

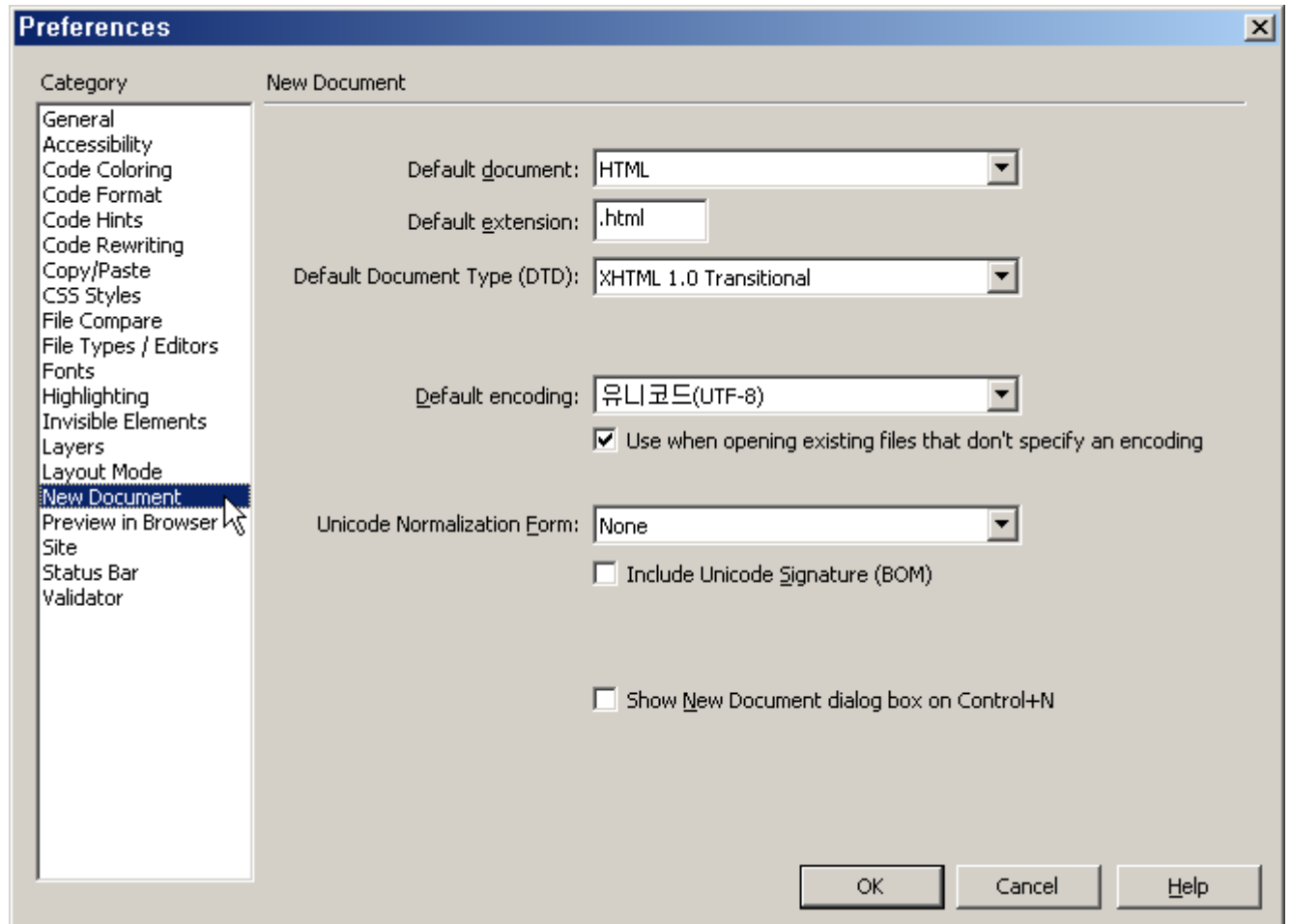
## DTD 관리 (DTD Setting & DTD Switching)

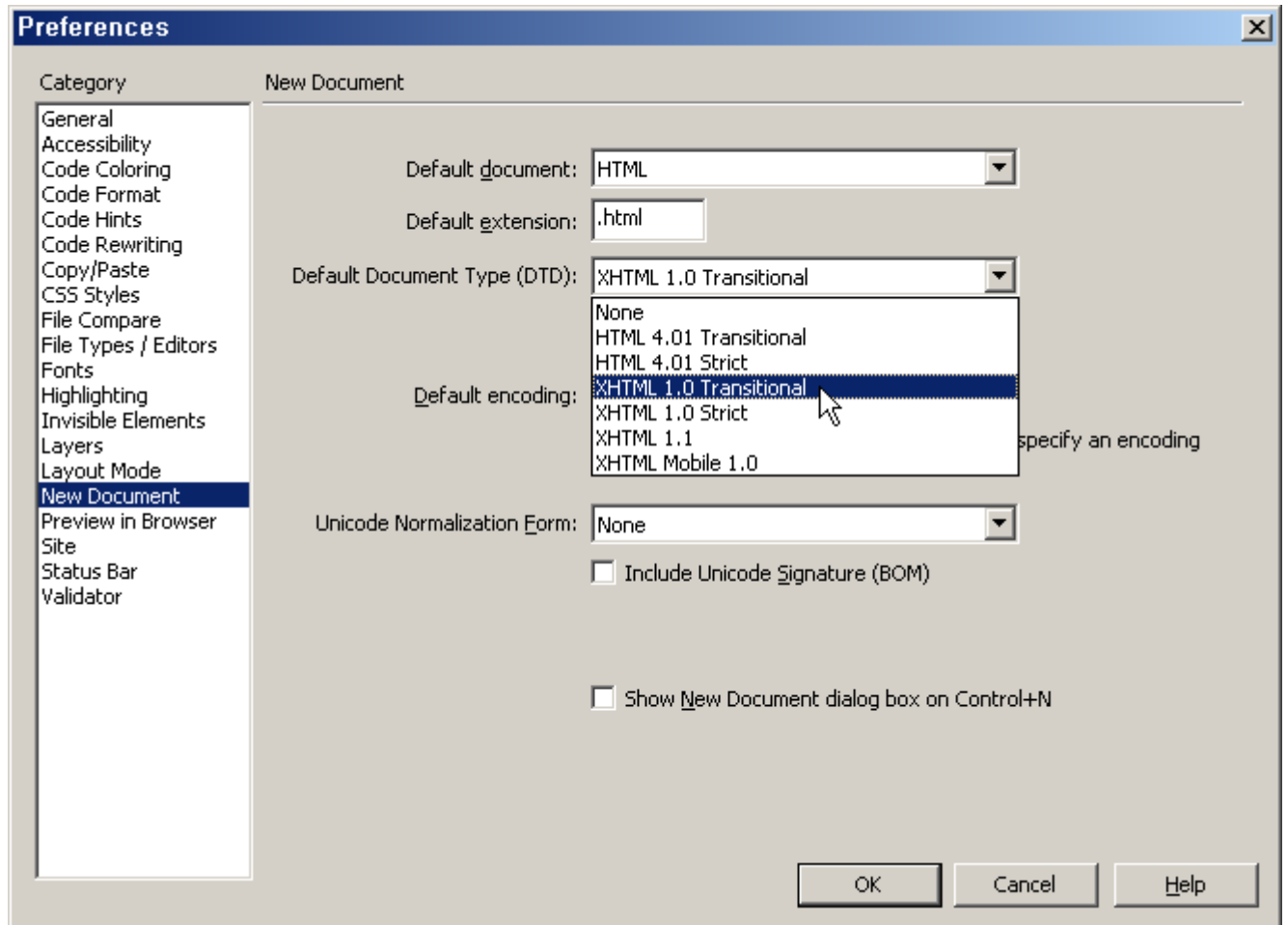
W3C에서 권고하고 있는 HTML DTD의 활성 표준은 XHTML 1.x이다. 표준 DTD를 사용하는 것은 최신의 브라우징 도구에서 더욱 안정적이며 Cross Browsing을 위한 첫 번째 원칙

