


Bsidesoft co.

since 2004


DELEGATED PROPERTY

Property?

```
class Test{

  #map = new Map;




}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
}
```

```
class Test{

  #map = new Map;                                    내부의 은닉된 상태

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
}
```

내부의 은닉된 상태

외부에 캡슐화된 인터페이스

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
}

const test = new Test;
console.log(test.name);

test.name = "hika";
console.log(test.name);
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
}

const test = new Test;
console.log(test.name);

test.name = "hika";
console.log(test.name);
```

```
no name
```

```
hika
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  set name(v){
    this.#map.set("name", v);
  }
  get name(){
    return this.#map.get("name") ?? "no name";
  }
  set company(v){
    this.#map.set("company", v);
  }
  get company(){
    return this.#map.get("company") ?? "no company";
  }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
  }
}
```

Property to Object

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
 }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
 }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
 }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
 }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
 }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{

  #map = new Map;

  _set(k, v){
    this.#map.set(k, v);
  }
  _get(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  set name(v){
    this._set("name", v);
  }
  get name(){
    return this._get("name");
  }
  set company(v){
    this._set("company", v);
  }
  get company(){
    return this._get("company");
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{
  #map = new Map;

  #delegate = new TestDelegate(this.#map);

  set name(v){
    this.#delegate.setValue("name", v);
  }
  get name(){
    return this.#delegate.getValue("name");
  }
  set company(v){
    this.#delegate.setValue("company", v);
  }
  get company(){
    return this.#delegate.getValue("company");
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{
  #map = new Map;

  #delegate = new TestDelegate(this.#map);

  set name(v){
    this.#delegate.setValue("name", v);
  }
  get name(){
    return this.#delegate.getValue("name");
  }
  set company(v){
    this.#delegate.setValue("company", v);
  }
  get company(){
    return this.#delegate.getValue("company");
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{
  #map = new Map;

  #name = new TestDelegate(this.#map);
  #company = new ValueDelegate("");

  set name(v){
    this.#name.setValue("name", v);
  }
  get name(){
    return this.#name.getValue("name");
  }
  set company(v){
    this.#company.setValue("company", v);
  }
  get company(){
    return this.#company.getValue("company");
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

DefineProperty

```
class Test{
  #map = new Map;

  #name = new TestDelegate(this.#map);
  #company = new ValueDelegate("");

  set name(v){
    this.#name.setValue("name", v);
  }
  get name(){
    return this.#name.getValue("name");
  }
  set company(v){
    this.#company.setValue("company", v);
  }
  get company(){
    return this.#company.getValue("company");
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{
  #map = new Map;

  constructor(){
    const name = new TestDelegate(this.#map);
    Object.defineProperty(this, "name", {
      get(){
        return name.getValue("name");
      },
      set(v){
        name.setValue("name", v);
      }
    });
    const company = new ValueDelegate("");
    Object.defineProperty(this, "company", {
      get(){
        return name.getValue("company");
      },
      set(v){
        name.setValue("company", v);
      }
    });
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{
  #map = new Map;

  constructor(){
    const name = new TestDelegate(this.#map);
    Object.defineProperty(this, "name", {
      get(){
        return name.getValue("name");
      },
      set(v){
        name.setValue("name", v);
      }
    });
    const company = new ValueDelegate("");
    Object.defineProperty(this, "company", {
      get(){
        return name.getValue("company");
      },
      set(v){
        name.setValue("company", v);
      }
    });
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
class Test{
  #map = new Map;

  constructor(){
    const name = new TestDelegate(this.#map);
    Object.defineProperty(this, "name", {
      get(){
        return name.getValue("name");
      },
      set(v){
        name.setValue("name", v);
      }
    });
    const company = new ValueDelegate("");
    Object.defineProperty(this, "company", {
      get(){
        return name.getValue("company");
      },
      set(v){
        name.setValue("company", v);
      }
    });
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(key);
    },
    set(v){
      delegator.setValue(key, v);
    }
  });
};
```

```
class Test{
  #map = new Map;

  constructor(){
    prop(this, "name", new TestDelegate(this.#map));
    prop(this, "company", new ValueDelegate(this.#map));
  }
}
```

