





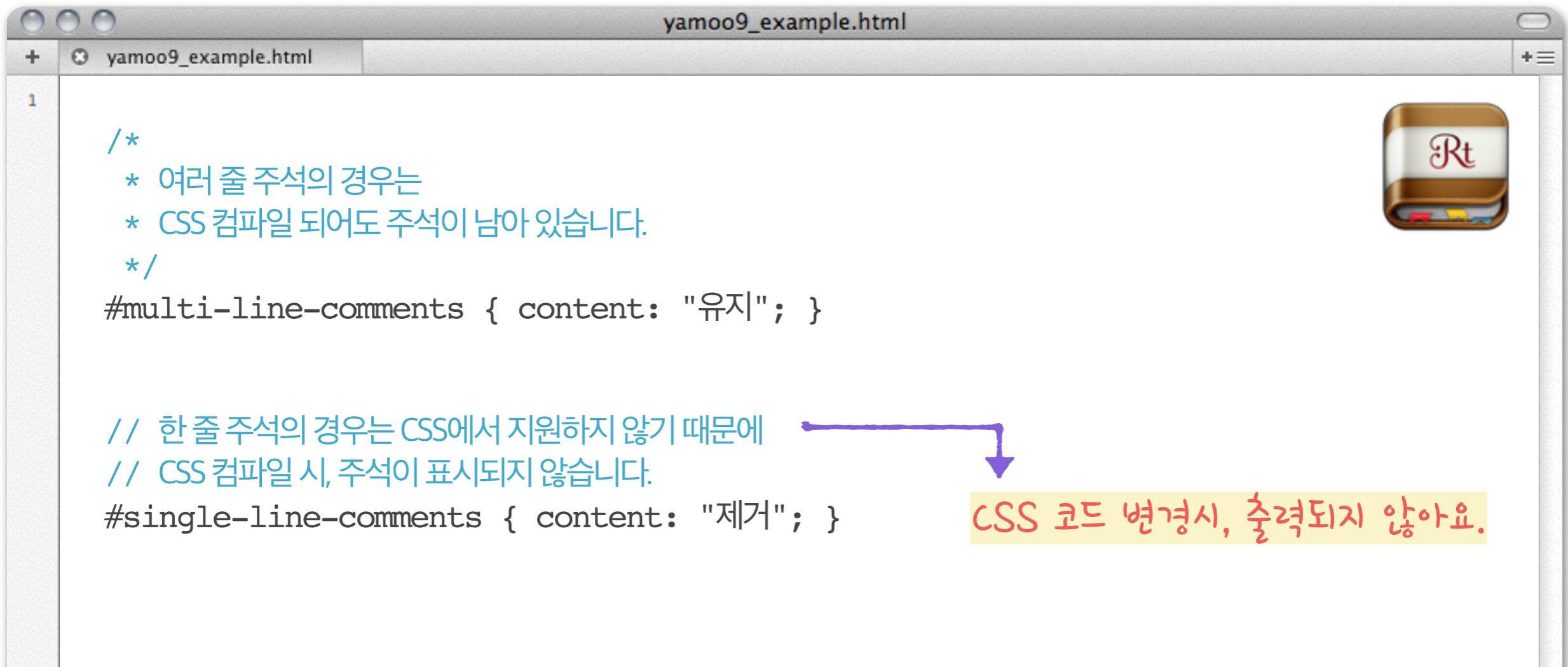
# Comments

`/**/, //`



## 주석(Comments) //, /\*\*/

CSS 멀티라인 주석 뿐만 아니라, JS 싱글라인 주석 역시 SASS에서 사용할 수 있습니다.



The screenshot shows a code editor window titled "yamoo9\_example.html". The code contains two examples of comments:

```
1
/*
 * 여러 줄 주석의 경우는
 * CSS 컴파일 되어도 주석이 남아 있습니다.
 */
#multi-line-comments { content: "유지"; }

// 한 줄 주석의 경우는 CSS에서 지원하지 않기 때문에
// CSS 컴파일 시, 주석이 표시되지 않습니다.
#single-line-comments { content: "제거"; }
```

A purple arrow points from the text "CSS 코드 변경시, 출력되지 않아요." to the single-line comment block at the bottom.

**CSS 코드 변경시, 출력되지 않아요.**



# 주석(Comments) //, /\*\*/

\*.sass 코드에서는 CSS 주석의 시작부분에서 Tab 한 후, 주석을 입력합니다.

```
// http://sass-
lang.com/documentation/file.SASS_REFERENCE.htm
l#comments

// SASS 한 줄 주석
// CSS 코드로 변경될 때, 해석되지 않습니다.

/*
 * CSS
 * 여러줄
 * 주석
 * CSS 코드로 변경될 때, 해석됩니다.

body
  font: 13px/1.5 "Nanum Gothic", Dotum, Sans-
  Serif

/*
 * Nested Rule: 중첩 규칙
```

```
// http://sass-
lang.com/documentation/file.SASS_REFERENCE.htm
l#comments

// SASS 한 줄 주석
// CSS 코드로 변경될 때, 해석되지 않습니다.

/*
 * CSS
 * 여러줄
 * 주석
 * CSS 코드로 변경될 때, 해석됩니다.

body {
  font: 13px/1.5 "Nanum Gothic", Dotum,
  Sans-Serief;
}

/*
 * Nested Rule: 중첩 규칙
```

SASS



SCSS



# 주석(Comments) 스니펫 만들기

Sublime Text에서 활용 가능한 주석 스니펫을 만들어 볼까요?,

```
1      /*
 * .SCSS, .sass, .css 파일에서 활용
 * 여러줄 주석 스니펫
 */

<snippet>
  <content><! [CDATA[/*
 * $1
 */
$2]]></content>
  <tabTrigger>cmt</tabTrigger>
  <scope>source.sass, source.scss, source.css</scope>
  <description>CSS 여러줄 주석</description>
</snippet>
```

```
comments.sass
```

```
1 // http://sass-lang.com/documentation/file.  
2 SASS_REFERENCE.html#comments  
3 // SASS 한 줄 주석  
4 // CSS 코드로 변경될 때, 해석되지 않습니다.  
5  
6 /*  
7 * CSS, SASS 여러 줄 주석  
8 * CSS 코드로 변경될 때, 해석됩니다.  
9 */
```

```
comments.css
```

```
1 /* CSS, SASS 여러 줄 주석  
2 * CSS 코드로 변경될 때, 해석됩니다.  
3 */  
4  
5  
6  
7  
8  
9  
10  
11
```



