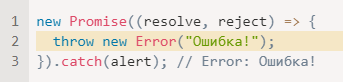
Цепочки промисов отлично подходят для перехвата ошибок. Если промис завершается с ошибкой, то управление переходит в ближайший обработчик ошибок.

.catch не обязательно должен быть сразу после ошибки, он может быть далее, после одного или даже нескольких .then

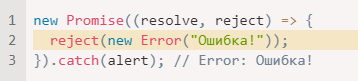
**[Неявный try…catch](https://learn.javascript.ru/promise-error-handling" \l "neyavnyy-try-catch)**

Вокруг функции промиса и обработчиков находится "невидимый try..catch". Если происходит исключение, то оно перехватывается, и промис считается отклонённым с этой ошибкой.

Например, этот код:



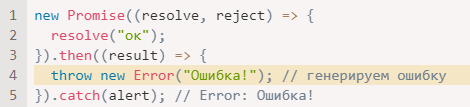
…Работает так же, как и этот:



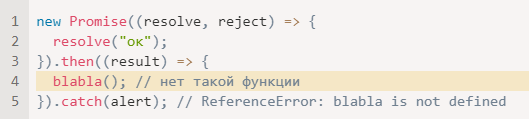
"Невидимый try..catch" вокруг промиса автоматически перехватывает ошибку и превращает её в отклонённый промис.

Это работает не только в функции промиса, но и в обработчиках. Если мы бросим ошибку (throw) из обработчика (.then), то промис будет считаться отклонённым, и управление перейдёт к ближайшему обработчику ошибок.

Пример:



Это происходит для всех ошибок, не только для тех, которые вызваны оператором throw. Например, программная ошибка:



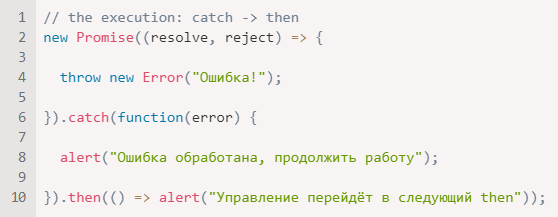
Финальный .catch перехватывает как промисы, в которых вызван reject, так и случайные ошибки в обработчиках.

## [Пробрасывание ошибок](https://learn.javascript.ru/promise-error-handling" \l "probrasyvanie-oshibok)

Мы можем иметь столько обработчиков .then, сколько мы хотим, и затем использовать один .catch в конце, чтобы перехватить ошибки из всех обработчиков.

Если мы пробросим (throw) ошибку внутри блока .catch, то управление перейдёт к следующему ближайшему обработчику ошибок. А если мы обработаем ошибку и завершим работу обработчика нормально, то продолжит работу ближайший успешный обработчик .then.

В примере ниже .catch успешно обрабатывает ошибку:



Здесь блок .catch завершается нормально. Поэтому вызывается следующий успешный обработчик .then.

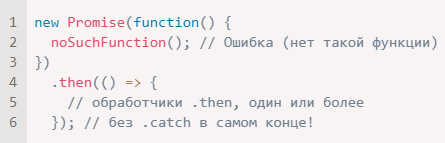
В примере ниже мы видим другую ситуацию с блоком .catch. Обработчик (\*) перехватывает ошибку и не может обработать её (например, он знает как обработать только URIError), поэтому ошибка пробрасывается далее:



Управление переходит от первого блока .catch (\*) к следующему (\*\*), вниз по цепочке.

## [Необработанные ошибки](https://learn.javascript.ru/promise-error-handling" \l "neobrabotannye-oshibki)

Что произойдёт, если ошибка не будет обработана? Например, мы просто забыли добавить .catch в конец цепочки, как здесь:

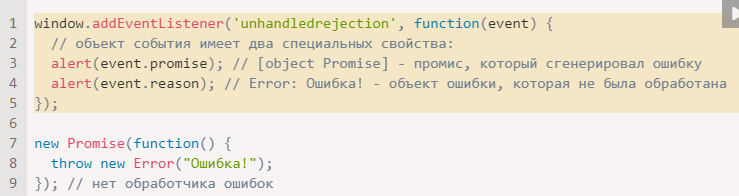


В случае ошибки выполнение должно перейти к ближайшему обработчику ошибок. Но в примере выше нет никакого обработчика. Поэтому ошибка как бы «застревает», её некому обработать.

На практике, как и при обычных необработанных ошибках в коде, это означает, что что-то пошло сильно не так.

Что происходит, когда обычная ошибка не перехвачена try..catch? Скрипт умирает с сообщением в консоли. Похожее происходит и в случае необработанной ошибки промиса.

В браузере мы можем поймать такие ошибки, используя событие unhandledrejection:



Это событие является частью [стандарта HTML](https://html.spec.whatwg.org/multipage/webappapis.html#unhandled-promise-rejections).

Если происходит ошибка, и отсутствует её обработчик, то генерируется событие unhandledrejection, и соответствующий объект event содержит информацию об ошибке.

Обычно такие ошибки неустранимы, поэтому лучше всего – информировать пользователя о проблеме и, возможно, отправить информацию об ошибке на сервер.

В не-браузерных средах, таких как Node.js, есть другие способы отслеживания необработанных ошибок.

## [Итого](https://learn.javascript.ru/promise-error-handling" \l "itogo)

* .catch перехватывает все виды ошибок в промисах: будь то вызов reject() или ошибка, брошенная в обработчике при помощи throw.
* .then также перехватывает ошибки таким же образом, если задан второй аргумент (который является обработчиком ошибок).
* Необходимо размещать .catch там, где мы хотим обработать ошибки и знаем, как это сделать. Обработчик может проанализировать ошибку (могут быть полезны пользовательские классы ошибок) и пробросить её, если ничего не знает о ней (возможно, это программная ошибка).
* Можно и совсем не использовать .catch, если нет нормального способа восстановиться после ошибки.
* В любом случае нам следует использовать обработчик события unhandledrejection (для браузеров и аналог для других окружений), чтобы отслеживать необработанные ошибки и информировать о них пользователя (и, возможно, наш сервер), благодаря чему наше приложение никогда не будет «просто умирать».