---

link: null
title: 珠峰架构师成长计划
description: 弹簧
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=113 sentences=273, words=2520

防抖和节流

- Underscore (https://underscorejs.net/)一个JavaScript实用库
    - debounce (https://underscorejs.net/#debounce)
    - debounce.js (https://github.com/jashkenas/underscore/blob/master/modules/debounce.js)
    - throttle (https://underscorejs.net/#throttle)
    - throttle.js (https://github.com/jashkenas/underscore/blob/master/modules/throttle.js)
- Lodash (https://www.lodashjs.com/)是一个高性能的JavaScript 实用工具库
    - lodash.debounce (https://www.lodashjs.com/docs/lodash.debounce)
    - debounce.js (https://github.com/lodash/lodash/blob/master/debounce.js)
    - lodash.throttle (https://www.lodashjs.com/docs/lodash.throttle)
    - throttle.js (https://github.com/lodash/lodash/blob/master/throttle.js)

1.防抖

1.1 直观感知

弹簧 (https://static.zhufengpeixun.com/tan_huang_dou_dong_qi_lai_1642683011827.mp4)

说话 (https://static.zhufengpeixun.com/fang_zhi_dou_dong_1642678682223.mp4)

1.2. 应用场景

- 输入框搜索
- 按钮的重复点击
- 上拉滚动加载
- 用户的缩放事件

1.3. 工作原理

- 当事件触发时并不会立即执行回调，而是会等待一段时间
- 如果在等待期间再次触发事件，会继续等待
- 只有等待期间无新的事件触发才会执行回调

1.4 代码演示

```
const { debounce } = require('lodash');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
}
const debounced = debounce(logger, 2000);
setTimeout(() => {
    debounced();
}, 1000);
setTimeout(() => {
    debounced();
}, 2000);
setTimeout(() => {
    debounced();
}, 3000);
setTimeout(() => {
    debounced();
}, 4000);
setTimeout(() => {
    debounced();
}, 5000);
```

2.节流

2.1. 直观感知

节流阀门 (https://static.zhufengpeixun.com/jie_liu_fa_men_1642683062075.mp4)

2.2. 应用场景

- 下拉刷新
- 鼠标移动
- 拖拽组件

2.3. 工作原理

2.4 代码演示

```
const { throttle } = require('lodash');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
}
const throttled = throttle(logger, 2990);
setTimeout(() => {
    throttled();
}, 1000);
setTimeout(() => {
    throttled();
}, 2000);
setTimeout(() => {
    throttled();
}, 3000);
setTimeout(() => {
    throttled();
}, 4000);
setTimeout(() => {
    throttled();
}, 5000);
setTimeout(() => {
    throttled();
}, 6000);
```

3.实现防抖

3.1 基本防抖功能

3.1.1 use.js

```
let debounce = require('./debounce');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
}
let debounced = debounce(logger, 2000);
setTimeout(() => {
    debounced();
}, 1000);
setTimeout(() => {
    debounced();
}, 2000);
setTimeout(() => {
    debounced();
}, 3000);
setTimeout(() => {
    debounced();
}, 4000);
setTimeout(() => {
    debounced();
}, 5000);
```

3.1.2 debounce.js

```
function debounce(fn, wait) {
    let timer;
    const debouncedFn = function () {
        if (timer) {
            clearTimeout(timer);
            timer = null;
        }
        timer = setTimeout(fn, wait);
    }
    return debouncedFn;
}

module.exports = debounce;
```

3.2 参数传递

3.2.1 use.js

```
let debounce = require('./debounce');
const start = Date.now();
+function logger(age) {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
+   console.log(this, age);
}
let debounced = debounce(logger, 2000);
+let obj = {
+    name: '张三',
+    debounced
+}
setTimeout(() => {
+   obj.debounced(1000);
}, 1000);
setTimeout(() => {
+   obj.debounced(2000);
}, 2000);
setTimeout(() => {
+   obj.debounced(3000);
}, 3000);
setTimeout(() => {
+   obj.debounced(4000);
}, 4000);
setTimeout(() => {
+   obj.debounced(5000);
}, 5000);
```

3.2.2 debounce.js

```
function debounce(fn, wait) {
    let timer;
    const debouncedFn = function (...args) {
        if (timer) {
            clearTimeout(timer);
            timer = null;
        }
+       timer = setTimeout(() => {
+           fn.apply(this, args);
+       }, wait);
    }
    return debouncedFn;
}
module.exports = debounce;
```

