## **1. applyModuleIds**

applyModuleIds() {

**const** unusedIds = [];//找到当前未使用的 id

**let** nextFreeModuleId = 0;//已经使用的最大的 id

**const** usedIds = **new** Set();

**if** (**this**.usedModuleIds) {

**for** (**const** id **of** **this**.usedModuleIds) {

usedIds.add(id);

}

}

**const** modules1 = **this**.modules;

**for** (**let** indexModule1 = 0; indexModule1 < modules1.length; indexModule1++) {

**const** module1 = modules1[indexModule1];

**if** (module1.id !== null) {

usedIds.add(module1.id);

}

}

**if** (usedIds.size > 0) {

**let** usedIdMax = -1;

**for** (**const** usedIdKey **of** usedIds) {

**if** (**typeof** usedIdKey !== "number") {

**continue**;

}

usedIdMax = Math.max(usedIdMax, usedIdKey);

}

**let** lengthFreeModules = (nextFreeModuleId = usedIdMax + 1);

**while** (lengthFreeModules--) {

**if** (!usedIds.has(lengthFreeModules)) {

unusedIds.push(lengthFreeModules);

}

}

}

**const** modules2 = **this**.modules;

**for** (**let** indexModule2 = 0; indexModule2 < modules2.length; indexModule2++) {

**const** module2 = modules2[indexModule2];

**if** (module2.id === null) {

**if** (unusedIds.length > 0) {

module2.id = unusedIds.pop();

} **else** {

module2.id = nextFreeModuleId++;

}

}

}

}

## **2. applyChunkIds**

applyChunkIds() {

/\*\* @type **{Set<number>}** \*/

**const** usedIds = **new** Set();

// Get used ids from usedChunkIds property (i. e. from records)

**if** (**this**.usedChunkIds) {

**for** (**const** id **of** **this**.usedChunkIds) {

**if** (**typeof** id !== "number") {

**continue**;

}

usedIds.add(id);

}

}

// Get used ids from existing chunks

**const** chunks = **this**.chunks;

**for** (**let** indexChunk = 0; indexChunk < chunks.length; indexChunk++) {

**const** chunk = chunks[indexChunk];

**const** usedIdValue = chunk.id;

**if** (**typeof** usedIdValue !== "number") {

**continue**;

}

usedIds.add(usedIdValue);

}

// Calculate maximum assigned chunk id

**let** nextFreeChunkId = -1;

**for** (**const** id **of** usedIds) {

nextFreeChunkId = Math.max(nextFreeChunkId, id);

}

nextFreeChunkId++;

// Determine free chunk ids from 0 to maximum

/\*\* @type {number[]} \*/

**const** unusedIds = [];

**if** (nextFreeChunkId > 0) {

**let** index = nextFreeChunkId;

**while** (index--) {

**if** (!usedIds.has(index)) {

unusedIds.push(index);

}

}

}

// Assign ids to chunk which has no id

**for** (**let** indexChunk = 0; indexChunk < chunks.length; indexChunk++) {

**const** chunk = chunks[indexChunk];

**if** (chunk.id === null) {

**if** (unusedIds.length > 0) {

chunk.id = unusedIds.pop();

} **else** {

chunk.id = nextFreeChunkId++;

}

}

**if** (!chunk.ids) {

chunk.ids = [chunk.id];

