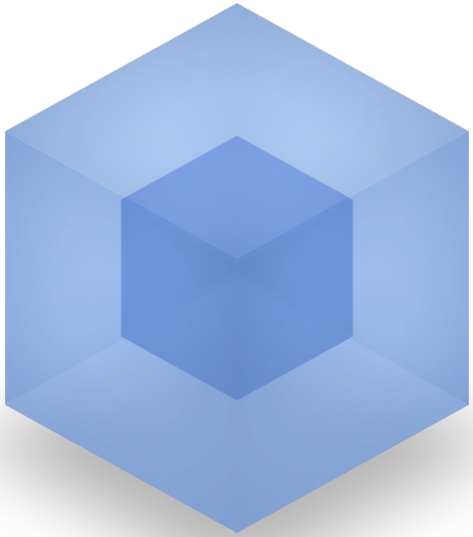
Introduction to Webpack

Cerize @ Bananatag L&L

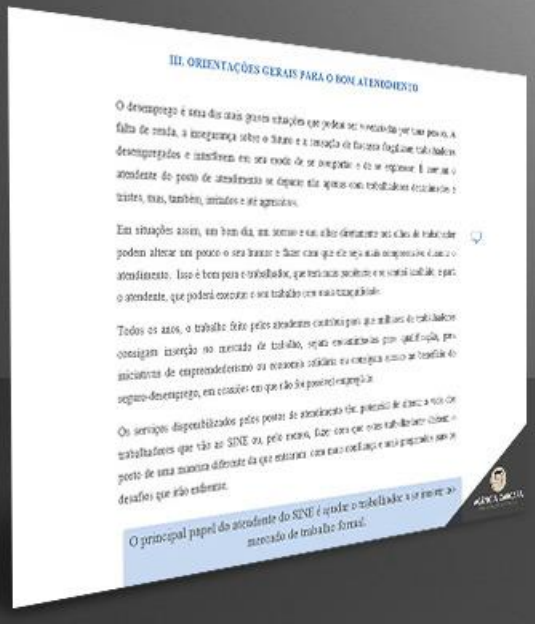


webpack

MODULE BUNDLER

Development vs Production: expectation

before

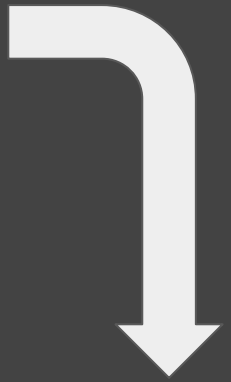


after

```
// This will be my standard 'hello'

var greetings = 'Hello Bananatag!';

// Don't touch here, it's magic

var n = 25200000;

var date = new Date('2016-09-29');

var dateAdjusted = new Date(date.getTime() + n);

// Wow such comment

console.log(greetings, 'Today is', dateAdjusted);
```

Reality

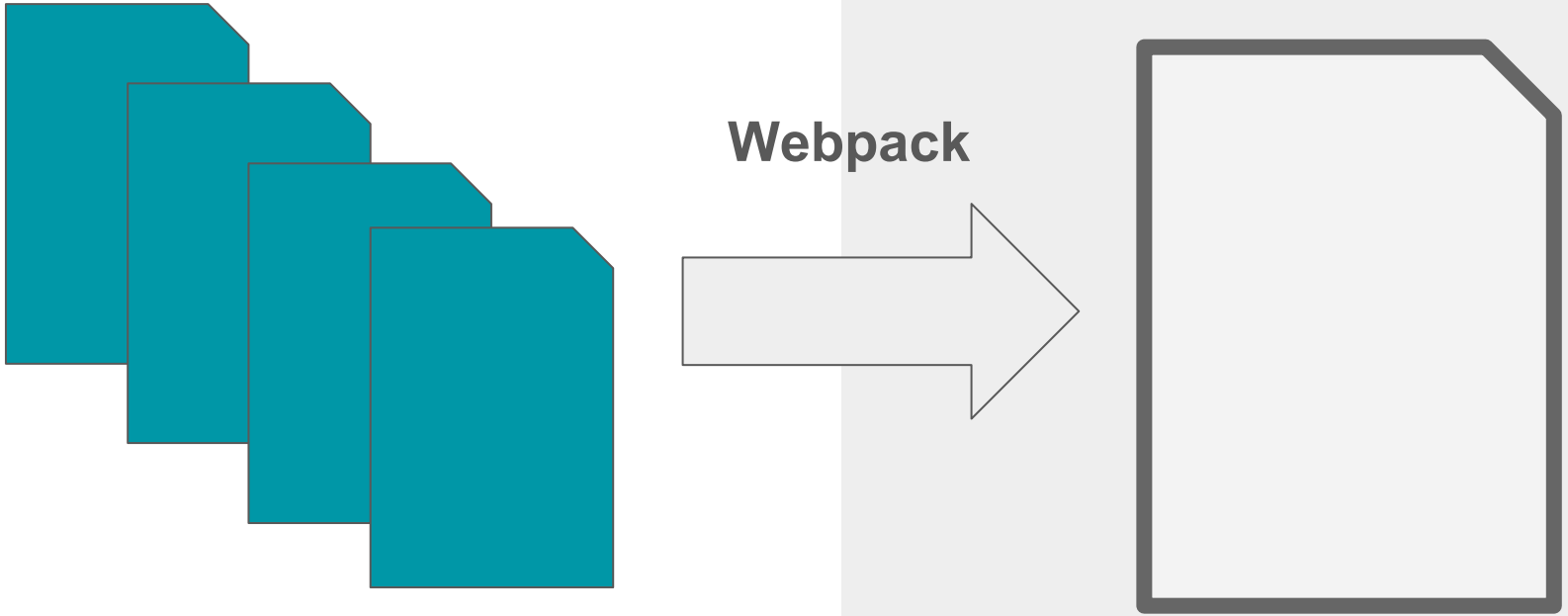
The minified output (156 bytes, saved 45.07%



```
var greetings="Hello Bananatag!",n=252e5,date=new
Date("2016-09-29"),dateAdjusted =new
Date(date.getTime()+n);console.log(greetings,"Today is",dateAdjusted);
```

