ОБРАБОТКА ИСКЛЮЧЕНИЙ (ОШИБОК)

THROW

```
// throw expression;

throw "some error!";                // Uncaught some error!
```

```
notDifenedVar(); // Uncaught ReferenceError: notDifenedVar is not def

3.14 = +Math.PI.toFixed(2)
// 3.14 === +Math.PI.toFixed(2)
// Uncaught ReferenceError: Invalid left-hand side in assignment

i ==== 2          // Uncaught SyntaxError: Unexpected token =
```

КОСТРУКТОР ERROR

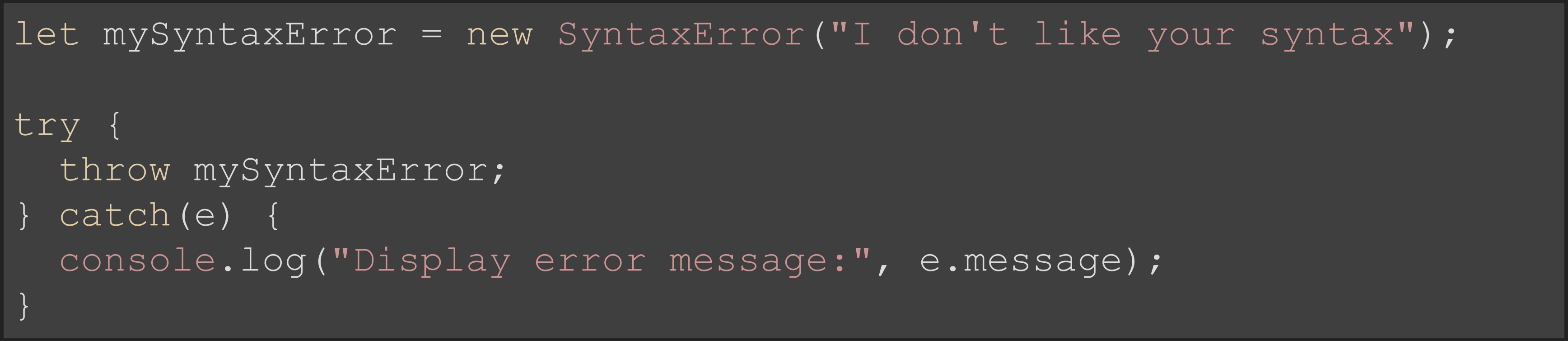
```
// new Error([message[, fileName[, lineNumber]]])

let someError = new Error("it's an error!")

throw someError;          // Uncaught Error: it's an error!
```

1. EvalError
2. InternalError
3. RangeError
4. ReferenceError
5. SyntaxError
6. TypeError
7. URIError

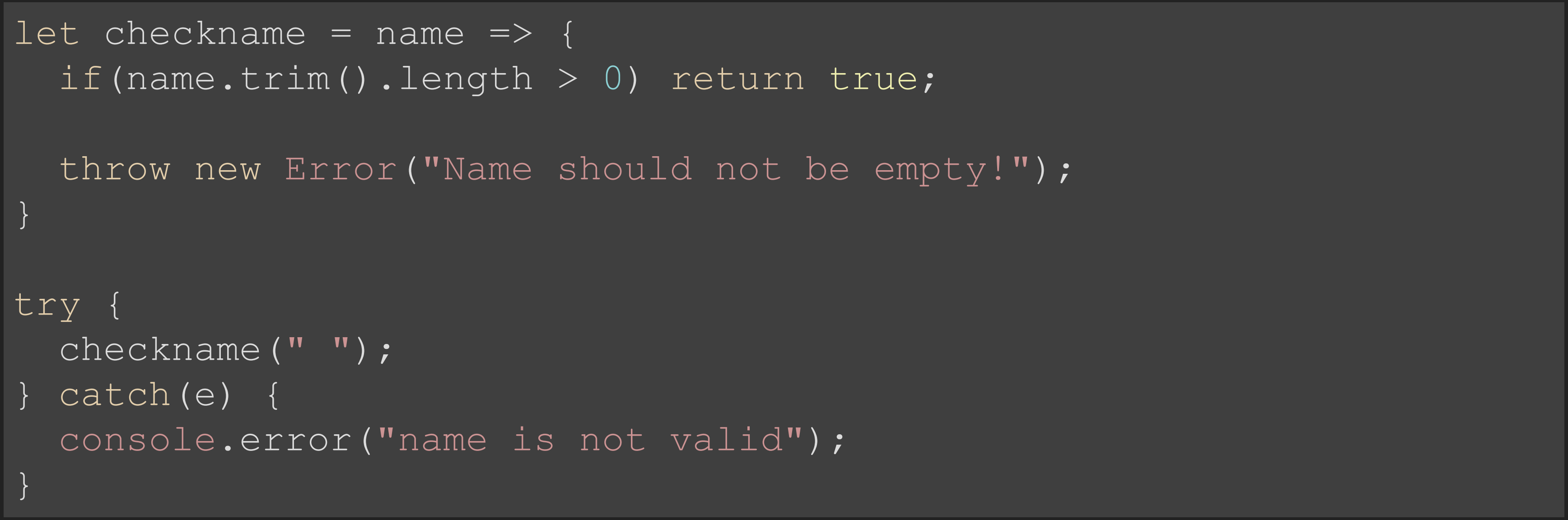
```
let mySyntaxError = new SyntaxError("I don't like your syntax");

throw mySyntaxError;
```

