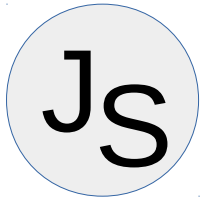
Исключения



- Понятие исключения
- throw
- try-catch-finally
- Использование исключений

Объект Error

```
{
   name: "ReferenceError",
   message: "Q is not defined"
}
```

Name	Описание
Error	Ошибка
EvalError	Ошибка в eval()
RangeError	Число за границами диапазона
ReferenceError	Неверная ссылка
SyntaxError	Синтаксическая ошибка в eval()
TypeError	Неожиданный тип значения
URIError	Ошибка при выполнении encodeURI()

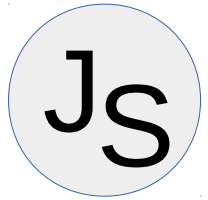
Исключение

throw *выражение*;

```
function factorial(x) {
   if (x < 0){
      throw new Error(
         "x не может быть отрицательным"
      );
   }
   for(var f = 1; x > 1; f *= x, x--);
   return f;
}
```

```
throw 20001;
throw "Invalid order number";
throw Math.cos;
```

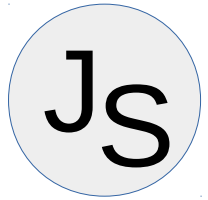
try – catch - finally



try
    инструкция;
catch (e)
    инструкция;
finally
    инструкция;

```
try {
   var n = prompt("Please enter an integer");
   var f = factorial(n);
   alert(n + "! = " + f);
}
catch (ex) {
   alert(ex);
}
```

finally

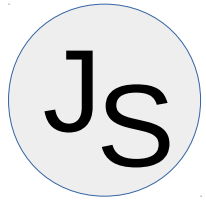


break, continue, return вызывают выполнение finally

```
for (i in new Array(4)){
   try{
      if (i%2 >0) break;
   }
   finally{
      i = 99;
   }
}

console.log(i);                          // 99
```

Использование исключений




```
function fn() {
   var foo = {};
   return foo.bar();
}
```

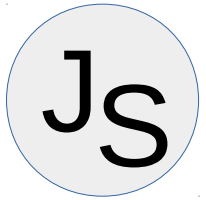
Плохо

```
try {
   return fn();
} catch (e) { }
```

Чуть лучше

```
try {
   return fn();
} catch (e) {
   throw Error('Другая ошибка');
}
```

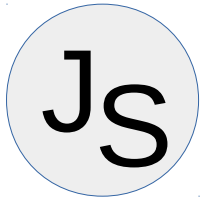
Использование исключений




```
for(let i=0; i<10000; i++){
   try {
      ...
   } catch (e) {
      ...
   }
}
```

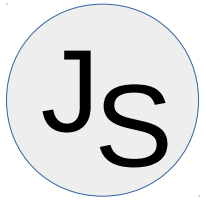
```
try {
   for(let i=0; i<10000; i++){
      ...
   }
}
catch (e) {
   ...
}
```

Использование исключений



1. Любой код может отказать
2. Журнал ошибок
3. Ошибки обрабатывает разработчик, а не среда
4. Понять, где могут возникать ошибки
5. Порождать собственные ошибки
6. Различать фатальные и нефатальные ошибки
7. Различать ошибки разработки и операционные ошибки
8. Обрабатывать ошибки централизованно
9. Обеспечить режим отладки

Проброс исключений

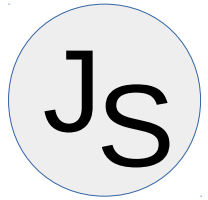


Ошибку, о которой catch не знает, он не должен обрабатывать.

```
var data = '{ "name": "Вася", "age": 30 }'; // данные корректны

try {
  var user = JSON.parse(data);
  if (!user.name) {
    throw new SyntaxError("Ошибка в данных");
  }
  blabla(); // произошла непредусмотренная ошибка
  alert( user.name );
} catch (e) {
  if (e.name == "SyntaxError") {
    alert( "Извините, в данных ошибка" );
  } else {
    throw e;
  }
}
```

Проброс исключений

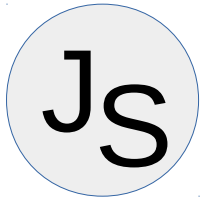



```
function readData() {
  var data = '{ "name": "Вася", "age": 30 }';

  try {
    // ...
    blabla(); // ошибка!
  } catch (e) {
    // ...
    if (e.name != 'SyntaxError') {
      throw e; // пробрасываем
    }
  }
}

try {
  readData();
} catch (e) {
  alert( "Поймал во внешнем catch: " + e ); // ловим
}
```

