCTION

프로그래밍 세계에서 유일하게 변하지 않는 원칙

프로그래밍 세계에서 유일하게 변하지 않는 원칙

"모든 프로그램은 변한다"

프로그래밍 세계에서 유일하게 변하지 않는 원칙

"모든 프로그램은 변한다"

▼

이미 작성된 복잡하고 거대한 프로그램을
어떻게 변경할 수 있을 것인가?

프로그래밍 세계에서 유일하게 변하지 않는 원칙

"모든 프로그램은 변한다"

▼

이미 작성된 복잡하고 거대한 프로그램을
어떻게 변경할 수 있을 것인가?

"격리(Isolation)"

결국 소프트웨어 공학의 상당 부분은

"격리 전략"

결국 소프트웨어 공학의 상당 부분은

"격리 전략"

▼

격리전략의 기본

결국 소프트웨어 공학의 상당 부분은

"격리 전략"

▼

격리전략의 기본

"변화율에 따라 작성하기"

변화율이란 시간적인 대칭성

"변화의 원인과 주기별로 정리"

변화율이란 시간적인 대칭성

"변화의 원인과 주기별로 정리"

▼

실천수칙

변화율이란 시간적인 대칭성

"변화의 원인과 주기별로 정리"

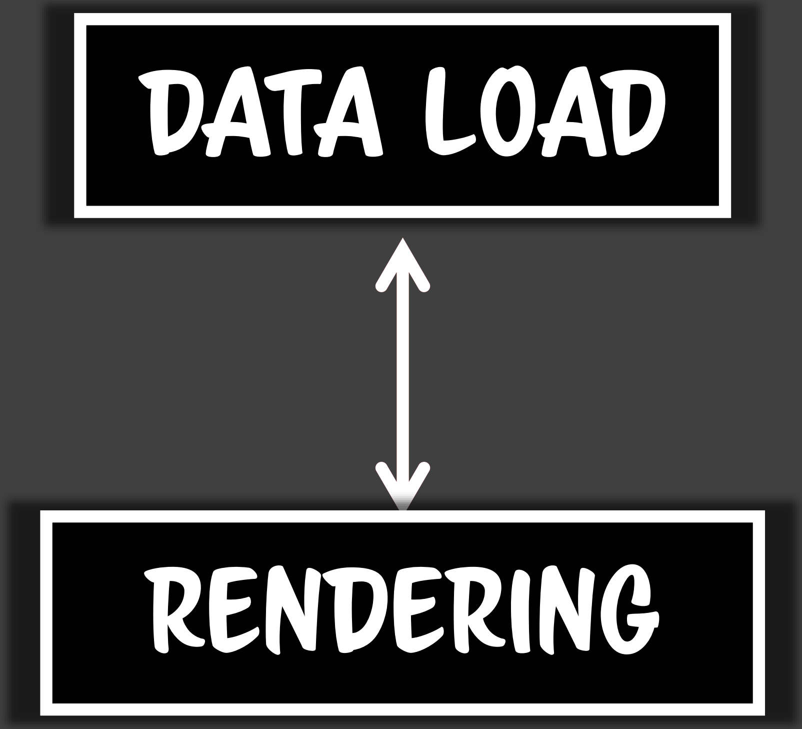
▼

실천수칙

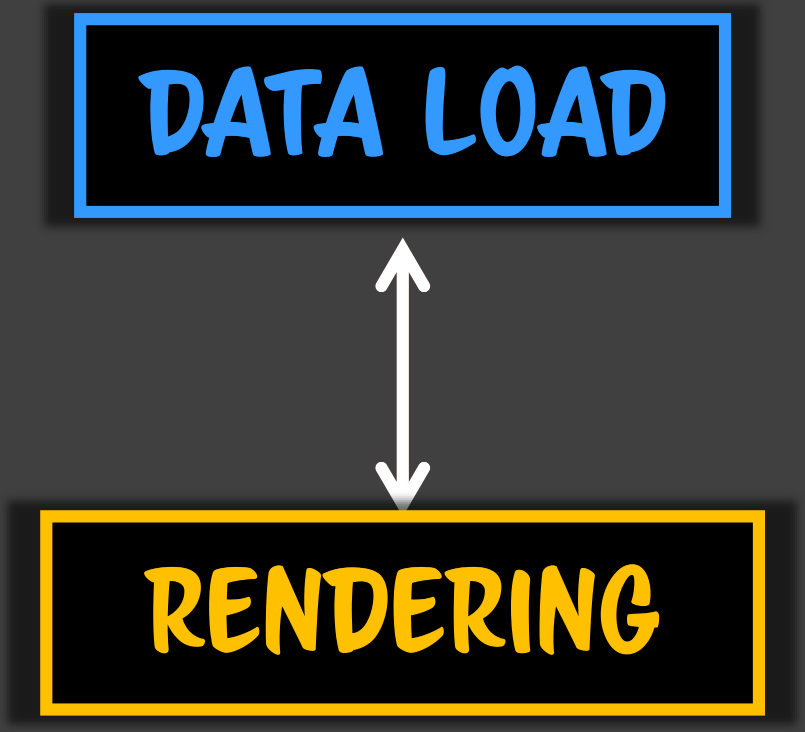
"강한 응집성" & "약한 의존성"

```
const Table =(_=>{
  const Private = Symbol();
  return class{
    constructor(parent){}
    async load(url){}
    _render(){}
  };
})();

const table = new Table("#data");
table.load("75_1.json");
```

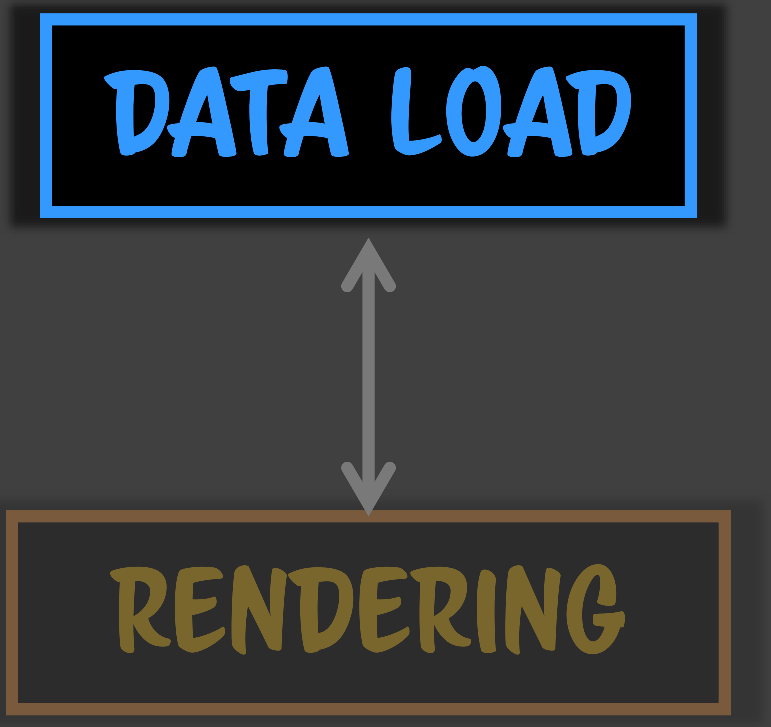
```
const Table =(_=>{
   const Private = Symbol();
   return class{
      constructor(parent){}
      async load(url){}
      _render(){}
   };
})();

const table = new Table("#data");
table.load("75_1.json");
```



