拉勾教育
互联网人实战大学

《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

拉勾教育

拉勾教育
—互联网人实战大学—

第19讲：把路由放在前端意味着什么？

L / A / G / O / U

路由

拉勾教育
互联网人实战大学

当浏览器地址栏中的 URL 发生变化时，会请求对应的网络资源，而负责响应这个网络资源的服务就称为**路由**

路由模块逐渐转移交给了前端进行控制

而路由转移到前端是前后端分离和单页应用架构的基石

拉勾教育

前端路由实现基础

实现前端路由的**重要基础**：

在修改 URL 时不引起浏览器向后端请求数据

前端路由实现基础

基于 hash 实现

hash 值的变化不会触发浏览器发起请求

hash 值是指 URL “#” 号后面的内容

通过 location.hash 属性可以读写 hash 值

这个值可以让浏览器将页面滚动到 ID 与 hash 值相等的 DOM 元素位置

不会传给服务端

前端路由实现基础

拉勾教育
互联网人实战大学

基于 hash 实现

通过监听 window 对象的 hashchange 事件就可以感知到hash 值的变化

这种实现方式占用了 hash 值

导致默认的页面滚动行为失效

对于有滚动定位需求的情况

需要自行手动获取元素的位置并调用 BOM 相关 API 进行滚动

前端路由实现基础

拉勾教育
互联网人实战大学

基于 history 实现

history 提供了**history.pushState()** 和 **history.replaceState()** 函数修改 URL

前者是新增一个历史记录

后者是直接替换当前的历史记录

前端路由实现基础

基于 history 实现

监听 URL 变化可以通过监听 window 对象上的 popstate 事件来实现

注意：

history.pushState() 或 history.replaceState() 不会触发 popstate 事件，这时我们需要手动触发页面渲染

前端路由实现基础

拉勾教育
互联网人实战大学

基于 history 实现

这种方式实现前端路由功能

但当用户修改地址栏网址时还是会向服务端发起请求

所以还需要服务端进行设置

将所有 URL 请求转向前端页面

交给前端进行解析

前端路由实现基础


```
location / {
  try_files $uri $uri/ /index.html;
}
```

拉勾教育

路由解析

拉勾教育
互联网人实战大学

vue-router 和 react-router 都同时依赖了一个第三方库

Path-to-RegExp 进行路由解析

路由解析

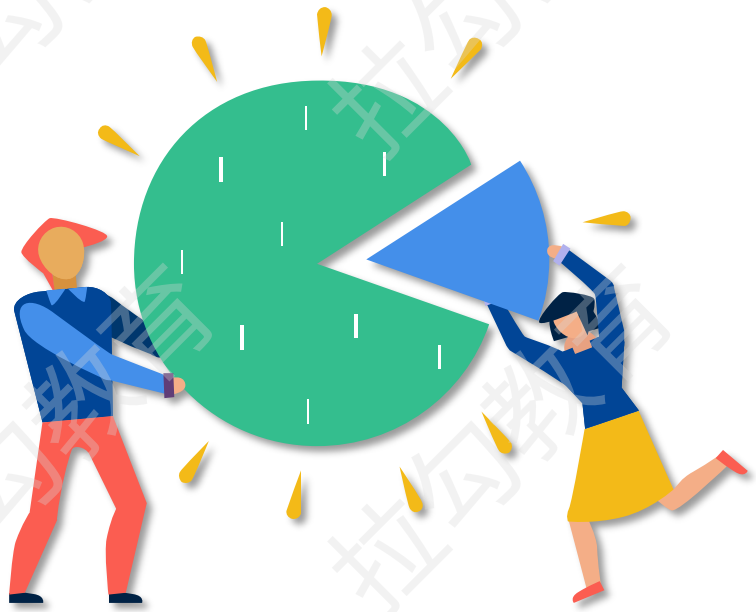
拉勾教育
互联网人实战大学

路由匹配

路由匹配就是当获取到请求路径后

如何找到对应的配置路径

在 path-to-regexp 源码中对应的是默认导出函数 **pathToRegexp()**



路由解析

拉勾教育
互联网人实战大学

路由匹配

- **path**：必传参数

  值可以为自定义的请求路径

  也可以是正则表达式

  还可以是两者组成的数组

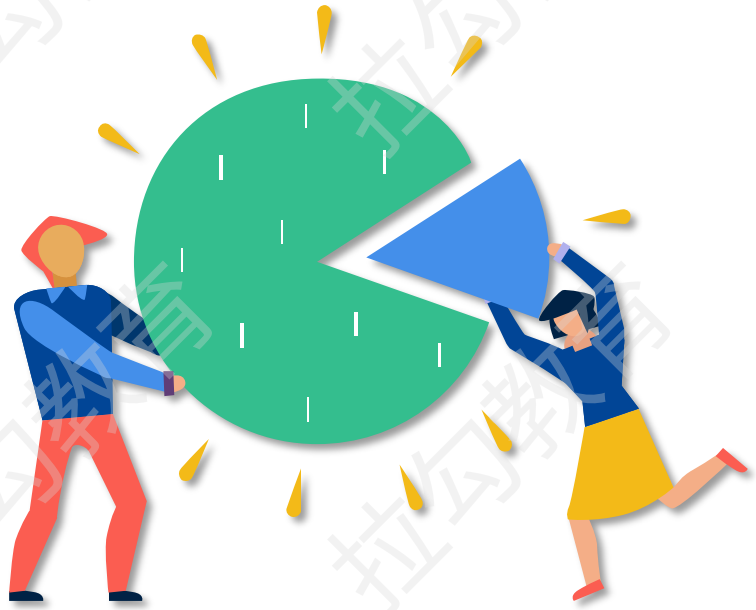
- **keys**：可选参数

  值为数组

  数组元素为解析正则表达式风格的字符串或冒号开头的占位符（下文简称为“特殊字符串”）生成的令牌

- **options**：可选参数

  执行过程中的配置信息



路由解析