Оборачивание исключений




```
function ReadError(message, cause) {
  this.message = message; this.cause = cause; this.name = 'ReadError';
}

function readData() {
  try {
    // ...
    JSON.parse('{ bad data }');
    // ...
  } catch (e) {
    // ...
    switch(e.name){
         case 'URIError': throw new ReadError("Ошибка в URI", e); break;
         case 'SyntaxError'):
          throw new ReadError("Синтаксическая ошибка в данных", e);break;
         default: throw e;// пробрасываем
     }
  }
}

try { readData(); }
catch (e) {
  if (e.name == 'ReadError') {
    alert( e.message + '\n + e.cause ); // оригинальная ошибка-причина
  }
  else {throw e;}
}
```

Исключения

- Понятие исключения
- throw
- try-catch-finally
- Использование исключений

Объект Error

```
{
   name: "ReferenceError",
   message: "Q is not defined"
}
```

Name	Описание
Error	Ошибка
EvalError	Ошибка в eval()
RangeError	Число за границами диапазона
ReferenceError	Неверная ссылка
SyntaxError	Синтаксическая ошибка в eval()
TypeError	Неожиданный тип значения
URIError	Ошибка при выполнении encodeURI()

Исключение

throw *выражение*;

```
function factorial(x) {
   if (x < 0){
      throw new Error(
         "x не может быть отрицательным"
      );
   }
   for(var f = 1; x > 1; f *= x, x--);
   return f;
}
```

```
throw 20001;
throw "Invalid order number";
throw Math.cos;
```

try – catch - finally

try
    инструкция;
catch (e)
    инструкция;
finally
    инструкция;

```
try {
   var n = prompt("Please enter an integer");
   var f = factorial(n);
   alert(n + "! = " + f);
}
catch (ex) {
   alert(ex);
}
```

finally

break, continue, return вызывают выполнение finally

```
for (i in new Array(4)){
   try{
      if (i%2 >0) break;
   }
   finally{
      i = 99;
   }
}

console.log(i);                    // 99
```

Использование исключений

```
function fn() {
   var foo = {};
   return foo.bar();
}
```

Плохо

```
try {
   return fn();
} catch (e) { }
```

Чуть лучше

```
try {
   return fn();
} catch (e) {
   throw Error('Другая ошибка');
}
```

Использование исключений

```
for(let i=0; i<10000; i++){
   try {
      ...
   } catch (e) {
      ...
   }
}
```

```
try {
   for(let i=0; i<10000; i++){
      ...
   }
}
catch (e) {
   ...
}
```

Использование исключений

1. Любой код может отказать
2. Журнал ошибок
3. Ошибки обрабатывает разработчик, а не среда
4. Понять, где могут возникать ошибки
5. Порождать собственные ошибки
6. Различать фатальные и нефатальные ошибки
7. Различать ошибки разработки и операционные ошибки
8. Обрабатывать ошибки централизованно
9. Обеспечить режим отладки

Проброс исключений

Ошибку, о которой catch не знает, он не должен обрабатывать.

```
var data = '{ "name": "Вася", "age": 30 }'; // данные корректны

try {
  var user = JSON.parse(data);
  if (!user.name) {
    throw new SyntaxError("Ошибка в данных");
  }
  blabla(); // произошла непредусмотренная ошибка
  alert( user.name );
} catch (e) {
  if (e.name == "SyntaxError") {
    alert( "Извините, в данных ошибка" );
  } else {
    throw e;
  }
}
```

Проброс исключений

```
function readData() {
  var data = '{ "name": "Вася", "age": 30 }';

  try {
    // ...
    blabla(); // ошибка!
  } catch (e) {
    // ...
    if (e.name != 'SyntaxError') {
      throw e; // пробрасываем
    }
  }
}

try {
  readData();
} catch (e) {
  alert( "Поймал во внешнем catch: " + e ); // ловим
}
```

Оборачивание исключений

```
function ReadError(message, cause) {
  this.message = message; this.cause = cause; this.name = 'ReadError';
}

function readData() {
  try {
    // ...
    JSON.parse('{ bad data }');
    // ...
  } catch (e) {
    // ...
    switch(e.name){
         case 'URIError': throw new ReadError("Ошибка в URI", e); break;
         case 'SyntaxError':
          throw new ReadError("Синтаксическая ошибка в данных", e);break;
         default: throw e;// пробрасываем
     }
  }
}

try { readData(); }
catch (e) {
  if (e.name == 'ReadError') {
    alert( e.message + '\n + e.cause ); // оригинальная ошибка-причина
  }
  else {throw e;}
}
```