# Nesting Parent Selector

&



# 중첩 규칙(Nested Rules)

부모 선택자 **내부에** 자식 선택자를 포함하는 구조 형태로 작성이 가능합니다.

```
yahoo9_example.html
```

```
#page {   ..... ➤ #page
  content: "페이지";
}

.container {   ..... ➤ #page .container
  content: "콘테이너";
}

.section {   ..... ➤ #page .container .section
  content: "섹션"; }

} // .container

} // #page
```

The code illustrates nested CSS rules. The first rule, '#page { ... }', is a parent selector. Inside it, the 'content' property is set to '페이지'. The second rule, '.container { ... }', is a child selector nested within the '#page' block. Inside it, the 'content' property is set to '콘테이너'. The third rule, '.section { ... }', is another child selector nested within the '.container' block. Inside it, the 'content' property is set to '섹션'. Purple arrows point from the right side of each selector to the corresponding nested rule, and red arrows point from the right side of each nested rule to its parent selector. The final two lines of code close the '#page' block and the entire '#page' block.



## 중첩 규칙(Nested Rules)

CSS 선택자 코드 작성은 효율성이 떨어지는 반면 SASS 중첩규칙은 매우 효율성이 높습니다.

```
1
nav {float: right;}
nav li {float: left;}
nav li a {color: #666;}
nav li a:hover {color: #333;}
nav li.current {font-weight: bold;}
```

VS

```
nav {
  float: right;
}
li {
  float: left;
}
a {
  color: #666;
}
&:hover {
  color: #333; }
}
&.current {
  font-weight: bold;
}}
```



yamoo9\_example.html

```
1  div.container {  
2      div.content {  
3          div.articles {  
4              & > div.post {  
5                  div.title {  
6                      h1 {  
7                          a {  
8                          }  
9                      }  
10                     }  
11                 }  
12             }  
13             div.content {  
14                 ul {  
15                     li { ... }  
16                 }  
17             }  
18         }  
19     }  
20 }
```

**Bad Nesting!**

중첩 코드는 적당한게 좋아요.  
너무 많은 중첩은 안 좋습니다. 왜냐하면...

SCSS



yahoo9\_example.html

```
1 div.content div.container { ... }
div.content div.container div.articles { ... }
div.content div.container div.articles > div.post { ... }
div.content div.container div.articles > div.post div.title { ... }
div.content div.container div.articles > div.post div.title h1 { ... }
div.content div.container div.articles > div.post div.title h1 a { ... }
div.content div.container div.articles > div.post div.content { ... }
div.content div.container div.articles > div.post div.content ul { ... }
div.content div.container div.articles > div.post div.content ul li { ... }
```

적절하지 않은 중첩 코드는  
쓸데없는 CSS 선택자를 남용하여 생성하기 때문입니다.

**Oops!**

CSS



## 부모, 참조 선택자 (&)

앤퍼센트(&) 심볼은 부모를 참조하는 선택자 역할을 수행합니다.

yahoo9\_example.html

```
1 p {  
  font-weight: bold;  
  text-decoration: none;  
  
  &:hover a { text-decoration: underline; }  
  #page.firefox &.showcase a { font-weight: normal; }  
}  
  
p:hover a  
#page.firefox p.showcase a
```

The code editor window shows a snippet of CSS. A red arrow points from the '&' symbol in the selector `&:hover` to the word 'parent' in the explanatory text above. A purple box highlights the `&` symbol, and a purple arrow points from this box to the resulting selector `#page.firefox p.showcase a` at the bottom right. Another purple arrow points from the closing brace `}` back up to the `&` symbol. The file tab in the window is labeled 'yahoo9\_example.html'.



yahoo9\_example.html

```
1
a
  position: relative
  color: #495054

  &:hover
    color: #5f666d

  → &.external-link,
  &[target=_blank]:before
    content: ''
    background: url(external_link.gif) no-repeat
    position: absolute
    right: 100%
    width: 10px
    height: 10px

  → .article & .bold
    font-weight: bold
```

& 뒤에 '공간'이 있을 때와 없을 때,  
& 앞에 선택자가 있을 경우 유의하여 보세요!



yahoo9\_example.html

```
1
a {
  position: relative;
  color: #495054;
  &:hover {
    color: #5f666d;
  }
  &.external-link,
  &[target=_blank]:before {
    content: "";
    background: url(external_link.gif) no-repeat;
    position: absolute;
    right: 100%;
    width: 10px;
    height: 10px;
  }
  .article & .bold {
    font-weight: bold;
  }
}
```

→ & 뒤에 '공간'이 있을 때와 없을 때,  
→ 앞에 선택자가 있을 경우 유의하여 보세요!





yahoo9\_example.html

```
1
a {
  position: relative;
  color: #495054;
}
a:hover {
  color: #5f666d;
}
→ a.external-link, a[target=_blank]:before {
  content: "";
  background: url(external_link.gif) no-repeat;
  position: absolute;
  right: 100%;
  width: 10px;
  height: 10px;
}
→ .article a .bold {
  font-weight: bold;
}
```

& 뒤에 '공간'이 있을 때와 없을 때,  
& 앞에 선택자가 있을 경우 유의하여 보세요!