### 새 문서의 기본 DTD 설정(Ctrl+U)

Preferences > New Document

1. New Document
2. Default Document Type(DTD)
3. HTML 4.01 Transitional → XHTML 1.0 Transitional(권장)





Default Document Type(DTD) 항목에서 원하는 DTD를 선택

```

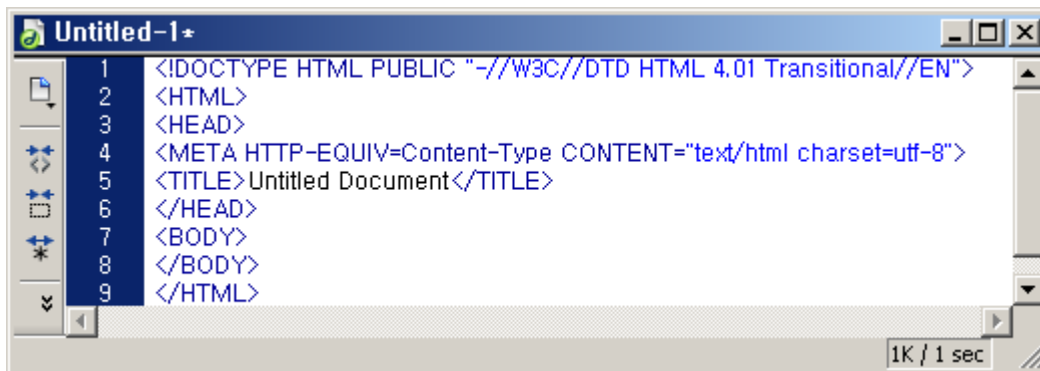
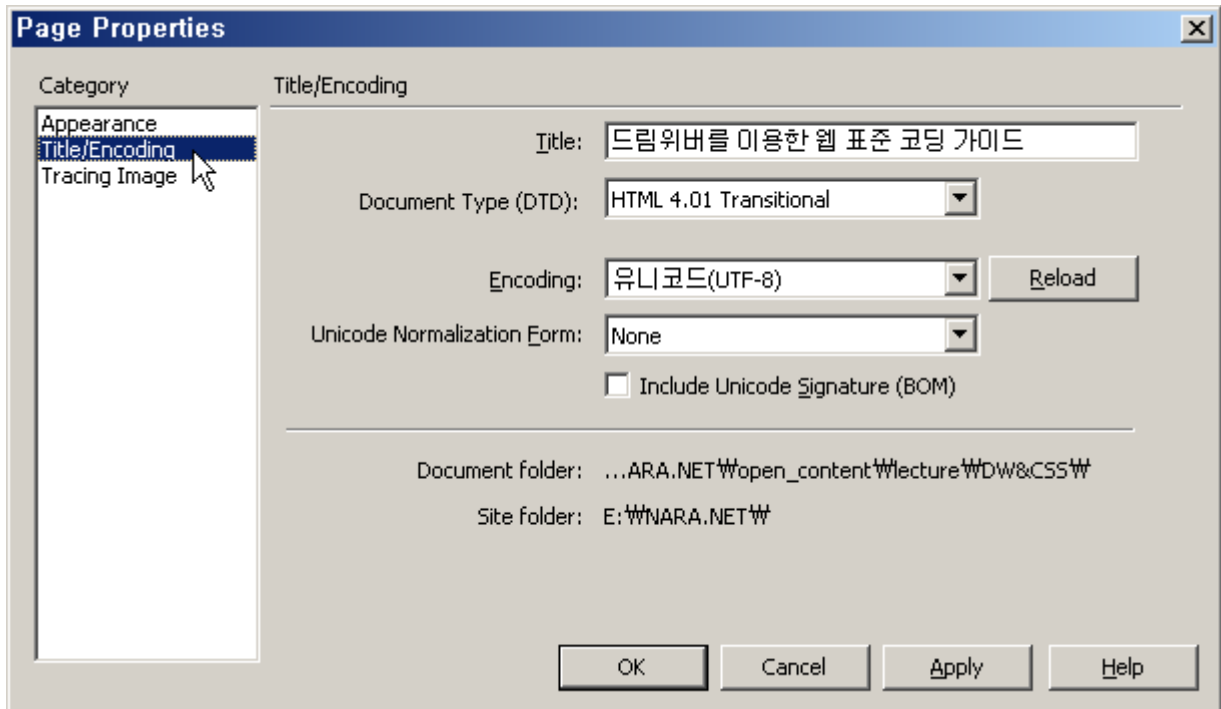
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-t
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Untitled Document</title>
6 </head>
7
8 <body>
9 </body>
10 </html>

```

### 기존 문서의 DTD를 변경하기(Ctrl+J)

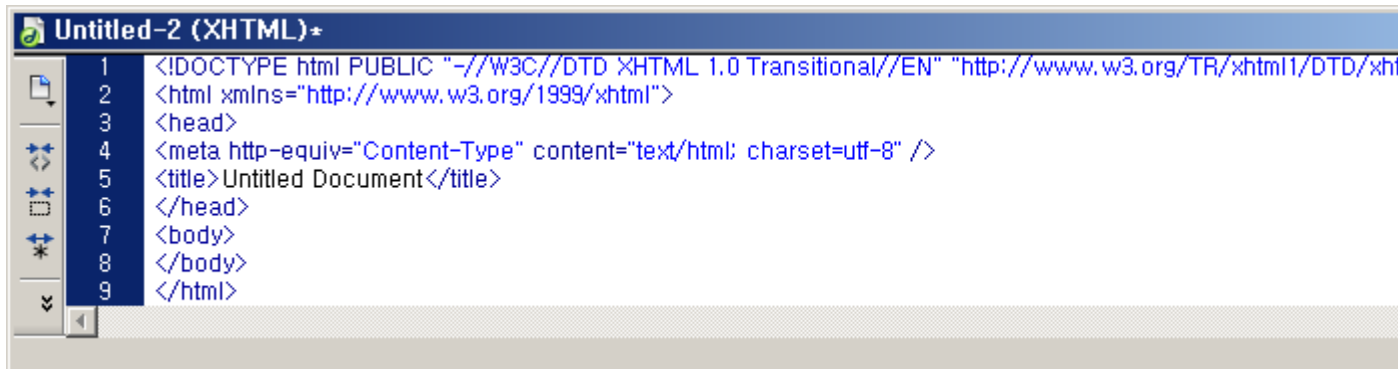
웹사이트를 유지보수 하는 과정에서 기존에 이미 코딩되어 있는 HTML 4.01 문서를 현재의 활성 버전인 XHTML 1.0 문서로 변경하는 방법

1. Page Properties(Ctrl+J)
2. Title/Encoding
3. Document Type(DTD)
4. HTML 4.01 Transitional → XHTML 1.0 Transitional(권장)



HTML 4.01 Transitional 상태로 되어있는 문서의 DTD 변경 전 코드

요소와 속성등이 대문자로 코딩 되어 있으며 속성값에 따옴표 처리가 되어 있지 않는등 다소 느슨한 문법을 보여주고 있다. 하지만 이 문서는 문법적으로는 문제는 없다. 단지 현재의 DTD가 웹 브라우저의 Quirks Mode를 유발하는 DTD 이며 웹 브라우저 마다 심한 렌더링 차이를 보이기 때문에 Cross Browsing과 최신의 웹 표준 권고안을 따르기 위하여 DTD를 변경할 수 밖에 없는 페이지 이다. DTD와 함께 <meta> 태그의 닫기 형식도 눈여겨 본다.