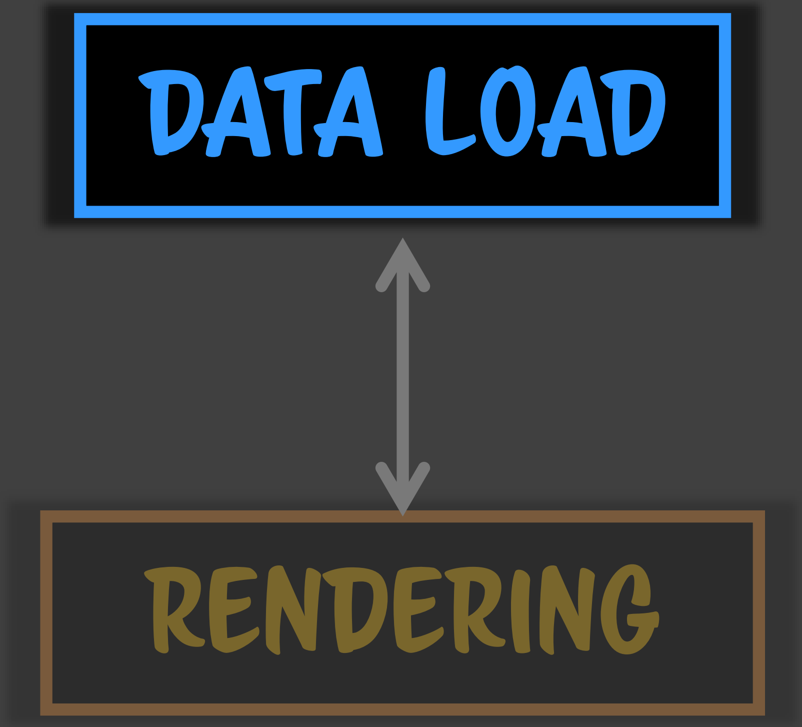
```
const Table =(_=>{
   const Private = Symbol();
   return class{
      constructor(parent){}
      async load(url){}
      _render(){}
   };
})();

const table = new Table("#data");
table.load("75_1.json");
```



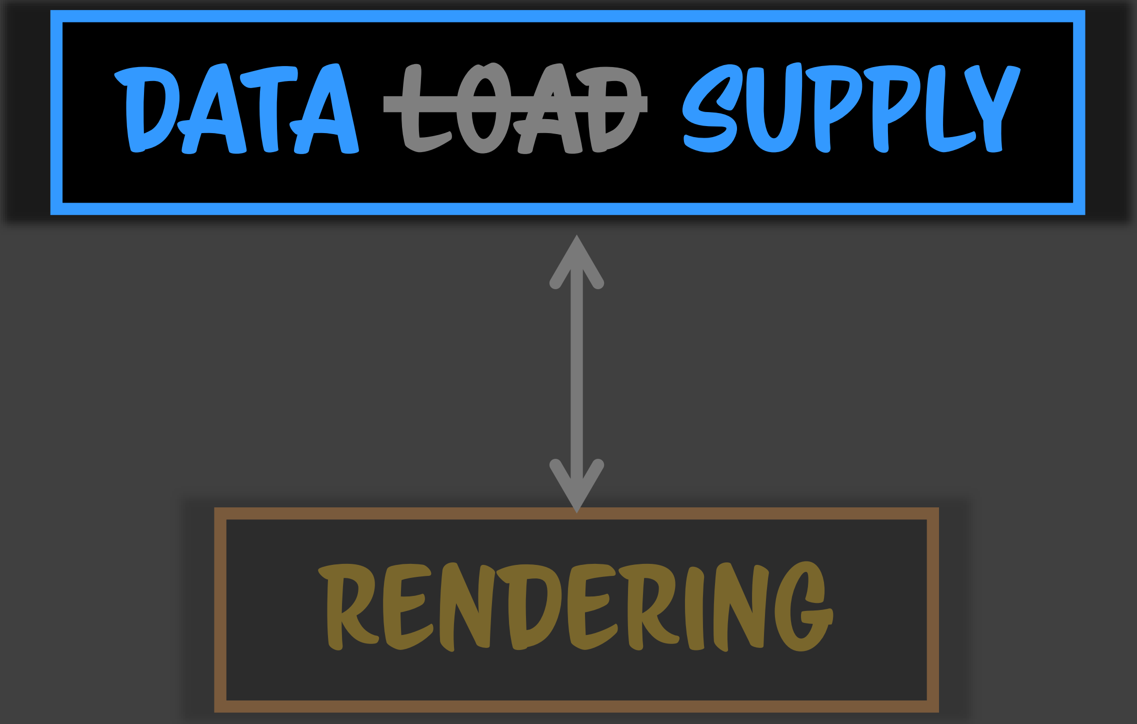
```
const loader = new Loader("75_1.json");
```



```
const loader = new Loader("75_1.json");
loader.load(json=>{
  const renderer = new Renderer();
  renderer.setData(json);
  renderer.render();
});
```

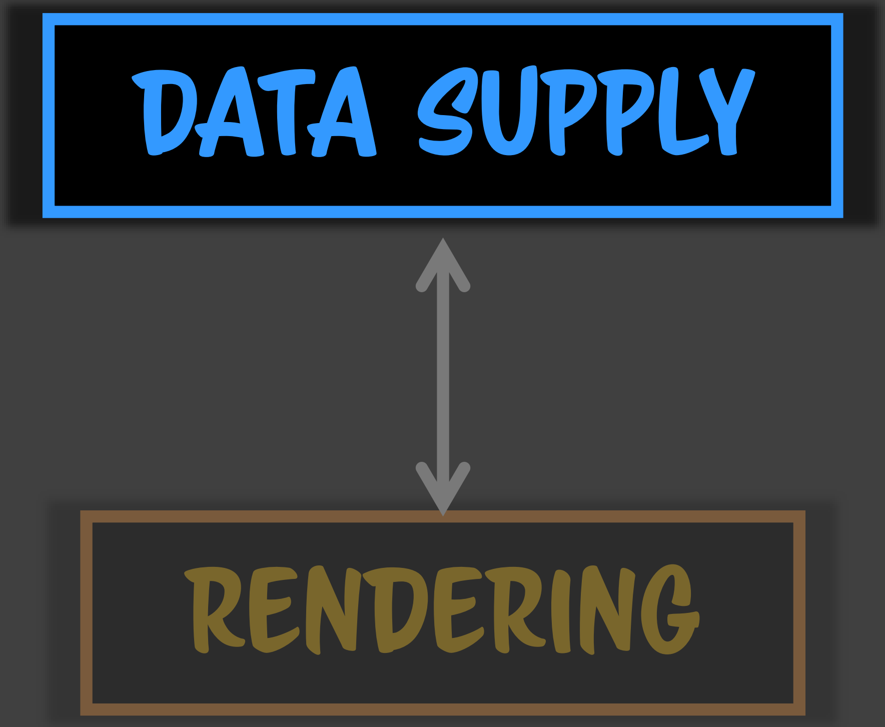


```
const loader = new Loader("75_1.json");
loader.load(json=>{
  const renderer = new Renderer();
  renderer.setData(json);
  renderer.render();
});
```



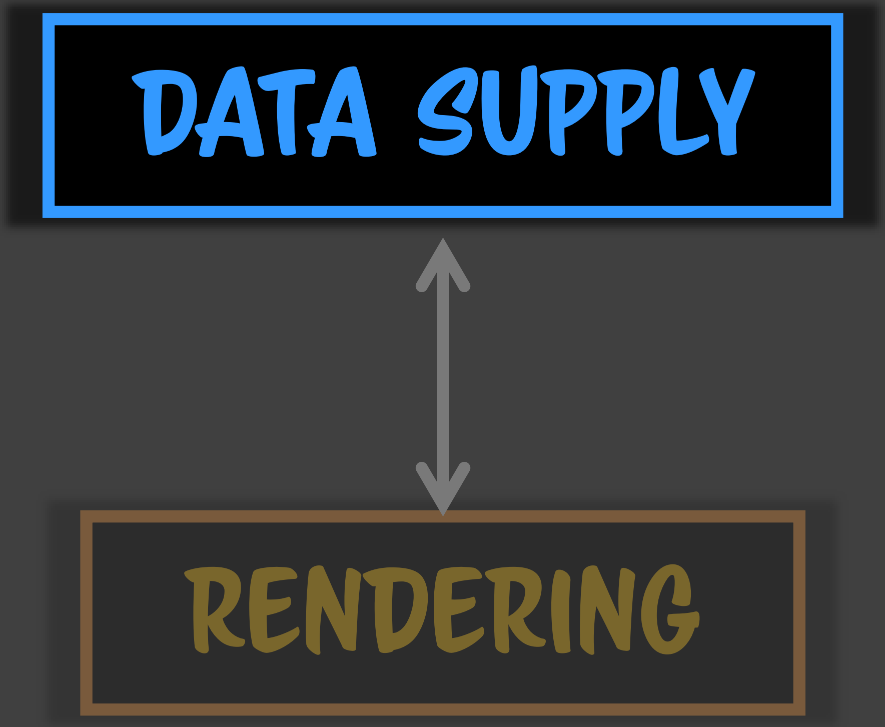
```
const loader = new Loader("75_1.json");
loader.load(json=>{
   const renderer = new Renderer();
   renderer.setData(json);
   renderer.render();
});
```

