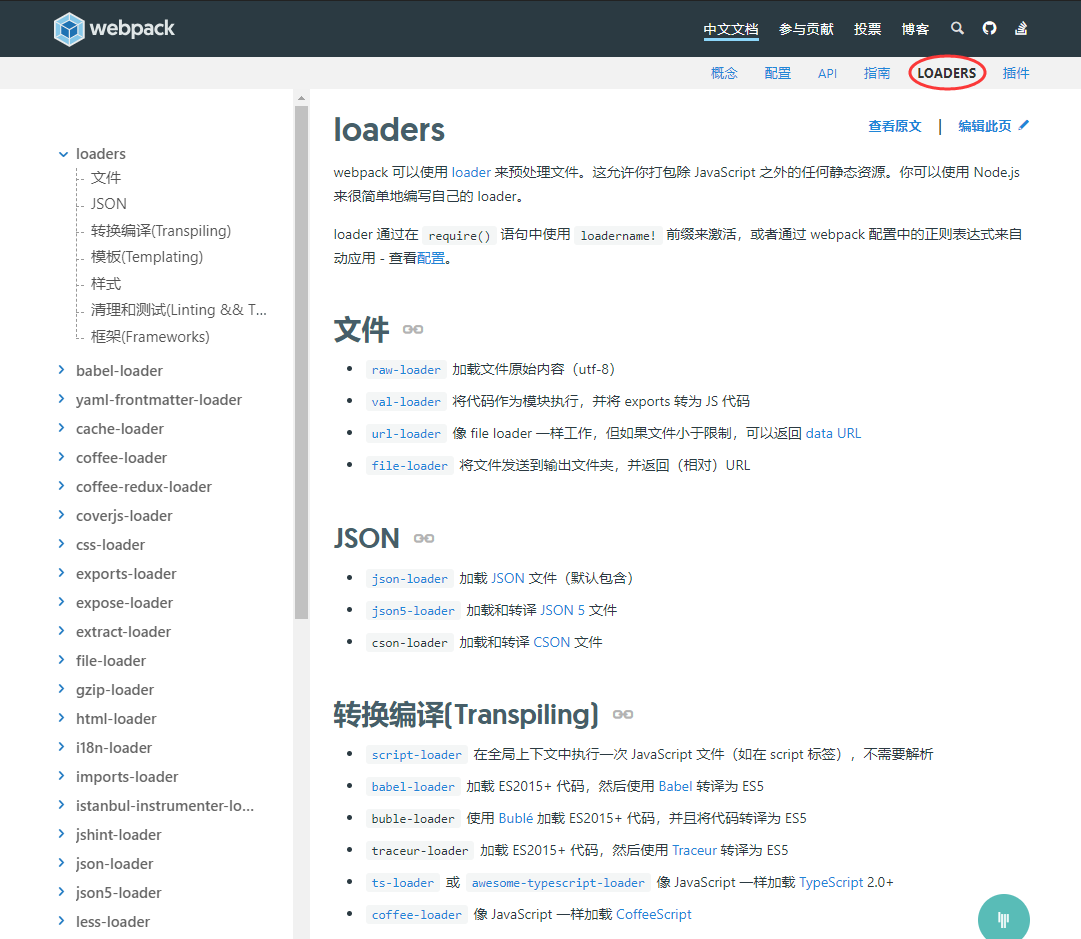
当我们打包一些非js模块的时候，比如jpg，png我们就需要去调查一下打包此类文件需要什么loader。

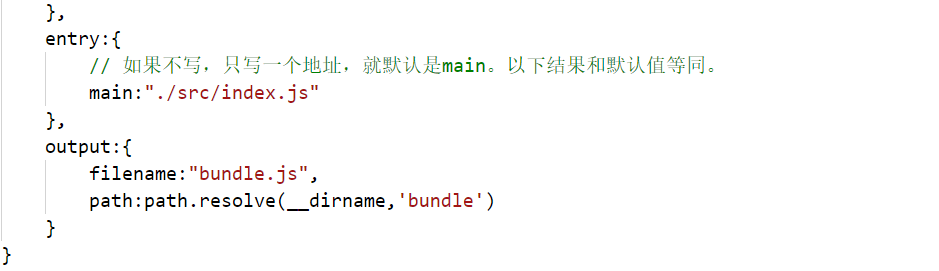
查阅方式为，进入官方文档，



每种Loader的配置都有很多说道，想要完全掌握是不可能的，网页左边就是官方推荐的一群loader，点开就可以查看详细配置说明。

练习时用的demo：





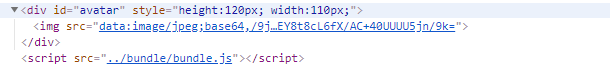
值得注意的是，file-loader会打包图片，打包后返回的是图片的名称（打包后的图片就会‘复制’一份到生产环境下，只不过文件可能是经过webpack处理过的，而且名字也是通过webpack配置文件中配置的，事实上这个经过打包过后，出现在生产环境下的图片和原来的图片就是不同的图片）。这也就意味着，如果是通过src + 图片的名称的方式引用图片，会产生一次http请求。如果图片较多，请求的次数也一定会变多，因此会造成性能下降的问题。

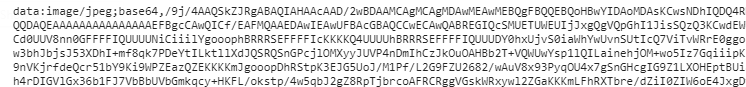
因此我们可以尝试用url-loader来代替file-loader。 因为url-loader不仅仅可以实现返回图片名称（和file-loader效果一模一样），而且还能返回base64代码。如果能使用base64代码替代图片索引式的src就能减少一次http请求。

不过依旧存在缺点，如果图片较大，生成的base64就会非常的长，导致js文件过大，加载缓慢。效果还不如直接索引，去进行http请求。

厉害的是我们可以通过限制图片大小，让url-loader智能判断到底应该是返回图片名称还是base64代码。

返回base64的样子





返回图片名称的样子



Sass插值 #{}，虽然可以插值，但是只可以作为属性或者选择器来进行插值，而不能企图通过插值拼装成某个变量或者混合宏。但是可以在@entend继承中拼接一个类。

// 拼接成属性

$properties:(margin,padding);

@mixin set-value($side,$value){

@each $prop in $properties{

#{$side}-#{$value}:$value;

}

}

.login-box{

@include set-value(top,14px);

}

// 拼接成选择器

@mixin generate-sizes($class, $small, $medium, $big) {

.#{$class}-small { font-size: $small; }

.#{$class}-medium { font-size: $medium; }

.#{$class}-big { font-size: $big; }

}

@include generate-sizes("header-text", 12px, 20px, 40px);

// 不能用于拼接变量或者宏

$margin-big: 40px;

$margin-medium: 20px;

$margin-small: 12px;

@mixin set-value($size) {

margin-top: $margin-#{$size};

}

.login-box {

@include set-value(big);

}

或者

@mixin updated-status {

margin-top: 20px;

background: #F00;

}

$flag: "status";

.navigation {

@include updated-#{$flag};

}

// 但是可以再继承@extend中通过插值拼接成一个继承类

说白了就是只能拼接选择器啊、占位符的类啊、属性啊之类的，不能替换变量和宏。

%updated-status {

margin-top: 20px;

background: #F00;

}

.selected-status {

font-weight: bold;

}

$flag: "status";

.navigation {

@extend %updated-#{$flag};

@extend .selected-#{$flag};

}