高级node第六单元（6.3）

2020年6月2日 16:05

2,8,16,10进制数字的表示方法

```
// 十进制
const num10 = 129;
// 二进制 以0b开头
const num2 = 0b110;
// 八进制 以0开头
const num8 = 076;
// 十六进制 以0x开头
const num16 = 0xf23;
```

js中内置的进制相互转换的方法

数字.toString(目标进制)
Return <string>

```
// 16进制转10进制
console.log(num16.toString(10))

// 16进制转2进制
console.log(num16.toString(2));

// 10进制转换成2进制
console.log(num10.toString(2));
```

```
100011

D:\software\ph
λ node 15.不同
35
100011
100011

D:\software\ph
λ
```

Buffer相关的api

Buffer是一个单独的内存区域，Buffer类提供了可以直接操作字节的方式，Buffer不需要引入

静态方法和属性

1. Buffer.alloc(size) 申请一块指定大小的存储空间
2. Buffer.byteLength(string | buffer[,encoding]) 获取字节长度
3. Buffer.compare(buf1, buf2) => 1 0 -1 可以对buf进行排序
4. Buffer.concat(list[,length]) 合并buf
5. Buffer.from(string | array | buffer) => buffer 转换并生成buffer
6. Buffer.poolSize 默认为8192 缓冲区buffer实例的默认大小
7. Buffer.isEncoding(encoding) 判断是否支持该编码格式
8. Buffer.isBuffer(obj) 判断该对象是否是buffer实例

实例方法和属性

1. Buf.fill(string | buffer[,offset[,end[,encoding]]]) => buffer 给buffer中填充内容
2. Buf[index] 通过下标的方式可以获取也可以修改字节
3. Buf.copy(target[,targetStart[,sourceStart[,sourceEnd]]]) 把buf中的内容拷贝进target中
4. Buf.entries() buf的迭代器

```
// buf迭代器

const buf11 = Buffer.from('abc');
let itr = buf11.entries() // 返回一个迭代器对象
// 该迭代器对象有一个next方法，该方法可以不断的调用
// 1. 使用下标进行遍历
for (let i = 0; i < Buffer.byteLength(buf11); i++) {
  console.log(buf11[i]);
}
// 2. 使用迭代器进行遍历
let doneG = null;
do {
  let { value, done } = itr.next()
  doneG = done;
  console.log(value);
} while (!doneG)
// 3. 使用专用的for of语句处理迭代器
for (let b of buf11.entries()) {
  console.log(b);
}
```

5. Buf.includes(string | buffer[,byteoffset[,encoding]]) => boolean 用于匹配内容在buffer中是否存在
6. Buf.indexOf(string | buffer[,byteoffset[,encoding]]) => number 用于匹配内容在buffer中的索引
7. Buf.keys() 一个迭代器，但是仅仅遍历出字节所对应的索引
8. Buf.length 返回buf的字节长度 同 Buffer.byteLength(buf)
9. Buf.slice/subArray([start[,end]]) 截取buffer，相当于浅拷贝
10. Buf.toString() 转换buffer为字符串
11. Buf.write(string, [start[,end][,encoding]]) => number 给buffer中写入字符串

简单使用ws模块，实现websocket功能

服务端代码

```
const WebSocket = require('ws');

const wsServer = new WebSocket.Server({ port: 8080 }); // 创建一个ws服务器
const clients = [];
wsServer.on('connection', (client) => {
  // console.log('已经与浏览器建立链接了');
  clients.push(client)
  console.log(`当前链接人数是${clients.length}`)
  client.on('message', (msg) => {
    // 得到某一方发送的消息后，发给另外的人
    clients.filter(item => item !== client).forEach(item => {
      item.send(msg);
    });
  })
})
```

浏览器端代码

```
function createConection() {
  console.log('aaaa')
  // 与服务器建立链接
  client = new WebSocket('ws://localhost:8080')
  console.log(oBox);
  oBox.style.display = 'block'

  client.onmessage = (msg) => {
    console.log(msg.data);
  }
}
function sendMsg() {
  // 给服务器发消息
  client.send(oText.value);
}
```