변경 후 코드. 문서의 DTD가 변경되면서 모든 코드가 XHTML 문법에 맞게 변경 되었다.

<meta> 태그의 닫기 형식도 변경됨. 대문자로 코딩되어 있는 Element, Property 등은 모두 소문자로 변경되었다. 속성값에 따옴표가 " " 없는 값은 모두 따옴표 " " 처리. XHTML 문법이 요구하는 엄격한 코드로 자동 변환 됨.

**이 기능은 현재 열려있는 문서에만 적용이 가능하며 사이트 전체의 페이지를 모두 자동으로 변환해 주지는 않는다. 또한 HTML 4.0 버전부터 폐기된 엘리먼트 <applet> <basefont> <center> <dir> <font> <isindex> <menu> <s> <strike> <u> <xmp> 등을 자동으로 제거해 주지는 않는다.** 해당 엘리먼트가 문서에서 어떤 역할을 하고 있는지 판단할 수 없기 때문입니다. 만약 폐기된 엘리먼트에 레이아웃을 위한 스타일시트가 적용되어 있다면 해당 엘리먼트를 제거했을 때 레이아웃이 깨질수도 있다.

#### » 실습용 예제 코드

아래 HTML 4.01 Quirks Mode 버전의 문서 코드를 드림위버를 이용하여 XHTML 1.0 Transitional 버전의 코드로 변경해 본다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV=Content-Type CONTENT="text/html; charset=utf-8">
<TITLE>Untitled Document</TITLE>
</HEAD>
<BODY>
<IMG SRC=blank.gif WIDTH=100 HEIGHT=100 ALT=>
</BODY>
</HTML>
```

#### » 보충설명 : HTML과 XHTML의 차이

1. 문법적으로 엄격하게 구성되어 있어야 한다.  
HTML 은 종료태그가 없는 것을 허용하였으나 XHTML은 반드시 종료태그를 갖는다.  
HTML 은 태그의 중첩이 잘못된 것을 허용하였으나 XHTML은 잘못된 중첩을 허용하지 않는다. 잘못된 중첩은 화면표시(렌더링)에 직접적인 영향을 주기도 한다.
2. 요소와 속성은 소문자로 표기되어야 한다.  
HTML 은 요소(=엘리먼트, 태그)와 속성에 대소문자를 함께 사용하는 것을 허용하였으나 XHTML의 마크업 '요소'와 '속성'들은 반드시 소문자로 표기한다. 단, 속성의 '값'에는 대소문자 혼합 표기가 가능하다. 하지만 대소문자를 명확하게 구분하기 때문에 대문자로 구성된 '값'과 소문자로 구성된 '값'은 동일하지 않고 확실히 구별된다.
3. 모든 태그는 종료태그를 갖는다.  
HTML 의 경우 <p>, <td> 등의 태그에서 종료태그를 생략하는 것을 허용하였지만 XHTML 의 경우 반드시 닫아야 한다.
4. 속성 '값'들은 항상 따옴표로 감싸주어야 한다.  
HTML 의 경우 속성 값들을 따옴표로 감싸지 않는 것을 허용하였지만 XHTML 에서는 반드시 속성 "값"은 따옴표 안에 있어야 한다.
5. 속성과 값의 단축표기를 허용하지 않는다.

HTML에서는 속성과 속성 값의 단축표기를 허용하였으나 XHTML에서는 단축표기 하는 것을 허용하지 않는다. <input checked> 는 <input checked="checked"> 와 같이 표기되어야 한다.

6. 비어있는 태그(종료태그가 없는 태그)도 종료 되어야 한다.

HTML에서 <br>, <hr> 과 같이 콘텐츠를 담지 않는 빈 태그들은 <br />, <hr /> 과 같이 표기하여 시작태그에서 곧 종료됨을 표기해 주어야 한다.

7. a, applet, frame, iframe, img, map 에서 name 속성은 다음 버전부터 지원하지 않는다. id 와 name 을 함께 사용하던 마크업의 name 속성은 모두 id 속성으로 교체되어야 한다. **name** 속성은 공식적으로 폐기하였지만 여전히 **XHTML 1.0 Transitional 버전의 문서까지는 지원**하고 있다. 하지만 XHTML 1.0 Strict 버전에서 지원하지 않으며 다음 버전에서는 분명히 폐기된다.

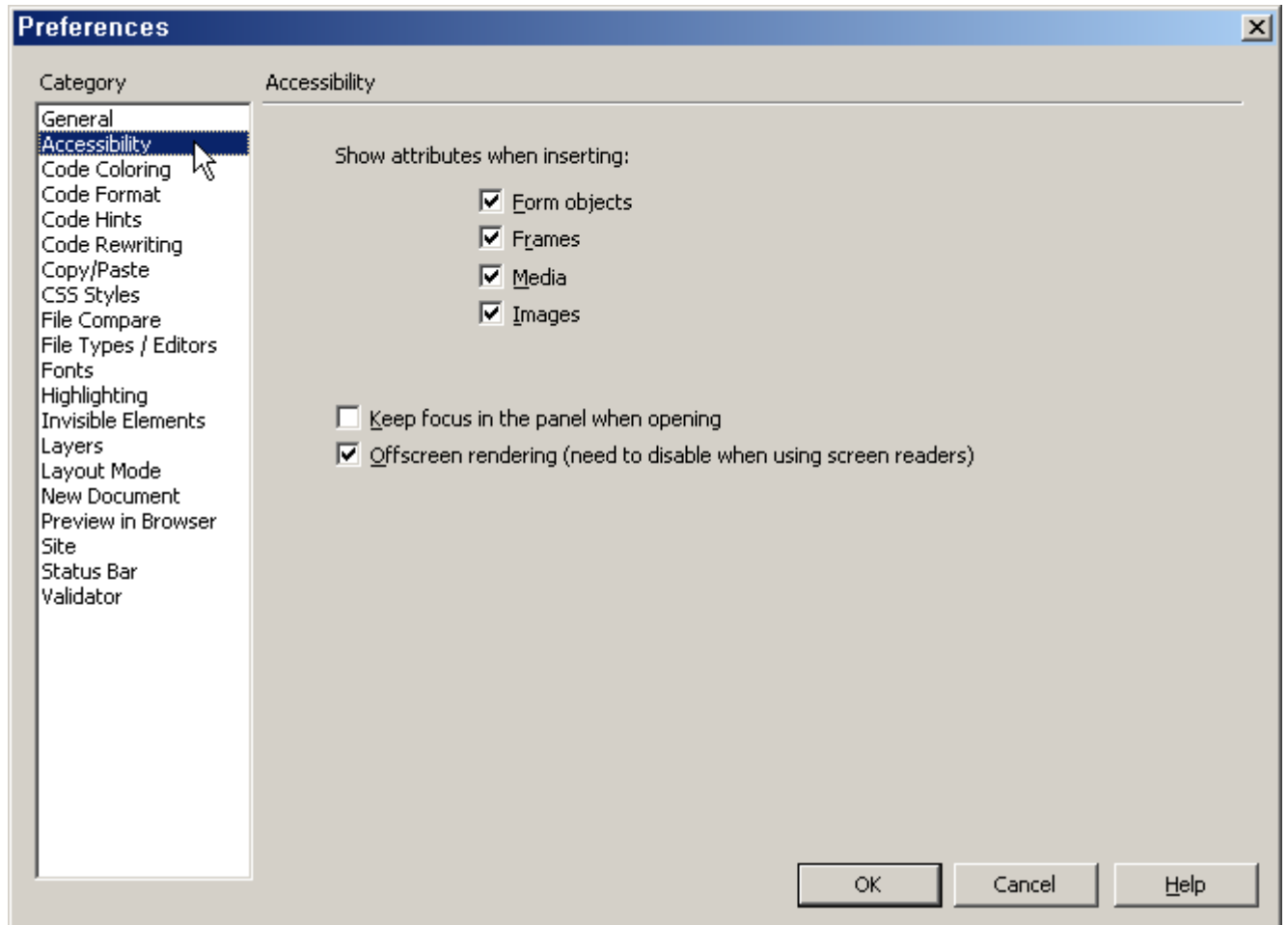
### 접근성 대화상자 (Accessibility Dialog Box)

접근성 대화상자는 웹 문서의 접근성과 사용성을 높이기 위한 코드의 생성을 지원하고 있다.

#### 접근성 대화상자 환경설정(Ctrl+U)

Preferences > Accessibility 에서 접근성 대화상자의 사용 여부를 체크 한다.

1. Preferences(Ctrl+U)
2. Accessibility
3. Show attributes when inserting:
  - Form object
  - Frames
  - Media
  - Images



Show attributes when inserting 항목에 모두 체크하면 접근성 속성을 지닌 엘리먼트 <form> <frame> <object> <img> 요소를 삽입할 때 접근성 관련 속성의 입력을 도와주는 대화상자가 나타난다. **모두 체크하는 것을 권장**

- Keep focus in the panel when opening  
패널이 열릴 때 포커스 유지하기. 시각장애인이 드림위버를 사용하는 경우 패널이 새로 열릴 때 포커스를 새 패널에 맞히도록 한다.
- Offscreen rendering (need to disable when using screen readers)  
오프스크린 렌더링. 스크린리더를 사용하는 경우 체크를 해제 한다.

**Form 접근성 대화상자**

**Input Tag Accessibility Attributes**

Label:

Style: ☐ Wrap with label tag  
☒ Attach label tag using 'for' attribute  
☐ No label tag

Position: ☒ Before form item  
☐ After form item

Access key:  Tab Index:

If you don't want to enter this information when inserting objects, [change the Accessibility preferences.](#)

OK Cancel Help

**입력 태그 액세스 가능성 속성**

ID:

레이블:

스타일: ☐ 레이블 태그로 줄 바꿈  
☒ 'for' 속성을 사용하여 레이블 태그 첨부  
☐ 레이블 태그 없음

위치: ☒ 양식 항목 이전  
☐ 양식 항목 다음

선택 키:  탭 인덱스:

객체 삽입 시 정보를 입력하지 않으려면 [액세스 가능성 환경 설정을 변경하십시오.](#)

확인 취소 도움말

<label for="textfield">아이디</label>

<input type="text" name="textfield" id="textfield" />

접근성 속성 대화상자. Form 요소의 하나인 <input type="text">와 같은 요소를 삽입하고자 할 때 Label 텍스트를 함께 입력할 수 있다.

» **Style:**

- Wrap with label tag [권장안함]

for 속성을 사용하지 않고 label 요소가 텍스트와 input을 모두 감싸도록 하는 방법