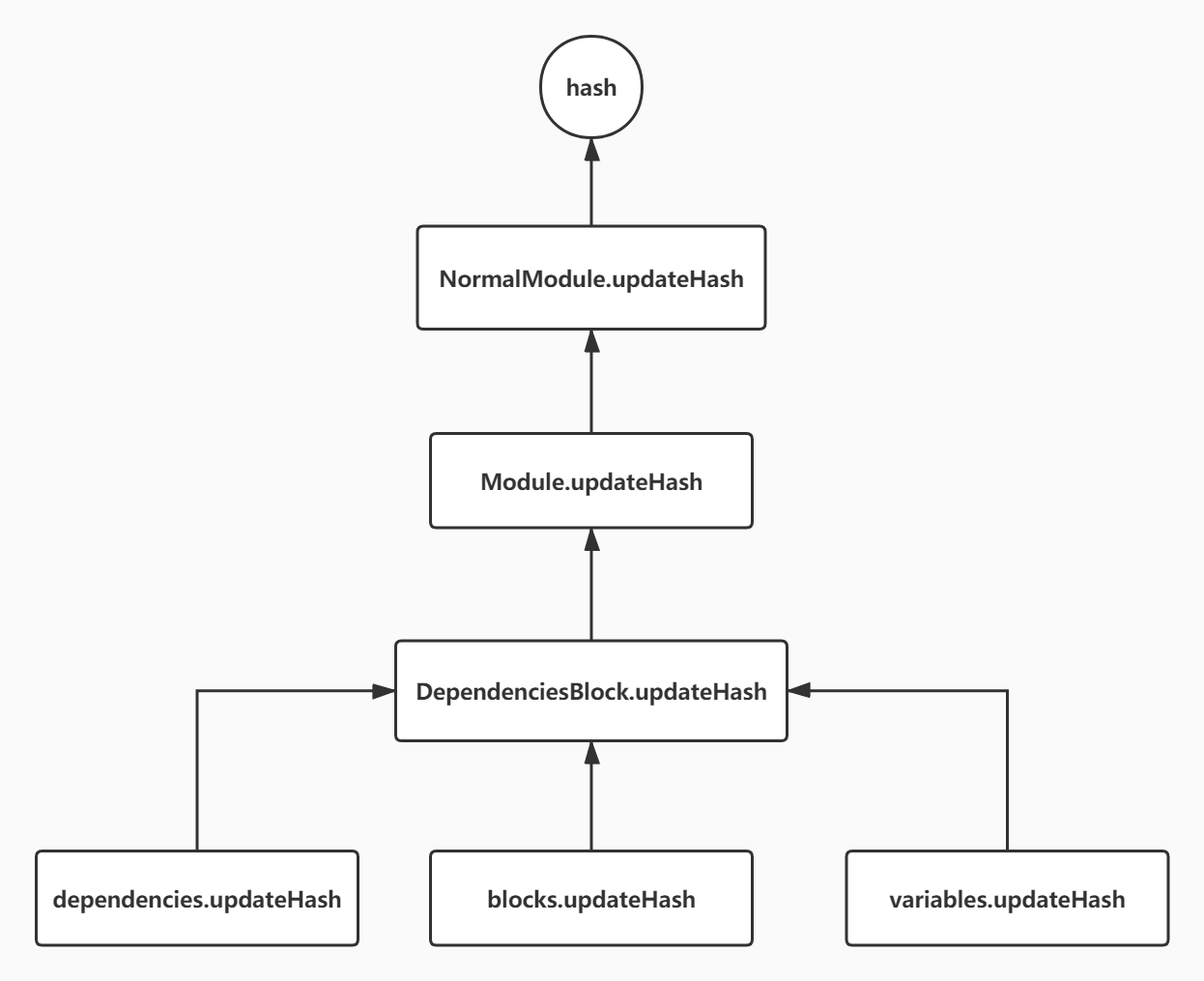
}

}

}

## **3. hash**

### **3.1 module hash**



Module.js

updateHash(hash) {

hash.update(`${**this**.id}`);

hash.update(JSON.stringify(**this**.usedExports));

**super**.updateHash(hash);

}

### **3.2 chunk hash**

Chunk.js

updateHash(hash) {

hash.update(`${**this**.id} `);

hash.update(**this**.ids ? **this**.ids.join(",") : "");

hash.update(`${**this**.name || ""} `);

**for** (**const** m **of** **this**.\_modules) {

hash.update(m.hash);

