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=82 sentences=140, words=856

1.核心概念

1.1 设备物理像素

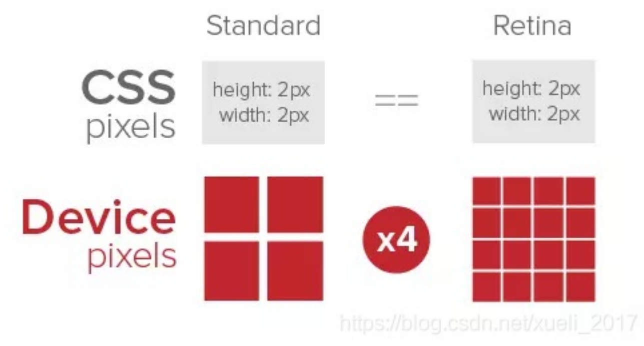
- 是一个物理概念，是显示器显示的最小物理单位
- `iPhone6`的像素分辨率是750*1334`
- `px`是一个相对单位，相对的是设备像素(device pixel)

1.2 设备独立像素

- 是一个逻辑概念,用于向CSS中的宽度、高度等提供信息
- iPhone6的逻辑分辨率是 `375*667`
- iPhone6: `window.screen.width=375,window.screen.height=667`

1.3 设备像素比

- DPR(设备像素比) = 设备像素/CSS像素
- 设备像素比 `window.devicePixelRatio`



1.4 移动端适配

- 一般由设计师按照设备像素(device pixel)为单位制作设计稿
- 然后由前端工程师参照设备像素比(device pixel ratio)进行换算

1.4.1 rem

- 参照根元素的字体大小
- 适配就是让根元素的字体大小根据分辨率进行动态改变
- [px2rem-loader (https://www.npmjs.com/package/px2rem-loader)](https://www.npmjs.com/package/px2rem-loader)

1.4.2 vw和vh

- 参照的是viewport视口
- vw参照的是视口的宽度(1vw=视口宽度/100)
- vh参照的是视口的高度(1vh=视口高度/100)
- iPhone6 1vw=3.75px
- [postcss-px-to-viewport (https://www.npmjs.com/package/postcss-px-to-viewport)](https://www.npmjs.com/package/postcss-px-to-viewport)

型号 宽度 1vw iPhone6 375 3.75px

750px 75px

1vw=7.5px 10vw=75px

```
75/10
```

2.px2rem-loader实战

2.1 安装

- [lib-flexible (https://github.com/amfe/lib-flexible)](https://github.com/amfe/lib-flexible)
- [px2rem-loader (https://www.npmjs.com/package/px2rem-loader)](https://www.npmjs.com/package/px2rem-loader)

```
npm install webpack webpack-cli html-webpack-plugin style-loader css-loader amfe-flexible px2rem-loader --save-dev
```

2.2 src\index.js

```
import './base.css'
```

2.3 src\base.css

```
#root{
    width:750px;
    height:750px;
}
```

2.4 src\index.html

src\index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>webpacktitle>
head>
<body>
    <div id="root">div>
body>
html>
```

2.5 webpack.config.js

webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
    mode: 'development',
    devtool: false,
    entry: './src/index.js',
    output: {
        path: path.resolve(__dirname, 'dist'),
        filename: '[name].js'
    },
    module: {
        rules: [{
            test: /\.css$/,
            use: [{
                loader: 'style-loader'
            }, {
                loader: 'css-loader'
            }, {
                loader: 'px2rem-loader',
                options: {
                    remUni: 75,
                    remPrecision: 8
                }
            }]
        }]
    },
    plugins: [
        new HtmlWebpackPlugin({ template: './src/index.html' })
    ]
};
```

2.6 package.json

package.json

```
{
   "scripts": {
    "build": "webpack"
   }
}
```

3. loader

- loader 用于对模块的源代码进行转换
- loader 可以使你在 `import` 模块时预处理文件
- loader 可以将文件从不同的语言(如TypeScript)转换为 JavaScript

loaders\px2rem-loader.js

```
function loader(source){
    console.log('px2rem-loader');
    return source;
}
module.exports = loader;
```

