**require with expression**

A context is created if your request contains expressions, so the **exact** module is not known on compile time.

Example, given we have the following folder structure including .ejs files:

例如有下面结构目录

example\_directory

│

└───template

│ │ table.ejs

│ │ table-row.ejs

│ │

│ └───directory

│ │ another.ejs

When following require() call is evaluated:

如果如下调用require()方法

require('./template/' + name + '.ejs');

webpack parses the require() call and extracts some information:

webpack就会用extract模块输出一下信息

Directory: ./template

Regular expression: /^.\*\.ejs$/

**context module**

**context模块**

A context module is generated. It contains references to **all modules in that directory** that can be required with a request matching the regular expression. The context module contains a map which translates requests to module ids.

Context模块本身就是一个map容器，记录着所有能有效引入的文件目录，如下：

{

"./table.ejs": 42,

"./table-row.ejs": 43,

"./directory/another.ejs": 44

}

The context module also contains some runtime logic to access the map.

This means dynamic requires are supported but will cause all matching modules to be included in the bundle.

如果context的调用是在bundle.js中生效的，那么，可以通过context模块来动态引入文件

**require.context**

You can create your own context with the require.context() function.

如何自定义添加一个context模块？通过require.context方法就可以了

It allows you to pass in a directory to search, a flag indicating whether subdirectories should be searched too, and a regular expression to match files against.

可以传参：目录，是否深挖，匹配正则

require.context(directory, useSubdirectories = true, regExp = /^\.\/.\*$/, mode = 'sync');

webpack parses for require.context() in the code while building.

在打包建立过程中webpack会执行该方法

**context module API**

A context module exports a (require) function that takes one argument: the request.

The exported function has 3 properties: resolve, keys, id.

* resolve is a function and returns the module id of the parsed request.
* keys is a function that returns an array of all possible requests that the context module can handle.

This can be useful if you want to require all files in a directory or matching a pattern, Example:

function importAll (r) {

r.keys().forEach(r);

}

importAll(require.context('../components/', true, /\.js$/));

const cache = {};

function importAll (r) {

r.keys().forEach(key => cache[key] = r(key));

}

importAll(require.context('../components/', true, /\.js$/));

// At build-time cache will be populated with all required modules.

* id is the module id of the context module. This may be useful for module.hot.accept.