Promise

JavaScript is asynchronous

```
(function(){
 setInterval(function(){ console.log('a'); }, 3000);

 setInterval(function(){ console.log('b'); }, 1000);

 setInterval(function(){ console.log('c'); }, 5000);
})();
```

I never wait for you I've my own way to go

I never wait for you I've my own way to go

```
> (function(){
    setInterval(function(){ console.log('a'); }, 3000);
    setInterval(function(){ console.log('b'); }, 1000);
    setInterval(function(){ console.log('c'); }, 5000);
  })();
< undefined
b
a
b
c
a
b
```

The Promise object is used for asynchronous computations. A Promise represents a value which may be available now, or in the future, or never.

Problem: JavaScript never wait for delay in response

```
function receiveFriend(){
 var isHeReached;
 setInterval(function(){
  // 5 sec delay due to trafic
  reached("yes");
 }, 5000);
 function reached(check){
   isHeReached = check;
 }
 return isHeReached;
}
```

```
> function receiveFriend(){
    var isHeReached;
    setInterval(function(){
      // 5 sec delay due to trafic
      reached("yes");
    }, 5000);
    function reached(check){
       isHeReached = check;
    }
    return isHeReached;
  }
< undefined
> console.log(receiveFriend());
  undefined
```

```
> function receiveFriend(){
    var isHeReached;
    //setInterval(function(){
      // no delay
      reached("yes");
    //}, 5000);
    function reached(check){
       isHeReached = check;
    }
    return isHeReached;
  }
< undefined
> console.log(receiveFriend());
  yes
```

The promise constructor takes one argument, a callback with two parameters, resolve and reject. Do something within the callback, perhaps async, then call resolve if everything worked, otherwise call reject.

Solution: Promise

```
function receiveFriend(){
 return new Promise(function(resolve){
 var isHeReached;
 setInterval(function(){
  // simulating delay
  resolve("yes");
 }, 5000);
});
}
```

```
> function receiveFriend(){
    return new Promise(function(resolve){
    var isHeReached;
    setInterval(function(){
      // simulating delay
      resolve("yes");
     }, 5000);
    });
  }
< undefined
> receiveFriend().then(function(res){ console.log("Response:", res); });
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue]]: undefined}
  Response: yes
```

```
> function receiveFriend(){
    return new Promise(function(resolve, reject){
    var isHeReached;
    setInterval(function(){
      // simulating delay
      resolve("yes");
     }, 5000);
     setInterval(function(){
       reject("Something went wrong, I'm unable to come")
     }, 2000);
    });
  }
< undefined
> receiveFriend().then(
    function(res){ console.log("Response:", res); },
    function(error){ console.log("Response:", error); }
  );
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue]]: undefined}
  Response: Something went wrong, I'm unable to come
```