互联网应用开发技术

Web Application Development

第3课
WEB前端-WEBPACK

Episode Three

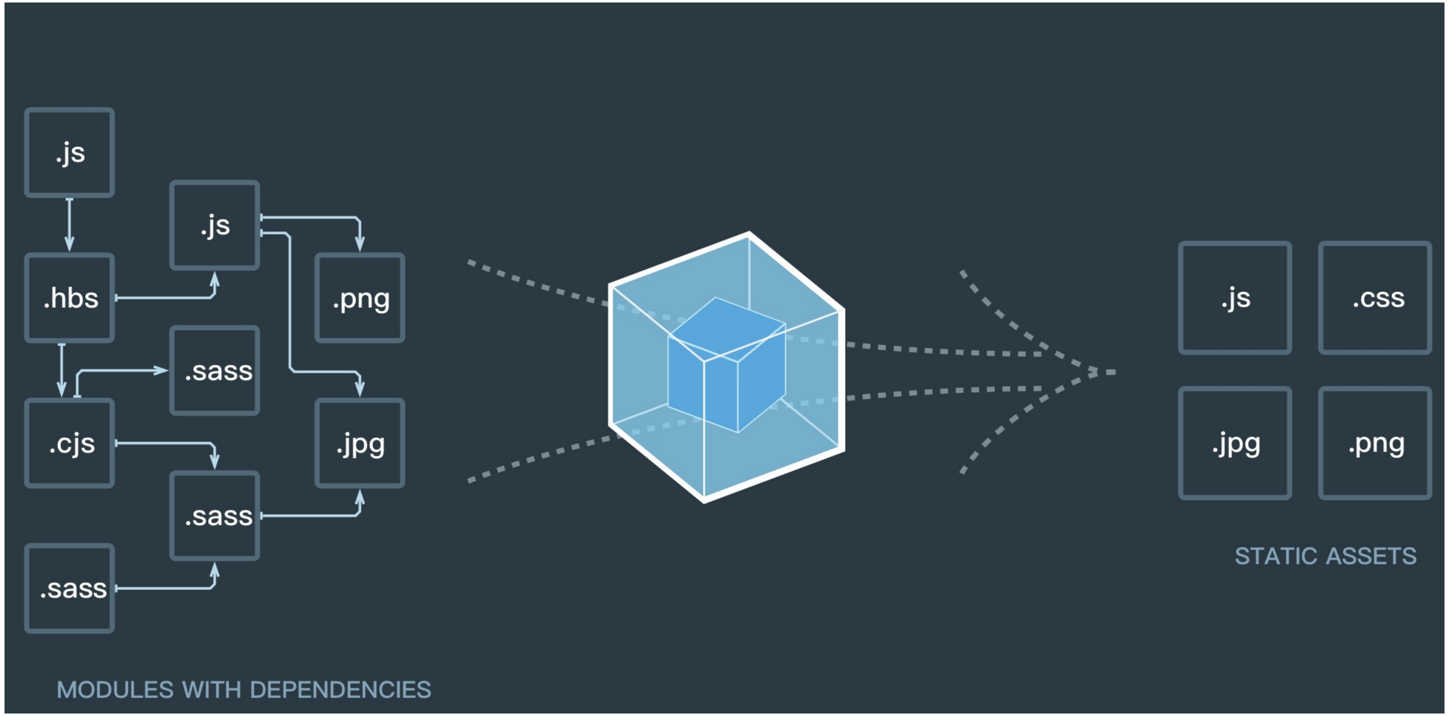
Webpack

陈昊鹏

chen-hp@sjtu.edu.cn

Web Application Development

Webpack


.js
.hbs
.js
.png
.sass
.cjs
.jpg
.sass
.sass
MODULES WITH DEPENDENCIES
.js
.css
.jpg
.png
STATIC ASSETS

Webpack

REliable, INtelligent & Scalable Systems

编写代码

src/index.js

```
import bar from './bar';

bar();
```

src/bar.js

```
export default function bar() {
  //
}
```

使用 webpack 打包

Without config or provide custom **webpack.config.js**

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
      path: path.resolve(__dirname, 'dist'),
    filename: 'bundle.js'
  }
};
```

page.html