TRY CATCH FINALLY

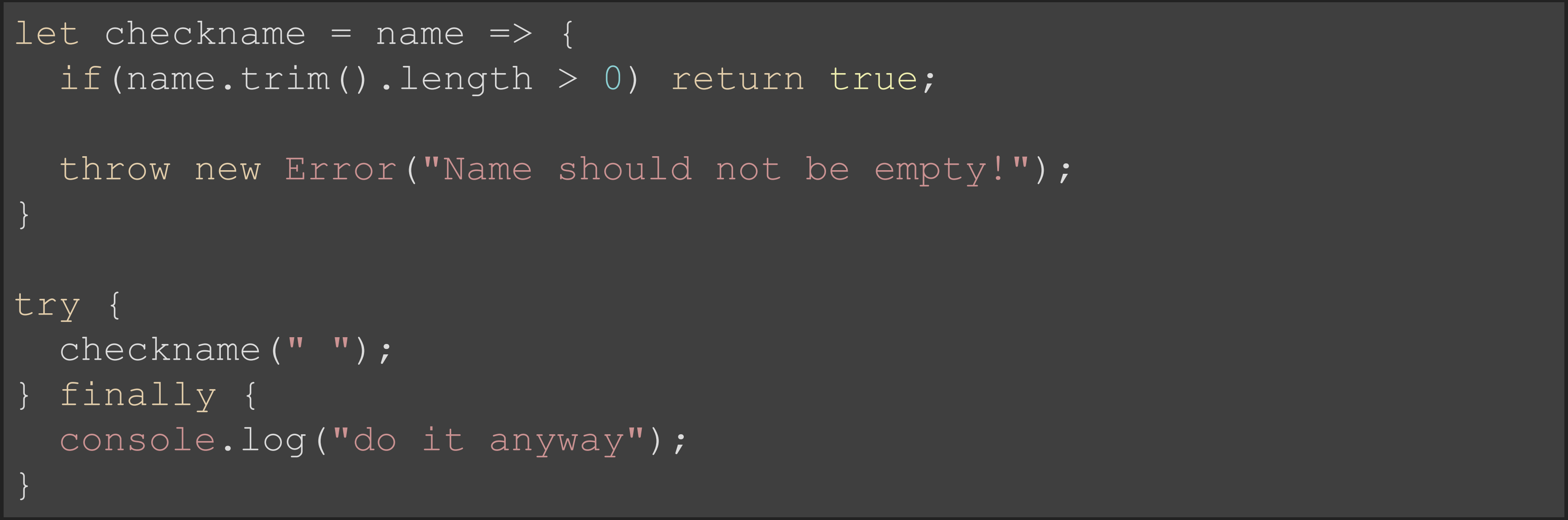
```
try {
   try_statements
}
[catch (exception_var_2) {
   catch_statements_2
}]
[finally {
   finally_statements
}]
```



```javascript
let mySyntaxError = new SyntaxError("I don't like your syntax");

try {
  throw mySyntaxError;
} catch(e) {
  console.log("Display error message:", e.message);
}
```



```
let checkname = name => {
  if(name.trim().length > 0) return true;

  throw new Error("Name should not be empty!");
}

try {
  checkname(" ");
} catch(e) {
  console.error("name is not valid");
}
```



```
let checkname = name => {
  if(name.trim().length > 0) return true;

  throw new Error("Name should not be empty!");
}

try {
  checkname(" ");
} finally {
  console.log("do it anyway");
}
```

