函数防抖和函数节流

防抖(debounce)

- 如果下达该命令后，在t毫秒内再次下达该命令，则取消刚刚下达的命令，只执行新命令
- 最终效果： 对于连续动作(动作间的时间间隔小于t)，以最后一次为准

范例

```
let command
document.body.onscroll = () => {
    console.log('这里每次都执行')
    if(command)
        clearTimeout(command)
    command = setTimeout(() => {
        console.log('这里只执行很少次')
    }, 1000)
}
```

查看完整代码

封装

```
function debounce(fn, wait) {
    let timer = null
    return function() {
        if(timer)
            clearTimeout(timer)
        timer = setTimeout(() => fn.apply(this, arguments), wait)
    }
}

let fn = () => console.log('这里只执行很少次')
fn = debounce(fn, 1000)

document.body.onscroll = fn
```

查看完整代码

节流(throttle)

- 从上一次命令结束开始的一定时间范围t内，如果多次连续下达命令，则只执行当前时间段t内第一次命令。
- 最终效果：对于连续动作，会过滤出部分动作，让这些过滤后的动作之间的执行间隔大于等于t

范例

```
let gapTime = 1000
let lastTime = null
let nowTime = null
let fn = () => console.log('我执行了')
document.body.onscroll = () => {
    nowTime = Date.now()
    if(!lastTime || nowTime - lastTime > gapTime) {
        fn()
        lastTime = nowTime
    }

}
```

完整代码

封装

```
function throttle(fn, gapTime) {
    let lastTime = null
    let nowTime = null
    return function() {
        nowTime = Date.now()
        if(!lastTime || nowTime - lastTime > gapTime) {
            fn()
            lastTime = nowTime
        }
    }
}

let fn = () => console.log('我执行了')
fn = throttle(fn, 1000)

document.body.onscroll = fn
```

完整代码

案例

用户在连续输入文字时如何实现自动保存，兼顾性能(保存频率不能太高)与可用性(边输入边保存)?

```
<!DOCTYPE html>
<html>
<head>
<meta name="description" content="节流自动保存文件案例" />
  <meta charset="utf-8">
  <title>节流自动保存文件案例</title>
</head>
<body>
  <textarea id="box" cols="50" rows="10"></textarea>
  <p>保存<span id="count">0</span>次</p>

  <script>
    const $ = s => document.querySelector(s)

    let count = 0
    save = throttle(save, 1000*3)
    $('#box').oninput = function() {
      save()
    }

    function save() {
      $('#count').innerText = ++count
    }

    function throttle(fn, gapTime) {
      let lastTime = null
      let nowTime = null
      return function() {
        nowTime = Date.now()
        if(!lastTime || nowTime - lastTime > gapTime) {
          fn()
          lastTime = nowTime
        }
      }
    }

    function debounce(fn, wait) {
      let timer = null
      return function() {
        if(timer)
          clearTimeout(timer)
          timer = setTimeout(() => {
            fn.apply(this, arguments)
```

```
                }, wait)
            }
        }
    </script>
</body>
</html>
```

查看效果

案例

用户在连续输入文字时如何实现自动保存，兼顾性能(保存频率不能太高)与可用性(边输入边保存)?

```
<!DOCTYPE html>
<html>
<head>
<meta name="description" content="节流自动保存文件案例" />
  <meta charset="utf-8">
  <title>节流自动保存文件案例</title>
</head>
<body>
  <textarea id="box" cols="50" rows="10"></textarea>
  <p>保存<span id="count">0</span>次</p>

  <script>
    const $ = s => document.querySelector(s)

    let count = 0
    save = throttle(save, 1000*3)
    $('#box').oninput = function() {
      save()
    }

    function save() {
      $('#count').innerText = ++count
    }

    function throttle(fn, gapTime) {
      let lastTime = null
      let nowTime = null
      return function() {
        nowTime = Date.now()
        if(!lastTime || nowTime - lastTime > gapTime) {
          fn()
          lastTime = nowTime
        }
      }
    }

    function debounce(fn, wait) {
      let timer = null
      return function() {
        if(timer)
          clearTimeout(timer)
          timer = setTimeout(() => {
            fn.apply(this, arguments)
```

```
              }, wait)
          }
        }
      </script>
    </body>
    </html>
```

查看效果