VALUE→OBJECT



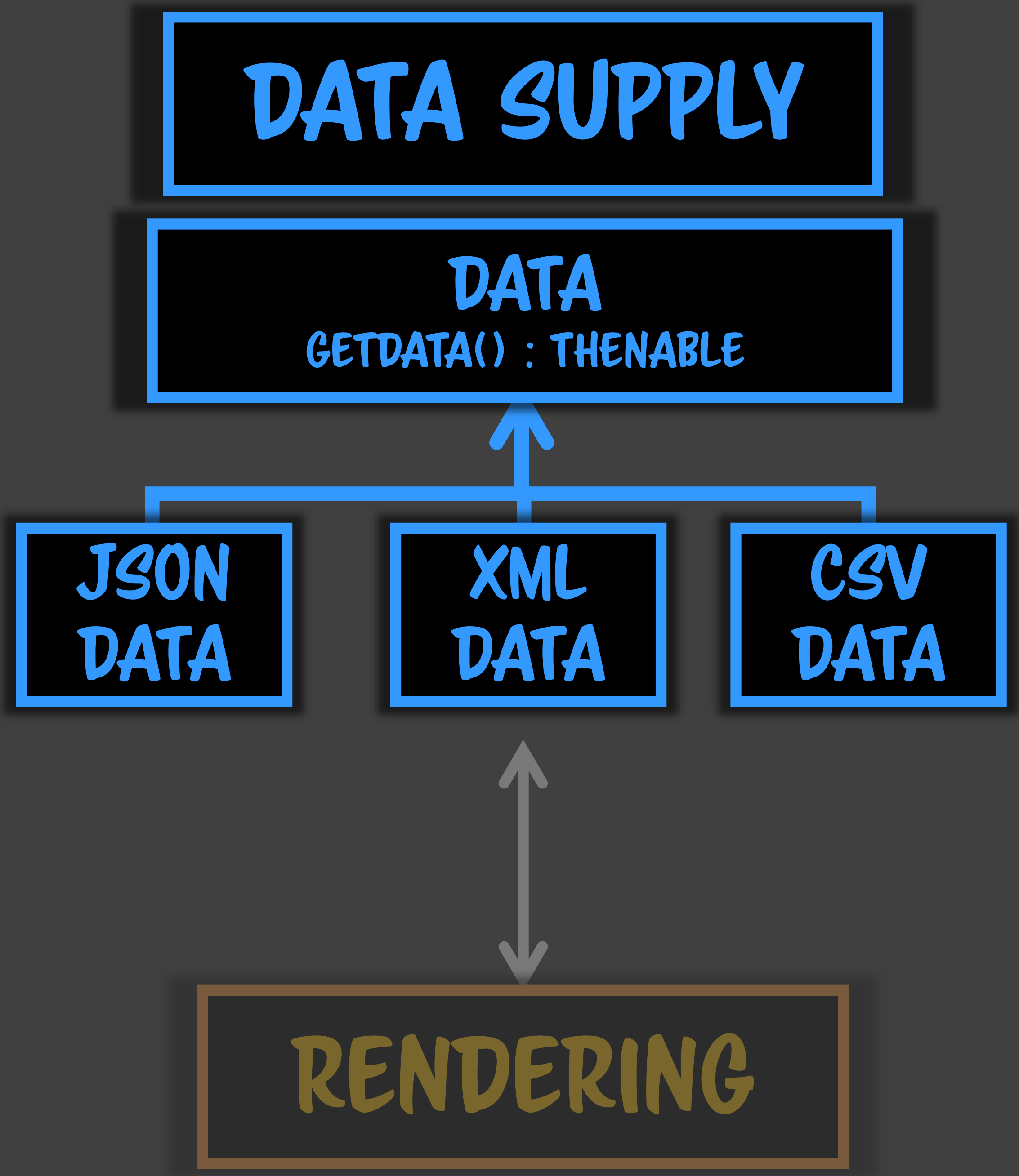
```
const loader = new Loader("75_1.json");
loader.load(json=>{
   const renderer = new Renderer();
   renderer.setData(json);
   renderer.render();
});
```

```
const data = new JsonData("75_1.json");
const renderer = new Renderer();
renderer.render(data);
```

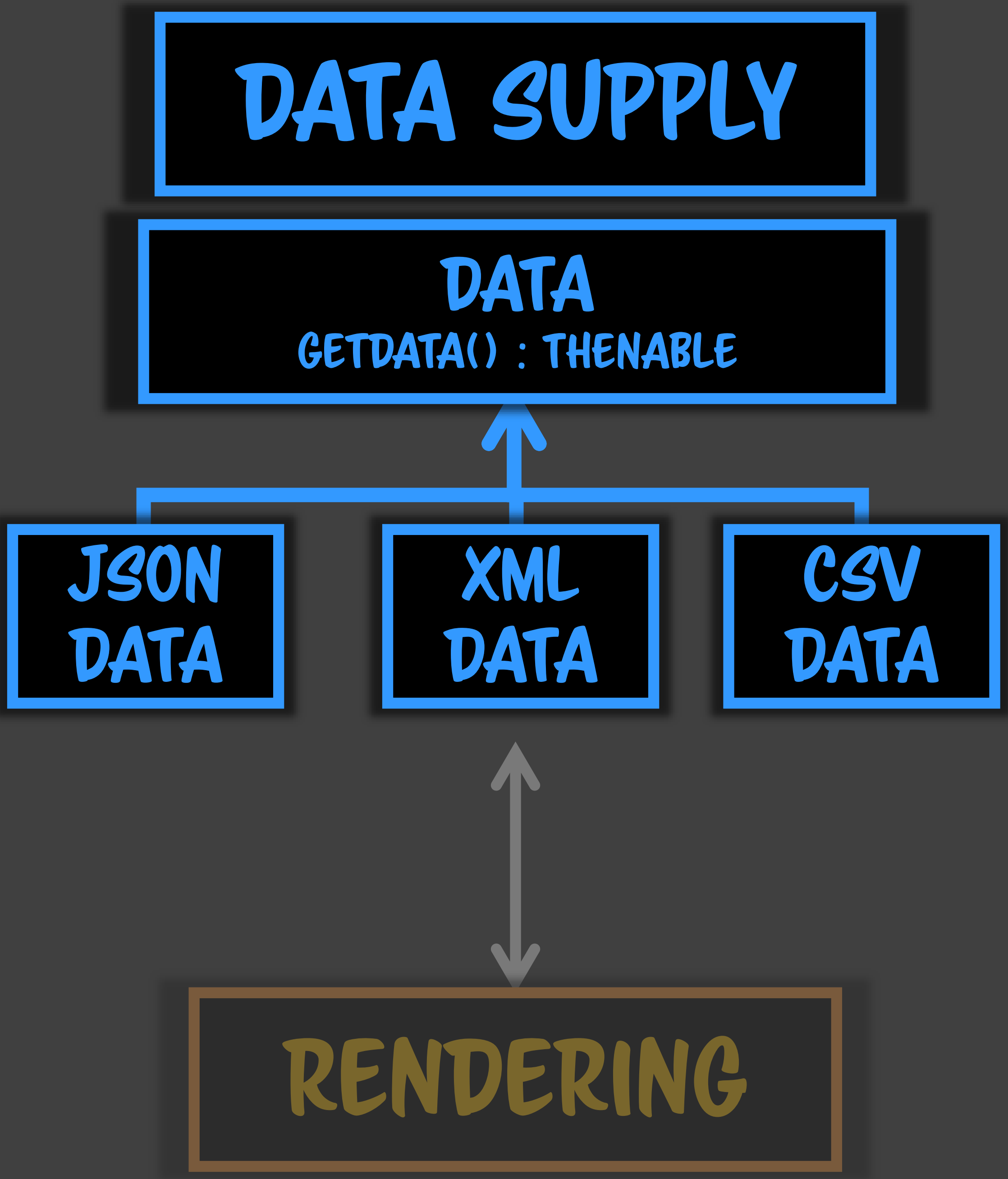


```
const loader = new Loader("75_1.json");
loader.load(json=>{
   const renderer = new Renderer();
   renderer.setData(json);
   renderer.render();
});
```