Development

Production



Motivation

Fact 1: More and more code on the client side….A big code base needs to be organized…

Fact 2: To be manageable, large applications must be separated in parts

Fact 3: Module systems offer the option to split your code base into modules.

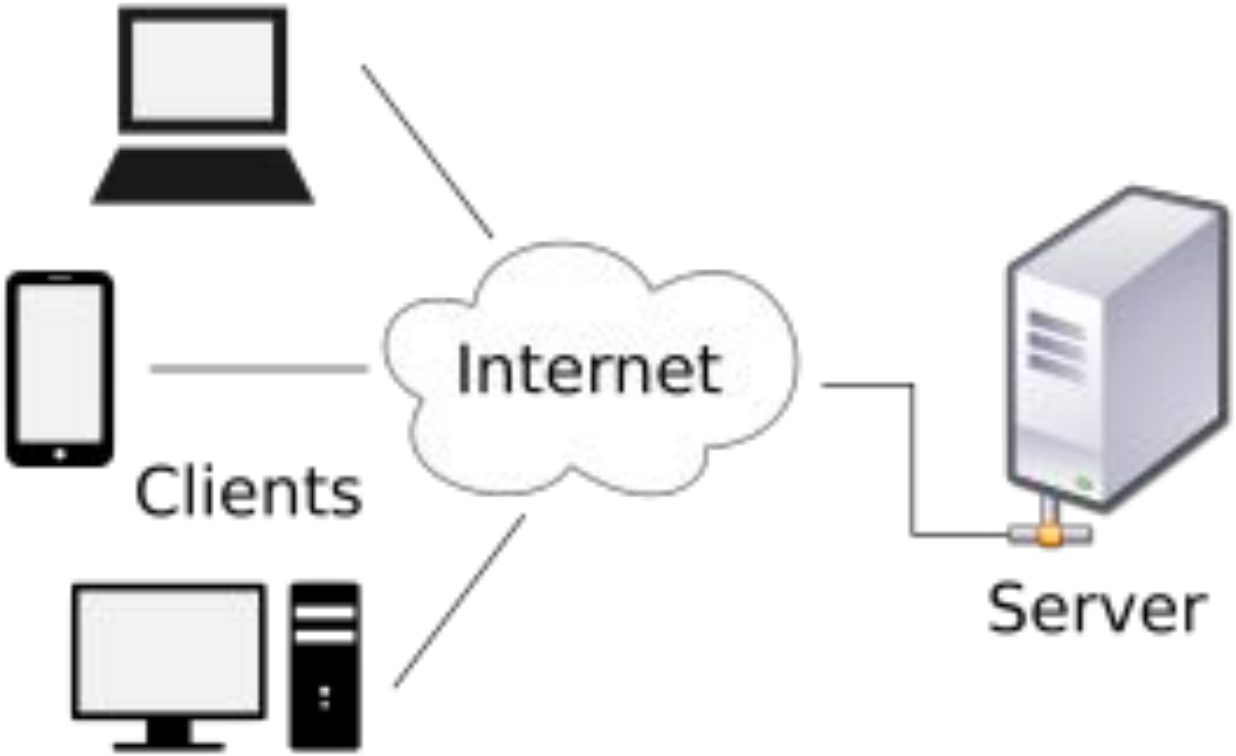
Fact 4: Module is the main **Javascript** design pattern

```
// CommonJs

    require("module");
    require("../file.js");
    exports.doStuff = function()
    {};
```

```
  // ES6

    export const sqrt = Math.sqrt;
    export function square(x) {
        return x * x;
    }
    export function diag(x, y) {
        return sqrt(square(x) +
square(y));
```

Motivation

Fact 4: Until ES6, JavaScript did not have built-in support for modules, which means: NO MODULES IN THE BROWSER FOR YOU :(


Internet
Clients
Server

Webpack

Packs CommonJs/AMD modules for the browser.