```
class ValueDelegate{
  #v;
  constructor(v){
    this.#v = v;
  }
  getValue(k){
    return this.#v;
  }
  setValue(k, v){
    this.#v = v;
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(key);
    },
    set(v){
      delegator.setValue(key, v);
    }
  });
};
```

DefineProperty to Prototype

```
class Test{
  #map = new Map;

  constructor(){
    prop(this, "name", new TestDelegate(this.#map));
    prop(this, "company", new TestDelegate(this.#map));
  }
}
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}

const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(key);
    },
    set(v){
      delegator.setValue(key, v);
    }
  });
};
```

```
class Test{
  #map = new Map;

  constructor(){
    prop(this, "name", new TestDelegate(this.#map));
    prop(this, "company", new TestDelegate(this.#map));
  }
}

prop(Test.prototype, "name", new TestDelegate(this.#map));
prop(Test.prototype, "company", new TestDelegate(this.#map));
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}

const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(key);
    },
    set(v){
      delegator.setValue(key, v);
    }
  });
};
```

```
class Test{
  #map = new Map;

  constructor(){
    prop(this, "name", new TestDelegate(this.#map));
    prop(this, "company", new TestDelegate(this.#map));
  }
}

prop(Test.prototype, "name", new TestDelegate);
prop(Test.prototype, "company", new TestDelegate);
```

```
class TestDelegate{
  #map;
  constructor(map){
    this.#map = map;
  }
  getValue(k){
    return this.#map.get(k) ?? `no ${k}`;
  }
  setValue(k, v){
    this.#map.set(k, v);
  }
}

const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(key);
    },
    set(v){
      delegator.setValue(key, v);
    }
  });
};
```

```
class Test{
  #map = new Map;

  constructor(){
    prop(this, "name", new TestDelegate(this.#map));
    prop(this, "company", new TestDelegate(this.#map));
  }
}

prop(Test.prototype, "name", new TestDelegate);
prop(Test.prototype, "company", new TestDelegate);
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
class Test{
  #map = new Map;

  constructor(){
    prop(this, "name", new TestDelegate(this.#map));
    prop(this, "company", new TestDelegate(this.#map));
  }
}

prop(Test.prototype, "name", new TestDelegate);
prop(Test.prototype, "company", new TestDelegate);
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
class Test{
  map = new Map;
}

prop(Test.prototype, "name", new TestDelegate);
prop(Test.prototype, "company", new TestDelegate);
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

Delegate Class Wrapper

```
class Test{
  map = new Map;
}

prop(Test.prototype, "name", new TestDelegate);
prop(Test.prototype, "company", new TestDelegate);
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
const by = (cls)=>{
```

```
};
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
const by = (cls)=>{
  Object.entries(cls)
```

```
};
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const by = (cls)=>{
  Object.entries(cls)
    .filter(([, v])=>typeof v.getValue == "function" && typeof v.setValue == "function")
```

```
};
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const by = (cls)=>{
  Object.entries(cls)
    .filter(([, v])=>typeof v.getValue == "function" && typeof v.setValue == "function")
    .reduce((proto, [key, delegator])=>{

      return proto;
    }, cls.prototype)

};
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const by = (cls)=>{
  Object.entries(cls)
    .filter(([, v])=>typeof v.getValue == "function" && typeof v.setValue == "function")
    .reduce((proto, [key, delegator])=>{
      Object.defineProperty(proto, key, {
        get(){
          return delegator.getValue(this, key);
        },
        set(v){
          delegator.setValue(this, key, v);
        }
      });
      return proto;
    }, cls.prototype)

};
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

```
const Test = by(class{
  static name = new TestDelegate;
  static company = new TestDelegate;
  map = new Map;
});
```

```
class TestDelegate{
  getValue(target, k){
    return target.map.get(k) ?? `no ${k}`;
  }
  setValue(target, k, v){
    target.map.set(k, v);
  }
}
```

```
const by = (cls)=>{
  Object.entries(cls)
    .filter(([, v])=>typeof v.getValue == "function" && typeof v.setValue == "function")
    .reduce((proto, [key, delegator])=>{
      Object.defineProperty(proto, key, {
        get(){
          return delegator.getValue(this, key);
        },
        set(v){
          delegator.setValue(this, key, v);
        }
      });
      return proto;
    }, cls.prototype)
  return cls;
};
```

