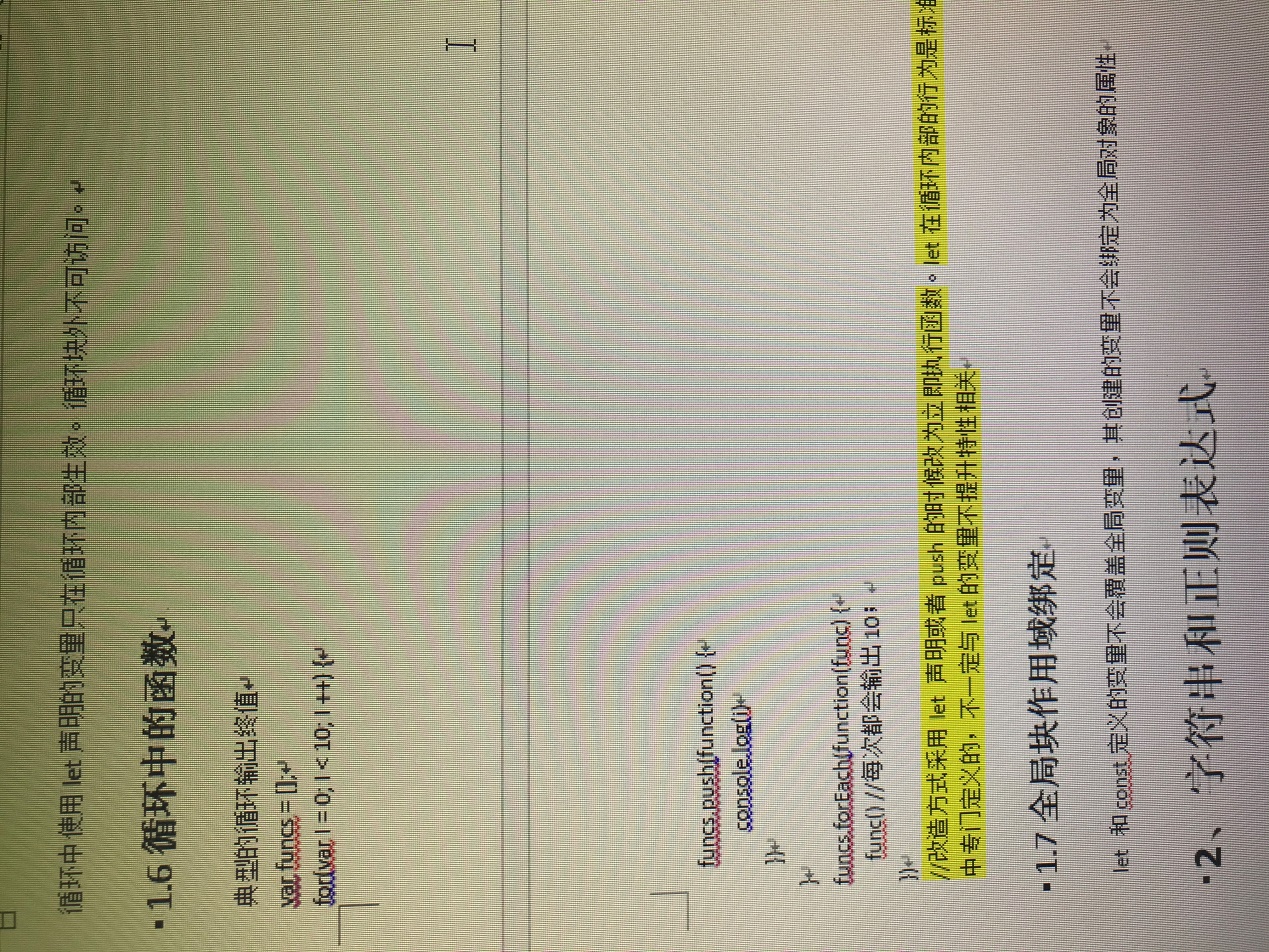