Allows to split your codebase into multiple bundles, which can be loaded on demand.

Support loaders to preprocess files, i.e. json, jsx, es7, css, less, ... and your custom stuff.

It was authored by Tobias Koppers @sokra on Mar, 2012.

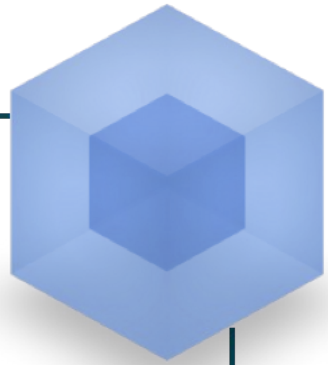
cats.js

```
var cats = ['dave', 'henry', 'martha'];
module.exports = cats;
```

app.js

```
var cats = require('./cats.js');
console.log(cats);
```

1 webpack reads the entry point and analyzes its dependencies, its dependencies' dependencies, and so on.



webpack
MODULE BUNDLER

2 webpack bundles the entry point and all its dependencies into a single file.

app.bundle.js

```
!function(r){function t(o){if(n[o])return n[o].exports;
var e=n[o]={i:o,l:!1,exports:{}};return r[o].call(
e.exports,e,e.exports,t),e.l=!0,e.exports}var n={};
return t.m=r,t.c=n,t.p="",t(t.s=1)}([function(r,t){
var n=["dave","henry","martha"];r.exports=n},function
(r,t,n){cats=n(0),console.log(cats)}]);
```

Is that all?

- Preloaders
- Loaders
  - Can be chained (pipeline)
  - Only the final load must return js
  - Can be bound to Regex/extensions
  - Examples: **babel, css, sass, flow**
- Plugins
- SourceMaps
- Code Splitting

https://webpack.github.io/docs/list-of-loaders.html

Installation and Use

```
$ npm install webpack -g

$ npm install webpack-dev-server --save-dev

    webpack ./app.js bundle.js
```

webpack.config.js

```
var webpack = require('webpack');
var path = require('path');
var config = require('./configs/config');

module.exports = {
    entry: [
        path.join(__dirname, 'web', 'src', 'js', 'client.js'),
        // path.join(__dirname, 'web', 'static', 'css', 'style.css')
    ],
    output: {
        path: path.join(__dirname, 'web', 'static', 'dist'),
        filename: 'bundle.js'
    },
    resolve: {
        extensions: ['', '.js', '.jsx']
    },
```

```
  module: {
      loaders: [
          {
              test: /.jsx?$/,
              loader: 'babel',
              exclude: /node_modules/
          },
          {
              test: /\.json$/, loader: 'json'
          },
          // {
          //     test: /\.css$/, loader: 'style!css?sourceMap'
          // },
          // {
          //     test: /\.s(a|c)ss$/, loader: 'style!css?sourceMap!sass?indentedSyntax'
          // },
          // {
          //     test: /\.(png|jpg)$/, loader: 'url?limit=8192' // inline base64 URLs for <=8k images,
direct URLs for the rest
          // }

      ]
  },
```

```
plugins: [
    new webpack.DefinePlugin({
        'process.env': {
            // This has effect on the react lib size
            'NODE_ENV': JSON.stringify('production'),
            'API_SERVER': JSON.stringify('http://' + config.apiHost + ':' + config.apiPort)
        }
    }),
    new webpack.optimize.OccurenceOrderPlugin(),
    new webpack.optimize.DedupePlugin(),
    new webpack.optimize.UglifyJsPlugin({
        minimize: true,
        warning: false,
        compress: {
            warnings: false,
            dead_code: true,
            loops: true,
            unused: true,
            if_return: true,
            join_vars: true
        }
    })
]
```

References

https://webpack.github.io/docs/

https://medium.com/@dabit3/beginner-s-guide-to-webpack-b1f1a3638460#.73w1zc23y

http://survivejs.com/webpack/developing-with-webpack/

My notes

CommonJS is a project with the goal of specifying an ecosystem for JavaScript outside the browser

The dev server uses Webpack's watch mode. It also prevents webpack from emitting the resulting files to disk. Instead it keeps and serves the resulting files from memory.

With Webpack dev server running, you will notice that if you go back to your app and make a change, the browser will automatically refresh (hot-loading).