```
const prop = (target, key, delegator)=>{
  Object.defineProperty(target, key, {
    get(){
      return delegator.getValue(this, key);
    },
    set(v){
      delegator.setValue(this, key, v);
    }
  });
};
```

Delegates

```
const lazy =f=>{
  let v;
  return {
    getValue(target, k){
      return v ?? (v = f(target));
    },
    setValue(target, k, v){}
  };
};
```

```
const lazy =f=>{
  let v;
  return {
    getValue(target, k){
      return v ?? (v = f(target));
    },
    setValue(target, k, v){}
  };
};

const Test2 = by(class{
  static element = lazy(({selector})=>document.querySelector(selector));
  selector;
  constructor(selector){
    this.selector = selector;
  }
});
```

```
const lazy =f=>{
  let v;
  return {
    getValue(target, k){
      return v ?? (v = f(target));
    },
    setValue(target, k, v){}
  };
};

const Test2 = by(class{
  static element = lazy(({selector})=>document.querySelector(selector));
  selector;
  constructor(selector){
    this.selector = selector;
  }
});


<body>
  <main></main>
  <script>
  const test = new Test2("#test");



</script>
</body>
```

```
const lazy =f=>{
  let v;
  return {
    getValue(target, k){
      return v ?? (v = f(target));
    },
    setValue(target, k, v){}
  };
};
const Test2 = by(class{
  static element = lazy(({selector})=>document.querySelector(selector));
  selector;
  constructor(selector){
    this.selector = selector;
  }
});

<body>
  <main></main>
  <script>
  const test = new Test2("#test");
  document.querySelector("main").innerHTML = `<div id="test">test</div>`;
  console.log(test.element.innerHTML);
</script>
</body>
```

```
const lazy =f=>{
  let v;
  return {
    getValue(target, k){
      return v ?? (v = f(target));
    },
    setValue(target, k, v){}
  };
};

const Test2 = by(class{
  static element = lazy(({selector})=>document.querySelector(sele
  selector;
  constructor(selector){
    this.selector = selector;
  }
});


<body>
  <main></main>
  <script>
  const test = new Test2("#test");
  document.querySelector("main").innerHTML = `<div id="test">test</div>`;
  console.log(test.element.innerHTML);
</script>
</body>
```

```
const lazy =(_=>{
  class Lazy{
    #f;
    #v;
    constructor(f){
      this.#f = f;
    }
    getValue(target, k){
      return this.#v ?? (this.#v = this.#f(target));
    }
    setValue(target, k, v){}
  }
  return f=>new Lazy(f);
})();
```

```
const observe =(_=>{
  class Observer{
    #value;
    #observer;
    constructor(value, observer){
      this.#value = value;
      this.#observer = observer;
    }
    getValue(target, k){return this.#value;}
    setValue(target, k, v){
      const old = this.#value;
      this.#value = v;
      this.#observer(target, k, old, v);
    }
  }
  return (value, observer)=>new Observer(value, observer);
})();
```

```
const observe =(_=>{
  class Observer{
    #value;
    #observer;
    constructor(value, observer){
      this.#value = value;
      this.#observer = observer;
    }
    getValue(target, k){return this.#value;}
    setValue(target, k, v){
      this.#observer(target, k, this.#value, this.#value = v);
    }
  }
  return (value, observer)=>new Observer(value, observer);
})();
```

```
<input id="name">
<input id="company">
<button id="log">log</button>
```

```
const observe =(_=>{
  class Observer{
    #value;
    #observer;
    constructor(value, observer){
      this.#value = value;
      this.#observer = observer;
    }
    getValue(target, k){return this.#value;}
    setValue(target, k, v){
      this.#observer(target, k, this.#value, this.#value = v);
    }
  }
  return (value, observer)=>new Observer(value, observer);
})();
```

```
const Test = by(class{
  static name = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#name").value = v;
  });
  static company = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#company").value = v;
  });
});
```

```
<input id="name">
<input id="company">
<button id="log">log</button>
```

```
const observe =(_=>{
  class Observer{
    #value;
    #observer;
    constructor(value, observer){
      this.#value = value;
      this.#observer = observer;
    }
    getValue(target, k){return this.#value;}
    setValue(target, k, v){
      this.#observer(target, k, this.#value, this.#value = v);
    }
  }
  return (value, observer)=>new Observer(value, observer);
})();
```