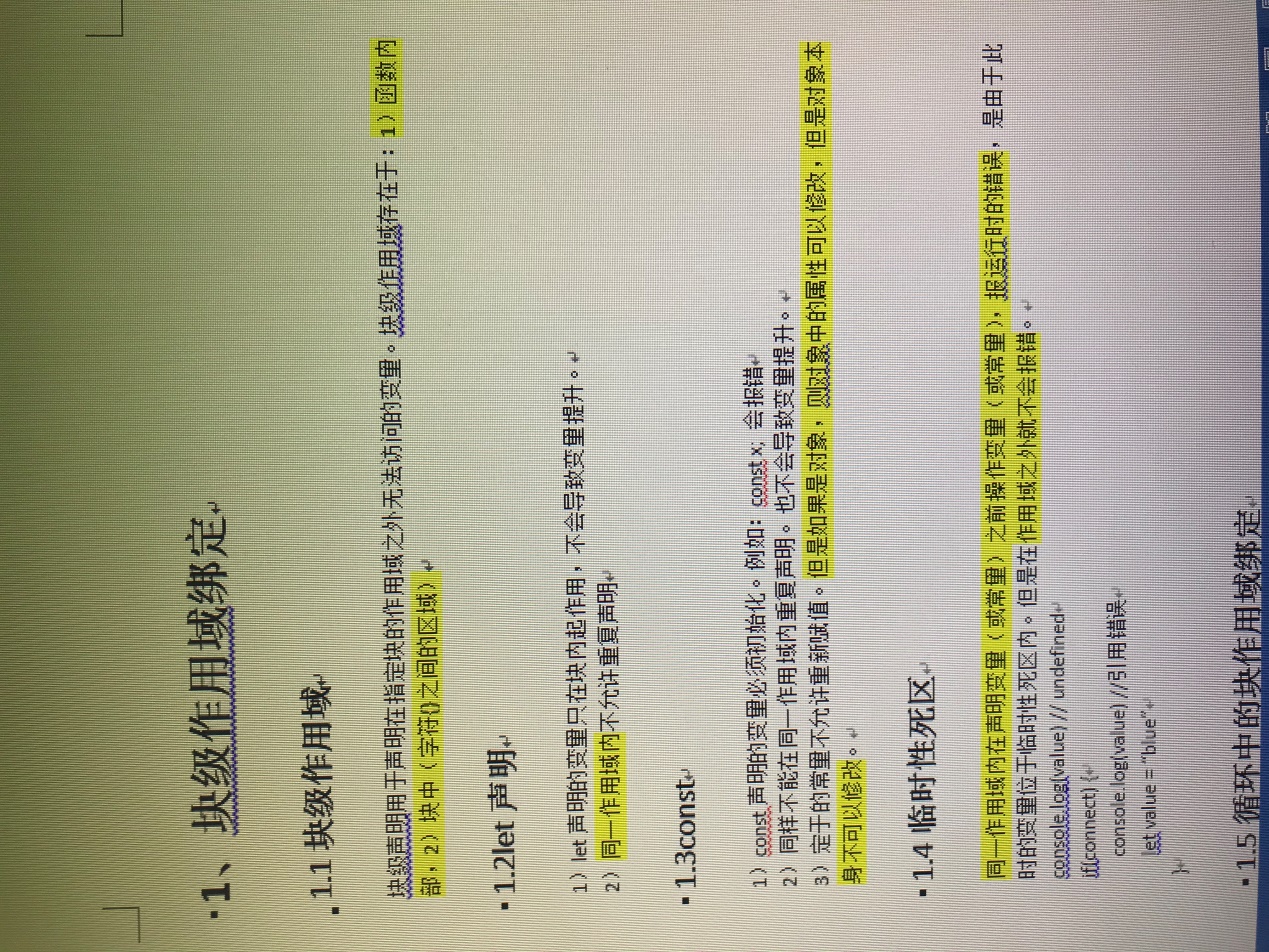
