JAVASCRIPT

Promises. Routing. Single Page Application. CDN

PROMISE

Объект **Promise (обещание)** используется для отложенных и асинхронных вычислений.

Promise может находиться в трёх состояниях:

вначале **pending («ожидание»)**: начальное состояние, не выполнено и не отклонено

затем – одно из:

fulfilled («выполнено успешно»): операция завершена успешно

или

rejected («выполнено с ошибкой»): операция завершена с ошибкой

На promise можно навешивать коллбэки двух типов:

onFulfilled – срабатывают, когда promise в состоянии «выполнен успешно».

onRejected – срабатывают, когда promise в состоянии «выполнен с ошибкой».

PROMISE

При создании обещание находится **в ожидании** (pending),

а затем может стать **выполнено** (fulfilled), вернув полученный результат (значение),
или **отклонено** (rejected), вернув причину отказа.

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

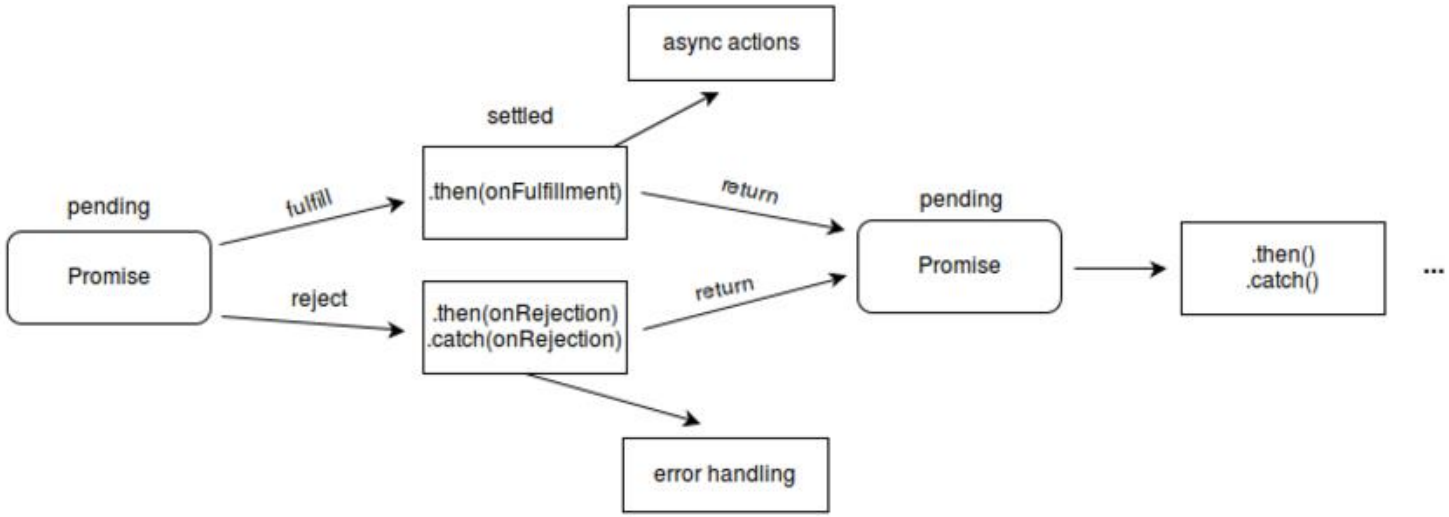
```
var promise = new Promise(function(resolve, reject) {
  // Эта функция будет вызвана автоматически

  // В ней можно делать любые асинхронные операции,
  // А когда они завершатся — нужно вызвать одно из:
  // resolve(результат) при успешном выполнении
  // reject(ошибка) при ошибке
  var somethingIsDone = true;
  var resultOfOperation = {};
  var anyErrorDuringOfOperation;

  if (somethingIsDone){
    resolve(resultOfOperation);
  } else {
    reject(anyErrorDuringOfOperation);
  }
});
```

```
promise.then(onFulfilled, onRejected);
// onFulfilled сработает при успешном выполнении
promise.then(onFulfilled)
// onRejected сработает при ошибке
promise.then(null, onRejected);
//это то же самое
promise.catch(onRejected);
```

PROMISE




```
function onFulfilled(resultOfPreviousOperation){
  console.log('OnFulfilled:', resultOfPreviousOperation);

}

function onRejected(anyErrorDuringOfOperation){
  console.log('onRejected:', anyErrorDuringOfOperation);
}
```

```
function myLongPromiseFunction (){
  return new Promise(function(resolve, reject){
    setTimeout(function(){
      reject('время вышло!');
    }, 1000);

  });
}
```

```
myLongPromiseFunction()
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected);
```

PROMISE

```
function onFulfilled(resultOfPreviousOperation){
  console.log('OnFulfilled:', resultOfPreviousOperation);

}

function onRejected(anyErrorDuringOfOperation){
  console.log('onRejected:', anyErrorDuringOfOperation);
}
```

```
Promise.resolve()
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected)
  .then(onFulfilled, onRejected);
```

Window.location

Свойство только для чтения Window.location возвращает объект **Location** с информацией о текущем расположении документа.

`Location.href`
`DOMString`, содержащий URL целиком. При изменении, соответствующий документ переходит на новую страницу.

`Location.hash`
`DOMString`, содержащий `'#'` с последующим идентификатором.

`Location.host`
`DOMString`, содержащий хост, а именно *имя хоста*, `':'` и *порт*.

`Location.hostname`
`DOMString`, содержащий домен текущего URL.

`Location.pathname`
`DOMString`, содержащий первый `'/'` после хоста с последующим текстом URL.

`Location.assign()`
Загружает ресурс по URL, указанному в качестве параметра.

`Location.reload()`
Перезагружает ресурс по текущему URL. Единственный опциональный параметр `Boolean` при значении `true` указывает, что страница должна быть заново загружена с сервера, при значении `false` страница может быть загружена из кэша.

`Location.replace()`
Заменяет текущий ресурс на новый по URL, указанному в качестве параметра. Отличие от `assign()` в том, что при использовании `replace()` текущая страница не будет сохранена в `History`, и пользователь не сможет использовать кнопку *назад*, чтобы вернуться к ней.

`Location.toString()`
Возвращает `DOMString`, содержащий URL целиком. Это синоним `URLUtils.href`, однако он не может использоваться для изменения значения.

Window.location

Пример №1: Переход на новую страницу

```
location.assign("http://www.mozilla.org"); // or
location = "http://www.mozilla.org";
```

Пример №2: Принудительная перезагрузка текущей страницы с сервера

```
location.reload(true);
```

```
function reloadPageWithHash() {
  var initialPage = location.pathname;
  location.replace('http://example.com/#' + initialPage);
}
```

JAVASCRIPT

Promises. Routing. Single Page Application. CDN