

ES6语法

禁止变量提升-----let 关键字

```
console.log(a);  
var a=10;  
//***等价于***  
var a;  
console.log(a);  
a=10;
```

引入 `let` 关键字，解决变量提升

注释：小程序中会将ES6转为ES5

- `let` 定义的变量只在当前代码块内有效

常量 `const` 关键字

定义常量 `const` 关键字

- 内容允许改变，引用不允许改变

例如：

```
//允许  
const mylist = [1,2,3];  
mylist.push(4);  
console.log(mylist);  
//不允许  
const PI = 3.14;  
PI = 3;
```

函数的默认参数

```
function person(name , age=18) {  
    .....  
}
```

`person('Li',20);`

- 传参数

`person('Li');`

- 使用默认参数

解构赋值

```
funcdemo: function(){  
  function person(name, { age = 18, gender = "男", height = 170 }){  
    console.log("=====");  
    console.log(name, age, gender, height);  
    console.log("=====");  
  }  
  person('知了', { height: 180 });  
},
```

参数为对象，传参数时也传对象----对象内可以改变里面属性的值。

- 形参为对象时必须传入实参

箭头函数


- 向函数传入函数为参数时

```
function request(url, success) {  
  .....  
  if (success) {  
    success([1, 2, 3, 4]);  
  }  
}
```

- 调用函数

```
request("[url]", res => {  
  console.log(res);  
});
```

- 结果

- 

promise风格

```

promisedemo: function(){
  let p = new Promise((resolve,reject) => {
    setTimeout(() => {
      // 模拟网络请求的回调
      resolve([1,2,3,4]);
    },1000);
  });

  p.then(res => {
    console.log(res);
    console.log("请求成功啦! ");
  });
},

```

- 成功

p.then(...)

- 失败

p.then().catch(...)

类 Class

```

Person : class {
  constructor() {}
  function1() {}
  function2() {}
  static function3() {}
}

```

模块导入

export ---- 导出

import ---- 导入

```

//utils.js
var name = "zhiliao";
function dateFormat(date) {
  .....
}

class Person {
  constructor(name, age) {
    .....
  }
}

```

```
}  
export {name, dateFormat, Person}
```

//from 后面是一个相对路径

```
import {name, dateFormat} from "utils.js"
```