```
<label>아이디
```

```
<input type="text" name="textfield" />
```

```
</label>
```

- Attach label tag using 'for' attribute [권장]

for 속성을 사용하여 label 요소를 마크업 하는 방법

```
<label for="textfield">아이디</label>
```

```
<input type="text" name="textfield" id="textfield" />
```

- No label tag [권장안함]

label 요소를 사용하지 않는 방법

```
<input type="text" name="textfield" />
```

이 가운데 for 속성을 사용하여 label 요소를 마크업 하는 두 번째 방법이 현실적으로 가장 접근성이 높습니다. label요소를 사용하더라도 for 속성을 사용하지 않는 경우 대부분의 스크린리더는 이것을 해석하지 못한다고 합니다. for 속성을 사용하여 label 요소를 마크업 하는 경우 input 요소에 포커스가 맺힐 때 스크린리더는 label 텍스트를 읽어줍니다. 장애를 갖지 않은 일반 사용자들은 label 텍스트를 클릭하는 것만으로도 form 요소를 클릭하게 되는 효과가 있어 사용성에도 도움이 됩니다. 이것은 특히

- input type="checkbox" 또는

☐ 이것은 라벨 텍스트입니다

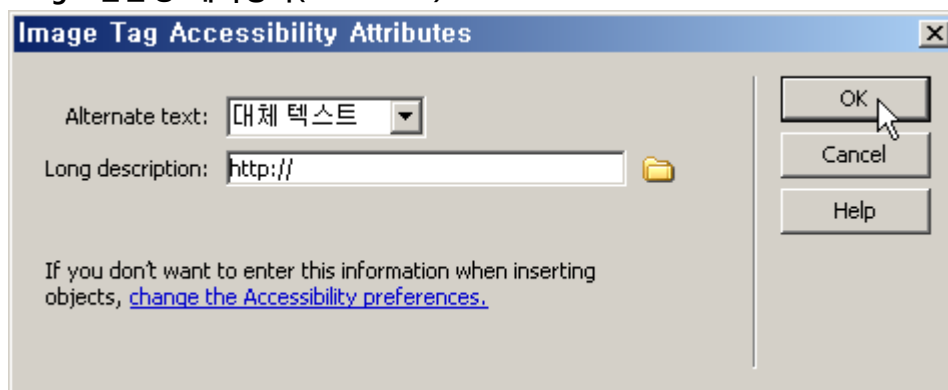
- input type="radio" 형식일 때 더욱 그렇습니다.

☒ 이것은 라벨 텍스트입니다

바로 위 예제로 제공된 체크박스과 라디오 버튼의 우측에 위치한 라벨 텍스트를 클릭해 봅니다. 텍스트를 클릭하여 form 요소를 제어할 수 있게 됩니다.

접근성 속성 대화상자에서 Position: 항목은 라벨텍스트를 <input> 요소의 앞에 넣을 것인지 뒤에 넣을 것인지를 묻습니다. Access key 항목은 브라우징 기기의 단축키 조합과 충돌할 가능성이 항상 존재하기 때문에 권장하지 않으며 Tab Index는 Tab키의 이동경로를 예측하는 사용자의 자연스러운 흐름을 방해할 우려가 있어 역시 권장되지 않습니다. Tab Index 속성을 사용하지 않은 상태에서 Tab 키의 순서가 사용자의 예측을 벗어나지 않고 논리적으로 흐르도록 순서에 유의하며 마크업 하는 방법이 가장 좋습니다.

#### Image 접근성 대화상자(Ctrl+Alt+I)

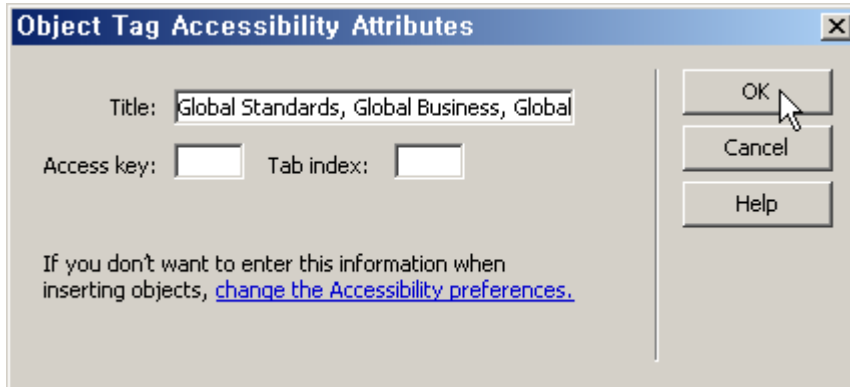






이미지 요소를 삽입할 때 만나는 접근성 속성 대화상자 입니다. 이미지에 대한 대체 텍스트 또는 Long Description 속성을 사용할 수 있습니다. 현실적으로 Long Description을 사용하는 경우는 매우 드문데 스크린 리더 조차 이것을 지원하는 경우도 드물다고 하는군요. 하지만 스크린 리더는 언제라도 개선될 수 있습니다.

#### Media 접근성 대화상자(Ctrl+Alt+F)



<script type="text/javascript">