→

→

CSS



# 중첩 부모 참조 선택자(&) 스니펫 만들기

Sublime Text에서 활용 가능한 중첩 부모 참조 선택자 스니펫을 만들어 볼까요?,



A screenshot of the Sublime Text editor showing a snippet definition for SASS nesting. The file is named "yamoo9\_example.html". The snippet content is as follows:

```
1 <snippet>
  <content><! [CDATA[&$1 {
    $2
  }
$0]]></content>
  <tabTrigger>nested-parent</tabTrigger>
  <scope>source.scss, source.sass</scope>
  <description>SASS - 중첩 규칙, 부모 참조</description>
</snippet>
```

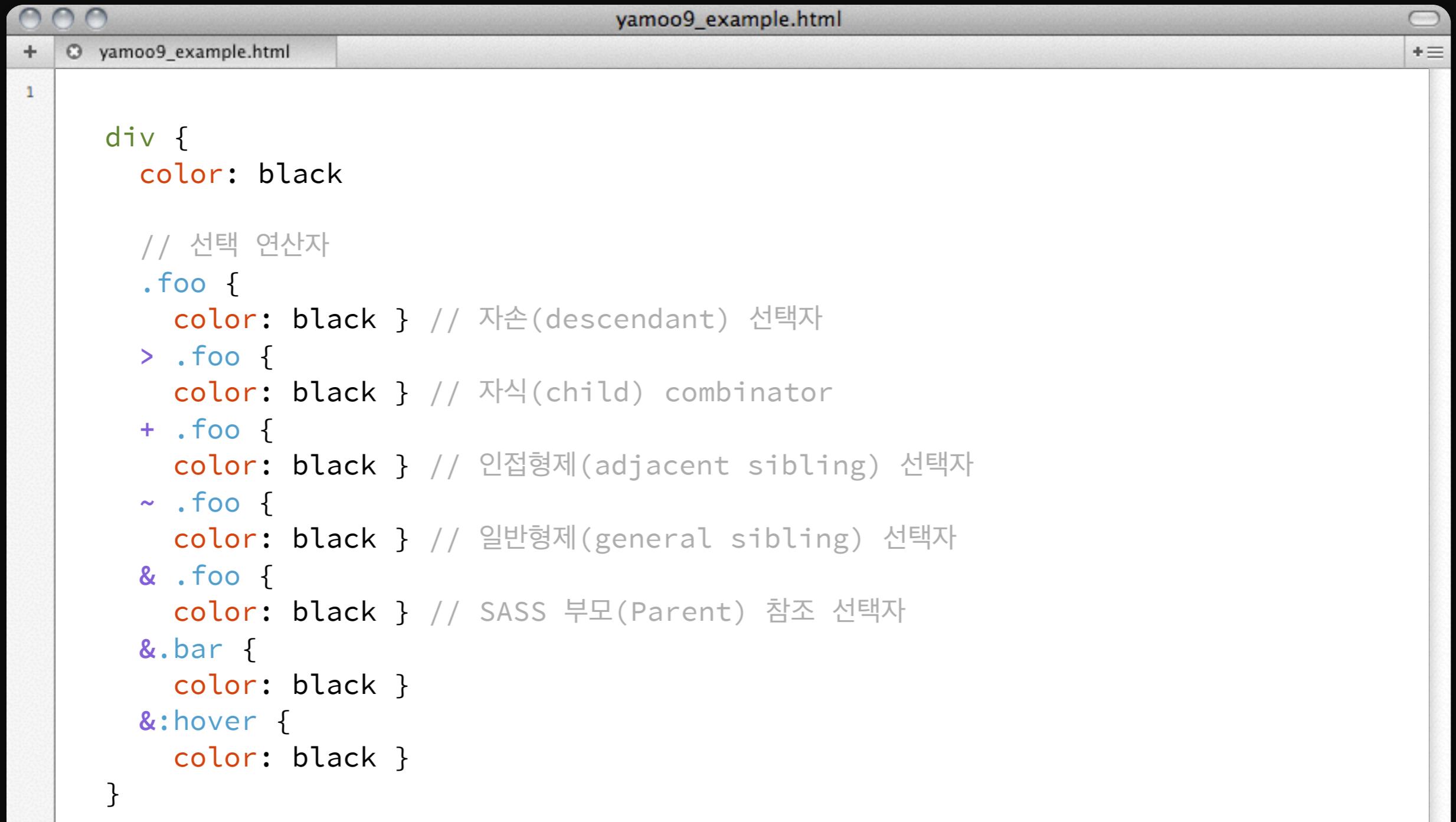


# 중첩 부모 참조 선택자(&) 스니펫 만들기

Sublime Text에서 활용 가능한 중첩 부모 참조 선택자 스니펫을 만들어 볼까요?,

```
1 <snippet>
  <content><! [CDATA[ ${1:상위 참조 선택자} &$2 {
    $3
  }
$0]]></content>
<tabTrigger>nested-parent-high</tabTrigger>
<scope>source.scss, source.sass</scope>
<description>SASS - 중첩 규칙, 상위 참조 선택자</description>
</snippet>
```

# Combining Selectors



The screenshot shows a web browser window with the title bar "yamoo9\_example.html". The main content area displays the following CSS code:

```
1
div {
  color: black

  // 선택 연산자
  .foo {
    color: black } // 자손(descendant) 선택자
  > .foo {
    color: black } // 자식(child) combinator
  + .foo {
    color: black } // 인접형제(adjacent sibling) 선택자
  ~ .foo {
    color: black } // 일반형제(general sibling) 선택자
  & .foo {
    color: black } // SASS 부모(Parent) 참조 선택자
  &.bar {
    color: black }
  &:hover {
    color: black }

}
```

The code illustrates various CSS selector combinators:

- `div`: Basic element selector.
- `.foo`: Class selector.
- `> .foo`: Descendant combinator (selects `.foo` elements that are immediate children of the current element).
- `+ .foo`: Adjacent sibling combinator (selects `.foo` elements that immediately precede the current element).
- `~ .foo`: General sibling combinator (selects `.foo` elements that share the same parent as the current element).
- `& .foo`: Parent reference combinator (selects `.foo` elements that are descendants of the current element's parent).
- `&.bar`: Parent reference class combinator (selects `.bar` elements that are descendants of the current element's parent).
- `&:hover`: Parent reference pseudo-class combinator (selects the current element when it is a descendant of an element that is being hovered over).



