자바스크립트 코드 축소

코드의 크기를 줄이는 것을 압축(compress) 또는 축소(minification)라고 한다. 또는 최적화(Optimize)라고 불리기도 한다. 그리고 코드를 못 읽게 만드는 것을 조금 애매하지만 암호화(obfuscate)라고 한다. 코드 압축 기능을 하는 것들을 축소기(minifier)라고 한다. 축소기는 코드의 기능을 유지하면서 필요 없는 부분인 주석, 빈칸, 필요 없는 괄호 등을 제거해서 크기를 줄인다. 압축기(compressor)는 축소기를 조금 더 확장한 개념인데, 지역 변수명을 짧게 바꾼다든지, 인코딩을 바꾼다든지 해서 코드의 크기를 조금 더 줄이는 것을 말한다.

구글느님에게 JavaScript compressor 또는 minifier로 검색을 하면 굉장히 많은 결과가 나온다. 유명한 자바스크립트 압축기(compressor)는 YUI compressor, Google Closure Compiler, *packer* 가 있다. 이외에 JSMin, 이제는 없지만 Dojo에 있었던 ShrinkSafe 등이 있다. YUI Compressor나 Closure Compiler는 자바로 쓰여져 있어서 jar 파일로 배포된다. 그래서 실행 할 때 약간 번거롭다. 그래서 다시 검색을 해봤더니 UglifyJS가 눈에 들어 왔다. UglifyJS는 잠시 뒤에 소개하고 우선은 축소를 했으니 이제 반대로 축소한것을 복구 시키는 것을 알아보자.

코드 서식 정리

프로그래밍을 할 때 신경 쓰이는 부분 중에 하나는 들여쓰기, 한 줄의 길이, 중괄호의 위치와 같은 코드의 서식이다. 보통 code formatting, programming style이라고 불린다. 각 언어마다 프로그래밍 스타일에 대한 지침을 코딩 관습(Coding Convention)이라고 한다. 관습은 사회마다 다르듯이 이 코딩 관습도 여러가지다. 결국은 속해있는 곳을 따라서, 대세를 따르는 것이 정식건강에 여러모로 좋다. 이 코드의 서식을 규칙에 맞게 바꿔 주는 도구들이 있다. 자동으로 들여쓰기를 바꿔주는 것이 대표적인데, 훌륭한 편집기에서는 Formatting이라든지 Align이라는 형태로 나타난다.

UglifyJS

코드를 압축하면 보기 싫기 때문에 이름이 UglifyJS인지는 모르지만 UglifyJS는 자바스크립트 코드의 파서, 축소, 압축, 코드 서식 정리를 하는 툴킷이다. 두가지 버전이 있다. UglifyJS와 UglifyJS v2다. 뭐가 다른지는 홈페이지에 가서 확인해 보자. 그냥 UglifyJS v2를 쓰도록 하자. UglifyJS도 나름 유명하다. jQuery, jQuery UI, jQuery Mobile 코드 압축할 때 UglifyJS가 사용된다고 한다.

1. UglifyJS 설치

UglifyJS2 에 가면 자세히 나와 있다. Node.js와 npm를 먼저 설치해야 한다. 그리면 간단하게 npm으로 UglifyJS2를 설치할 수 있다.

```
npm install -g uglify-js
```

기본 사용법

주로 사용할 압축하는 법과 코드 정리하는 것을 알아보자.

코드 축소

`**uglifyjs 입력파일경로 -c -o 결과파일경로**` 입력파일경로에는 하나 이상의 파일을 쓸 수 있고, 표준입력(STDIN)으로 대체하려면 파일 경로 대신에 -를 사용한다. -c(--compress)는 압축한다는 의미다. -c 플래그 뒤에 압축 옵션을 지정할 수 있다. 이 옵션은 아래에 나온다.

-o 플래그 뒤에 결과 파일의 경로를 입력하는데, -o를 사용하지 않으면 결과는 표준출력(STDOUT)으로 나온다.

조금 더 파일 크기를 줄이려면 변수 이름을 줄이는 -m(––mangle 변수명을 바꾼다는 의미) 플래그를 추가로 사용할 수 있다. -m 옵션을 사용하면 -r(––reserved) 플래그를 사용해서 맹글링하지 않을 이름을 지정할 수 있다. 예를들어 `-m -r '$,require,exports'` 를 사용하면 $, require, exports는 변환되지 않는다. [alert type='error']주의할 점은 mangler 옵션을 사용하면, 변수명은 원래로 돌아오지 않는다. 무슨 뜻이냐면 아래에서 beutify를 해도 변경된 변수명은 원래 변수명으로 복구되지 않는다는 말이다. 그러니 -m 플래그을 사용할 때는 눈을 크게 뜨고 있어야 한다. [/alert]