3.3 立即执行

3.3.1 debounce.js

```
+function debounce(fn, wait, immediate = false) {
    let timer;
+   let immediateInvoked = false;
    const debouncedFn = function (...args) {
        if (timer) {
            clearTimeout(timer);
            timer = null;
        }
+       if (immediate && !immediateInvoked) {
+           fn.apply(this, args);
+           immediateInvoked = true;
+       } else {
            timer = setTimeout(() => {
                fn.apply(this, args);
+               immediateInvoked = false;
            }, wait);
        }
    }
    return debouncedFn;
}
module.exports = debounce;
```

3.4 取消任务

3.4.1 use.js

```
let debounce = require('./debounce');
const start = Date.now();
function logger(age) {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
    console.log(this, age);
}
let debounced = debounce(logger, 2000, true);
let obj = {
    name: '张三',
    debounced
}
setTimeout(() => {
    obj.debounced(1000);
}, 1000);
setTimeout(() => {
    obj.debounced(2000);
}, 2000);
setTimeout(() => {
    obj.debounced(3000);
}, 3000);
setTimeout(() => {
    obj.debounced(4000);
}, 4000);
setTimeout(() => {
    obj.debounced(5000);
+   obj.debounced.cancel();
}, 5000);
```

3.4.2 debounce.js

debounce.js

```
function debounce(fn, wait, immediate = false) {
    let timer;
    let immediateInvoked = false;
    const debouncedFn = function (...args) {
        if (timer) {
            clearTimeout(timer);
            timer = null;
        }
        if (immediate && !immediateInvoked) {
            fn.apply(this, args);
            immediateInvoked = true;
        } else {
            timer = setTimeout(() => {
                fn.apply(this, args);
                immediateInvoked = false;
            }, wait);
        }
    }
+   debouncedFn.cancel = function () {
+       if (timer) {
+           clearTimeout(timer);
+           timer = null;
+           immediateInvoked = false;
+       }
+   }
    return debouncedFn;
}
module.exports = debounce;
```

3.5 获取返回值

3.5.1 use.js

```
let debounce = require('./debounce');
const start = Date.now();
function logger(age) {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
    console.log(this, age);
+   return Date.now();
}
+let debounced = debounce(logger, 2000, true, (err, data) => {
+    console.log('callback', data);
+});
let obj = {
    name: '张三',
    debounced
}
setTimeout(() => {
    obj.debounced(1000);
}, 1000);
setTimeout(() => {
    obj.debounced(2000);
}, 2000);
setTimeout(() => {
    obj.debounced(3000);
}, 3000);
setTimeout(() => {
    obj.debounced(4000);
}, 4000);
+setTimeout(() => {
+    obj.debounced(5000).then((data) => {
+        console.log('then', data);
+    });
+    //obj.debounced.cancel();
+}, 5000);
```

3.5.2 debounce.js

```
+function debounce(fn, wait, immediate = false, callback = () => { }) {
+   if (typeof immediate === 'function') {
+       callback = immediate;
+       immediate = false;
+   }
    let timer;
    let immediateInvoked = false;
    const debouncedFn = function (...args) {
+       return new Promise((resolve, reject) => {
            if (timer) {
                clearTimeout(timer);
                timer = null;
            }
            if (immediate && !immediateInvoked) {
                try {
+                   const result = fn.apply(this, args);
+                   callback(null, result);
+                   resolve(result);
+               } catch (error) {
+                   callback(error);
+                   reject(error);
+               } finally {
+                   immediateInvoked = true;
+               }
            } else {
                timer = setTimeout(() => {
                    try {
+                       const result = fn.apply(this, args);
+                       callback(null, result);
+                       resolve(result);
+                   } catch (error) {
+                       callback(error);
+                       reject(error);
+                   } finally {
+                       immediateInvoked = false;
+                   }
                }, wait);
            }
+       });

    }
    debouncedFn.cancel = function () {
        if (timer) {
            clearTimeout(timer);
            timer = null;
            immediateInvoked = false;
        }
    }
    return debouncedFn;
}

module.exports = debounce;
```

4.实现节流

4.1 基本节流功能

4.1.1 use.js

```
const throttle = require('./throttle');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
}
let throttled = throttle(logger, 2990);
setTimeout(() => {
    throttled();
}, 1000);
setTimeout(() => {
    throttled();
}, 2000);
setTimeout(() => {
    throttled();
}, 3000);
setTimeout(() => {
    throttled();
}, 4000);
setTimeout(() => {
    throttled();
}, 5000);
setTimeout(() => {
    throttled();
}, 6000);
```

4.1.2 throttle.js

```
function throttle(fn, wait) {
    let lastExecTime = 0;
    const throttledFn = function (...args) {
        const currentTime = Date.now();
        const nextExecTime = lastExecTime + wait;
        if (currentTime >= nextExecTime) {
            fn.apply(this, args);
            lastExecTime = currentTime;
        }
    }
    return throttledFn;
}
module.exports = throttle;
```