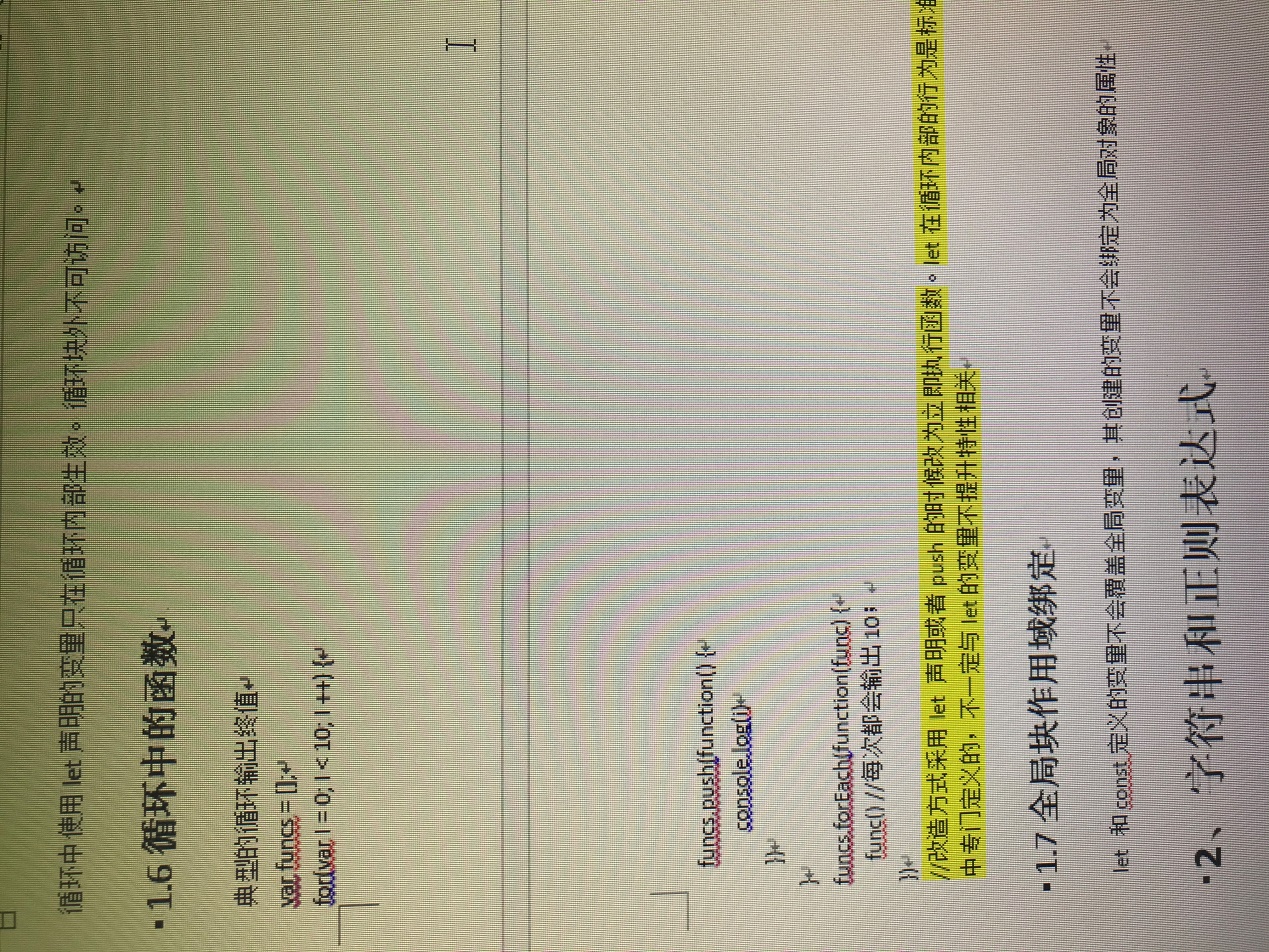