# 중첩 속성(Nested Properties)

세분화된 속성 여러 개를 설정할 경우, 네임스페이스(Namespace)를 사용하여 중첩 속성을 설정할 수 있습니다.

```
1  .yamoo9 {           → '내부공간' 네임스페이스
    padding: [
      top: 10px;
      bottom: 5px;
    ]
    font: [
      family: fantasy;
      size: 30em;
      weight: bold;
    ]
}
```

‘글꼴’ 네임스페이스

CSS 속기형 속성  
font, margin, border, padding, background,  
transition, animation, box, column, list, min,  
max, outline, text ⋯.



# 중첩 속성 스니펫 만들기

Sublime Text에서 활용 가능한 중첩 속성 스니펫을 만들어 볼까요?,

```
yamoo9_example.html
```

```
+ ① yahoo9_example.html +≡  
1 <snippet>  
  <content><! [CDATA[ ${1:margin,  
animation, box, border, background, column, font, list, max, min,  
outline, padding, text, transition}: {  
  $2  
}  
$0 ]></content>  
  <tabTrigger>nested-properties</tabTrigger>  
  <scope>source.scss, source.sass</scope>  
  <description>SASS - 중첩 속성 정의</description>  
</snippet>
```



# RWD, 부모 참조 선택자와 미디어쿼리

선언 구간 내부의 미디어쿼리는 외부로 분리됩니다.

Yahoo! UI Library screenshot showing the separation of media queries from their parent selector declarations.

The screenshot shows the code editor interface for "yamoo9\_example.html". The code is as follows:

```
1 aside {  
  width: 25%;  
  @media screen and (max-width: 480px) {  
    width: 100%;  
  }  
}  
  
aside {  
  width: 25%;  
}  
  
@media screen and (max-width: 480px) {  
  aside {  
    width: 100%;  
  }  
}
```

A purple box highlights the media query block: `@media screen and (max-width: 480px) { width: 100%; }`. A yellow box highlights the text "CSS 미디어쿼리". A purple bracket groups the entire media query block. A purple arrow points from the bottom of this bracket to the equivalent media query declaration in the bottom half of the code editor, which is enclosed in a gray box. A red arrow points to the word "aside" in the media query declaration, indicating its reference to the parent selector.

```
nesting-&.sass
```

```
nesting-&.css
```

```
1 /*  
2 * Nesting  
3 * 중첩 규칙을 사용하면 구조를 쉽게 파악하면서  
4 * 반복적인 선택자 코드 사용량을 대폭 줄일 수 있어  
5 * 매우 유용합니다.  
6 */  
7  
8 a  
9   position: relative  
10  color: #495054  
11  padding:  
12    top: 20px  
13    bottom: 10px  
14  
15 @media only screen and (max-width: 480px)  
16  color: red  
17  
18 &:hover  
19  color: #5f666d  
20  
21 &.external-link,  
22 &[target=_blank]:before  
23  content: ''  
24  background: url(external_link.gif) no-repeat  
25  position: absolute  
26  right: 100%  
27  width: 10px  
28  height: 10px  
29  
30 @media only screen and (max-width: 480px)  
31  position: static  
32  
33 & .bold  
34  font-weight: bold  
35
```

```
1 /* Nesting  
2 * 중첩 규칙을 사용하면 구조를 쉽게 파악하면서  
3 * 반복적인 선택자 코드 사용량을 대폭 줄일 수 있어  
4 * 매우 유용합니다.  
5 */  
6 a {  
7  position: relative;  
8  color: #495054;  
9  padding-top: 20px;  
10  padding-bottom: 10px;  
11 }  
12 @media only screen and (max-width: 480px) {  
13  a {  
14    color: red;  
15  }  
16 }  
17 a:hover {  
18  color: #5f666d;  
19 }  
20 a.external-link, a[target=_blank]:before {  
21  content: "";  
22  background: url(external_link.gif) no-repeat;  
23  position: absolute;  
24  right: 100%;  
25  width: 10px;  
26  height: 10px;  
27 }  
28 @media only screen and (max-width: 480px) {  
29  a.external-link, a[target=_blank]:before {  
30    position: static;  
31  }  
32 }  
33 a .bold {  
34  font-weight: bold;  
35 }
```



# Selector Inheritance

@extend, %



# 선택자 상속(Selector Inheritance)

@extend를 사용해 선언된 다른 규칙의 내용을 상속할 수 있습니다.

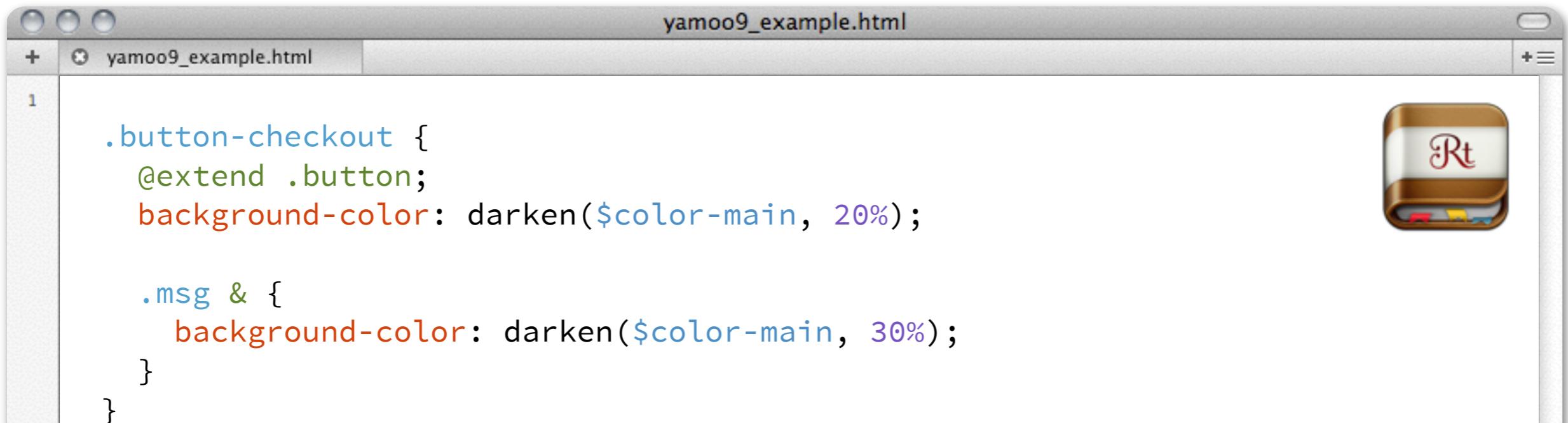
```
yahoo9_example.html
```

```
1 .button {  
background-color: $color-main;  
font-weight: bold;  
color: white;  
padding: 5px;    재사용 가능한 .button 클래스  
}  
  
.button-checkout {  
@extend .button; ← .button 클래스 상속  
background-color: darken($color-main, 20%);  
}
```