拉勾教育
互联网人实战大学

```
function pathToRegexp (path, keys, options) {
  if (!isarray(keys)) {
    options = /** @type {!Object} */ (keys || options)
    keys = []
  }
  options = options || {}
  if (path instanceof RegExp) {
    return regexpToRegexp(path, /** @type {!Array} */ (keys))
  }
```

拉勾教育

路由解析


```
}
options = options || {}
if (path instanceof RegExp) {
  return regexpToRegexp(path, /** @type {!Array} */ (keys))
}
if (isarray(path)) {
  return arrayToRegexp(/** @type {!Array} */ (path), /** @type {!Array} */ (keys), options)
}
return stringToRegexp(/** @type {string} */ (path), /** @type {!Array} */ (keys), options)
```

拉勾教育

路由解析

拉勾教育
互联网人实战大学

```
function arrayToRegexp (path, keys, options) {
  var parts = []
  for (var i = 0; i < path.length; i++) {
    parts.push(pathToRegexp(path[i], keys, options).source)
  }
  var regexp = new RegExp('(?:' + parts.join('|') + ')', flags(options))
  return attachKeys(regexp, keys)
}
```

路由解析


```
function regexpToRegexp (path, keys) {
  var groups = path.source.match(/\((?!\?)/g)
  if (groups) {
    for (var i = 0; i < groups.length; i++) {
      keys.push({
        name: i,
        prefix: null,
        delimiter: null,
        optional: false,
        repeat: false,
```

拉勾教育

路由解析

拉勾教育
互联网人实战大学

```
      optional: false,
      repeat: false,
      partial: false,
      asterisk: false,
      pattern: null
    })
  }
}
return attachKeys(path, keys)
}
```

路由解析

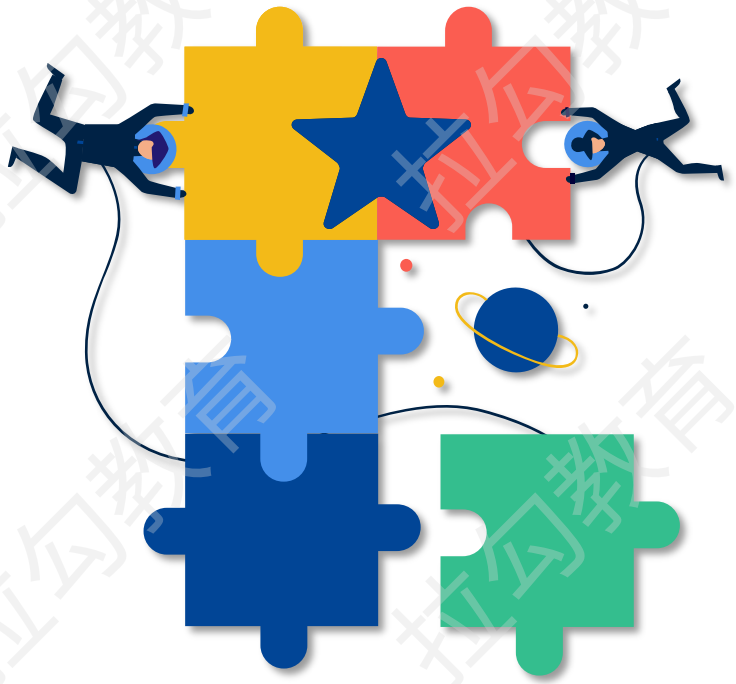

```
function stringToRegexp (path, keys, options) {
  return tokensToRegExp(parse(path, options), keys, options)
}
```