```
AC_FL_RunContent( 'codebase','http://download.macromedia.com/pub/shockwave/cabs/flash/swflas  
h.cab#version=7,0,19,0','width','400','height','325','title','Global Standards, Global Business, Global  
Mind!','src','mainVisual','quality','high','pluginspage','http://www.macromedia.com/go/getflashplayer',  
'movie','mainVisual' ); //end AC code
```

</script>

<noscript>

<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,  
19,0" width="400" height="325" title="Global Standards, Global Business, Global Mind!">

<param name="movie" value="mainVisual.swf" />

<param name="quality" value="high" />

<embed src="mainVisual.swf" quality="high"

pluginspage="http://www.macromedia.com/go/getflashplayer" type="application/x-shockwave-  
flash" width="400" height="325"></embed>

</object>

</noscript>

Media 요소를 삽입할 때 만나는 접근성 속성 대화상자 입니다. Flash등의 Object를 삽입할 때 <object> 태그에 title 속성을 추가해 줍니다. Flash는 제작단계에서 버튼 심볼 등에 description속성을 추가해 주면 스크린리더가 그것을 대체텍스트로 인지하는 것이 가능합니다. 하지만 아직 국산 스크린리더들은 대부분의 경우 Flash 오브젝트를 인지하지 못하고 그냥 건너 뛰기 때문에 Flash를 주요 네비게이션에 사용하는 것은 바람직하지 않습니다.

#### Frame 접근성 대화상자