4. 使用自定义loader

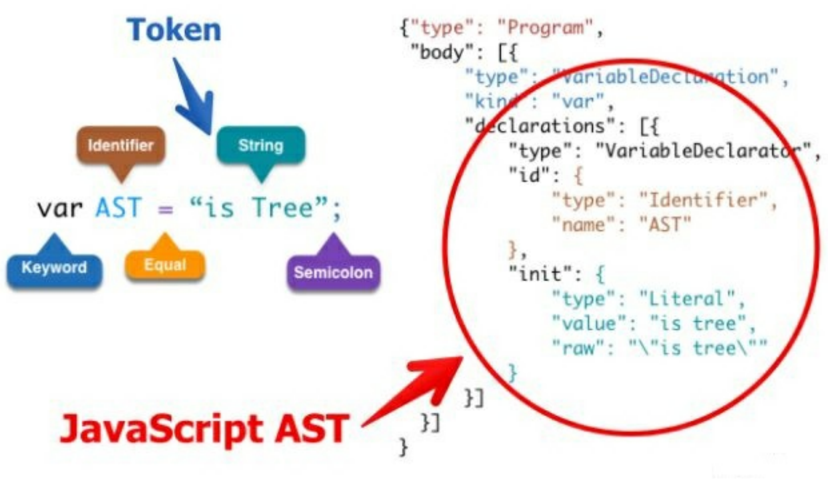
webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
    mode: 'development',
    devtool: false,
    entry: './src/index.js',
    output: {
        path: path.resolve(__dirname, 'dist'),
        filename: '[name].js'
    },
+   resolveLoader: {
+       alias: {
+           "px2rem-loader": path.resolve('./loaders/px2rem-loader.js')
+       },
+       modules: [path.resolve('./loaders'), 'node_modules']
+   },
    module: {
        rules: [{
            test: /\.css$/,
            use: [
                { loader: 'style-loader' },
                { loader: 'css-loader' },
                {
+                   loader: path.resolve(__dirname, 'loaders/px2rem-loader.js'),
                    options: {
                        remUni: 75,
                        remPrecision: 8
                    }
                }]
        }]
    },
    plugins: [
        new HtmlWebpackPlugin({ template: './src/index.html' })
    ]
};
```

5 css

5.1 AST

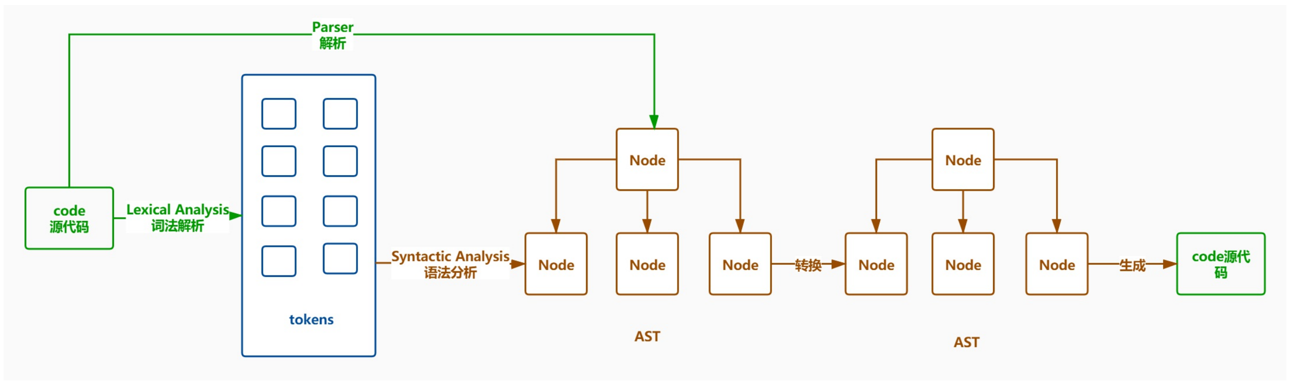
- astexplorer (https://astexplorer.net)
- JavaScript Parser可以把源代码转化为一颗抽象语法树（AST），这颗树定义了代码的结构