4.2 leading

4.2.1 use.js

```
//let { throttle } = require('lodash');
const throttle = require('./throttle');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
}
+let throttled = throttle(logger, 2990, false);
setTimeout(() => {
    throttled();
}, 1000);
setTimeout(() => {
    throttled();
}, 2000);
setTimeout(() => {
    throttled();
}, 3000);
setTimeout(() => {
    throttled();
}, 4000);
setTimeout(() => {
    throttled();
}, 5000);
setTimeout(() => {
    throttled();
}, 6000);
```

4.2.2 throttle.js

```
function throttle(fn, wait, options = { leading: true }) {
+   const { leading } = options;
    let lastExecTime = 0;
    const throttledFn = function (...args) {
        const currentTime = Date.now();
+       if (lastExecTime === 0 && !leading) {
+           lastExecTime = currentTime;
+       }
        const nextExecTime = lastExecTime + wait;
        if (currentTime >= nextExecTime) {
            fn.apply(this, args);
            lastExecTime = currentTime;
        }
    }
    return throttledFn;
}
module.exports = throttle;
```

4.3 trailing

4.3.1 use.js

```
//let { throttle } = require('lodash');
const throttle = require('./throttle');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
}
+let throttled = throttle(logger, 2990, { leading: false, trailing: false });
setTimeout(() => {
    throttled();
}, 1000);
setTimeout(() => {
    throttled();
}, 2000);
setTimeout(() => {
    throttled();
}, 3000);
setTimeout(() => {
    throttled();
}, 4000);
setTimeout(() => {
    throttled();
}, 5000);
setTimeout(() => {
    throttled();
}, 6000);
```

4.3.2 throttle.js

```
+function throttle(fn, wait, options = { leading: true, trailing: true }) {
+   const { leading, trailing } = options;
    let lastExecTime = 0;
+   let timer;
    const throttledFn = function (...args) {
        const currentTime = Date.now();
       if (lastExecTime
            lastExecTime = currentTime;
        }
        const nextExecTime = lastExecTime + wait;
+       if (currentTime >= nextExecTime) {
+           if (timer) { clearTimeout(timer); timer = null; }
            fn.apply(this, args);
            lastExecTime = currentTime;
+       } else {
+           if (trailing) {
+               if (timer) { clearTimeout(timer); timer = null; }
+               timer = setTimeout(() => {
+                   fn.apply(this, args);
+                   lastExecTime = Date.now();
+               }, (nextExecTime - currentTime));
+           }
+       }
    }
    return throttledFn;
}
module.exports = throttle;
```

4.4 取消和返回值

4.4.1 use.js

```
//let { throttle } = require('lodash');
const throttle = require('./throttle');
const start = Date.now();
function logger() {
    console.log((Math.floor((Date.now() - start) / 1000)) + 's');
+   return Date.now();
}
let throttled = throttle(logger, 2990, {
    leading: true,
    trailing: true,
+   callback: (err, data) => {
+       console.log('callback', data);
+   }
});
setTimeout(() => {
    throttled();
}, 1000);
setTimeout(() => {
    throttled();
}, 2000);
setTimeout(() => {
    throttled();
}, 3000);
setTimeout(() => {
    throttled();
}, 4000);
setTimeout(() => {
    throttled();
}, 5000);
setTimeout(() => {
+   throttled().then(data => {
+       console.log('promise', data);
+   });
+   //throttled.cancel();
}, 6000);
```

4.4.2 throttle.js

```
+function throttle(fn, wait, options = { leading: true, trailing: true, callback: () => { } }) {
+    const { leading, trailing, callback } = options;
    let lastExecTime = 0;
    let timer;
+   const throttledFn = function (...args) {
+       return new Promise((resolve, reject) => {
            const currentTime = Date.now();
            if (lastExecTime
                lastExecTime = currentTime;
            }
            const nextExecTime = lastExecTime + wait;
            if (currentTime >= nextExecTime) {
                if (timer) { clearTimeout(timer); timer = null; }
+                try {
+                    const result = fn.apply(this, args);
+                    callback(null, result);
+                    resolve(result);
+                } catch (error) {
+                    callback(error);
+                    reject(error)
+                }
                lastExecTime = currentTime;
            } else {
                if (trailing) {
                    if (timer) { clearTimeout(timer); timer = null; }
                    timer = setTimeout(() => {
+                       try {
+                           const result = fn.apply(this, args);
+                           callback(null, result);
+                           resolve(result);
+                       } catch (error) {
+                           callback(error);
+                           reject(error)
+                       }
                        lastExecTime = Date.now();
                    }, (nextExecTime - currentTime));
                }
            }
+       });

    }
    throttledFn.cancel = function () {
        if (timer) {
            clearTimeout(timer);
            timer = null;
        }
    }
    return throttledFn;
}
module.exports = throttle;
```