```
<frameset rows="80,*" frameborder="no" border="0" framespacing="0">
<frame src="menu.html" name="menu" scrolling="No" noresize="noresize" id="menu" title="사이트 메뉴" />
<frame src="main.html" name="content" id="content" title="나라디자인-홈" />
</frameset>
```

Frames 요소를 삽입할 때 만나는 접근성 속성 대화상자 입니다. Frameset문서에서 각각의 프레임에 대한 title 속성을 입력하도록 지원합니다. 하지만 이러한 프레임 구조는 접근성을 치명적으로 훼손시킬 수 있기 때문에 권장하지 않습니다.

#### Table 접근성 대화상자(Ctrl+Alt+T)

#### XHTML 유효성 검사 (XHTML Validation Check)

The W3C Markup Validation Service 가 단 한 개의 페이지에 대한 유효성 검사만을 지원해 주는데 반하여 드림위버가 제공하는 유효성 검사 기능은 지정폴더 또는 현재 로컬 사이트 전체 페이지에 대한 유효성 검사까지 지원해 주고 있습니다. 단, XHTML의 표준 문법만을 검증하며 의미론적으로 적합한 마크업을 지녔는지에 대하여는 검증해 주지 못하므로 이 검증결과를 통과했다고 하여 완전히 유효한 코드를 작성했다고 보기는 어렵습니다. 이것은 현존하는 모든 유효성 검사도구의 한계이며 앞으로도 마크업의 의미가 적합한지에 대하여 검증해 주는 도구는 나타나기 어려울 것으로 보입니다. 왜냐하면 도구는 콘텐츠의 의미를 파악할 수 없기 때문에 어떤 콘텐츠가 제목요소로 마크업이 되어야 하는지 문단요소로 마크업이 되어야 하는지 가늠하기 어려운 이유입니다. 웹 표준 마크업 이란 결국 '문법적 유효성 + 시멘틱 마크업' 이 병행 되어야 하며, 시멘틱 마크업을 생성하고 검증하는 것은 오직 사람만이 할 수 있는 일입니다. 실제로 모든 문서에는 마땅히 제목이 있어야 하지만 유효성 검사 도구들은 문서 내에서 제목요소를 발견하지 못했다고 해서 그것에 대하여 오류 메시지를 출력하지는 않습니다.

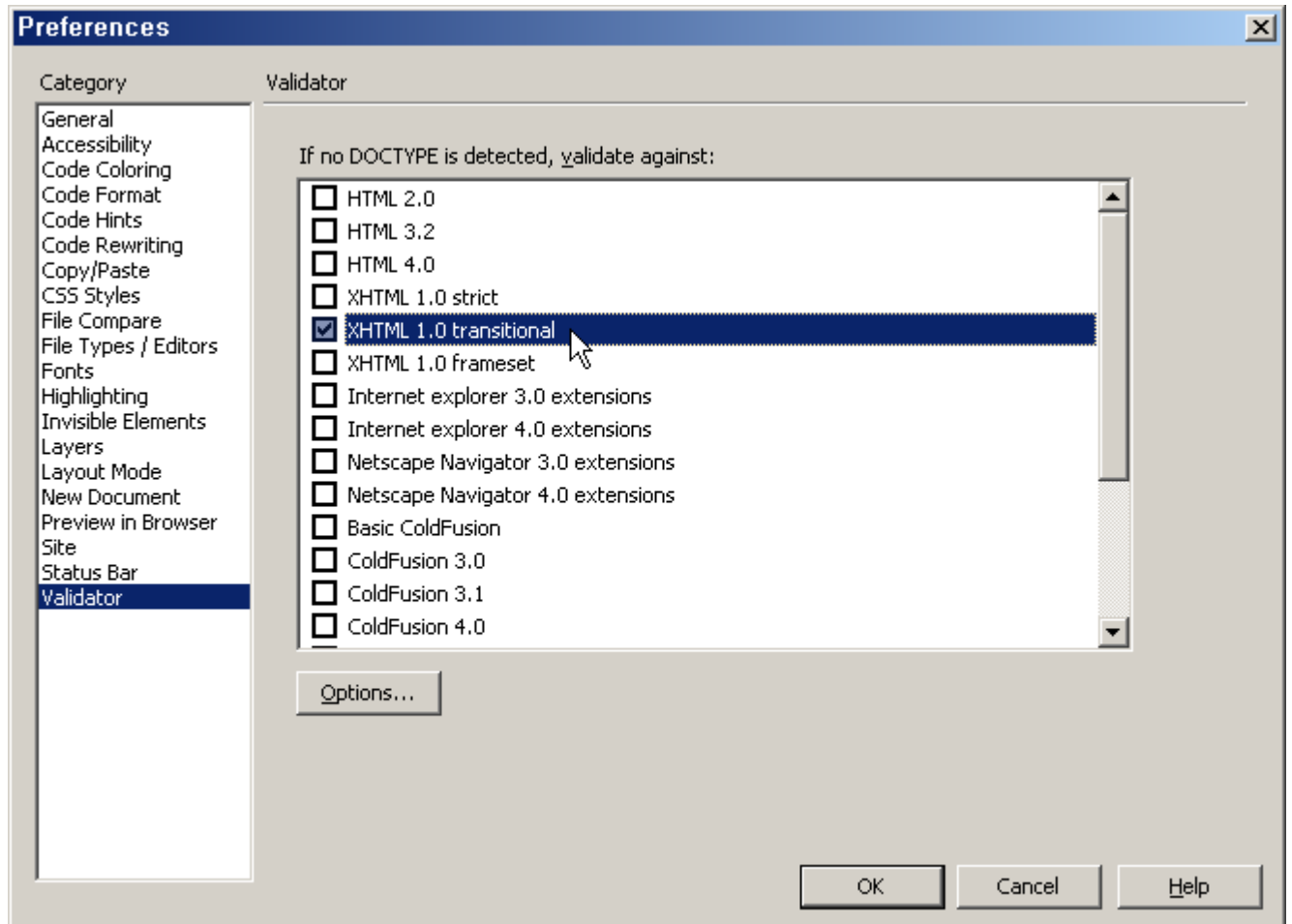