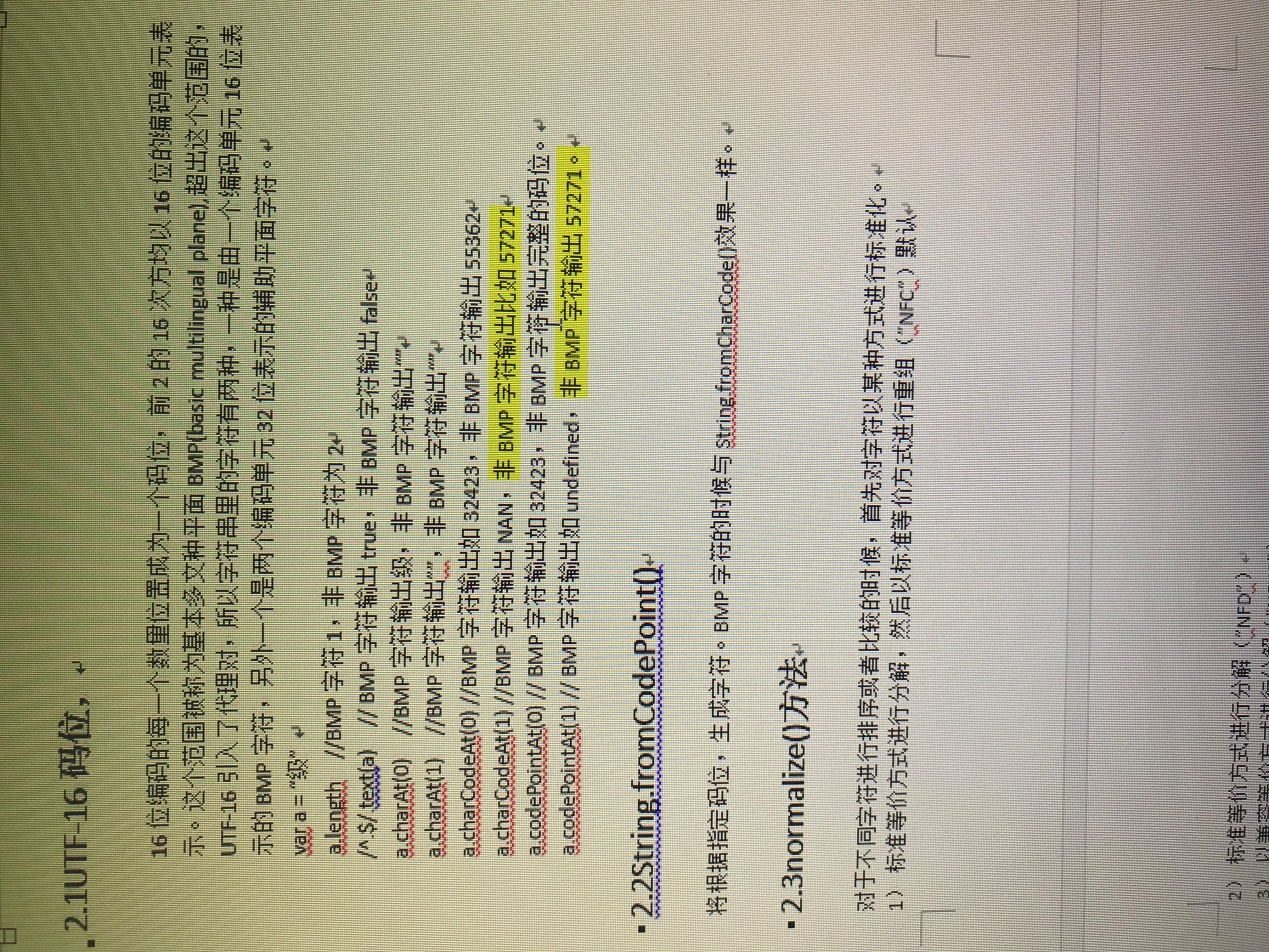
Es6笔记

