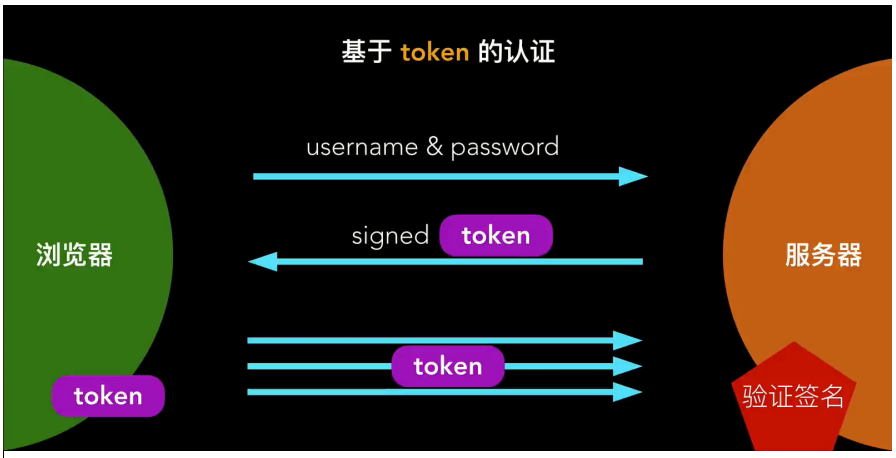

实现基于 JWT 的用户认证接口

安装 jsonwebtoken 包

```
npm install --save jsonwebtoken
```

jsonwebtoken 包可以生成、验证和解码 JWT 认证码

生成 JWT 认证码

打开 `server/routes.js` 文件，首先导入 `jsonwebtoken` 模块：

```
var jwt = require('jsonwebtoken');
```

然后，定义生成 JWT 的 `generateToken` 方法，把下面代码放到 `module.exports ...` 语句之上：

```
var generateToken = function(user) {
  return jwt.sign(user, 'hand-in-hand', {
    expiresIn: 3000
  });
}
```

调用 `jsonwebtoken` 模块提供的 sign() 接口生成 JWT。

其中， `hand-in-hand` 是生成 JWT 认证码的秘钥，为了安全，最好把秘钥放到配置文件中。

`user` 是要传递给前端的信息，前端可以利用工具解码 JWT 认证码，从而得到 `user` 数据。

`expiresIn` 选项用来指定认证码自生成到失效的时间间隔（过期间隔），上述代码中数字 `3000` 的单位是秒，意思说这个认证码自生成后，再过50分钟就失效了。认证码失效之后，客户端就不能使用失效的认证码访问服务器端的受保护资源了。后续视频会介绍认证码过期时间的用处。

秘钥放入配置文件

修改 `config.js` 文件，再添加一个键值对：

```
module.exports = {
```

```
  ...
  secret: 'hand-in-hand'
};
```

相应的修改 `server/routes.js` 文件中，导入配置文件：

```
var secret = require('./config.js').secret;
```

用到 `hand-in-hand` 秘钥的地方修改为 `secret`，如下所示：

```
var generateToken = function(user) {
  return jwt.sign(user, secret, {
    expiresIn: 3000
  });
}
```

最后，别忘了修改一下可以开源的 `config.default.js` 文件：

```
module.exports = {
  ...
  secret: 'xxx'
};
```

服务器端响应 JWT 认证码

修改 `/auth/login` 接口的 JSON 返回数据，添加一对新的键值对：

```
  user.comparePassword(req.body.password, funct
    ...
    return res.json({
      token: generateToken({name: user.username
      user: { name: user.username }
    });
  });
```

token 属性对应的值就是 JWT 认证码

测试用户认证接口

```
curl -H "Content-Type: application/json" -X P
```

上节课程中我们已经保存了一条用户名为 billie，密码为 cccccc 的信息到 MongoDB 数据库中，所以能够得到正确的 JSON 数据：

小总结

这样，服务器端关于认证部分的 API 基本上就写到这里的，客户端部分后续章节中我们一起来实现。