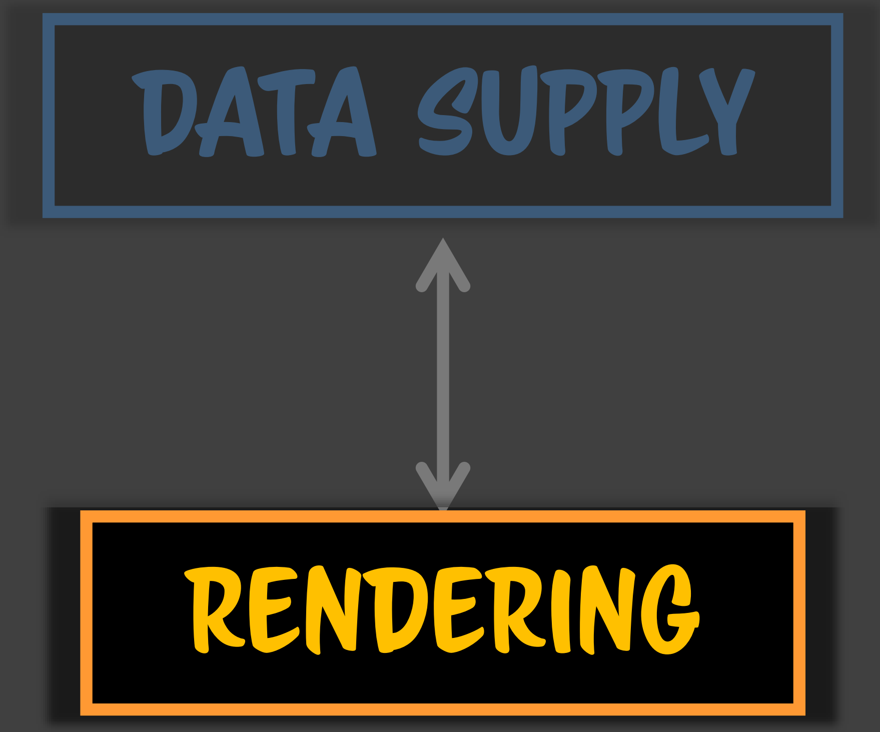
```
const data = new JsonData("75_1.json");
const renderer = new Renderer();
renderer.render(data);
```



```
const Data = class{
   async getData(){throw "getData must override";}
};


const JsonData = class extends Data{
   constructor(data){
      super();
      this._data = data;
   }
   async getData(){
      if(typeof this._data == 'string'){
         const response = await fetch(this._data);
         return await response.json();
      }else return this._data;
   }
};
```



```
const Data = class{
    async getData(){throw "getData must override";}
};

const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async getData(){
        if(typeof this._data == 'string'){
            const response = await fetch(this._data);
            return await response.json();
        }else return this._data;
    }
};

const Renderer = class{
    constructor(){}
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        const json = await data.getData();
        console.log(json);
    }
}

const data = new JsonData("75_1.json");

const renderer = new Renderer();
renderer.render(data);
```



```
const JsonData = class extends Data{
   constructor(data){
      super();
      this._data = data;
   }
   async getData(){
      if(typeof this._data == 'string'){
         const response = await fetch(this._data);
         return await response.json();
      }else return this._data;
   }
};



const Renderer = class{
   constructor(){}
   async render(data){
      if(!(data instanceof Data)) throw "invalid data type";
      const json = await data.getData();
      console.log(json);
   }
}
```

```
const JsonData = class extends Data{
   constructor(data){
      super();
      this._data = data;
   }
   async getData(){
      if(typeof this._data == 'string'){
         const response = await fetch(this._data);
         return await response.json();
      }else return this._data;
   }
};
```

VALUE → OBJECT

```
const Renderer = class{
   constructor(){}
   async render(data){
      if(!(data instanceof Data)) throw "invalid data type";
      const json = await data.getData();
      console.log(json);
   }
}
```

```
const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async getData(){
        if(typeof this._data == 'string'){
            const response = await fetch(this._data);
            return await response.json();
        }else return this._data;
    }
};
```

VALUE → OBJECT

INFO
TITLE
HEADER
ITEMS

```
const Renderer = class{
    constructor(){}
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        const json = await data.getData();
        console.log(json);
    }
}
```

```
const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async getI
        if(typ
            con

}

const Rendere
    construct
    async rend
        if(!(d
        const json = await data.getData();
        console.log(json);
    }
}
```

INFO

TITLE
HEADER
ITEMS

```
const Info = class{
    constructor(json){
        const {title, header, items} = json;
        if(typeof title != 'string' || !title) throw "invalid title";
        if(!Array.isArray(header) || !header.length) throw "invalid header";
        if(!Array.isArray(items) || !items.length) throw "invalid items";
        this._private = {title, header, items};
    }
    get title(){return this._private.title;}
    get header(){return this._private.header;}
    get items(){return this._private.items;}
};
```

```
const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async getData(){
        let json;
        if(typeof this._data == 'string'){
            const response = await fetch(this._data);
            json = await response.json();
        }else json = this._data;
        return new Info(json);
    }
};
```

```
const Info = class{
    constructor(json){
        const {title, header, items} = json;
        if(typeof title != 'string' || !title) throw "invalid title";
        if(!Array.isArray(header) || !header.length) throw "invalid header";
        if(!Array.isArray(items) || !items.length) throw "invalid items";
        this._private = {title, header, items};
    }
    get title(){return this._private.title;}
    get header(){return this._private.header;}
    get items(){return this._private.items;}
};
```

