


```
<script>
export default {
  data() {
    return {
      name: "John Doe"
    };
  },
  render(h) {
    return h("div", this.name);
  }
};
</script>
```

```
let originalValue = this.data.name;
```

```
Object.defineProperty(this.data, 'name', {  
  get() {  
    // do something  
    return originalValue;  
  },
```

```
    set(newValue) {  
      // do something  
      originalValue = newValue;  
    }  
  });
```



this is data name

this.data.name == 'Richard Roe'





1. The first step in the process of creating a new product is to identify a market need. This involves conducting market research to determine what consumers want and what problems they are trying to solve. Once a need is identified, the next step is to develop a concept that addresses that need. This is often done through brainstorming sessions with a team of designers and engineers. The concept is then refined through prototyping and testing, with feedback from potential users being used to make improvements. Once the concept is finalized, the next step is to develop a business plan that outlines the costs of production, the pricing strategy, and the marketing plan. This plan is then used to secure funding from investors or lenders. Finally, the product is manufactured and distributed to the market. Throughout the process, it is important to maintain communication with potential users and to be flexible in making changes as needed. The goal is to create a product that is not only useful and innovative, but also commercially viable.

this name = Richard Roe!


```
{
```

```
  tag: 'div',
```

```
  children: ['Richard Roe']
```

```
}
```

~~<div>~~Richard</div>~~<div>~~



verified updates DDM

```
<script>
export default {
  data() {
    return {
      name: "John Doe"
    };
  },
  render(h) {
    return h("div", this.name);
  }
};
</script>
```

```
let originalValue = this.data.name;

Object.defineProperty(this.data, 'name', {
  get() {
    // do something
    return originalValue;
  },

  set(newValue) {
    // do something
    originalValue = newValue;
  }
});
```

```
/**
 * Define a reactive property on an Object.
 */
export function defineReactive (
  obj: Object,
  key: string,
  val: any,
  customSetter?: ?Function,
  shallow?: boolean
) {
  const dep = new Dep()

  const property = Object.getOwnPropertyDescriptor(obj, key)
  if (property && property.configurable === false) {
    return
  }

  // cater for pre-defined getter/setters
  const getter = property && property.get
  const setter = property && property.set
  if ((!getter || setter) && arguments.length === 2) {
    val = obj[key]
  }

  let childOb = !shallow && observe(val)
  Object.defineProperty(obj, key, {
    enumerable: true,
    configurable: true,
    get: function reactiveGetter () {
      const value = getter ? getter.call(obj) : val
      if (Dep.target) {
        dep.depend()
      }
    },
    set: function reactiveSetter (newVal) {
      if (getter && getter.call(obj) === newVal) {
        return
      }
      if (!setter) {
        console.warn('setter is not defined, property changed from outside')
        return
      }
      setter.call(obj, newVal)
      if (!shallow && childOb) {
        childOb.value = newVal
      }
    }
  })
}
```