起始状态：import("greeting").then(…)

↓ 进入 __webpack_require.f.j 方法

记录 installChunks、

↳ 初始时只有主入口 { main : 0 }

记录局部的 installedChunkData.

↳ 结构为：[resolve, reject, promise]

用于保存加载这个 chunk 时的 promise 实例，以及控制其状态的两个函数引用.

初始化 endLoaded 方法.

主要作用就是如果没有加载完成，则手动置空之前的 installedChunks.

并使用 installedChunkData 中的 reject 将加载错误给抛出去

↓ 进入 __webpack_require__.l 方法

初始化 inProgress ⇒ 一个 hashmap ⇒ { key：资源的链接.
value：加载好之后的回调函数列表.

查找当前是否已经在加载了.

⇒ { 是：回调里 push 一下当前的 callback.
否：

新建一个标签，inProgress 标志位设置

绑定 onScriptComplete 方法到 script 标签的 load 事件上.

script 标签插入到文档中.

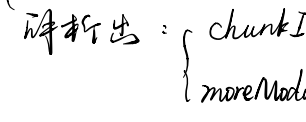
加载脚本，执行脚本

加载：浏览器行为

执行：

webpackChunkWebpack 数组的 push 方法被重新定义

调用时会默认触发 webpackJsonpCallback



解析出：{ chunkIds => ["greeting"]
moreModules => greeting 模块的 namespace

装载 module 到 __webpack_require__.m 中

一个大的 hashmap => 添加进来其实就表示装载了这个模块

执行原生的 push 操作

chunk 对应的 promise 执行 resolve

将 installedChunks 中对应的 chunkId 置为 0，表示这个 chunk 已经装好了

回到 f.j 的 loadingEnd 方法中

一些错误的处理