link null title: 珠峰架构师成长计划 description: null keywords: null author: null date: null publisher: 珠峰架构师成长计划 stats: paragraph=15 sentences=77, words=641

## 1. 模块化

模块化是指把一个复杂的系统分解到多个模块以方便编码。

开发网页要通过命名空间的方式来组织代码

```
:/code>:)命名空间冲突,两个库可能会使用同一个名称:大ode>:无法合理地管理项目的依赖和版本;:大法方便地控制依赖的加载顺序。:/口i>CommonJS是一种使用广泛的
  code>JavaScript</code>模块化规范,核心思想是通过<code>require</code>
代码如下: 
   edule.c
                                          span class="hljs-keyword">lett/span> path = <span class="hljs-built_in">require</span>(<span class="hljs-string">'path'</span>);
 <span class="hljs-keyword">let</span> b = reg(<span class="hljs-string">'./b.js'</span>);
     come where "hip-respond where filename = path.join( dirname, mod);
      cyspan class="hljs-keyword">let/span class="hljs-keyword">let</span> content = fs.readFileSync(filename, <span class="hljs-string">'utf8'</span>);
cyspan class="hljs-keyword">let</span> fn = <span class="hljs-keyword">new</span> <span class="hljs-built_in">Function</span>(<span class="hljs-</pre>
 string")'exports''</pan>, <span class="hljs-string">'require'</span>, <span class="hljs-string">'module'</span>, <span class="hljs-string">'_filename'</span>, <span class="hljs-string">'_filename'</span>, <span class="hljs-string">'_filename'</span>, <span class="hljs-string">'_filename'</span>, content + <span class="hljs-string">'\n return module.exports;'</span>);
      <span class="hljs-keyword">let</span> <span class="hljs-built_in">module</span> = {
            capan schane-Thills-activity-appointer/apano: {}
      <span class="hljs-keyword">return</span> fn(<span class="hljs-built_in">module</span>. __filename,
  dirname);
 magnarian magna
 requirejs。AMD 的优点可在不转換代码的情况下直接在浏览器中运行可加载多个依赖代码可运行在浏览器环境和 Node.js 环境下大/p>AMD 的缺点大/p>JavaScript 运行环境没有原生支持 AMD,需要先导入实现了 AMD 的库后才能正常使用。JavaScript 运行环境没有原生支持 AMD,需要先导入实现了 AMD 的库后才能正常使用。
   r[imm[<span class="hljs-string">'a'</span>, [], <span class="hljs-function"><span class="hljs-keyword">function</span> (<span class="hljs-params"></span>)
 </span>{
      <span class="hljs-keyword">return</span> <span class="hljs-string">'a'</span>;
 define(<span class="hljs-string">'b'</span>, [<span class="hljs-string">'a'</span>], <span class="hljs-function"><span class="hljs-keyword">function</span>
 (<span class="hljs-params">a</span>) </span>{
      <span class="hljs-keyword">return</span> a + <span class="hljs-string">'b'</span>;
 <span class="hljs-built in">require</span>([<span class="hljs-string">'b'</span>], <span class="hljs-function"><span class="hljs-keyword">function</span> (<span)</pre>
 class="hljs-params">b</span>) </span>{
     capan alama = 11 pa-hailE in = 200 annalac/apano.log(b);
 //span>{
      factory.dependencies = dependencies;
      factories[modName] = factory;
          //span>{
      <span class="hljs-keyword">let</span> loadedModNames = modNames.map(<span class="hljs-function"><span class="hljs-keyword">function</span> (<span class="hljs-keyword")</pre>
 class="hljs-params">modName</span>) </span>{
           Capus olars Thips require Thirty apart factory = factories[modName];
            capan plana - 11 pa-mayan - 12 pa-mayan - 2 part dependencies = factory.dependencies;
            capan miana Thips-negatifizing/apano exports;
  exports = factory.apply(capan mlans in principle of the paper of the paper);
            capan mlana Tilja-mayanni vantauns/apano exports;
      callback.apply(come days);
          >ES6 模块化是<code>ECMA</code>提出的<code>JavaScript</code>模块化規范,它在语言的层面上实现了模块化。浏览器厂商和<code>Node.j</code>s<code>都宣布要原生支持该
機范。它将逐渐取代</code>CommonJS<code>和</code>AMD、機范,成为浏览器和服务器通用的模块解决方案。 采用 ES6 模块化导入及导出时的代码如下
 <span class="hljs-keyword">import</span> { name } <span class="hljs-keyword">from</span> <span class="hljs-string">'./person.js'</span>;

機定文到哲库則而要校验不時是自行官规范。以及平元獨或差合理近。</fl>
《/li>
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
》
<
# 安装指定版本
 npm i -D webpack@<version>
# 安装最新体验版本
```

```
npm i -D webpack@beta
  /code><blookquote>npm i -D 是 <code>npm install --save-dev</code> 的简写,是指安装模块并保存到 <code>
| Amazon |
x/h2>ch2 id="t153.1.3 使用 Webpack">3.1.3 使用 Webpack </h2>cyne>ccode class="lang-javascript">(<span class="hljs-function"><span class="hljs-keyword">function</span> (<span class="hljs-params">modules</span>) </span>(
</span>{
                <span class="hljs-keyword">var</span> <span class="hljs-built_in">module</span> = {
               modules[moduleId].call(<span class="hljs-built in">module</span>.exports, <span class="hljs-built in">module</span>, <span class="hljs-built in">module</span class="hljs-built in">module</span class="hljs-built in"</span class="hljs-built i
