Map

Map是一组键值对的结构，用于解决以往不能用对象做为键的问题

- 具有极快的查找速度
- 函数、对象、基本类型都可以作为键或值

#声明定义

可以接受一个数组作为参数，该数组的成员是一个表示键值对的数组。

```
let m = new Map([
  ['houdunren', '后盾人'],
  ['hdcms', '开源系统']
]);

console.log(m.get('houdunren')); //后盾人

```

使用`set`方法添加元素，支持链式操作

```
let map = new Map();
let obj = {
  name: "后盾人"
};

map.set(obj, "houdunren.com").set("name", "hdcms");

console.log(map.entries()); //MapIterator {{…} => "houdunren.com", "name" => "hdcms"}

```

使用构造函数`new Map`创建的原理如下

```
const hd = new Map();
const arr = [["houdunren", "后盾人"], ["hdcms", "开源系统"]];

arr.forEach(([key, value]) => {
  hd.set(key, value);
});
```

```
console.log(hd);

```

对于键是对象的`Map`， 键保存的是内存地址，值相同但内存地址不同的视为两个键。

```
let arr = ["后盾人"];
const hd = new Map();
hd.set(arr, "houdunren.com");
console.log(hd.get(arr)); //houdunren.com
console.log(hd.get(["后盾人"])); //undefined

```

#获取数量

获取数据数量

```
console.log(map.size);

```

#元素检测

检测元素是否存在

```
console.log(map.has(obj1));

```

#读取元素

```
let map = new Map();

let obj = {
 name: '后盾人'
}

map.set(obj, 'houdunren.com');
console.log(map.get(obj));

```

#删除元素

使用 `delete()` 方法删除单个元素

```
let map = new Map();
let obj = {
        name: '后盾人'
}

map.set(obj, 'houdunren.com');
console.log(map.get(obj));

map.delete(obj);
console.log(map.get(obj));

```

使用`clear`方法清除Map所有元素

```
let map = new Map();
let obj1 = {
 name: 'hdcms.com'
}

let obj2 = {
 name: 'houdunren.com'
}

map.set(obj1, {
 title: '内容管理系统'
});

map.set(obj2, {
 title: '后盾人'
});

console.log(map.size);
console.log(map.clear());
console.log(map.size);

```

#遍历数据

使用 `keys()/values()/entries()` 都可以返回可遍历的迭代对象。

```
let hd = new Map([["houdunren", "后盾人"], ["hdcms", "开源系统"]]);
console.log(hd.keys()); //MapIterator {"houdunren", "hdcms"}
console.log(hd.values()); //MapIterator {"后盾人", "开源系统"}
console.log(hd.entries()); //MapIterator {"houdunren" => "后盾人", "hdcms" => "开源系统"}

```

可以使用`keys/values` 函数遍历键与值

```
let hd = new Map([["houdunren", "后盾人"], ["hdcms", "开源系统"]]);
for (const key of hd.keys()) {
  console.log(key);
}
for (const value of hd.values()) {
  console.log(value);
}

```

使用`for/of`遍历操作，直播遍历Map 等同于使用`entries()` 函数

```
let hd = new Map([["houdunren", "后盾人"], ["hdcms", "开源系统"]]);
for (const [key, value] of hd) {
  console.log(`${key}=>${value}`);
}

```

使用`forEach`遍历操作

```
let hd = new Map([["houdunren", "后盾人"], ["hdcms", "开源系统"]]);
hd.forEach((value, key) => {
  console.log(`${key}=>${value}`);
});

```

#数组转换

可以使用`展开语法`或`Array.form`静态方法将Set类型转为数组，这样就可以使用数组处理函数了

```
let hd = new Map([["houdunren", "后盾人"], ["hdcms", "开源系统"]]);

console.log(...hd); //(2) ["houdunren", "后盾人"] (2) ["hdcms", "开源系统"]
console.log(...hd.entries()); //(2) ["houdunren", "后盾人"] (2) ["hdcms", "开源系统"]
console.log(...hd.values()); //后盾人 开源系统
console.log(...hd.keys()); //houdunren hdcms

```

检索包含`后盾人`的值组成新Map

```
let hd = new Map([["houdunren", "后盾人"], ["hdcms", "开源系统"]]);

let newArr = [...hd].filter(function(item) {
  return item[1].includes("后盾人");
});

hd = new Map(newArr);
console.log(...hd.keys());

```

#节点集合

map的key可以为任意类型，下面使用DOM节点做为键来记录数据。

```
<body>
  <div desc="后盾人">houdunren</div>
  <div desc="开源系统">hdcms</div>
</body>

<script>
  const divMap = new Map();
  const divs = document.querySelectorAll("div");

  divs.forEach(div => {
```

```
    divMap.set(div, {
      content: div.getAttribute("desc")
    });
  });
  divMap.forEach((config, elem) => {
    elem.addEventListener("click", function() {
      alert(divMap.get(this).content);
    });
  });
</script>

```

#实例操作

当不接受协议时无法提交表单，并根据自定义信息提示用户。

```
<form action="" onsubmit="return post()">
    接受协议:
    <input type="checkbox" name="agreement" message="请接受接受协议" />
    我是学生:
    <input type="checkbox" name="student" message="网站只对学生开放" />
    <input type="submit" />
  </form>
</body>

<script>
  function post() {
    let map = new Map();

    let inputs = document.querySelectorAll("[message]");
    //使用set设置数据
    inputs.forEach(item =>
      map.set(item, {
        message: item.getAttribute("message"),
        status: item.checked
      })
    );

    //遍历Map数据
```

```
    return [...map].every(([item, config]) => {
      config.status || alert(config.message);
      return config.status;
    });
  }
</script>