플래그 -c 뒤에 압축 옵션을 지정할 수 있는데 조금 많다. 아래 옵션은 기본값이 hoist_vars를 제외하고 전부 true다. 따라서 필요한 옵션만 false로 지정해 주면 된다. 여러 옵션을 전달할 때에는 쉼표(,)로 분리해서 전달한다. 예를 들어 sequence=false,join_vars=fals

옵션	설명
sequences	연속된 간단한 문장을 컴마(,)로 합친다.
properties	프로퍼티 접근을 닷(.) 표기로 바꾼다. foo[“bar”] → foo.bar
dead_code	닿지않는(unrechable) 코드를 제거한다. 이런 코드는 return, break, continue를 쓸 때 생길 수 있다.
drop_debugger	debugger; 문장을 제거한다. debugger는 본래 없는 문장이지만 자바스크립트 엔진 대부분이 지원하는 것 같다.
unsafe	말 그대로 ‘안전하지 않은’ 변환을 적용한다.예를들어 var a = new Array(1,2,3,4); → var a=[1,2,3,4];
conditionals	조건 표현을 최적화한다
comparisons	바이너리 노드에 최적화를 한다고 하는데, 조금 애매하다. 일단은 a<=b를 b>=a로 바꾸기를 한다.
evaluate	상수 표현을 평가한다. var a = 10+20+30; 이면 var = 70;으로 바꾼다.
booleans	!a ? foo : bar → a?bar:foo로 바꾼다.
loops	루프를 최적화 한다. while(true) 를 for(;;) 이런식으로 바꾼다.
unused	참조되지 않는 변수와 함수를 제거한다. 당연하겠지만 전역(global) 변수와 함수는 제외한다.
hoist_funs	함수 선언을 끌어 올린다(hoist). 본래 함수 선언은 Hoist 되는데, 소스 자체에서 함수 선언을 위로 올려준다는 것이다. 예를 들어 foo 함수 선언을 소스 코드 맨 아래에 해둬도 이 옵션 값이 true면 결과에 foo 함수 선언을 코드의 앞으로 이동시킨다는 뜻이다
hoist_vars	위 hoist_funs 처럼 변수를 끌어 올린다. 기본값은 false
if_return	if/return과 if/continue를 최적화한다.
join_vars	연속된 var 문장을 합친다. var a=10; var b=20; 이면 var a=10,b=20;로 바꾼다.
cascade	x=foo(),x를 x=foo()로 바꾼다.
warning	위에서 dead_code, unused를 사용할 때 제거되는 것에 대한 경고를 표시한다. 이 메시지는 표준에러(STDERR)로 나온다.

코드 정리

`**uglifyjs 입력파일경로 -b -o 결과파일경로**` 위 코드 축소에서 -c 대신 -b 플래그를 사용한것 빼곤 같다. 마찬가지로 플래그 -b 뒤에도 -c 처럼 옵션을 지정할 수 있다. 각 옵션을 쉼표(,)로 분리해서 전달한다.

옵션	기본값	설명
beautify	true	최소화만 할 경우에 false로 사용한다.
indent-level	4	들여쓰기를 얼마나 할 것인지를 지정한다.
indent-start	0	들여쓰기 전에 빈칸을 얼마나 넣을 것인지 지정한다.
quote-keys	false	키를 인용부호(“,”)로 감싼다.
space-colon	true	콜론(:)뒤에 빈칸을 넣는다.
ascii-only	false	문자열과 정규식에서 유니코드 문자를 eacape한다.예를들어 문자열 “가”를 “\uac00"으로 바꾼다.
inline-script	false	문자열의 </script 에서 슬러시(/)를 이스케이프한다.
width	80	한 행의 길이를 지정한다. 잘 안 된다.
max-line-len	32000	행의 길이의 최대값
ie-proof	true	IE-proof 코드를 생성한다.
bracketize	false	if, for, do, while, with의 문장에 bracket({,})을 항상 넣는다.
semicolons	true	세미콜론(;)으로 문장을 구분한다.

사용예

```
$ type hello.js
function hello(name){
  console.log('Hi, '+name)
}
hello("NodeJS");

$ uglifyjs Hello.js
function hello(name){console.log("Hi, "+name)}hello("NodeJS");

$ uglifyjs Hello.js -m
function hello(o){console.log("Hi, "+o)}hello("NodeJS");

$ uglifyjs Hello.js -m -o Hello.min.js

$ dir Hello*.*
2018-05-08  오후 05:01                70 Hello.js
```

```
2018-05-08  오후 05:15                56 Hello.min.js
```