# 선택자 상속(Selector Inheritance)

@extend를 사용해 선언된 다른 규칙의 내용을 상속할 수 있습니다.



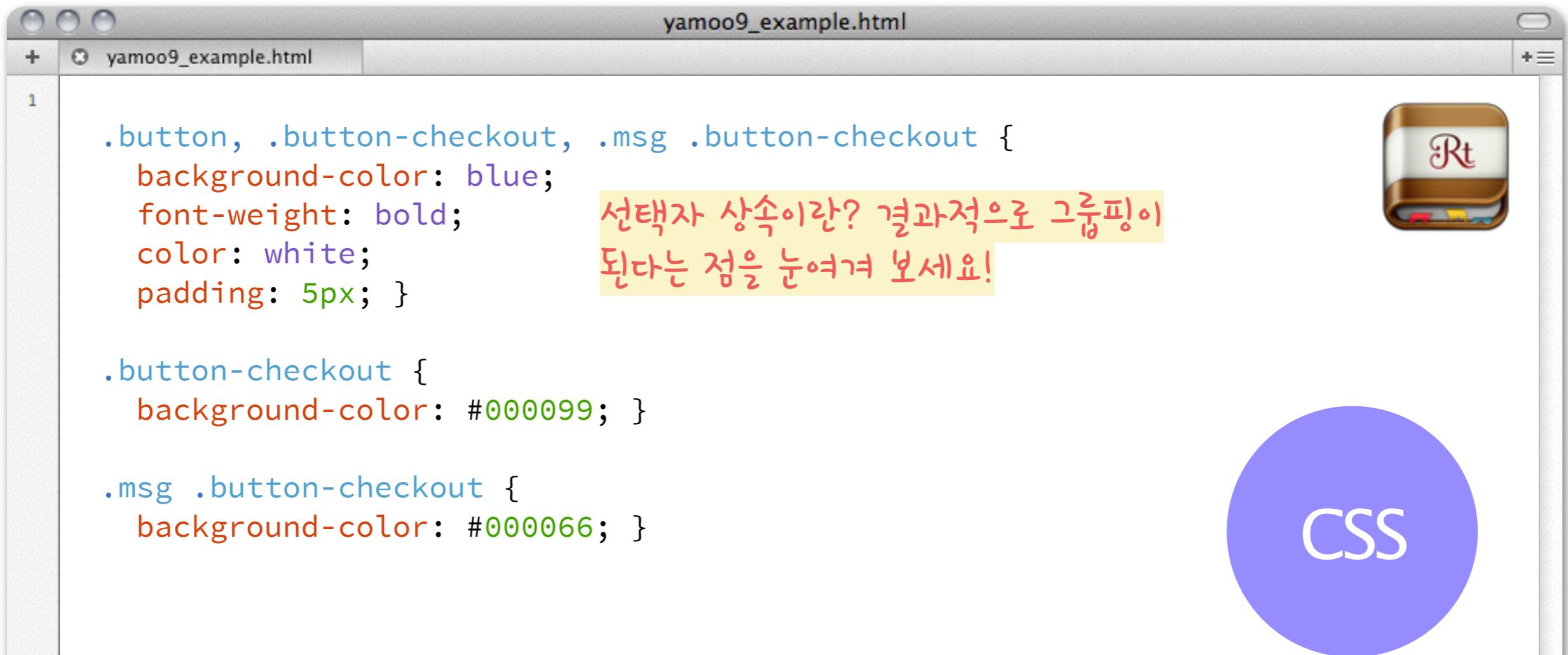
```
1 .button-checkout {
  @extend .button;
  background-color: darken($color-main, 20%);

  .msg & {
    background-color: darken($color-main, 30%);
  }
}
```



# 선택자 상속(Selector Inheritance)

@extend를 사용해 선언된 다른 규칙의 내용을 상속할 수 있습니다.



The screenshot shows a browser window titled "yamoo9\_example.html". The code editor contains the following CSS:

```
1 .button, .button-checkout, .msg .button-checkout {  
background-color: blue;  
font-weight: bold;  
color: white;  
padding: 5px; }  
  
.button-checkout {  
background-color: #000099; }  
  
.msg .button-checkout {  
background-color: #000066; }
```