拉勾教育

路由解析

非特殊字符串：不需要做任何处理，直接以字符串形式放入数组

特殊字符串：需要依赖一个正则表达式来进行处理



路由解析


```
var PATH_REGEXP =
/(\\.)|([\/.])?(?:(?:\:(\w+)(?:\(((?:\\.|[^\\()])+)\))?|\(((?:\\.|[^\\()])+)\))([+*?])?|(\*))/g
```

路由解析


```
PATH_REGEXP.exec("/:test(\\d+)?") // ["/:test(\d+)?", undefined, "/", "test", "\d+", undefined, "?", undefined]
PATH_REGEXP.exec("/route(\\d+)") // ["(\d+)", undefined, undefined, undefined, undefined, "\d+", undefined, undefined]
PATH_REGEXP.exec("/*") // ["/*", undefined, "/", undefined, undefined, undefined, undefined, "*"]
```

拉勾教育

路由解析

拉勾教育
— 互联网人实战大学 —

```
["/user",
 {
   asterisk: false
   delimiter: "/"
   name: "id"
   optional: false
   partial: false
   pattern: "[^\/]+?"
   prefix: "/"
   repeat: false
 }
]
```

拉勾教育

路由解析


```
...
if (typeof token === 'string') {
  route += escapeString(token)
}
...
function escapeString (str) {
  return str.replace(/([.+*?=^!:${}()[\]|\/\\])/g, '\\$1')
}
```

路由解析

```
...
var prefix = escapeString(token.prefix)
var capture = '(?:' + token.pattern + ')'
keys.push(token)
if (token.repeat) {
  capture += '(?:' + prefix + capture + ')*'
}
if (token.optional) {
  if (!token.partial) {
    capture = '(?:' + prefix + '(' + capture + '))?'
  } else {
    capture = prefix + '(' + capture + ')?'
```

拉勾教育
互联网人实战大学
拉勾教育
L / A / G / O / U

路由解析

```
if (token.optional) {
 if (!token.partial) {
  capture = '(?:' + prefix + '(' + capture + '))?'
 } else {
  capture = prefix + '(' + capture + ')?'
 }
} else {
 capture = prefix + '(' + capture + ')'
}
route += capture
...
return attachKeys(new RegExp('^' + route, flags(options)), keys)
```

