

این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.

این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.



این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.

این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.

این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.

این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.

این مطلب یکی از مجموعه مقالات پرونده ویژه «آینده برنامه‌نویسی» است. برای دانلود کل پرونده ویژه کلیک کنید.


```
stang.printCurrentSpeed();
```

ES6의 Promise

ES6의 Promise는 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다.

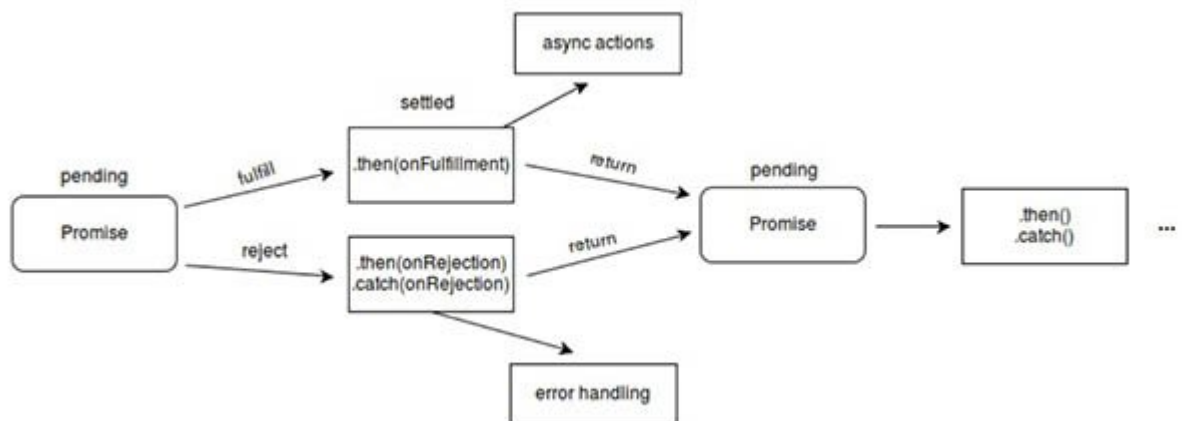
```
var num = 0; //globally scoped
```

```
for (let i = 0; i < 10; i++) { //i is block scoped
  num += i;
  console.log('value of i in block: ' + i);
}
```

```
console.log('Is i defined here?: ' + (typeof i !== 'undefined')); //Is i defined here?: false
```

Promises

Promises는 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다.



Promise 객체는 1개의 값을 반환합니다.

Promise는 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다. Promise는 ES6에서 도입된 새로운 객체로, 비동기 작업을 동기화하는 데 사용됩니다.

```
getJSON("/api/employee/1").then(function(post) {
  return getJSON(post.commentURL);
}).then(function(comments) { //you could chain multiple then statements
  // proceed with access to employee
}).catch(function(error) { //succinct error handling
  // handle errors in either of the two requests
});
```

```
var promises = [2, 3, 5, 7, 11, 13].map(function(id){
  return getJSON("/post/" + id + ".json");
});
```

```
RSVP.all(promises).then(function(posts) {
```

```
// posts contains an array of results for the given promises
}).catch(function(reason){
  // if any of the promises fails.
});
```

Multi Thread) Promise.all(iterable) Promise.race(iterable) Promise.reject(reason) Promise.resolve(value)

JavaScript Robotics

JavaScript Robotics is a framework for building robot-like applications. It is designed to be easy to use and to integrate with other JavaScript frameworks. The framework is built on top of Node.js and uses the same syntax as JavaScript. It provides a simple way to create and manage robots, and it allows you to interact with the robots in a variety of ways. The framework is designed to be flexible and to support a wide range of use cases. It is a great choice for anyone who wants to build robot-like applications in JavaScript.

JavaScript Robotics

JavaScript Robotics is a framework for building robot-like applications. It is designed to be easy to use and to integrate with other JavaScript frameworks. The framework is built on top of Node.js and uses the same syntax as JavaScript. It provides a simple way to create and manage robots, and it allows you to interact with the robots in a variety of ways. The framework is designed to be flexible and to support a wide range of use cases. It is a great choice for anyone who wants to build robot-like applications in JavaScript.

JavaScript Robotics

JavaScript Robotics is a framework for building robot-like applications. It is designed to be easy to use and to integrate with other JavaScript frameworks. The framework is built on top of Node.js and uses the same syntax as JavaScript. It provides a simple way to create and manage robots, and it allows you to interact with the robots in a variety of ways. The framework is designed to be flexible and to support a wide range of use cases. It is a great choice for anyone who wants to build robot-like applications in JavaScript.

```
var Cylon = require("cylon");
```

```
// Initialize the robot
Cylon.robot({
```

```
// Change the port to the correct port for your Arduino.
connections: {
  arduino: { adaptor: 'firmata', port: '/dev/ttyACM0' }
},

devices: {
  led: { driver: 'led', pin: 13 }
},

work: function(my) {
  every((1).second(), function() {
    my.led.toggle();
  });
}
}).start();
```

مقاله ها

مقاله ها در زمینه های مختلف فناوری اطلاعات و اینترنت. در این مقاله به بررسی تغییرات و بهروزرسانی های جدید در استاندارد ECMAScript 6 می پردازیم. این تغییرات شامل افزودن ویژگی های جدید و بهبود کارایی کد است. همچنین به بررسی نحوه استفاده از این ویژگی ها در برنامه های وب می پردازیم. این تغییرات در سال 2015 به تصویب رسید و در سال 2016 به طور رسمی به عنوان استاندارد پذیرفته شد. این تغییرات شامل افزودن ویژگی های جدید و بهبود کارایی کد است. همچنین به بررسی نحوه استفاده از این ویژگی ها در برنامه های وب می پردازیم. این تغییرات در سال 2015 به تصویب رسید و در سال 2016 به طور رسمی به عنوان استاندارد پذیرفته شد. این تغییرات شامل افزودن ویژگی های جدید و بهبود کارایی کد است. همچنین به بررسی نحوه استفاده از این ویژگی ها در برنامه های وب می پردازیم. این تغییرات در سال 2015 به تصویب رسید و در سال 2016 به طور رسمی به عنوان استاندارد پذیرفته شد.

:مقاله ها

مقاله ها

:مقاله ها

15:36 - 05/07/1394

:مقاله ها

مقاله ها - مقاله ها - مقاله ها - مقاله ها - مقاله ها