#### 유효성 검사를 위한 환경설정(Ctrl+U)

Validation 검사는 기본적으로 해당 문서의 DTD를 기준으로 진행 됩니다. 하지만 DTD가 없는 문서를 만났을 때에는 어떤 DTD를 기준으로 유효성 검사를 진행할 것인지 미리 설정해 두어야 합니다. 흔히 Include 되는 페이지(예: Header, Global Navigation, Local Navigation, Footer) 들은 DTD가 없는 상태로 존재하기 때문에 Preferences > Validator 항목에서 DTD가 없는 문서를 발견했을 때 기준이 되는 DTD를 설정해 두어야 합니다.

1. Preferences(Ctrl+U)

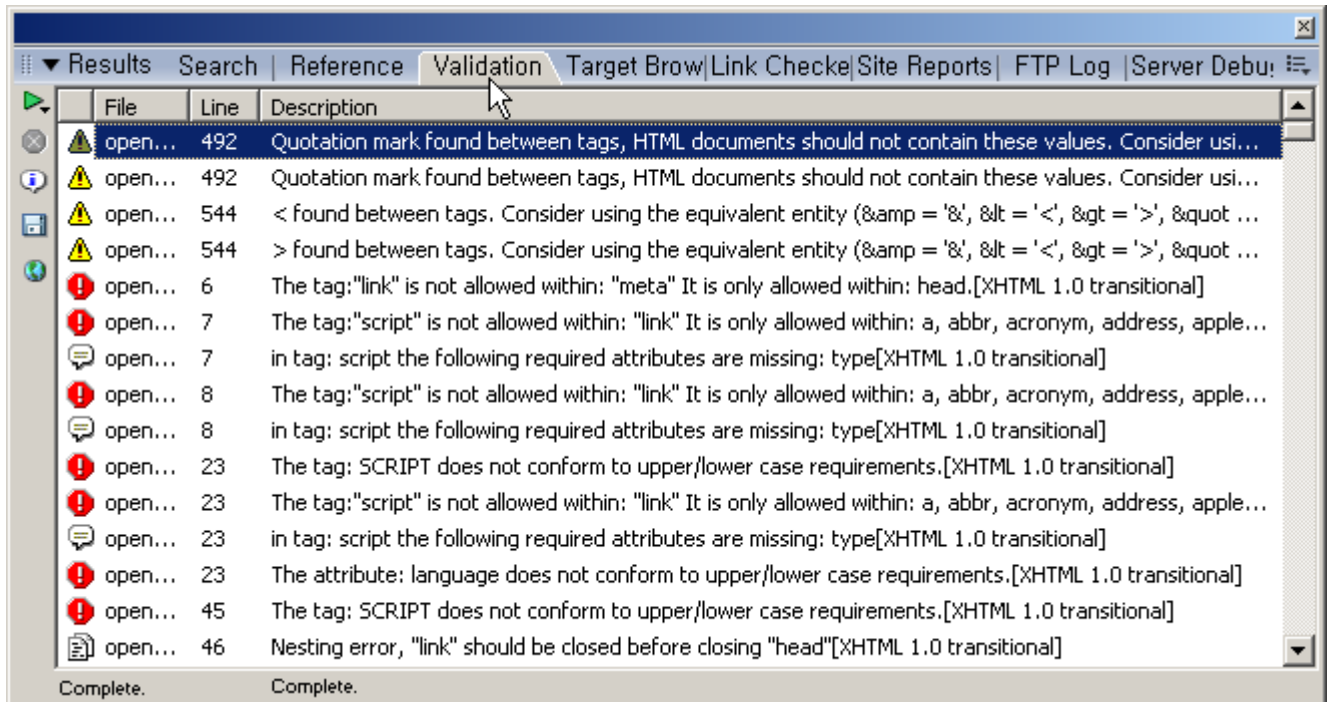
2. Validator

- If no DOCTYPE is detected, validate against (DTD가 없는 문서는 다음 DTD에 준하여 검증할 것)



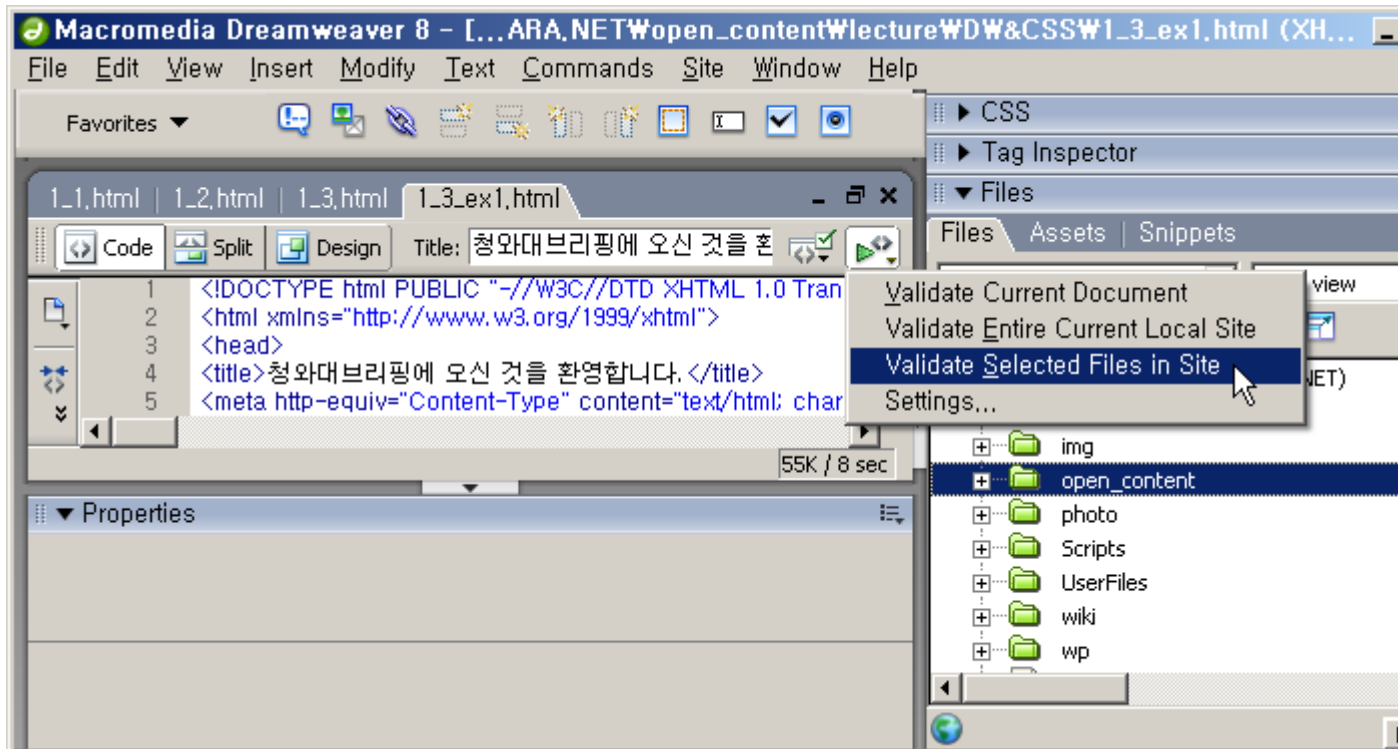
만약 현재 사이트의 모든 문서에 DTD가 포함되어 있다면 이 설정은 무시됩니다. 현재의 설정은 DTD가 없는 문서를 만났을 때 기준이 되는 DTD를 선택하는 작업입니다.

현재 페이지 유효성 검사(Shift+F6)



단축키를 이용합니다. Shift+F6을 누르면 현재 열려있는 문서에 대한 Validation 검사결과를 보여줍니다. Description 항목에는 오류코드에 대한 설명이 적혀 있으며 HTML관련지식이 있는 사람이라면 영문이라도 판독이 가능한 수준입니다.

전체 페이지(또는 특정 폴더) 유효성 검사



Document Panel 에 위치한 Validate Markup 메뉴를 이용하면 Files Panel에서 선택한 폴더, 또는 로컬 사이트 전체에 대한 유효성 검증이 가능 합니다.

» Index

Table 삽입 대화상자는 기본적으로 접근성 관련속성을 함께 입력하도록 구성되어 있으며 환경설정에서는 이것을 제어하는 별도의 옵션을 제공하고 있지 않습니다. Header 셀(th)을 지정하고 Caption 요소와 Summary 속성을 한꺼번에 입력할 수 있도록 지원하고 있는데 이 대화상자에서 Header 셀을 지정하게 되면 th 엘리먼트에 scope 속성이 함께 작성되어 접근성이 매우 높은 표가 생성 됩니다.

### 자장면과 짬뽕의 가격과 열량 비교

구분	자장면	짬뽕
가격	3,000	3,500
열량	300	350