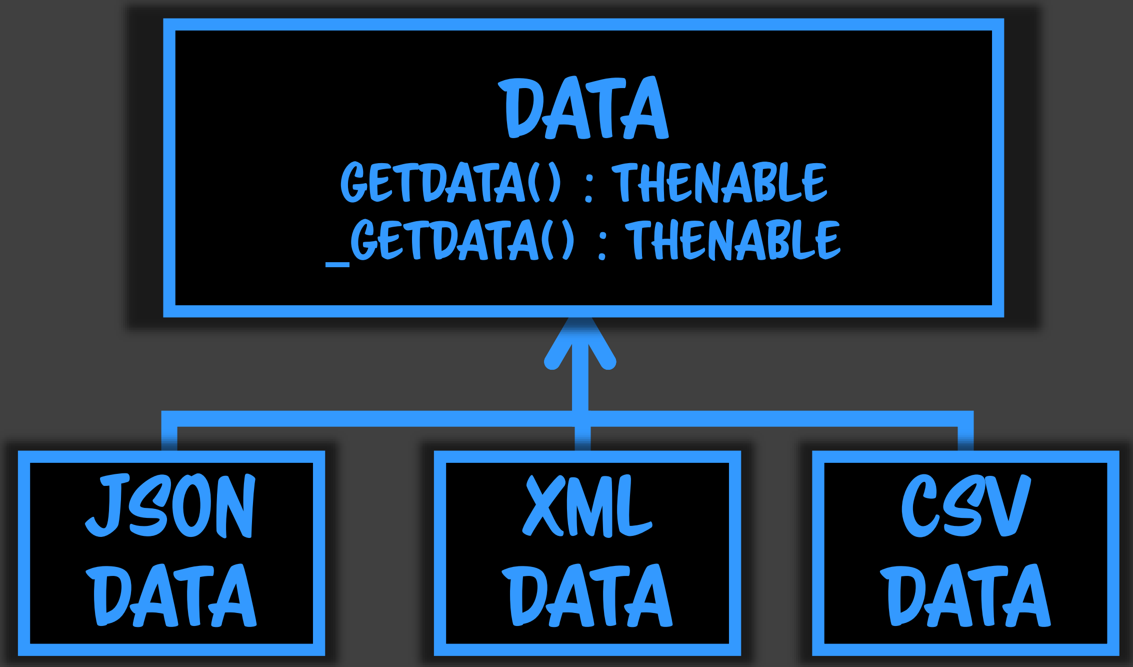
```
const Renderer = class{
    constructor(){}
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        const info = await data.getData();
        console.log(info.title, info.header, info.items);
    }
}
```

```
const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async getData(){
        let json;
        if(typeof this._data == 'string'){
            const response = await fetch(this._data);
            json = await response.json();
        }else json = this._data;
        return new Info(json);
    }
};
```

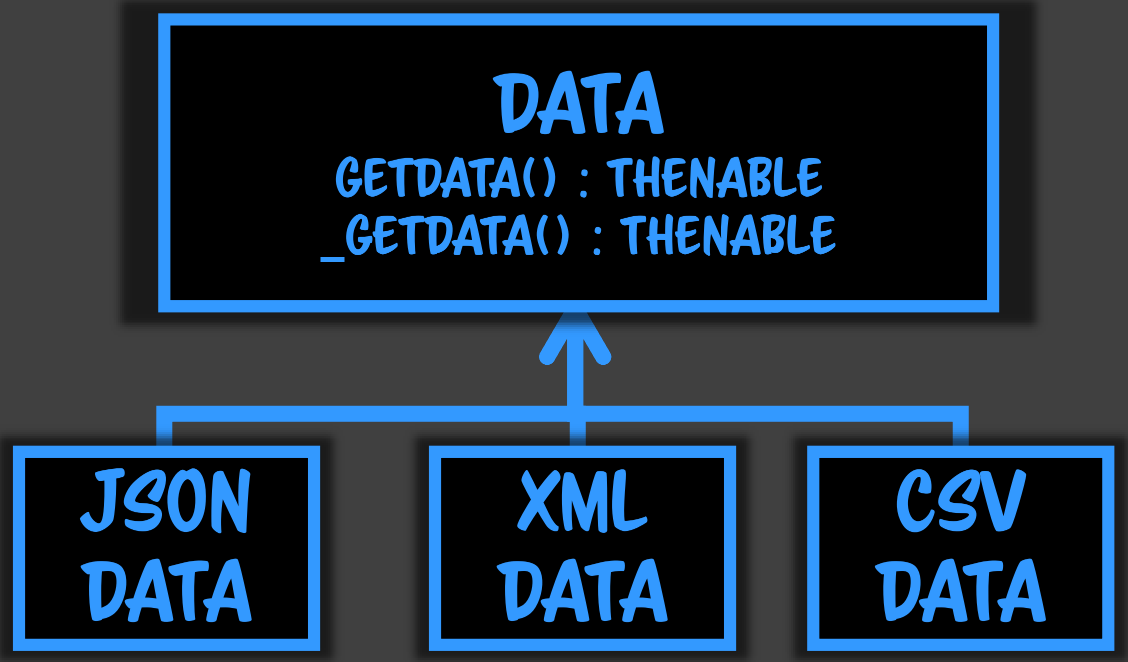
```
const Data = class{
    async getData(){throw "getData must override";}
};
```

```
const Renderer = class{
    constructor(){}
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        const info = await data.getData();
        console.log(info.title, info.header, info.items);
    }
}
```

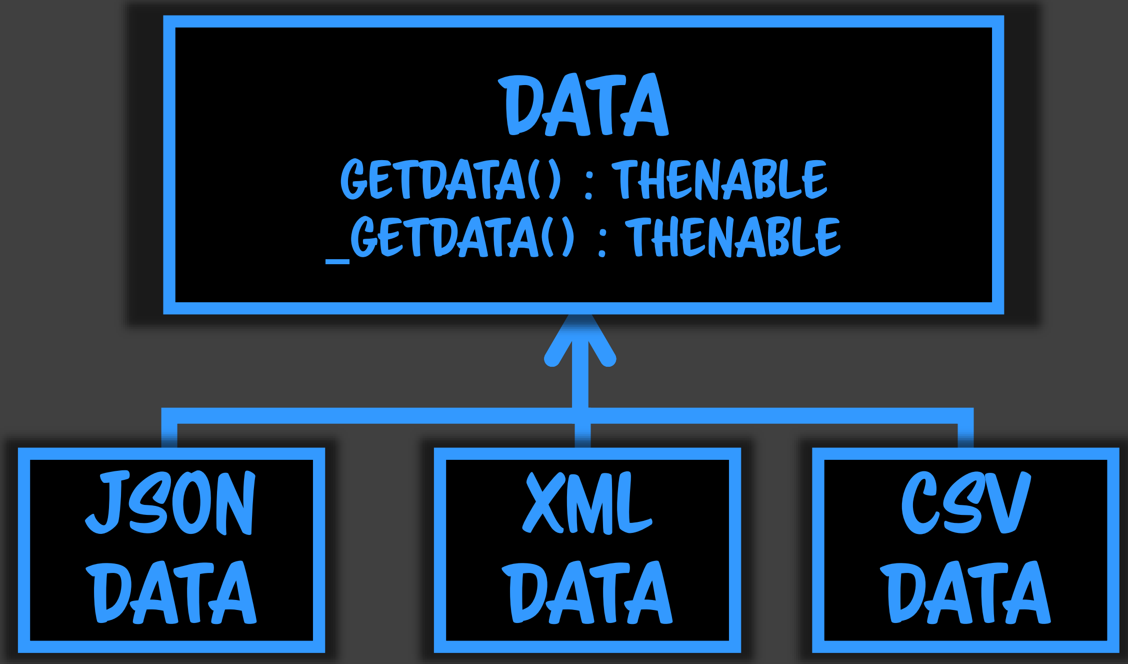
```
const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async getData(){
        let json;
        if(typeof this._data == 'string'){
            const response = await fetch(this._data);
            json = await response.json();
        }else json = this._data;
        return new Info(json);
    }
};


const Renderer = class{
    constructor(){}
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        const info = await data.getData();
        console.log(info.title, info.header, info.items);
    }
}
```