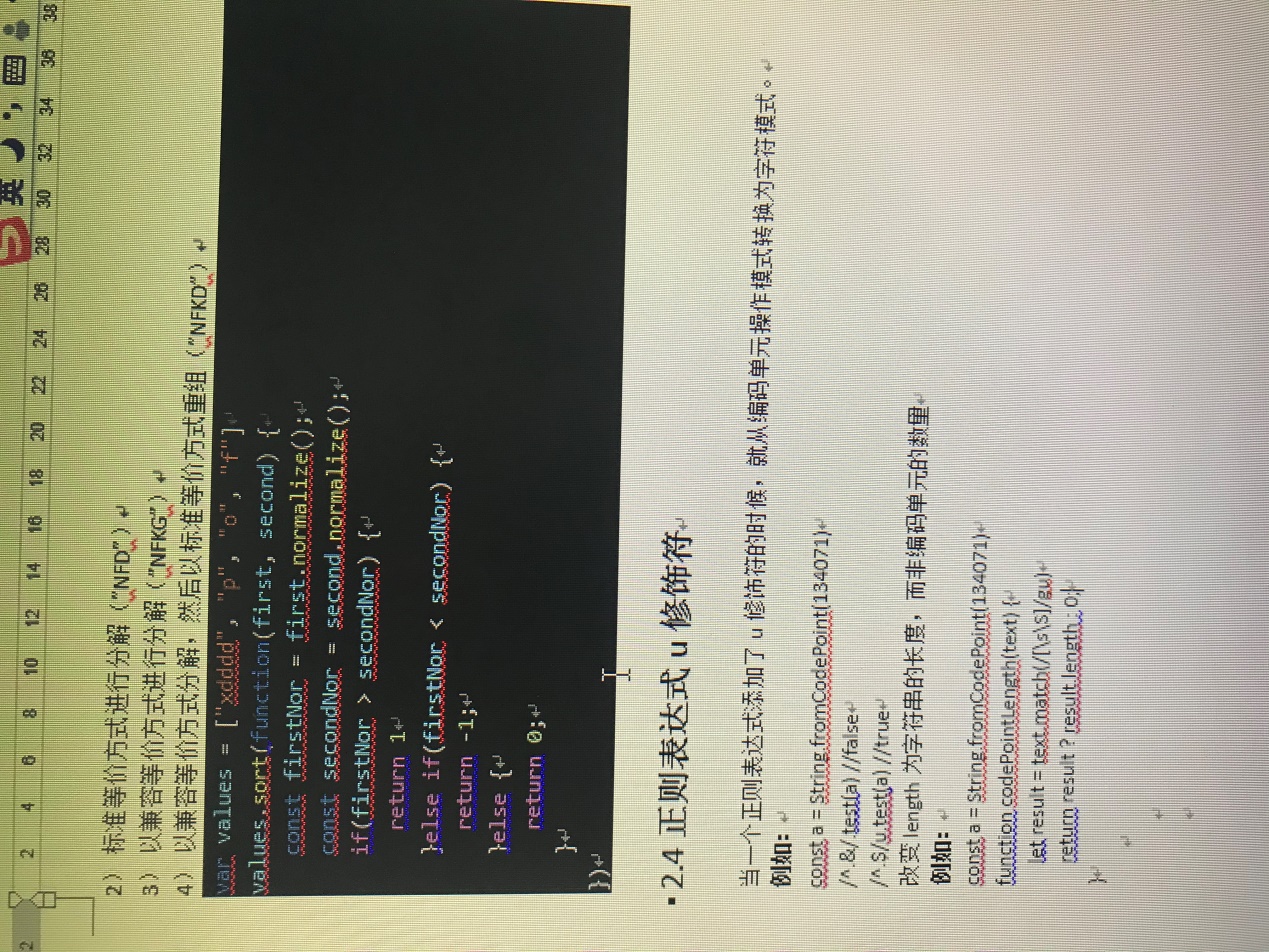
# 1、块级作用域绑定





# 2、字符串和正则表达式





**2.5其他字符串变更**

1）str.includes(childstr,startIndex) //返回true或者false

2）startWith(childStr, startIndex) //true或者false

3）endWith(childStr,index) //true或者false 如果传第二个参数，则从index-childStr.length的位置开始正向匹配