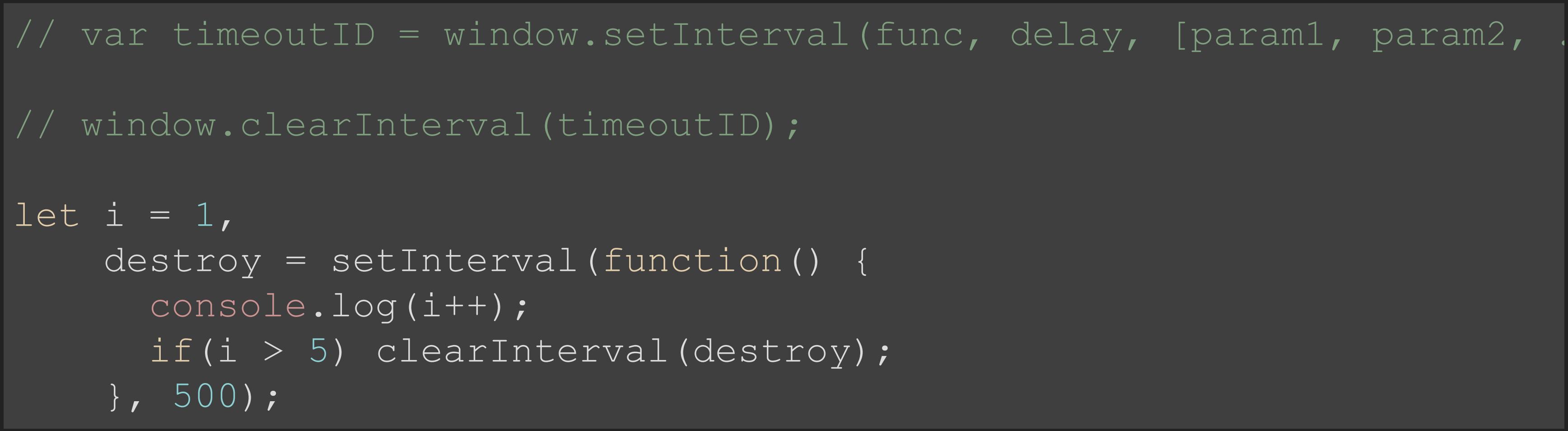
ОБЕЩАНИЕ (PROMISE)

СИНХРОННОСТЬ И АСИХРОННОСТЬ

```
console.log(1);
console.log(2);

setTimeout(function() {
  console.log(3);
});

console.log(4);
```