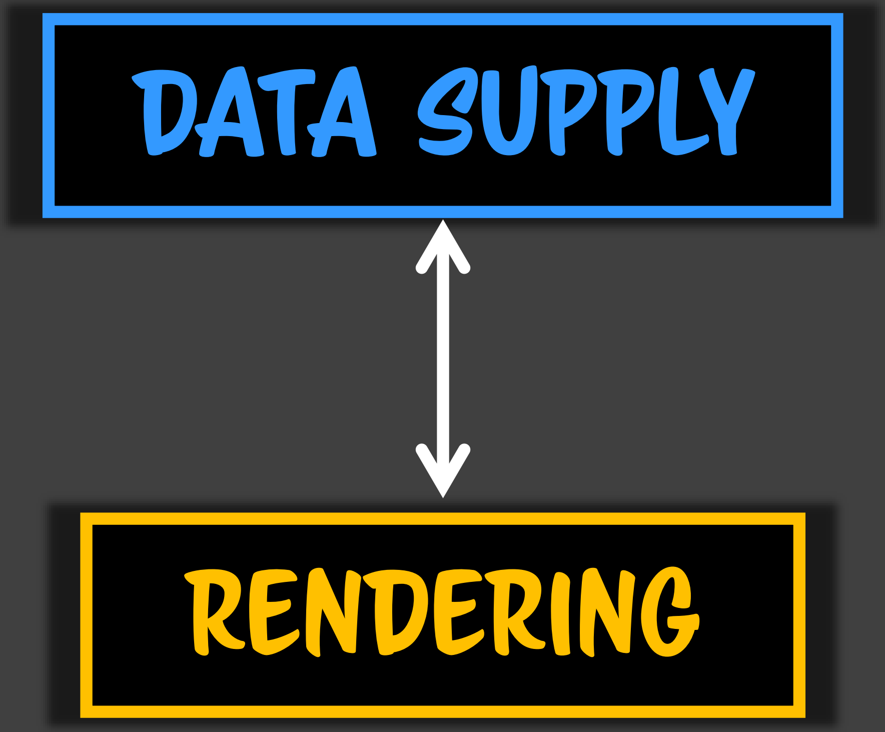
```
const Data = class{
    async getData(){
        const json = await this._getData();
        return new Info(json);
    }
    async _getData(){
        throw "_getData must overrided";
    }
};
const JsonData = class extends Data{
    constructor(data){
        super();
        this._data = data;
    }
    async _getData(){
        let json;
        if(typeof this._data == 'string'){
            const response = await fetch(this._data);
            return await response.json();
        }else return this._data;
    }
};
```

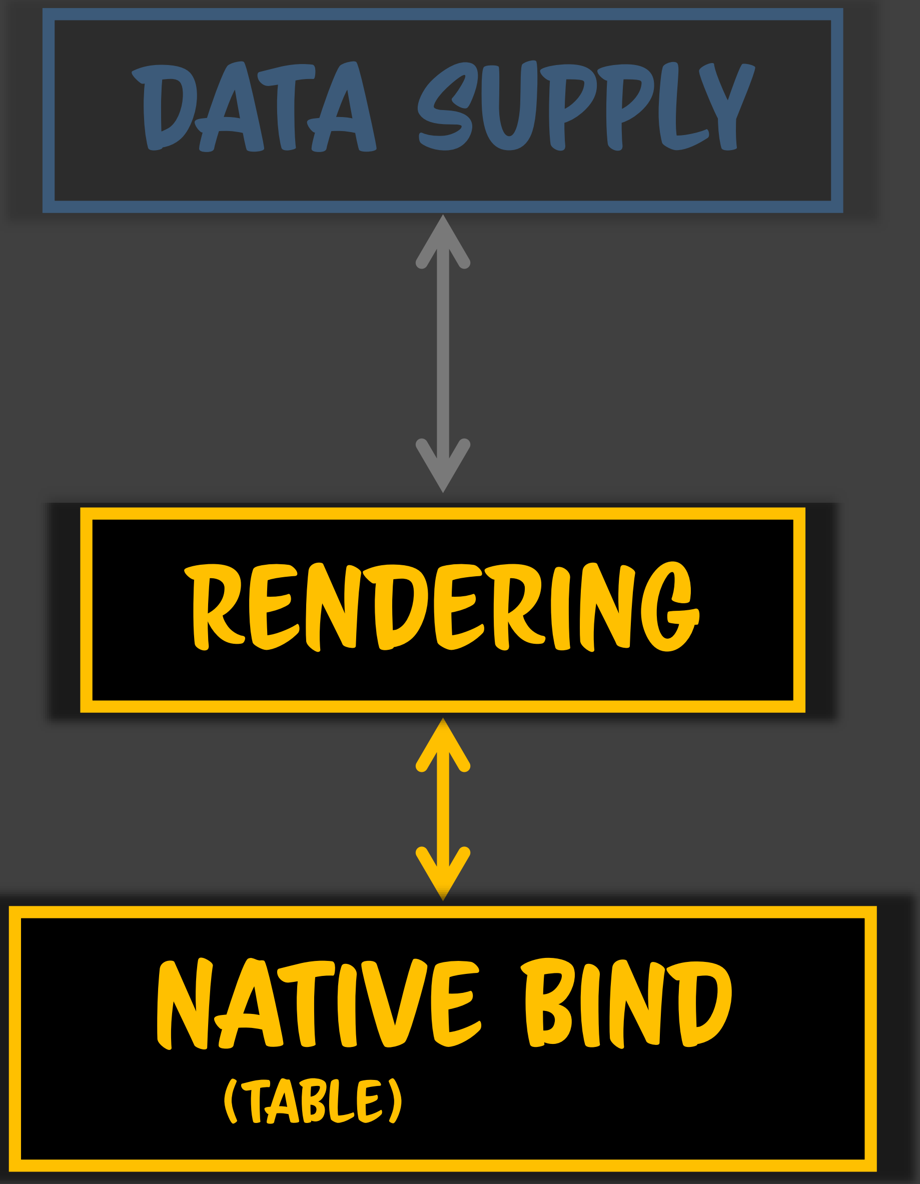


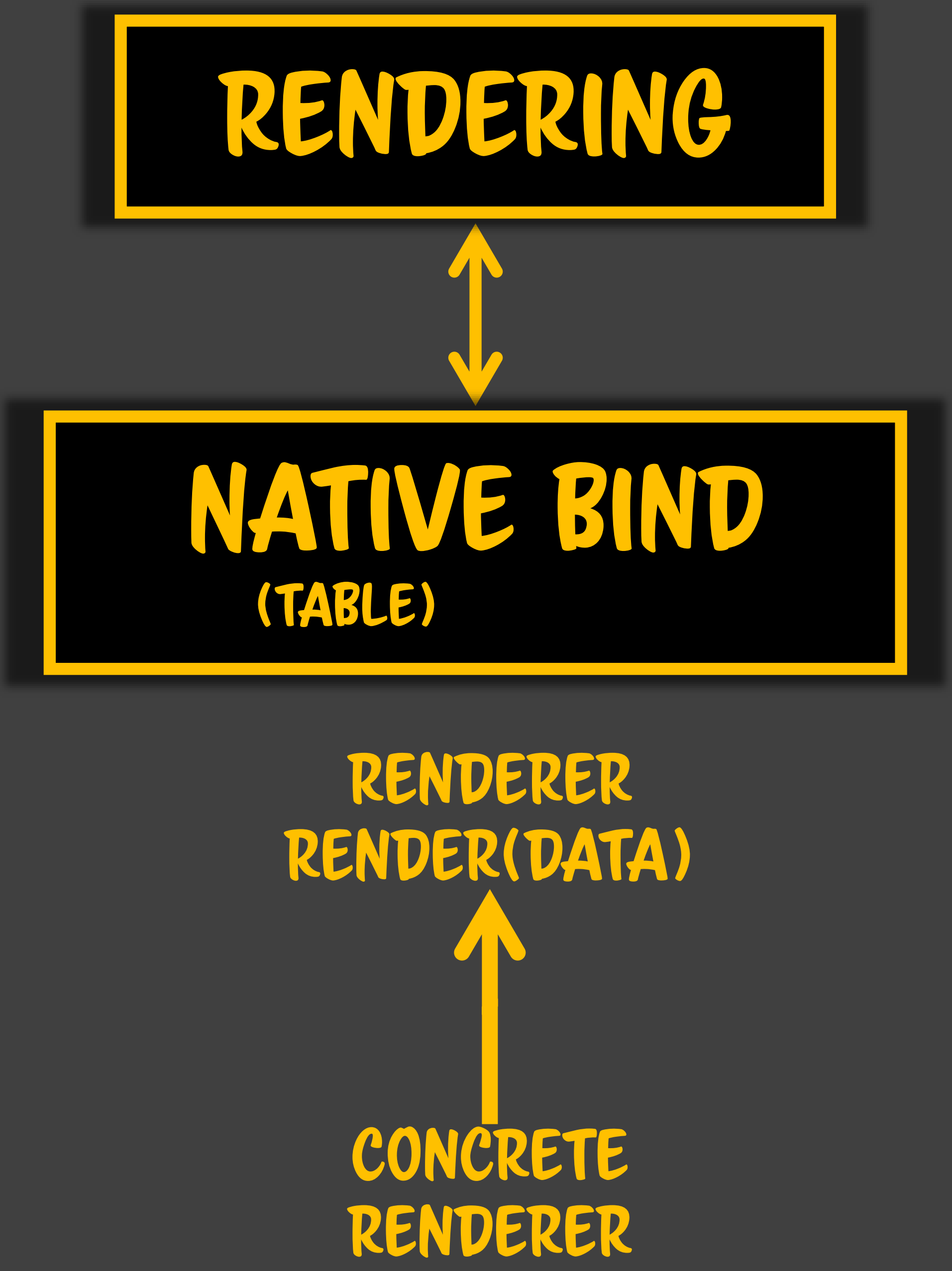
```
const Data = class{
     async getData(){
          const json = await this._getData();
          return new Info(json);
     }
     async _getData(){
          throw "_getData must overrided";
     }
};
const JsonData = class extends Data{
     constructor(data){
          super();
          this._data = data;
     }
     async _getData(){
          let json;
          if(typeof this._data == 'string'){
               const response = await fetch(this._data);
               return await response.json();
          }else return this._data;
     }
};
```