 ouilt_in">module</span>.exports, <span class="hljs-built_in">require</span>);
               <span class="hljs-keyword">return</span> <span class="hljs-built in">module</span>.exports;
       . (span class="hljs-keyword">return</span> (span class="hljs-built_in">require</span>((span class="hljs-string">"./index.js"</span>);
               capan schang-Thillip-actingToT./index.jaTc/apano;
                       (capua alama-11)
                               <span class="hljs-built_in">eval</span>(<span class="hljs-string">"console.log('hello');\n\n"</span>);
<span class="hljs-keyword">const</span> pathLib = <span class="hljs-built in">require</span>(<span class="hljs-string">rpath'</span>);
cspan class="hljs-keyword">const</span> fs = <span class="hljs-built_in">require</span>(<span class="hljs-string">'fs'</span>);
cspan class="hljs-keyword">let</span> ejs = <span class="hljs-built_in">require</span>(<span class="hljs-string">'ejs'</span>);
       e class milje-regeoremsteds/e
                                                                      while cwd = process.cwd();
cypan class="hljs-keyword">let</span> { entry, <span class="hljs-attr">output</span>: { filename, path } } = <span class="hljs-built_in">require</span>
(pathiib.join(cwd, <span class="hljs-string">'./webpack.config.js'</span>));
<span class="hljs-keyword">let</span> script = fs.readFileSync(entry, <span class="hljs-string">'utf8'</span>);
       function require(moduleId) {
               var module = {
                       exports: {}
                moduleo[moduleOd].coll(module.coporto, module, module.coporto, sequise);
               setum module.coporto,
        return regular ("ct-entryto") ;
                       (function (module, exports, require) {
   eval("<%-script%>");
</span>
                           bundlejs = ejs.render(bundle, {
      entry,
     script
       fs.writeFileSync(pathLib.join(path, filename), bundlejs);
                                         mayaand Daataba (e) {
       <span class="hljs-built_in">console</span>.error(<span class="hljs-string">'编译失败 '</span>, e);
span class="hljs-built_in">console</span>.log(<span class="hljs-string">'compile sucessfully!'</span>);
</code><h2 id="t163.1.4 依赖其它模块">3.1.4 依赖其它模块 </h2><code class="lang-javascript">
<span class="hljs-keyword">const</span> pathLib = <span class="hljs-built_in">require</span>(<span class="hljs-string">'path'</span>);<span class="hljs-keyword">const</span> fs = <span class="hljs-built_in">require</span>(<span class="hljs-string">'fs'</span>);
<span class="hljs-keyword">let</span> ejs = <span class="hljs-built_in">require</span>(<span class="hljs-string">'ejs '</span>);
                            (span class="hljs-keyword">let//span > (span class="hljs-keyword">let//span > (span class="hljs-built_in">require</span> (pathLib.join(cwd, <span class="hljs-string">'./webpack.config.js'</span>));
 span class="hljs-keyword">let</span> script = fs.readFileSync(entry, <span class="hljs-string">'utf8'</span>);
       n alama "hija-manana" kimiki/apana modules = [];
script.replace(<span class="hljs-regexp">/require\(['"](.+?)['"]\)/g</span>, <span class="hljs-function"><span class="hljs-keyword">function</span> (<span
class="hljs-params"></span>) </span>{
     <span class="hljs-keyword">let</span> name = <span class="hljs-built_in">arguments</span>[<span class="hljs-number">l</span>];
<span class="hljs-keyword">let</span> script = fs.readFileSync(name, <span class="hljs-string">'utf8'</span>);
       modules.push({
              name,
               script
                    a-fid pa-seyaon@xlads/aparo madde - capan atasa-fid pa-asiring
                    modulesci 4
       function require(moduleId) {
               var module = {
                     exports: {}
                modules [modulets] - call p
                                                                      dule.coporto, module, module.coporto, req
               setuan module.comovios
      refuse regular (**CE-enCayEO*) a
              "<%-entry%>":
                       (function (module, exports, require) {
                              eval(\`<%-script%>\`);
             <%if(modules.length>0){%>,<%}\lim
               cufaction in a removales, length, length
                        let module = modules[i];%>
                          "<%-module.name%>":
                        (function (module, exports, require) {
                                eval(\'<%-module.script%>\');
               <% }%>
```

```
| ));

'</span>

Compared and and the companied of the c
```