4）lastIndexOf(childStr)和indexOf(chilsStr)能够传入正则表达式作为参数，1）2）3）则不可以。

5）str.repeat(number) //返回字符串重复number次以后的字符串

**2.6正则表达式y修饰符**

1）

let text = "hello1 hello2 hello3";

let parttern = /hello\d\s?/;

let result = parttern.exec(text);

console.log(parttern.lastIndex) //此时parttern.lastIndex为0；

parttern.lastIndex = 1; //修改lastIndex为1，parttern会忽略；

result = parttern.exec(text);

console.log(parttern.lastIndex) //为0

console.log(result[0]) //parttern会忽略，所以返回还是"hello1 "

let globalPattern = /hello\d\s?/g;

let globalResult = globalPattern.exec(text);

console.log(globalPattern.lastIndex) //此时globalPattern的lastIndex为7；

globalPattern.lastIndex = 1; //修改lastIndex为1，globalPattern不会忽略，而是更改为1；

globalResult = globalPattern.exec(text); //从字母e开始匹配，往后找到"hello2 "匹配成功

console.log(globalPattern.lastIndex) //此时为14；

console.log(globalResult[0]) //"hello2 "

let stickyPattern = /hello\d\s?/y;

let stickyResult = stickyPattern.exec(text);

console.log(stickyPattern.lastIndex) //此时stickyPattern的lastIndex为7；

stickyPattern.lastIndex = 1; //修改lastIndex为1，stickyPattern不会忽略，而是更改为1；

stickyResult = stickyPattern.exec(text) //从字母e开始匹配，没有匹配到，返回null,如果上一步lastIndex不做修改，则此时lastIndex为14；

console.log(stickyPattern.lastIndex); //此时为0

console.log(stickyResult) //为null

注：y修饰符只对正则表达式的方法exec()，test()方法生效，对字符串的方法match()不起作用,对于粘制表达式（含有y修饰符）而言，如果正则中使用^来匹配，则只会从字符串的起始位置或者多行的首行开始匹配。

2）检测y修饰符是否存在

let pattern = /hello\d/y;

console.log(pattern.sticky)如果支持粘制修饰符，则sticky为true,此属性为只读属性。

function hasRegExp() { //判断js引擎是否支持y修饰符。

try{

new RegExp(“.”, y)

return true;

}catch(err) {return false}

}

**2.7复制正则表达式**

es6中

let reg1 = /ab/i;

let reg2 = new RegExp(reg1, “g”) //可以更改修饰符，不更改的话和reg1保持一样。

2.8 flags和source

let reg1 = /ab/i reg1.source //”ab” reg1.flags //i

**2.8模板字面量``**

1）实现多行字符串：var messge = `Hello

world`

2）占位符：`${任意的js表达式}`

3）模板标签:标签例如下面的tag就是

const fs = require("fs");

const path = require("path")

const writeFile = path.resolve(\_\_dirname, "./text.txt");

const name ="zhang"

const age = 10;

function tag(iterals, ...subArr) { //iterals数组有三个参数：a第一个`后面第一个占位符前面的””

//b第一个占位符后面和第二个占位符之间的字符串

//c第二个占位符后面的

//当然如果占位符有多个，数组元素个数也随之更改

//iteral有个等价iteral.raw数组，包含它的原生字符串信息。

// subArr有两个元素，分别对应占位符解析后的值

let str = "";

for(let i = 0; i < subArr.length; i++) {

str+=iterals[i]

str+=subArr[i]

}

str+= iterals[iterals.length-1];

fs.writeFile(writeFile, str, {encoding: "utf-8"}, function(err, data) {

console.log(err)

console.log(data)

})

// return str;

}

tag `My name is ${name}, my age is ${age}` //注意这里的调用方式，不要加“()”

3）String.raw() 字面量等价的字符串的原生字符串，用于输出含有代码的字符串，代码中含有转义字符的时候，不需要在执行的时候转义。

如：String.raw `Multiline\nstrimg` //” Multiline\nstrimg”

# 3、函数

3.1es6的默认参数：默认参数可以是任意位置元素。并且null不会使用默认参数