赋值

① let [a, b, c] = [1, 2, 3] → 数组, Set, Iterator 两边形式相同.

初始化: let [a=1, b=a, c]=2]

有值取值, 全没值默认, 最后 undefined.

②

```
let { bar, foo } = { foo: 'aaa', bar: 'bbb' };
foo // "aaa"
bar // "bbb"
```

eg.

```
let obj = {
  p: [
    'Hello',
    { y: 'World' }
  ]
};

let { p, p: [x, { y }] } = obj;
x // "Hello"
y // "World"
p // ["Hello", {y: "World"}]
```

← 从obj中取值.

字符：

\u{xxxx} \u{xxxx} ……

eg.

```
'\z' === 'z'  // true
'\172' === 'z' // true
'\x7A' === 'z' // true
'\u007A' === 'z' // true
'\u{7A}' === 'z' // true
```

* 只能转义

U+005C：反斜杠（reverse solidus）
U+000D：回车（carriage return）
U+2028：行分隔符（line separator）
U+2029：段分隔符（paragraph separator）
U+000A：换行符（line feed）

模板字符串.

反引号``

变量名写在${}里.

数组

克隆数组.

eg.

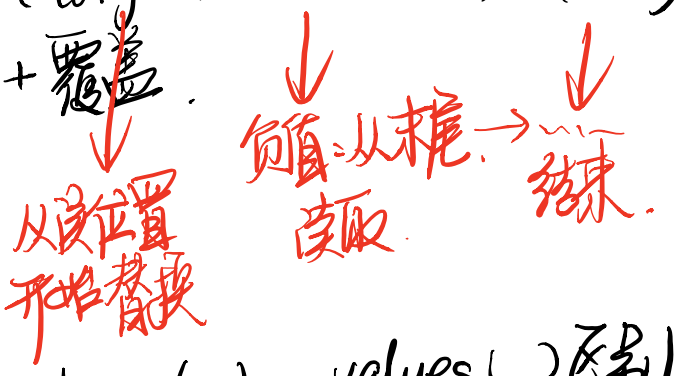
```
const a1 = [1, 2];
// 写法一
const a2 = [...a1];
// 写法二
const [...a2] = a1;
```

Array.from 类似数组. } → 数组.

~~~、of 一组值.

copy Within (target, start= , end=)

复制+覆盖.



entries(), keys(), values() 区别.

eg.

```
for (let index of ['a', 'b'].keys()) {
  console.log(index);
}
// 0
// 1

for (let elem of ['a', 'b'].values()) {
  console.log(elem);
}
// 'a'
// 'b'

for (let [index, elem] of ['a', 'b'].entries()) {
  console.log(index, elem);
}
// 0 "a"
// 1 "b"
```

flat()

eg.
~~~

```
[1, [2, [3]]].flat(Infinity)
// [1, 2, 3]
```

空位.

*

```
0 in [undefined, undefined, undefined] // true
0 in [, , ,] // false
```

Array.from( )
(...)
} ⇒ undefined.

对象

类似于Java.

* super

eg.

```
const proto = {
  foo: 'hello'
};

const obj = {
  foo: 'world',
  find() {
    return super.foo;
  }
};

Object.setPrototypeOf(obj, proto);
obj.find() // "hello"
```

函数.

与Java相似

* length
  有默认值后length失真

eg.

```
(function (a) {}).length // 1
(function (a = 5) {}).length // 0
(function (a, b, c = 5) {}).length // 2
```

箭头

eg.

```
var sum = (num1, num2) => { return num1 + num2; }
```

对应新增方法

Object.is()

类似于===

区别：

```
+0 === -0 //true
NaN === NaN // false

Object.is(+0, -0) // false
Object.is(NaN, NaN) // true
```

```
Object.assign([1, 2, 3], [4, 5])
// [4, 5, 3]
```

Promise

对象构造函数

eg、

```
const promise = new Promise(function(resolve, reject) {
  // ... some code

  if (/* 异步操作成功 */){
    resolve(value);
  } else {
    reject(error);
  }
});
```

方法n

Module

@export var firstName = ''

外部输出变量

or.

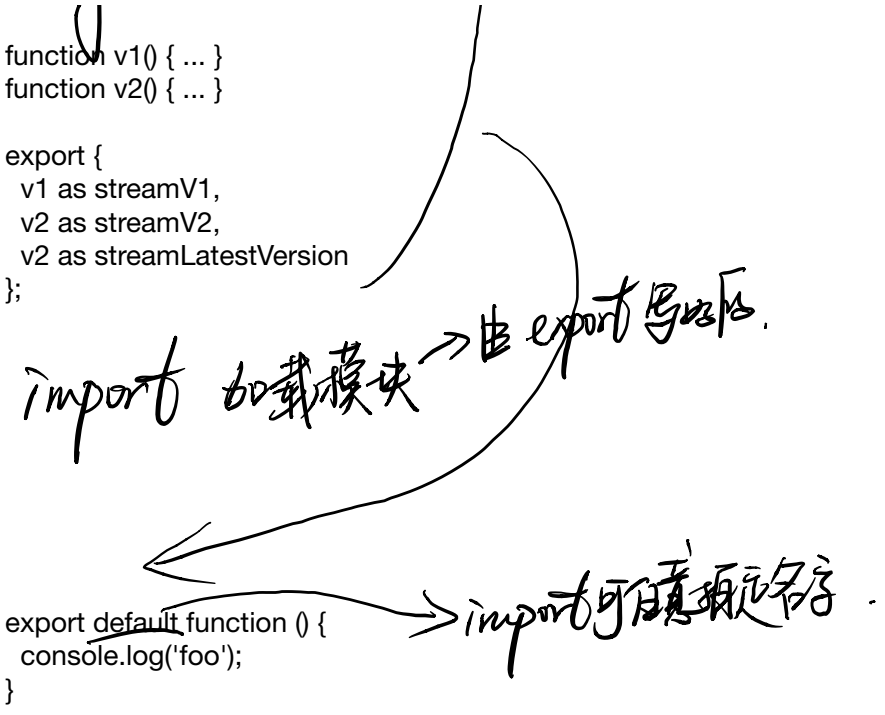
```
var firstName = 'Michael';
var lastName = 'Jackson';
var year = 1958;

export { firstName, lastName, year };
```

@函数/类

as重命名

eg、


function v1() { ... }
function v2() { ... }
export {
v1 as streamV1,
v2 as streamV2,
v2 as streamLatestVersion
};
import 加载模块 → 出 export 导出后.
export default function () {
console.log('foo');
}
import 可自定义名字.

Q & A.

①

```
function* fibs() {
  let a = 0;
  let b = 1;
  while (true) {
    yield a;
    [a, b] = [b, a + b];
  }
}

let [first, second, third, fourth, fifth, sixth] = fibs();
sixth // 5
```

} ⇒ 怎么运行的.

②扩展运算符是什么

简便 → 能简便什么..

③

```
// 将3号位复制到0号位
[].copyWithin.call({length: 5, 3: 1}, 0, 3)
// {0: 1, 3: 1, length: 5}
```

→ 怎么运行的.

④ Class 应用处. → 和function区别.