}

}

## **4. createChunkAssets**

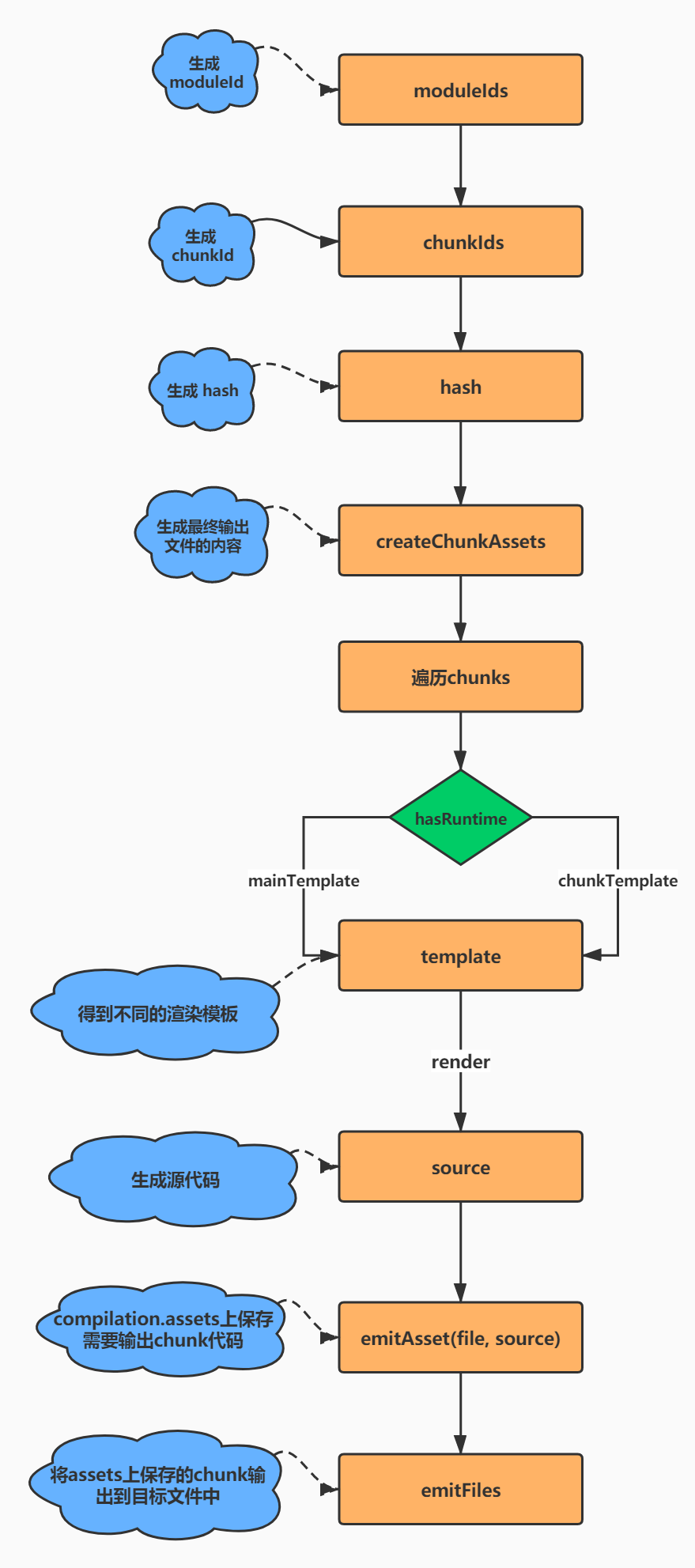
* [JavascriptModulesPlugin](https://zhufeng-document.vercel.app/html/JavascriptModulesPlugin)
* [MainTemplate.js](https://zhufeng-document.vercel.app/html/MainTemplate.js)

[JsonpMainTemplatePlugin.js](https://zhufeng-document.vercel.app/html/JsonpMainTemplatePlugin.js)

hash 值生成之后，会调用 createChunkAssets 方法来决定最终输出到每个 chunk 当中对应的文本内容

获取对应的渲染模板

* 然后通过 getRenderManifest 获取到 render 需要的内容
* 执行 render() 得到最终的代码
* 获取文件路径，保存到 assets 中



## **4.hash**

* hash 每次编译会生成一个hash,代表这次编译 代码: [https://github.com/webpack/webpack/blob/c9d4ff7b054fc581c96ce0e53432d44f9dd8ca72/lib/Compilation.js#L1985](https://github.com/webpack/webpack/blob/c9d4ff7b054fc581c96ce0e53432d44f9dd8ca72/lib/Compilation.js" \l "L1985)
* chunkhash 每个chunk代码块对应的哈希值，各个chunk之间独立 代码: [https://github.com/webpack/webpack/blob/c9d4ff7b054fc581c96ce0e53432d44f9dd8ca72/lib/Compilation.js#L1976](https://github.com/webpack/webpack/blob/c9d4ff7b054fc581c96ce0e53432d44f9dd8ca72/lib/Compilation.js" \l "L1976)
* contenthash 文件内容级别的哈希值,文件内容变了，那么hash值才改变 代码: [https://github.com/webpack/webpack/blob/c9d4ff7b054fc581c96ce0e53432d44f9dd8ca72/lib/Compilation.js#L1979](https://github.com/webpack/webpack/blob/c9d4ff7b054fc581c96ce0e53432d44f9dd8ca72/lib/Compilation.js" \l "L1979)