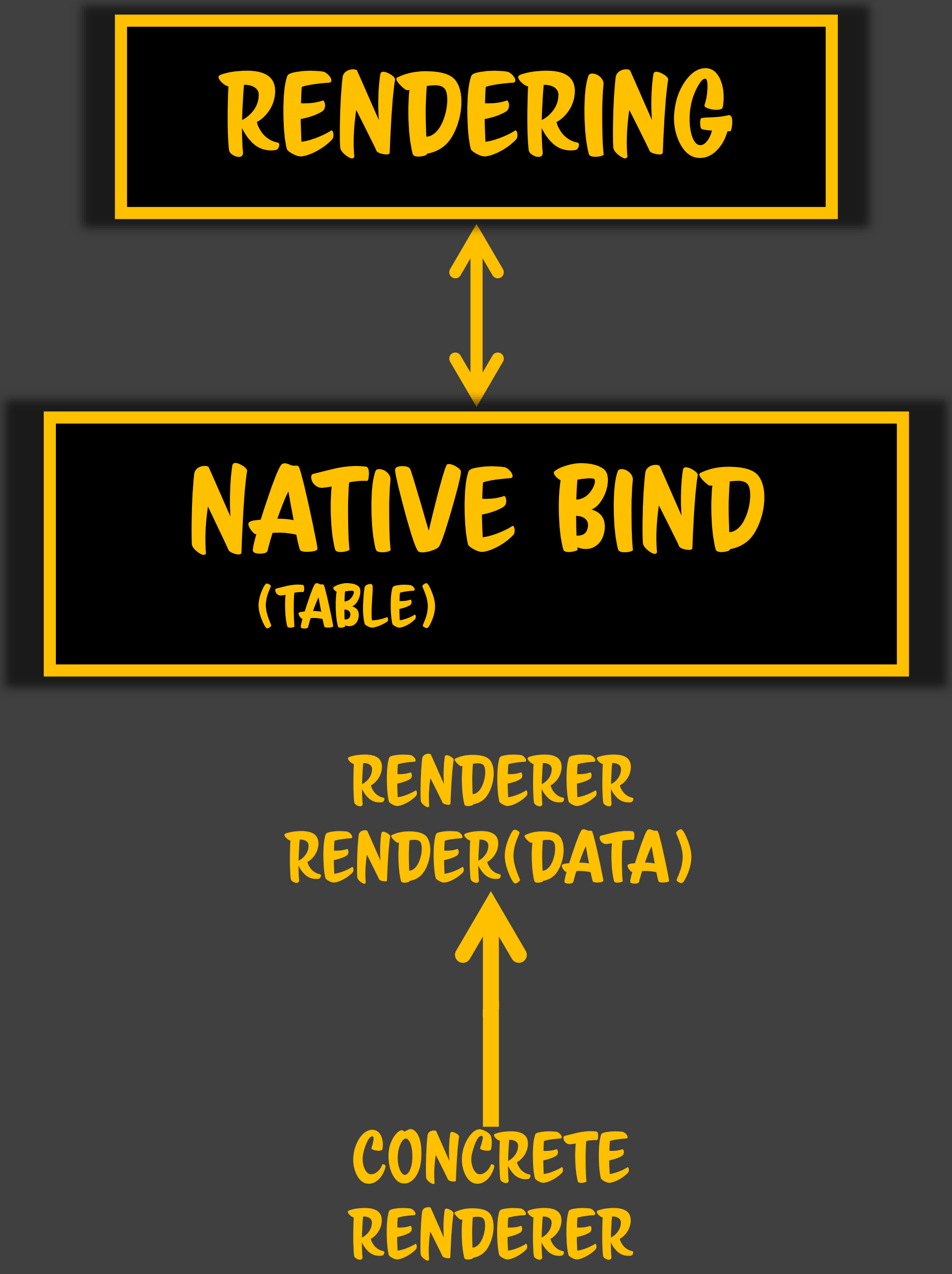
```
const Renderer = class{
   constructor(){}
   async render(data){
      if(!(data instanceof Data)) throw "invalid data type";
      const info = await data.getData();
      console.log(info.title, info.header, info.items);
   }
}
```