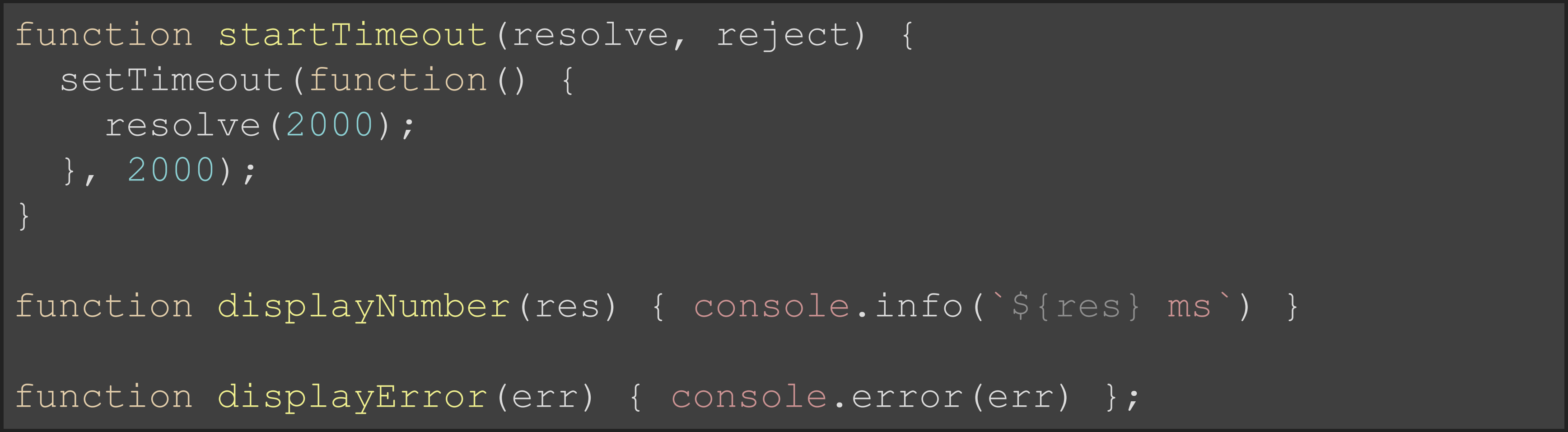
WINDOWTIMERS

```
// var timeoutID = window.setTimeout(func, delay, [param1, param2, ..

// window.clearTimeout(timeoutID);

setTimeout(function() {
  console.log("hello everybody!");
}, 1500);
```

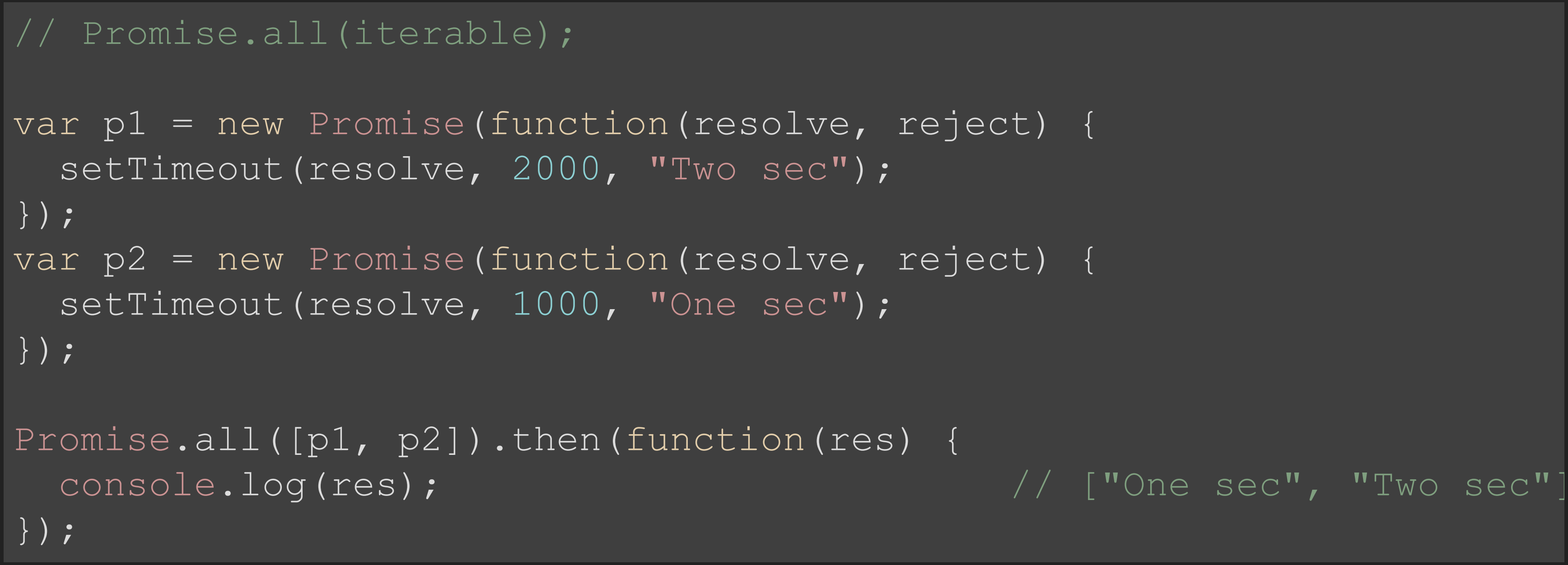


```
// var timeoutID = window.setInterval(func, delay, [param1, param2, .

// window.clearInterval(timeoutID);

let i = 1,
    destroy = setInterval(function() {
      console.log(i++);
      if(i > 5) clearInterval(destroy);
    }, 500);
```