```

#WeakMap

WeakMap 对象是一组键/值对的集

- 键名必须是对象
- WeaMap对键名是弱引用的，键值是正常引用
- 垃圾回收不考虑WeaMap的键名，不会改变引用计数器，键在其他地方不被引用时即删除
- 因为WeakMap 是弱引用，由于其他地方操作成员可能会不存在，所以不可以进行

forEach( )遍历等操作

- 也是因为弱引用，WeaMap 结构没有keys( )，values( )，entries( )等方法和 size 属性
- 当键的外部引用删除时，希望自动删除数据时使用

WeakMap

#声明定义

以下操作由于键不是对象类型将产生错误

```
new WeakSet("hdcms"); //TypeError: Invalid value used in weak set

```

将DOM节点保存到`WeakSet`

```
<body>
  <div>houdunren</div>
  <div>hdcms</div>
</body>
<script>
  const hd = new WeakMap();
  document
    .querySelectorAll("div")
    .forEach(item => hd.set(item, item.innerHTML));
  console.log(hd); //WeakMap {div => "hdcms", div => "houdunren"}
```

```
</script>

```

#基本操作

下面是WeakSet的常用指令

```
const hd = new WeakMap();
const arr = ["hdcms"];
//添加操作
hd.set(arr, "houdunren");
console.log(hd.has(arr)); //true

//删除操作
hd.delete(arr);

//检索判断
console.log(hd.has(arr)); //false

```

#垃圾回收

WakeMap的键名对象不会增加引用计数器，如果一个对象不被引用了会自动删除。

- 下例当

hd删除时内存即清除，因为WeakMap是弱引用不会产生引用计数

- 当垃圾回收时因为对象被删除，这时WakeMap也就没有记录了

```
let map = new WeakMap();
let hd = {};
map.set(hd, "hdcms");
hd = null;
console.log(map);

setTimeout(() => {
  console.log(map);
}, 1000);

```

#选课案例


```
<style>
  * {
    padding: 0;
    margin: 0;
  }
  body {
    padding: 20px;
    width: 100vw;
    display: flex;
    box-sizing: border-box;
  }
  div {
    border: solid 2px #ddd;
    padding: 10px;
    flex: 1;
  }
  div:last-of-type {
    margin-left: -2px;
  }
  ul {
    list-style: none;
    display: flex;
    width: 200px;
    flex-direction: column;
  }
  li {
```

```
    height: 30px;
    border: solid 2px #e67e22;
    margin-bottom: 10px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding-left: 10px;
    color: #333;
    transition: 1s;
  }
  a {
    border-radius: 3px;
    width: 20px;
    height: 20px;
    text-decoration: none;
    text-align: center;
    background: #16a085;
    color: white;
    cursor: pointer;
    display: flex;
    justify-content: center;
    align-items: center;
    margin-right: 5px;
  }
  .remove {
    border: solid 2px #eee;
    opacity: 0.8;
    color: #eee;
  }
  .remove a {
    background: #eee;
  }
  p {
    margin-top: 20px;
  }
  p span {
    display: inline-block;
    background: #16a085;
    padding: 5px;
```

```
    color: white;
    margin-right: 10px;
    border-radius: 5px;
    margin-bottom: 10px;
  }
</style>

<body>
  <div>
    <ul>
      <li><span>php</span> <a href="javascript:;">+</a></li>
      <li><span>js</span> <a href="javascript:;">+</a></li>
      <li><span>向军讲编程</span><a href="javascript:;">+</a></li>
    </ul>
  </div>
  <div>
    <strong id="count">共选了2门课</strong>
    <p id="lists"></p>
  </div>
</body>

<script>
  class Lesson {
    constructor() {
      this.lis = document.querySelectorAll("ul>li");
      this.countELem = document.getElementById("count");
      this.listElem = document.getElementById("lists");
      this.map = new WeakMap();
    }
    run() {
      this.lis.forEach(item => {
        item.querySelector("a").addEventListener("click", event => {
          const elem = event.target;
          const state = elem.getAttribute("select");
          if (state) {
            elem.removeAttribute("select");
            this.map.delete(elem.parentElement);
            elem.innerHTML = "+";
            elem.style.backgroundColor = "green";
          } else {
```

```
            elem.setAttribute("select", true);
            this.map.set(elem.parentElement, true);
            elem.innerHTML = "-";
            elem.style.backgroundColor = "red";
          }
          this.render();
        });
      });
    }
    count() {
      return [...this.lis].reduce((count, item) => {
        return (count += this.map.has(item) ? 1 : 0);
      }, 0);
    }
    lists() {
      return [...this.lis]
        .filter(item => {
          return this.map.has(item);
        })
        .map(item => {
          return `<span>${item.querySelector("span").innerHTML}</span>`;
        });
    }
    render() {
      this.countELem.innerHTML = `共选了${this.count()}课`;
      this.listElem.innerHTML = this.lists().join("");
    }
  }
  new Lesson().run();
</script>

```