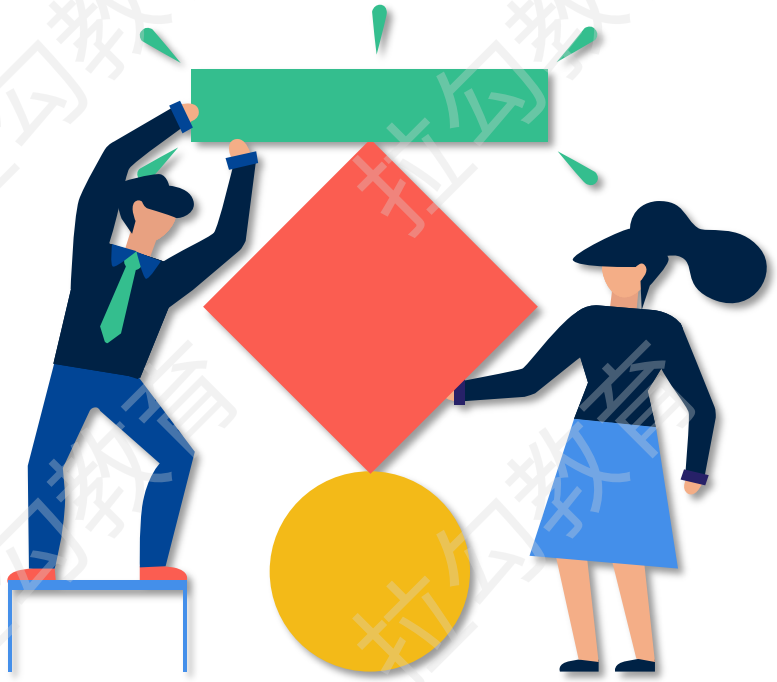
拉勾教育
互联网人实战大学
拉勾教育
L / A / G / O / U

路由解析

路由生成

路由生成是指通过配置的请求路径字符串和参数生成对应的请求路径

在 path-to-regexp 源码中对应的是函数 **compile()**



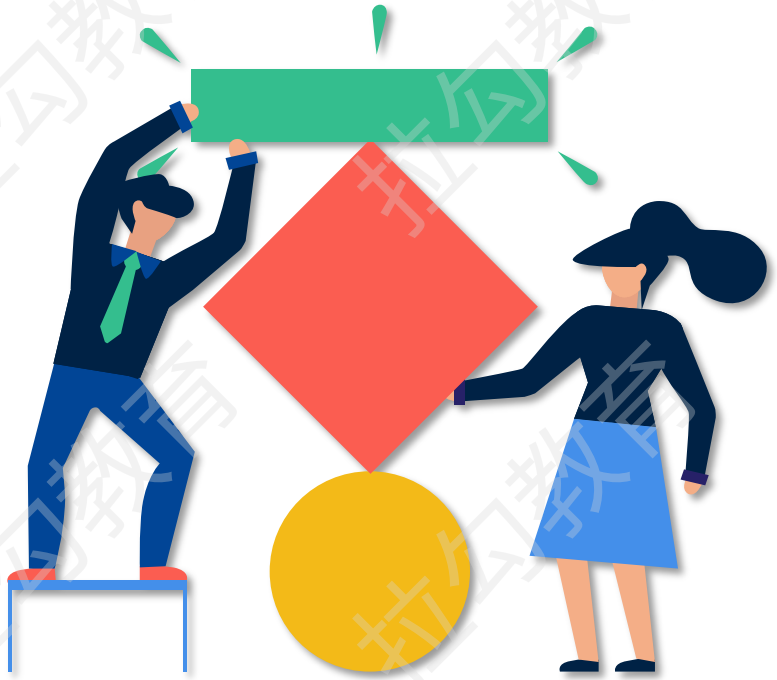
路由解析

路由生成

compile() 函数接收两个参数：**str** 和 **options**

str 为字符串，可能包含特殊字符串

options 同 pathToRegexp() 函数的 options 参数



路由解析

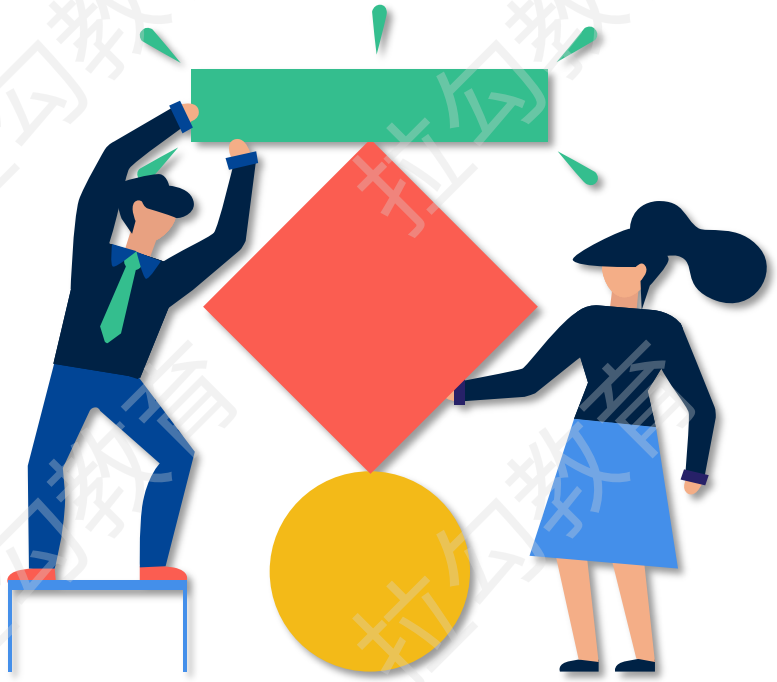
拉勾教育
互联网人实战大学

路由生成

compile() 函数并不直接生成结果字符串

而是返回一个生成函数

将参数传入这个函数中可以生成**结果字符串**



路由解析


```
function compile (str, options) {
  return tokensToFunction(parse(str, options), options)
}
```

拉勾教育

路由解析

拉勾教育
互联网人实战大学

```
for (var i = 0; i < tokens.length; i++) {
  var token = tokens[i]
  if (typeof token === 'string') {
    path += token
    continue
  }
  var value = data[token.name]
  var segment
  if (isarray(value)) {
    for (var j = 0; j < value.length; j++) {
      segment = encode(value[j])
```

拉勾教育

路由解析

拉勾教育
互联网人实战大学

```
  for (var j = 0; j < value.length; j++) {
    segment = encode(value[j])
    path += (j === 0 ? token.prefix : token.delimiter) + segment
  }
  continue
 }
 segment = token.asterisk ? encodeAsterisk(value) :
encode(value)
 path += token.prefix + segment
}
return path
```

拉勾教育

总结

拉勾教育
互联网人实战大学

前端路由的实现基础

分析了 vue-router 和 react-router 共同的依赖库 path-to-regexp 中的

两个核心函数 pathToRegexp() 和 compile()

总结

你在使用前端路由的时候碰到过哪些问题

又是怎么解决的呢？

Next：第20讲：《详解组件通信之状态管理》

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容

L / A / G / O / U

拉勾教育