JS模拟实现专题

1. 模拟实现bind

JavaScript

```javascript
Function.prototype.bindFn = function() {
  const self = this;
  const args = [].slice.call(arguments, 1);

  const obj = function() {};

  const bound = function() {
    const fnArgs = [].slice.call(arguments);
    return self.apply(
      this instanceof obj ? this : thisArgs,
      args.concat(fnArgs)
    );
  }

  if(this.prototype) {
    obj.prototype = this.prototype;
    bound.prototype = new obj();
  }

  return bound;
}
```

TypeScript

```typescript
Function.prototype.mybind = function(context, ...args) {
  if(typeof this !== 'function') throw new TypeError(this, 'is not a function');
  let _this = this;
  return function fun(...a) {
    return this instanceof fun ? new _this(...args, ...a) : _this.apply(context || window, [...args, ...a]);
  }
}
```

2. 模拟实现call、apply

TypeScript

```
Function.prototype.callFn = function(context, ...args) {
    context = context || window;
    context.fn = this;
    const res = context.fn(args);
    delete context.fn;
    return res;
}

Function.prototype.applyFn = function(context, arr) {
    context = context || window;
    context.fn = this;
    let res;
    if(!arr) res = context.fn();
    else res = context.fn(...arr);
    delete context.fn;
    return res;
}
```

3. 模拟实现Promise.all

TypeScript

```
Promise._all = function(promises) {
    const result = [];
    let promiseCount = 0;
    return new Promise((resolve, reject) => {
        for(let i=0; i<promises.length; i++) {
            Promise.resolve(promise[i]).then((res) => {
                ++promiseCount;
                result[i] = res;
                if(promiseCount === promises.length) {
                    resolve(result);
                }
            }, (err) => {
                reject(err);
            });
        }
    });
}
```

4. 模拟实现new

JavaScript

```
function myNew() {
    const obj = new Object();
    const Counstructor = [].shrift.apply(arguments);
    obj.__proto__ = Counstructor.prototype;
    const res = Counstructor.apply(obj, arguments);
    return typeof res === "object" ? res : obj;
}
```

5. 模拟实现instanceof

TypeScript

```
function _instanceof(A, B) {
    const o = B.prototype;
    A = A.__proto__;
    while(true) {
        if(A === null) return false;
        if(o === A) return true;
        A = A.__proto__;
    }
}
```