```
{"type": "Program",
 "body": [{
    "type": "VariableDeclaration",
    "kind": "var",
    "declarations": [{
       "type": "VariableDeclarator",
       "id": {
          "type": "Identifier",
          "name": "AST"
       },
       "init": {
          "type": "Literal",
          "value": "is tree",
          "raw": "\"is tree\""
       }
    }]
 }]
}
```

5.2 AST工作流

- Parse(解析) 将源代码转换成抽象语法树，树上有很多的estree节点
- Transform(转换) 对抽象语法树进行转换
- Generate(代码生成) 将上一步经过转换过的抽象语法树生成新的代码



5.3 px2rem.js

- px2rem (https://www.npmjs.com/package/px2rem)

px2rem.js

```
var css = require('css');
var pxRegExp = /\b(\d+(\.\d+)?)px\b/;
var pxGlobalRegExp = new RegExp(pxRegExp.source, 'g');
class Px2rem {
    constructor(config) {
        this.config = config;
    }
    generateRem(cssText) {
        let self = this;
        function processRules(rules) {
            for (var i = 0; i < rules.length; i++) {
                var rule = rules[i];
                var declarations = rule.declarations;
                for (var j = 0; j < declarations.length; j++) {
                    var declaration = declarations[j];
                    if (declaration.type === 'declaration' && pxRegExp.test(declaration.value)) {
                        declaration.value = self._getCalcValue('rem', declaration.value);
                    }
                }
            }
        }
        var astObj = css.parse(cssText);

        processRules(astObj.stylesheet.rules);
        return css.stringify(astObj);
    }
    _getCalcValue(type, value) {
        var { remUnit, remPrecision } = this.config;
        return value.replace(pxGlobalRegExp, (_, $1) => {
            let val = parseFloat($1) / remUnit.toFixed(remPrecision);
            return val + type;
        });
    }
}
module.exports = Px2rem;
```

5.4 usePx2rem.js

usePx2rem.js

```
let Px2rem = require('./px2rem');
let px2rem = new Px2rem({
    remUnit: 75,
    remPrecision: 8
});
let cssText = `
#root{
    width:750px;
    height:750px;
}
`;
let newCSS = px2rem.generateRem(cssText);
console.log(newCSS);
```

6. px2rem-loader.js

- loader-utils (https://www.npmjs.com/package/loader-utils)是一个webpack工具类
- px2rem-loader (https://www.npmjs.com/package/px2rem-loader)
- 直接写px，编译后会直接转化成rem
    - 在px后面添加/no/，不会转化px，会原样输出 一般border需用这个
    - 在px后面添加/px/,会根据dpr的不同，生成三套代码 一般字体需用这个

loaders\px2rem-loader.js

```
var loaderUtils = require('loader-utils');
var Px2rem = require('./px2rem');
function loader(source) {
  var options = loaderUtils.getOptions(this);
  var px2remIns = new Px2rem(options);
  let targetSource = px2remIns.generateRem(source);
  return targetSource;
}
module.exports = loader;
```

7. lib-flexible

src\index.js

```
import './base.css';
import 'amfe-flexible/index.js';
```

```
<script>
     (function flexible (window, document) {
         var docEl = document.documentElement;

         function setRemUnit () {
             var rem = docEl.clientWidth / 10;
             docEl.style.fontSize = rem + 'px';
         }
         setRemUnit();
         window.addEventListener('resize', setRemUnit);
         }(window, document))
 script>
```

7. 第三方框架样式问题

- 如果第三方组件已经是为移动端做了适配, px2rem又转成了 rem就导致其样式变的很小

7.1 index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'antd/dist/antd.css';
import {Button} from 'antd';
ReactDOM.render(<div>
    <Button type="primary">按钮Button>
div>, document.getElementById('root'));
```

7.2 webpack.config.js

webpack.config.js

```
 {
            test: /\.css$/,
            use: [
                { loader: 'style-loader' },
                { loader: 'css-loader' },
                {
                    loader: path.resolve(__dirname, 'loaders/px2rem-loader.js'),
                    options: {
                        remUnit: 75,
                        remPrecision: 8,
+                       exclude:/antd\.css/
                    }
                }],
```

7.3 px2rem-loader.js

loaders\px2rem-loader.js

```
var loaderUtils = require('loader-utils');
var Px2rem = require('./px2rem');

module.exports = function (source) {
  var options = loaderUtils.getOptions(this);
+ if(options.exclude && options.exclude.test(this.resource)){
+   return source;
+ }
  var px2remIns = new Px2rem(options);
  let targetSource = px2remIns.generateRem(source);
  return targetSource;
}
```