A yellow callout box with red text points to the third rule: ".msg .button-checkout { background-color: #000066; }". The text reads: "선택자 상속이란? 결과적으로 그룹핑이 된다는 점을 눈여겨 보세요!" (What is selector inheritance? It's grouping that results from inheritance!).



```
Inheritance.sass
```

```
Inheritance.css
```

```
1 /*  
2 * Selector Inheritance  
3 * 재사용 가능한 클래스 선택자를 상속할 수 있습니다.  
4 * CSS 코드 변경 시, 그룹핑으로 처리됩니다.  
5 */  
6 // button 클래스  
7 .button  
8 margin: 0 auto 25px;  
9 padding: 15px;  
10 border-radius: 6px;  
11 em  
12 text-decoration: underline;  
13 // 실패 버튼  
14 .button-fail  
15 @extend .button;  
16 background-color: #fac9c7;  
17 border-color: #ef625a;  
18 // 성공 버튼  
19 .button-success  
20 @extend .button;  
21 background-color: #1be4b7;  
22 border-color: #bef8eb;  
23 // 비활성화 버튼  
24 .button-disabled  
25 cursor: default;  
26 @extend .button;  
27 background-color: #c9c9c9;  
28 border-color: #6c6c6c;  
29 .msg &  
30 background-color: darken(#fac9c7, 30%)
```

```
1 /* Selector Inheritance  
2 * 재사용 가능한 클래스 선택자를 상속할 수 있습니다.  
3 * CSS 코드 변경 시, 그룹핑으로 처리됩니다.  
4 */  
5 .button, .button-fail, .button-success, .button-disabled {  
6 margin: 0 auto 25px;  
7 padding: 15px;  
8 border-radius: 6px;  
9 }  
10 .button em, .button-fail em, .button-success em, .button-disabled em {  
11 text-decoration: underline;  
12 }  
13 .button-fail {  
14 background-color: #fac9c7;  
15 border-color: #ef625a;  
16 }  
17 .button-success {  
18 background-color: #1be4b7;  
19 border-color: #bef8eb;  
20 }  
21 .button-disabled {  
22 cursor: default;  
23 background-color: #c9c9c9;  
24 border-color: #6c6c6c;  
25 }  
26 .msg .button-disabled {  
27 background-color: #d58508;  
28 }
```



# 익스텐드(@extend) 스니펫 만들기

Sublime Text에서 활용 가능한 익스텐드 스니펫을 만들어 볼까요?,

```
yahoo9_example.html
```

```
+ ① yahoo9_example.html +≡  
1 <snippet>  
  <content><! [CDATA[@extend ${1:상속할 선택자 이름}]]></content>  
  <tabTrigger>/</tabTrigger>  
  <scope>source.scss, source.sass</scope>  
  <description>SASS - @extend 호출</description>  
</snippet>  
  
'/' 단축 키워드 말고 직접 'extend'로  
등록하셔도 좋겠죠? ^ - ^
```



## 선택자 상속, 옵션(!optional) 설정

@extend 오류 발생시, 오류가 발생되지 않도록 옵션 설정을 할 수 있습니다.

```
.button-checkout {
  @extend .button !optional; ←
  background-color: darken($color-main, 20%);

  .msg & {
    background-color: darken($color-main, 30%);
  }
}
```

@extend 상속 시, 선언된 선택자가  
존재하지 않을 경우 '옵션' 처리하여 오류를  
발생시키지 않습니다.



# 옵션(!optional) 플래그 스니펫 만들기

Sublime Text에서 활용 가능한 옵션 플래그 스니펫을 만들어 볼까요?,

```
yahoo9_example.html
```

```
+ ① yahoo9_example.html +≡  
1 <snippet>  
  <content><! [CDATA[!optional]]></content>  
  <tabTrigger>optional</tabTrigger>  
  <description>SASS - !optional</description>  
  <scope>source.scss, source.sass</scope>  
</snippet>
```



# 대체 선택자(Placeholder Selector, %)

SASS는 특수한 선택자 ‘%대체 선택자’를 지원합니다. 해석된 CSS에서는 출력되지 않습니다.

The screenshot shows a code editor window titled "yamoo9\_example.html". The code is as follows:

```
%reset-mp {  
  margin: 0;  
  padding: 0; }  
  
#app {  
  @extend %reset-mp; }  
  
#design {  
  @extend %reset-mp; }
```

A purple arrow points from the "%reset-mp" declaration to the first "@extend" statement. A yellow callout box contains the Korean text: "% 선택자 코드는 CSS 코드 변경시, 충격되지 않아요." (The % placeholder selector code does not affect the CSS code when it changes). An icon of a book with "Rt" on it is also visible in the sidebar.

placeholder.sass

```
1 /*  
2  * Placeholder % 선택자  
3  * CSS 코드 변경 시, 해석되지 않습니다.  
4  * HTML에 class 속성을 부여하지 않고도  
5  * 100% 시멘틱한 마크업이 가능합니다.  
6 */  
7  
8 %button  
9 margin: 0 auto 25px  
10 padding: 15px  
11 border-radius: 6px  
12 em  
13   text-decoration: underline  
14  
15 .button-fail  
16 @extend %button  
17 background-color: #fac9c7  
18 border-color: #ef625a  
19  
20 .button-success  
21 @extend %button  
22 background-color: #1be4b7  
23 border-color: #bef8eb  
24  
25 .button-disabled  
26 cursor: default  
27 @extend %button  
28 background-color: #c9c9c9  
29 border-color: #6c6c6c  
30 .msg &  
31   background-color: darken(#fac9c7, 30%)  
32
```

placeholder.css

```
1 /* Placeholder % 선택자  
2  * CSS 코드 변경 시, 해석되지 않습니다.  
3  * HTML에 class 속성을 부여하지 않고도  
4  * 100% 시멘틱한 마크업이 가능합니다.  
5 */  
6 .button-fail, .button-success, .button-disabled {  
7   margin: 0 auto 25px;  
8   padding: 15px;  
9   border-radius: 6px;  
10 }  
11 .button-fail em, .button-success em, .button-disabled em {  
12   text-decoration: underline;  
13 }  
14  
15 .button-fail {  
16   background-color: #fac9c7;  
17   border-color: #ef625a;  
18 }  
19  
20 .button-success {  
21   background-color: #1be4b7;  
22   border-color: #bef8eb;  
23 }  
24  
25 .button-disabled {  
26   cursor: default;  
27   background-color: #c9c9c9;  
28   border-color: #6c6c6c;  
29 }  
30 .msg .button-disabled {  
31   background-color: #d58508;  
32 }
```



# 플레이스홀더(%) 스니펫 만들기

Sublime Text에서 활용 가능한 플레이스홀더 스니펫을 만들어 볼까요?,

```
yamoo9_example.html
```

```
+ ① yahoo9_example.html +≡  
1 <snippet>  
  <content><! [CDATA[%"{$1:플레이스홀더 이름} {  
    $2  
  }  
$0]]></content>  
  <tabTrigger>, </tabTrigger>  
  <scope>source.scss, source.sass</scope>  
  <description>SASS - %placeholder 정의</description>  
</snippet>  
  
'; 단축 키워드 말고 직접 'placeholder'로  
등록하셔도 좋겠죠? ^ - ^
```

# OOCSS + Sass = The best way to CSS

October 2012

Object-oriented CSS is awesome. But littering your markup with non-semantic classes is *not* awesome. Those classes sprinkled all over your HTML are going to change, and that's not gonna be fun. But if you combine OOCSS and Sass you get the best of both worlds: modular CSS without bloated, hard-to-maintain HTML.