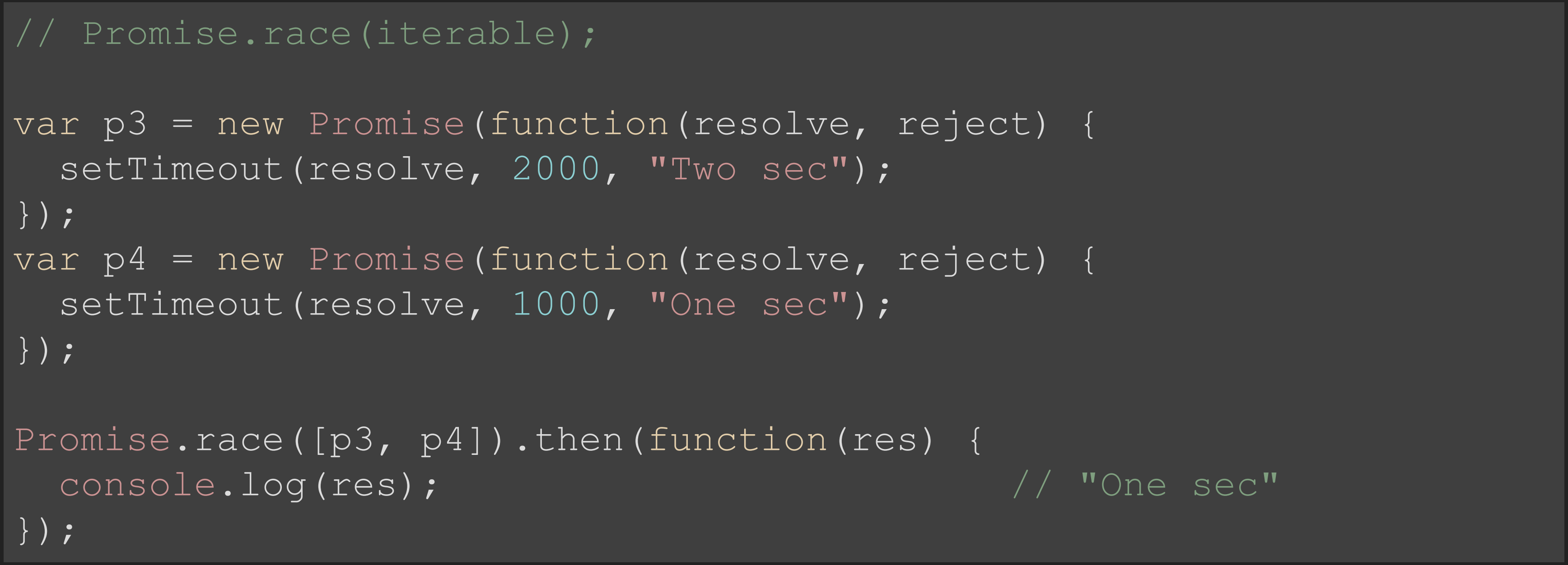
```
// new Promise( /* executor */ function(resolve, reject) { ... } );

let promise = new Promise(startTimeout);

promise
  .then(displayNumber)
  .catch(displayError);

// or

promise
  .then(displayNumber, displayError);
```



```
function startTimeout(resolve, reject) {
  setTimeout(function() {
    resolve(2000);
  }, 2000);
}

function displayNumber(res) { console.info(`${res} ms`) }

function displayError(err) { console.error(err) };
```



```
// Promise.all(iterable);

var p1 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 2000, "Two sec");
});
var p2 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 1000, "One sec");
});

Promise.all([p1, p2]).then(function(res) {
  console.log(res);                         // ["One sec", "Two sec"]
});
```



```
// Promise.race(iterable);

var p3 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 2000, "Two sec");
});
var p4 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 1000, "One sec");
});

Promise.race([p3, p4]).then(function(res) {
  console.log(res);                         // "One sec"
});
```