```
<!doctype html>
<html>
  <head>
    ...
  </head>
  <body>
    ...
    <script src="dist/bundle.js"></script>
  </body>
</html>
```

Webpack

- What is Webpack
  - a static module bundler for modern JavaScript applications
  - building a dependency graph
  - generating one or more bundles

- Why Webpack
  - let you bundle your JavaScript applications
  - solve dependency issues in building JavaScript APPs
  - can be extended to support many different assets like CSS
  - can be integrated with mainstream frontend frameworks

An example without Webpack

project

```
webpack-demo
|- package.json
|- index.html
|- /src
  |- index.js
```

src/index.js

```
function component() {
  const element = document.createElement('div');

  // Lodash, currently included via a script, is required for this line to work
  element.innerHTML = _.join(['Hello', 'webpack'], ' ');

  return element;
}

document.body.appendChild(component());
```

An example without Webpack

index.html

```html
<!doctype html>
<html>
  <head>
    <title>Getting Started</title>
    <script src="https://unpkg.com/lodash@4.16.6"></script>
  </head>
  <body>
    <script src="./src/index.js"></script>
  </body>
</html>
```

An example without Webpack

- Drawbacks
  - It is not immediately apparent that the script depends on an external library.
  - If a dependency is missing, or included in the wrong order, the application will not function properly.
  - If a dependency is included but not used, the browser will be forced to download unnecessary code.

How about Webpack?

An example with Webpack

project

```
webpack-demo
|- package.json
|- webpack.config.js
|- /dist
  |- bundle.js
  |- index.html
|- /src
  |- style.css
  |- index.js
|- /node_modules
```

node modules

```
npm install --save lodash
```

An example with Webpack

src/index.js

```
+ import _ from 'lodash';
+ import './style.css';

  function component() {
    const element = document.createElement('div');

+   // Lodash, now imported by this script
    element.innerHTML = _.join(['Hello', 'webpack'], ' ');
+   element.classList.add('hello');

    return element;
  }

  document.body.appendChild(component());
```

An example with Webpack

REliable, INtelligent & Scalable Systems

src/style.css

```
.hello {
  color: red;
}
```

dist/index.html

```
  <!doctype html>
  <html>
   <head>
     <title>Getting Started</title>
-    <script src="https://unpkg.com/lodash@4.16.6"></script>
   </head>
   <body>
-    <script src="./src/index.js"></script>
+    <script src="bundle.js"></script>
   </body>
  </html>
```

webpack.config.js

REliable, INtelligent & Scalable Systems

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  module: {
    rules: [
      {
        test: /\.css$/,
        use: [
          'style-loader',
          'css-loader',
        ],
      },
    ],
  },
};
```

webpack.config.js

- Entry
  - indicate which module webpack should use to begin building out its internal dependency graph
  - webpack will figure out which other modules and libraries that entry point depends on (directly and indirectly)
- Output
  - tell webpack where to emit the bundles it creates and how to name these files
- Loaders
  - allow webpack to process other types of files and convert them into valid modules and added to the dependency graph

packages.json

```json
{
    "name": "webpack-demo",
    "version": "1.0.0",
    "description": "",
    "scripts": {
      "test": "echo \"Error: no test specified\" && exit 1",
      "build": "webpack"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "devDependencies": {
      "webpack": "^4.20.2",
      "webpack-cli": "^3.1.2"
    },
    "dependencies": {
      "lodash": "^4.17.5"
    }
 }
```

packages.json

- Concept
  - it is core to the Node.js ecosystem and is a basic part of understanding and working with Node.js, npm, and even modern JavaScript
  - it is a manifest about applications, modules and packages
  - it is a tool to that's used to make modern development streamlined, modular, and efficient
  - can be generated by npm init

- Property
  - name: the name of the module
  - version: the current version of the module
  - license: the license of the module
  - main: a direction to the entry point to the module
  - scripts: used as a build tool to execute pre-defined command
  - dependencies: the dependencies that a module needs to run in production
  - devDependencies: the dependencies the module needs to run in development

Webpack

- run the scripts

```
npm run build

...
    Asset      Size  Chunks             Chunk Names
bundle.js  76.4 KiB       0  [emitted]  main
Entrypoint main = bundle.js
...
```

- the bundle.js has emitted in ./dist

- the index page can access the JavaScript file now !

References

- Webpack
  - https://webpack.js.org
- Webpack中文文档
  - https://www.webpackjs.com

- *Web开发技术*
- *Web Application Development*

Thank You!