IAN STORM TAYLOR  
Co-founder @ Segment.io

모듈러 CSS

의미적이지 않은  
표현 클래스 속성 문제



Import  
@import



## 호출(Import on Steroids)

CSS 뿐만 아니라 \*.scss, \*.sass 파일을 @import 문을 사용하여 호출 할 수 있습니다.

```
yahoo9_example.html
```

+ yahoo9\_example.html +≡

1 // scss, sass 확장자의 경우는 생략할 수 있습니다.  
@import "team-works.scss";  
@import "team-works";

css 확장자는 필히 명시해야 합니다.

2 // 여러 개의 파일을 불러들일 때는 콤마()로 구분하여 불러들일 수 있습니다.  
@import "team-works", "team-test";



import.sass

```
1 /*  
2  * @import  
3  * SASS 코드 상에서는 콤마(,)를 이용하여 구분자를 추가하면  
4  * 자동으로 @import 구문을 생성해줍니다.  
5  * 주석이 @import 문보다 먼저 위치한 경우에도  
6  * CSS 해석시에는 주석이 아래 존재하게 됩니다.  
7 */  
8  
9 @import "reset.css", "typography.css", "layout.css", "ie.css", "print.css"  
10 /*  
11  * SASS 파일을 호출할 경우에는  
12  * 확장자를 생략해도 됩니다.  
13 */  
14  
15 @import "placeholder"  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25
```

import.css

```
1 @import url(reset.css);  
2 @import url(typography.css);  
3 @import url(layout.css);  
4 @import url(ie.css);  
5 @import url(print.css);  
6 /* @import  
7  * SASS 코드 상에서는 콤마(,)를 이용하여 구분자를 추가하면  
8  * 자동으로 @import 구문을 생성해줍니다.  
9  * 주석이 @import 문보다 먼저 위치한 경우에도  
10 * CSS 해석시에는 주석이 아래 존재하게 됩니다.  
11 */  
12 /* SASS 파일을 호출할 경우에는  
13  * 확장자를 생략해도 됩니다.  
14 */  
15 /* Placeholder % 선택자  
16  * CSS 코드 변경 시, 해석되지 않습니다.  
17  * HTML에 class 속성을 부여하지 않고도  
18  * 100% 시멘틱한 마크업이 가능합니다.  
19 */  
20 .button-fail, .button-success, .button-disabled {  
21  margin: 0 auto 25px;  
22  padding: 15px;  
23  border-radius: 6px;  
24 }  
25 .button-fail em, .button-success em, .button-disabled em {  
26  text-decoration: underline;  
27 }  
28  
29 .button-fail {  
30  background-color: #fac9c7;  
31  border-color: #ef625a;  
32 }  
33  
34 .button-success {  
35  background-color: #1be4b7;  
36  border-color: #bef8eb;  
37 }  
38  
39 .button-disabled {  
40  cursor: default;
```



# 임포트(@import) 스니펫 만들기

Sublime Text에서 활용 가능한 임포트 스니펫을 만들어 볼까요?,

```
yahoo9_example.html
```

```
+ ① yahoo9_example.html +≡  
1 <snippet>  
  <content><! [CDATA[@import "$0";]]></content>  
  <tabTrigger>import</tabTrigger>  
  <scope>source.css, source.scss, source.sass</scope>  
  <description>SASS - @import</description>  
</snippet>
```



## \_호출 제어 (Partials)

\*.scss, \*.sass 파일 중 CSS 파일로 변경되지 않게 하는 방법은 파일 명 앞에 \_을 붙여주면 됩니다.

```
yamoo9_example.html
```

```
+ ① yamoo9_example.html +≡
```

```
1 // 기본적으로 SCSS, SASS 파일은 해석되어 CSS로 변경됩니다.  
compile-to-css.sass  
  
// SCSS, SASS 파일 앞에 밑줄()이 붙어 있으면 이를 CSS로 해석 변환하지 않습니다.  
_decompile-to-css.sass
```

스타일 모듈 관리에 용이합니다.



# 스타일 모듈 관리 (Manage Style Modules)

SASS에서는 각각의 역할을 분리하여 필요에 따라 재사용할 수 있도록 모듈 관리하는 것이 가능합니다.

yahoo9\_example.html

```
# style.sass
@import modules/normalize
@import modules/base
@import modules/mixins
@import partials/header
@import partials/footer
@import partials/ie
@import partials/print
```

**style.sass**

**modules**

- normalize.sass
- base.sass
- mixins.sass

**partials**

- header.sass
- footer.sass
- ie.sass
- print.sass



# 스타일 모듈 관리 (Manage Style Modules)

SASS에서는 각각의 역할을 분리하여 필요에 따라 재사용할 수 있도록 모듈 관리하는 것이 가능합니다.

```
yahoo9_example.html
```

```
// 라이브러리: Libraries
@import "compass"
@import "public/config"
@import "public/reset"
@import "shared/avantgarde"
@import "public/mixins"
@import "public/sprites"
@import "shared/payment_types"

// 타이포그래피: Typography
@import "public/type_styles"
@import "public/type_layout"
@import "public/links"
@import "public/pagination"

// 폼: Forms
@import "public/form_styles"
@import "public/form_layout"
@import "public/messaging"

// 레이아웃: Layout
@import "public/layout"
@import "public/lightbox"
@import "public/images"
```

