#### ES2019提案预览

##### 之集合操作

摘要

TypeScript语言特性——集合操作

@link <https://github.com/tc39/Array.prototype.includes>

@link <https://github.com/tc39/proposal-collection-normalization>

@link <https://github.com/tc39/proposal-upsert>

正文

Array.prototype.includes

示例：

assert([1, 2, 3].includes(2) === true);

assert([1, 2, 3].includes(4) === false);

assert([1, 2, NaN].includes(NaN) === true);

assert([1, 2, -0].includes(+0) === true);

assert([1, 2, +0].includes(-0) === true);

assert(["a", "b", "c"].includes("a") === true);

assert(["a", "b", "c"].includes("a", 1) === false);

Collection {coerceKey, coerceValue}

由于JavaScript是弱类型语言，因此及时TypeScript具有强类型特性，在编译为JS后类型限制得不到保证。该提议基于此提出了强键值特性，即显式声明Map、Set对象的键、值变量类型，在执行insert()、update()方法时进行类型检查或强制类型转换，示例如下：

const map = new Map([], {

coerceKey: String

});// stored using { [[Key]]: "1", [[Value]]: "one" } in map.[[MapData]]map.set(1, 'one');// looks for corresponding { [[Key]]: "1" } in map.[[MapData]]map.has(1); // true// functions directly exposing the underlying entry list are unaffected

[...map.entries()]; // [["1", "one"]]

const set = new Set([], {coerceValue: JSON.stringify});// stored using { [[Value]]: '{"path": "/foo"}' } in set.[[SetData]]set.add({path: '/foo'});// looks for corresponding { [[Value]]: '{"path": "/foo"}' } in set.[[SetData]]set.has({path: '/foo')};// functions directly exposing the underlying entry list are unaffected

[...set]; // ['{"path": "/foo"}']

Map.prototype.upsert

We propose the addition of a method that will add a value to a map if the map does not already have something at key, and will also update an existing value at key. It’s worthwhile having this API for the average case to cut down on lookups. It is also worthwhile for developer convenience and expression of intent.

□