```
const Test = by(class{
  static name = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#name").value = v;
  });
  static company = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#company").value = v;
  });
});
```

```
<input id="name">
<input id="company">
<button id="log">log</button>
<script>
const test = new Test;
document.querySelector("#name").onchange =({target:{value}})=>test.name = value;
document.querySelector("#company").onchange =({target:{value}})=>test.company = value;


</script>
```

```
const observe =(_=>{
  class Observer{
    #value;
    #observer;
    constructor(value, observer){
      this.#value = value;
      this.#observer = observer;
    }
    getValue(target, k){return this.#value;}
    setValue(target, k, v){
      this.#observer(target, k, this.#value, this.#value = v);
    }
  }
  return (value, observer)=>new Observer(value, observer);
})();
```

```
const Test = by(class{
  static name = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#name").value = v;
  });
  static company = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#company").value = v;
  });
});
```

```
<input id="name">
<input id="company">
<button id="log">log</button>
<script>
const test = new Test;
document.querySelector("#name").onchange =({target:{value}})=>test.name = value;
document.querySelector("#company").onchange =({target:{value}})=>test.company = value;
test.name = "hika"

</script>
```

```
const observe =(_=>{
  class Observer{
    #value;
    #observer;
    constructor(value, observer){
      this.#value = value;
      this.#observer = observer;
    }
    getValue(target, k){return this.#value;}
    setValue(target, k, v){
      this.#observer(target, k, this.#value, this.#value = v);
    }
  }
  return (value, observer)=>new Observer(value, observer);
})();
```

```
const Test = by(class{
  static name = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#name").value = v;
  });
  static company = observe("", (target, key, old, v)=>{
    if(old === v) return;
    document.querySelector("#company").value = v;
  });
});
```

```
<input id="name">
<input id="company">
<button id="log">log</button>
<script>
const test = new Test;
document.querySelector("#name").onchange =({target:{value}})=>test.name = value;
document.querySelector("#company").onchange =({target:{value}})=>test.company = value;
test.name = "hika"
document.querySelector("#log").onclick =_=>console.log(test.name, test.company);
</script>
```

```
class NetDelegate{









}
```

```
class NetDelegate{
}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }




}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }
  async getValue(target, k){




  }

}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }
  async getValue(target, k){
    if(!NetDelegate.loaded.has(this.#url)){
      NetDelegate.loaded.set(this.#url, await (await fetch(this.#url)).json());
    }

  }

}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }
  async getValue(target, k){
    if(!NetDelegate.loaded.has(this.#url)){
      NetDelegate.loaded.set(this.#url, await (await fetch(this.#url)).json());
    }
    return NetDelegate.loaded.get(this.#url)[k] ?? "no data";
  }

}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }
  async getValue(target, k){
    if(!NetDelegate.loaded.has(this.#url)){
      NetDelegate.loaded.set(this.#url, await (await fetch(this.#url)).json());
    }
    return NetDelegate.loaded.get(this.#url)[k] ?? "no data";
  }
  setValue(target, k, v){}
}
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }
  async getValue(target, k){
    if(!NetDelegate.loaded.has(this.#url)){
      NetDelegate.loaded.set(this.#url, await (await fetch(this.#url)).json());
    }
    return NetDelegate.loaded.get(this.#url)[k] ?? "no data";
  }
  setValue(target, k, v){}
}

const Test = by(class{
  static name = new NetDelegate("test.json");
  static company = new NetDelegate("test.json");
});
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```

```
class NetDelegate{
  static loaded = new Map;
  #url;
  constructor(url){
    this.#url = url;
  }
  async getValue(target, k){
    if(!NetDelegate.loaded.has(this.#url)){
      NetDelegate.loaded.set(this.#url, await (await fetch(this.#url)).json());
    }
    return NetDelegate.loaded.get(this.#url)[k] ?? "no data";
  }
  setValue(target, k, v){}
}

const Test = by(class{
  static name = new NetDelegate("test.json");
  static company = new NetDelegate("test.json");
});


(async()=> {
  const test = new Test();
  console.log(await test.name, await test.company);
})()
```

test.json

```
{
  "name": "hika",
  "company": "bsidesoft"
}
```