GROUP 1

- import-modules.sass

GROUP 2

- import-modules.css

FOLDERS

- sass-compass-example
- .sass-cache
- css
- modules
  - \_base.sass
  - \_mixins.sass
  - \_normalize.sass
- partials
  - \_footer.sass
  - \_header.sass
  - \_ie.sass
  - \_print.sass
- sass
  - cmt.sublime-snippet
  - compass\_app\_log.txt
  - config.rb
  - index.html

import-modules.sass

```
1 @import "../modules/normalize"
2 @import "../modules/base"
3 @import "../modules/mixins"
4 @import "../partials/header"
5 /*
6   * import-modules
7   */
8 body
9   font: 13px/1.5 'Nanum Gothic', Dotum, Sans-Serief
10 @import "../partials/footer"
11 @import "../partials/ie"
12 @import "../partials/print"
13
```

import-modules.css

```
1 /* Normalize Module
2 */
3 /* Base Module
4 */
5 /* Mixins Module
6 */
7 /* Header Partial Module
8 */
9 header {
10   margin: 10px auto 15px;
11 }
12
13 /* import-modules
14 */
15 body {
16   font: 13px/1.5 "Nanum Gothic", Dotum, Sans-Serief;
17 }
18
19 /* Footer Partial Module
20 */
21 footer address a {
22   text-decoration: none;
23 }
24
25 /* IE Partial Module
26 */
27 .lt-ie9 body::before {
28   content: "old IE";
29 }
30
31 /* Print Partial Module
32 */
33 @media print {
34   body {
35     color: #333333 !important;
36     background-color: white;
37   }
38 }
```

GROUP 1

import-modules.sass

GROUP 2

import-modules.css

FOLDERS

sass-compass-example

.sass-cache

css

modules

\_base.sass

\_mixins.sass

\_normalize.sass

partials

\_footer.sass

\_header.sass

\_ie.sass

\_print.sass

sass

cmt.sublime-snippet

compass\_app\_log.txt

config.rb

index.html

import-modules.sass

```
1 @import "../modules/normalize"
2 @import "../modules/base"
3 @import "../modules/mixins"
4 @import "compass"
5 /* * import modules
6 */
7 body
8 font: 1em/1.5 "Nanum Gothic", Dotum, Sans-Serif;
9
10 @import "modules/_base"
11 @import "modules/_mixins"
12 @import "modules/_normalize"
13
```

import-modules.css

```
1 /* Normalize Module
2 */
3 /* Base Module
4 */
5 /* Mixins Module
6 */
7
8 body
9 font: 1em/1.5 "Nanum Gothic", Dotum, Sans-Serif;
10
11
12
13
```

~/사용자계정/.compass

custom-project-template

stylesheets

modules

\_base.sass

\_mixins.sass

\_normalize.sass

partials

\_footer.sass

\_header.sass

\_ie.sass

\_print.sass

templates

templates

print.css

```
1 /* Print Partial Module
2 */
3 @media print {
4   body {
5     color: #333333 !important;
6     background-color: white;
7   }
8 }
```

Rt

import-project-modules.sass

```
1 @import "modules/base"  
2  
3 @import "partials/header"  
4  
5 @import "partials/print"  
6  
7 @import "modules/normalize"
```

import-project-modules.css

```
1 /* Base Module  
2 */  
3 /* Header Partial Module  
4 */  
5 header {  
6   margin: 10px auto 15px;  
7 }  
8  
9 /* Print Partial Module  
10 */  
11 ▼ @media print {  
12 ▼   body {  
13     color: #333333 !important;  
14     background-color: white;  
15   }  
16 }  
17 /* Normalize Module  
18 */
```

compass-reset.sass

```
1 // Compass reset 모듈 호출
2 @import "compass/reset"
```

compass-reset.css

```
1 html, body, div, span, applet, object, iframe,
2 h1, h2, h3, h4, h5, h6, p, blockquote, pre,
3 a, abbr, acronym, address, big, cite, code,
4 del, dfn, em, img, ins, kbd, q, s, samp,
5 small, strike, strong, sub, sup, tt, var,
6 b, u, i, center,
7 dl, dt, dd, ol, ul, li,
8 fieldset, form, label, legend,
9 table, caption, tbody, tfoot, thead, tr, th, td,
10 article, aside, canvas, details, embed,
11 figure, figcaption, footer, header, hgroup,
12 menu, nav, output, ruby, section, summary,
13 time, mark, audio, video {
14   margin: 0;
15   padding: 0;
16   border: 0;
17   font: inherit;
18   font-size: 100%;
19   vertical-align: baseline;
20 }
21
22 html {
23   line-height: 1;
24 }
25
26 ol, ul {
27   list-style: none;
28 }
29
30 table {
```

Rt



## 중첩 @import

@import를 선언 구문 내부에 사용하면 별도로 분리된 파일에서 코드를 불러와 선택자 뒤에 불러온 모듈 선택자가 추가됩니다. (선택자 상속)

The screenshot shows a web browser window with the title "yamoo9\_example.html". The code editor tab also displays "yamoo9\_example.html". The code itself is as follows:

```
1 .button {  
    background-color: $color-main;  
    font-weight: bold;  
    color: white;  
    padding: 5px;  
}  
  
#footer { ●.....► #footer .button  
    @import button;  
}
```

Two dark gray file icons are shown on the right. The top one is labeled "\_button.sass" and has a red arrow pointing from its bottom edge to the bottom icon. The bottom one is labeled "other.sass" and has a red arrow pointing from its top edge to the bottom edge of the "other.sass" icon.

The image shows a code editor interface with three tabs:

- `_box.sass`: Contains the following code:

```
1 .box
2 margin: 10px auto 20px
3 padding: 10px
```
- `nested-import.sass` (active tab): Contains the following code:

```
1 .demo
2 @import box
3 font: 1em Arial, sans-serif
```
- `nested-import.css`: Contains the generated CSS output:

```
1 .demo {
2   font: 1em Arial, sans-serif;
3 }
4 .demo .box {
5   margin: 10px auto 20px;
6   padding: 10px;
7 }
```