5.lodash

5.1 isObject.js

```
export default function isObject(obj) {
    var type = typeof obj;
    return type === 'function' || type === 'object' && !!obj;
}
```

5.2 debounce.js

```
import isObject from './isObject.js'

function debounce(func, wait, options) {
    let lastArgs,
        lastThis,
        maxWait,
        result,
        timerId,
        lastCallTime

    let lastInvokeTime = 0
    let leading = false
    let maxing = false
    let trailing = true

    const useRAF = (!wait && wait !== 0 && typeof requestAnimationFrame === 'function')

    if (typeof func !== 'function') {
        throw new TypeError('Expected a function')
    }
    wait = +wait || 0
    if (isObject(options)) {
        leading = !!options.leading
        maxing = 'maxWait' in options
        maxWait = maxing ? Math.max(+options.maxWait || 0, wait) : maxWait
        trailing = 'trailing' in options ? !!options.trailing : trailing
    }

    function invokeFunc(time) {
        const args = lastArgs
        const thisArg = lastThis

        lastArgs = lastThis = undefined
        lastInvokeTime = time
        result = func.apply(thisArg, args)
        return result
    }

    function startTimer(pendingFunc, wait) {
        if (useRAF) {
```

```
            cancelAnimationFrame(timerId)
            return requestAnimationFrame(pendingFunc)
        }
        return setTimeout(pendingFunc, wait)
    }

    function cancelTimer(id) {
        if (useRAF) {
            return cancelAnimationFrame(id)
        }
        clearTimeout(id)
    }

    function leadingEdge(time) {

        lastInvokeTime = time

        timerId = startTimer(timerExpired, wait)

        return leading ? invokeFunc(time) : result
    }

    function remainingWait(time) {
        const timeSinceLastCall = time - lastCallTime
        const timeSinceLastInvoke = time - lastInvokeTime
        const timeWaiting = wait - timeSinceLastCall

        return maxing
            ? Math.min(timeWaiting, maxWait - timeSinceLastInvoke)
            : timeWaiting
    }

    function shouldInvoke(time) {
        const timeSinceLastCall = time - lastCallTime
        const timeSinceLastInvoke = time - lastInvokeTime

        return (lastCallTime === undefined || (timeSinceLastCall >= wait) ||
            (timeSinceLastCall < 0) || (maxing && timeSinceLastInvoke >= maxWait))
    }

    function timerExpired() {
        const time = Date.now()
        if (shouldInvoke(time)) {
            return trailingEdge(time)
        }

        timerId = startTimer(timerExpired, remainingWait(time))
    }

    function trailingEdge(time) {
        timerId = undefined

        if (trailing && lastArgs) {
            return invokeFunc(time)
        }
        lastArgs = lastThis = undefined
        return result
    }

    function cancel() {
        if (timerId !== undefined) {
            cancelTimer(timerId)
        }
        lastInvokeTime = 0
        lastArgs = lastCallTime = lastThis = timerId = undefined
    }

    function flush() {
        return timerId === undefined ? result : trailingEdge(Date.now())
    }

    function pending() {
        return timerId !== undefined
    }

    function debounced(...args) {
        const time = Date.now()
        const isInvoking = shouldInvoke(time)

        lastArgs = args
        lastThis = this
        lastCallTime = time

        if (isInvoking) {
            if (timerId === undefined) {
                return leadingEdge(lastCallTime)
            }
            if (maxing) {

                timerId = startTimer(timerExpired, wait)
                return invokeFunc(lastCallTime)
            }
        }
        if (timerId === undefined) {
            timerId = startTimer(timerExpired, wait)
        }
        return result
    }
    debounced.cancel = cancel
    debounced.flush = flush
    debounced.pending = pending
    return debounced
}
```

```
export default debounce
```

5.3 throttle.js

```
import debounce from './debounce.js'
import isObject from './isObject.js'

function throttle(func, wait, options) {
    let leading = true
    let trailing = true

    if (typeof func !== 'function') {
        throw new TypeError('Expected a function')
    }
    if (isObject(options)) {
        leading = 'leading' in options ? !!options.leading : leading
        trailing = 'trailing' in options ? !!options.trailing : trailing
    }
    return debounce(func, wait, {
        leading,
        trailing,
        'maxWait': wait
    })
}

export default throttle
```