DATA SUPPLY
RENDERING

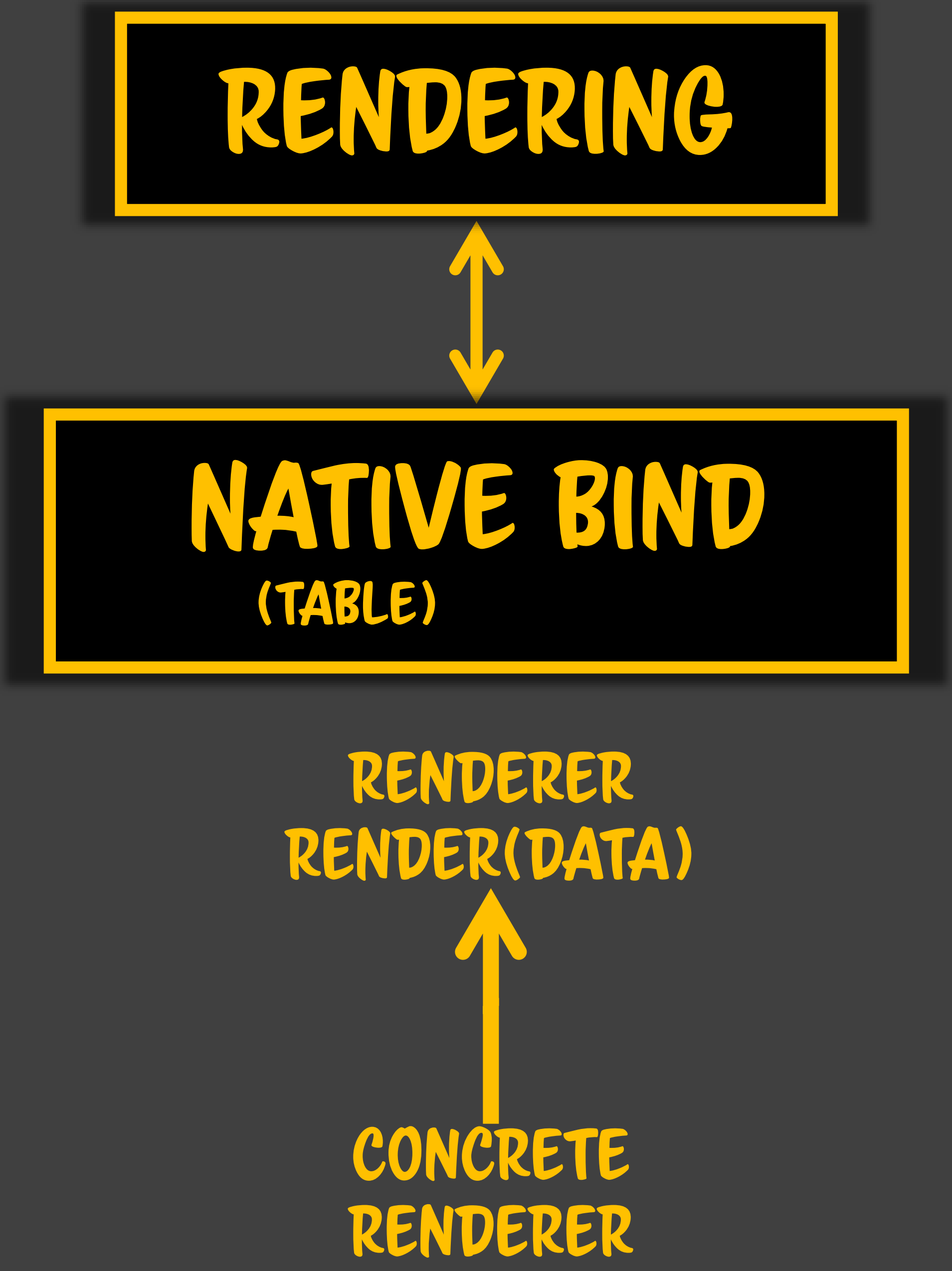

DATA SUPPLY
RENDERING
NATIVE BIND
(TABLE)


RENDERING
NATIVE BIND
(TABLE)
RENDERER
RENDER(DATA)
CONCRETE
RENDERER

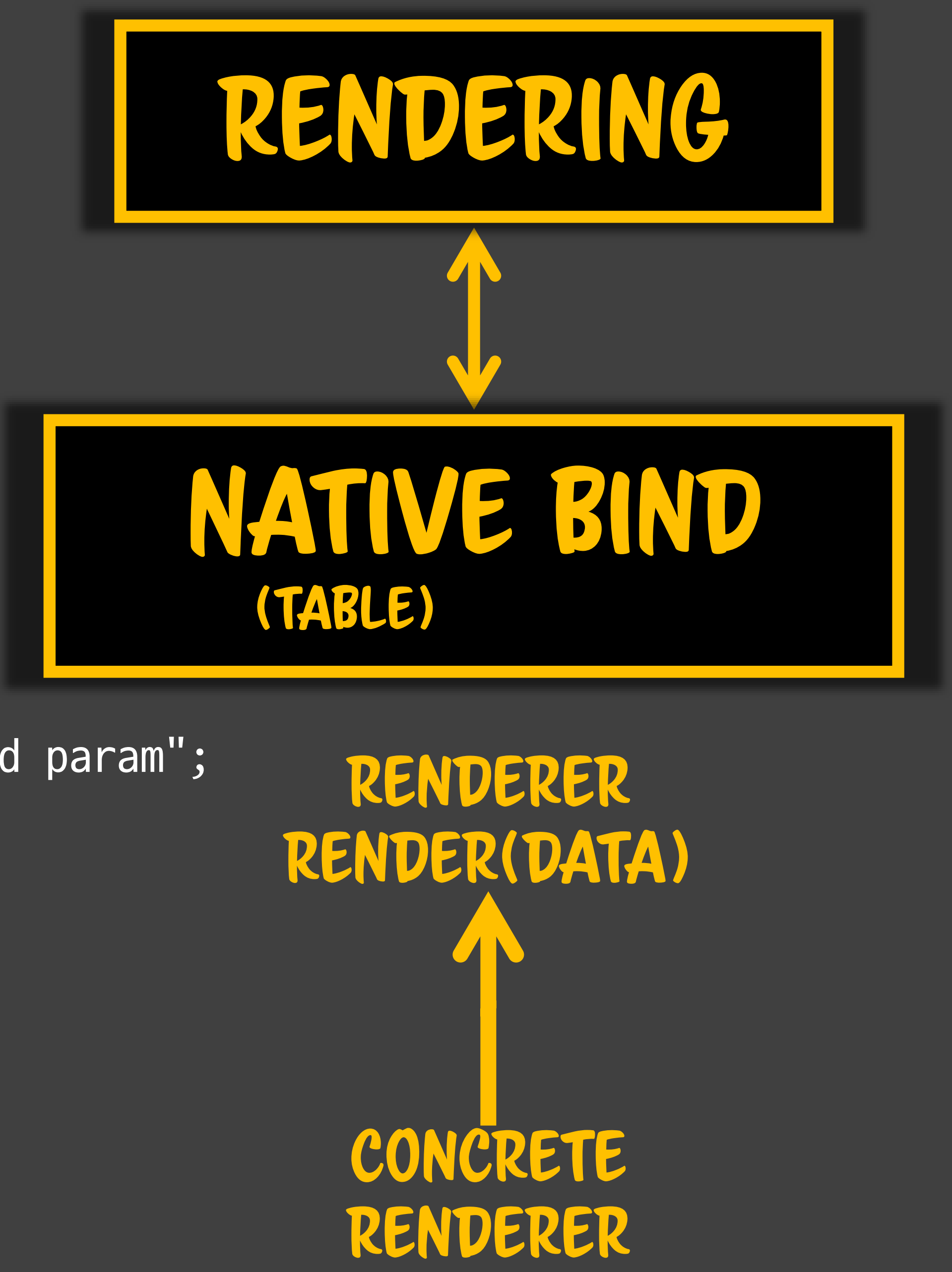
```
const Renderer = class{
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        this._info = await data.getData();
        this._render();
    }
    _render(){
        throw "_render must overrided";
    }
}
```



```
const Renderer = class{
    async render(data){
        if(!(data instanceof Data)) throw "invalid data type";
        this._info = await data.getData();
        this._render();
    }
    _render(){
        throw "_render must overrided";
    }
}


const TableRenderer = class extends Renderer{
    constructor(parent){}
    _render(){
    }
}
```



```
const Renderer = class{
   async render(data){
      if(!(data instanceof Data)) throw "invalid data type";
      this._info = await data.getData();
      this._render();
   }
   _render(){
      throw "_render must overrided";
   }
}


const TableRenderer = class extends Renderer{
   constructor(parent){
      if(typeof parent != 'string' || !parent) throw "invalid param";
      super();
      this._parent = parent;
   }
   _render(){
   }
}
```



```
const TableRenderer = class extends Renderer{
    constructor(parent){...}
    _render(){
        const parent = document.querySelector(this._parent);
        if(!parent) throw "invaild parent";
        parent.innerHTML = "";
        const [table, caption] = "table,caption".split(",").map(v=>document.createElement(v));
        caption.innerHTML = this._info.title;
        table.appendChild(caption);
        table.appendChild(
            this._info.header.reduce(
                (thead, data)=>(thead.appendChild(document.createElement("th")).innerHTML = data, thead),
                document.createElement("thead"))
        );
        parent.appendChild(
            this._info.items.reduce(
                (table, row)=>(table.appendChild(
                    row.reduce(
                        (tr, data)=>(tr.appendChild(document.createElement("td")).innerHTML = data, tr),
                        document.createElement("tr"))
                ), table),
                table)
        );
    }
}
```

PRACTICE #1

Q. 실제 코드를 구현하고 실행하면 예외가 발생한다.
예외의 지점을 찾고 수정하여 완성하라.

PRACTICE #2

지금까지 전개한 객체협력모델에서는 여전히 문제가 남아있다.
Info는 Data와 Renderer 사이에 교환을 위한 프로토콜인데
Renderer의 자식인 TableRenderer도 Info에 의존적인 상태다.
이를 개선하라.