```
<table cellpadding="0" summary="짬뽕은 자장면보다 500원이 비싸고 열량이 50 높다">
```

```
<caption>
```

```
자장면과 짬뽕의 가격과 열량 비교
```

```
</caption>
```

```
<thead>
```

```
<tr>
```

```
<th scope="col"> 구분</th>
```

```
<th scope="col"> 자장면</th>
```

```

<th scope="col"> 짬뽕</th>
</tr>
</thead>
<tbody>
<tr>
<th scope="row"> 가격</th>
<td class="ta_right"> 3,000</td>
<td class="ta_right"> 3,500</td>
</tr>
<tr>
<th scope="row"> 열량</th>
<td class="ta_right"> 300</td>
<td class="ta_right"> 350</td>
</tr>
</tbody>
</table>

```

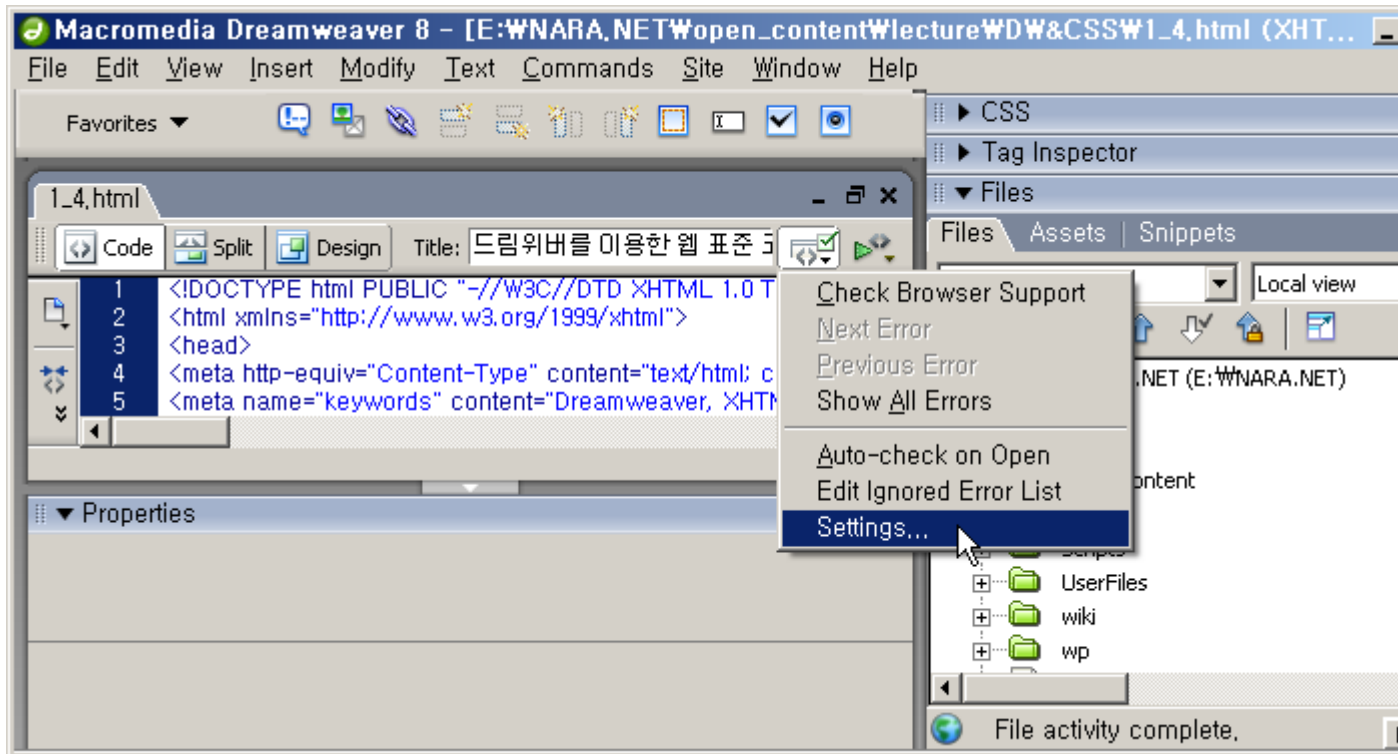
시각 장애인들이 사용하는 스크린리더는 이 표를 이렇게 읽어줄 것입니다.

1. 표 제목 - 자장면과 짬뽕의 가격과 열량 비교
2. 표 요약 - 짬뽕은 자장면보다 500원이 비싸고 열량이 50 높다
3. 표 머릿글 - 구분, 자장면, 짬뽕
4. 표 내용 - 가격, 자장면 가격 3,000원, 짬뽕 가격 3,500원, 열량, 자장면 열량 300, 짬뽕 열량 350

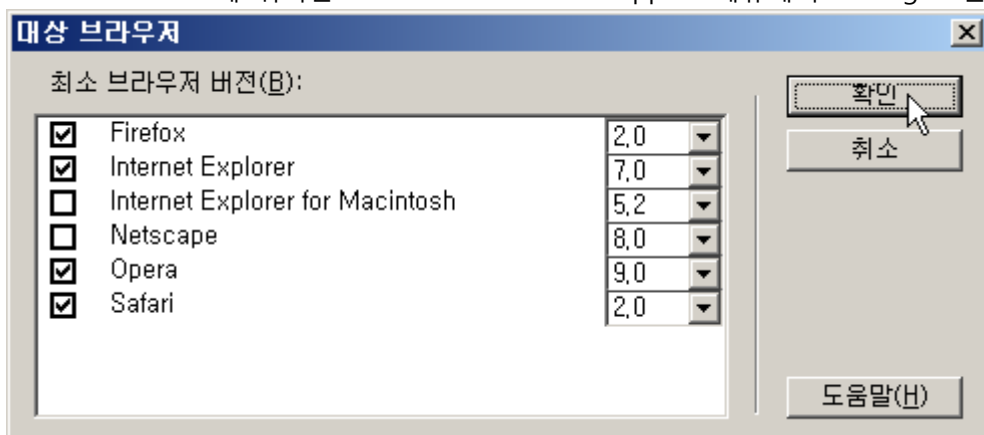
### CSS 유효성 및 Cross Browsing 검사 (CSS Validation & Cross Browsing Check)

CSS의 속성과 값을 현존하는 웹 브라우저들이 제대로 지원하는지 확인하는 기능입니다.

#### CSS 유효성 검사를 위한 환경설정

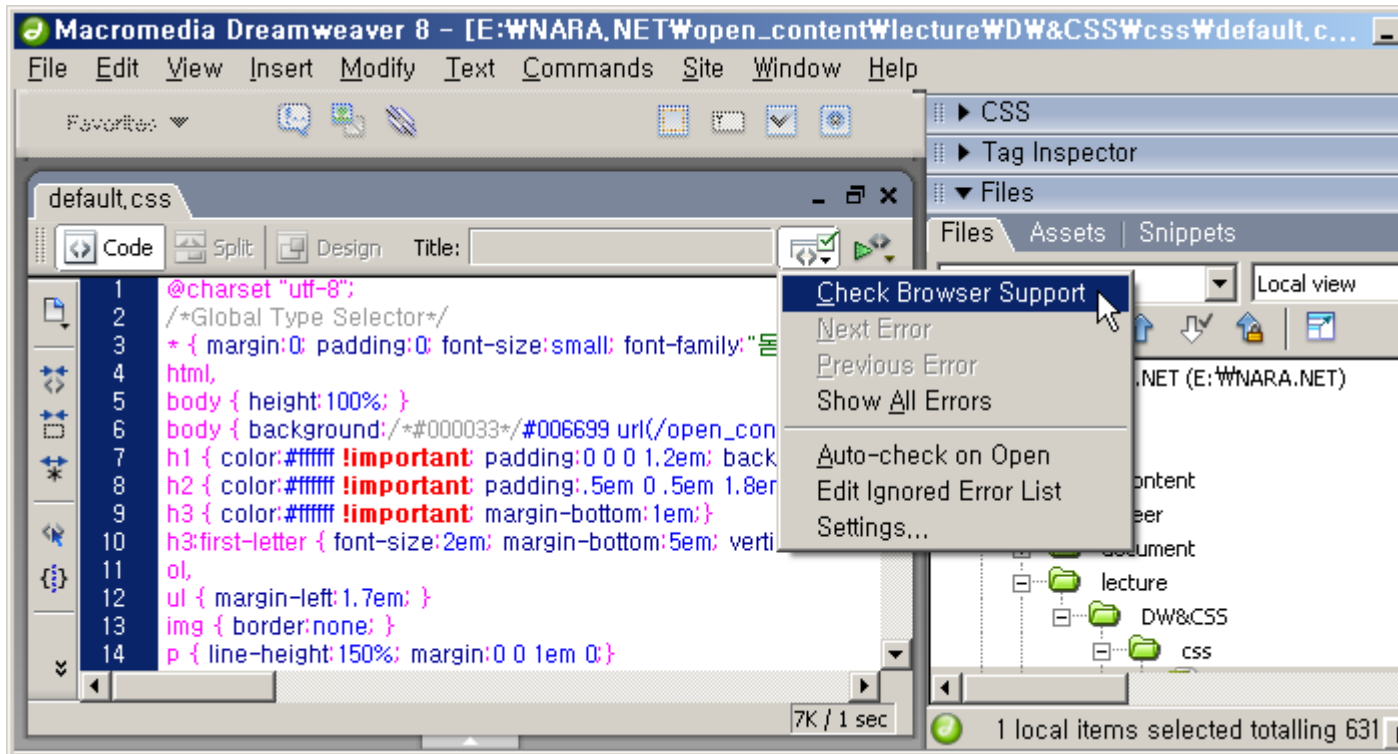



Document Panel 에 위치한  Check Browser Support 메뉴에서 Settings... 를 선택 합니다.

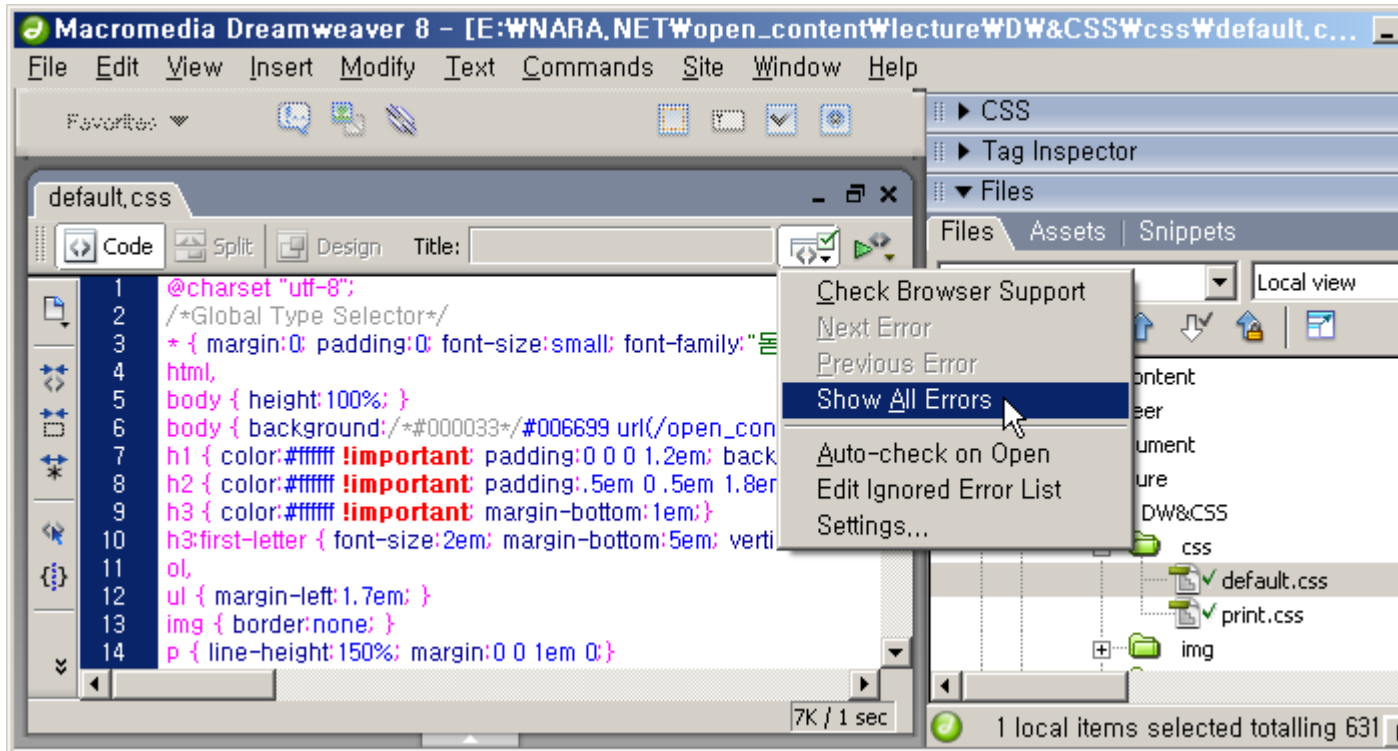



지원하고자 하는 브라우저와 버전을 선택 합니다.

**CSS 유효성 및 Cross Browsing 검사**

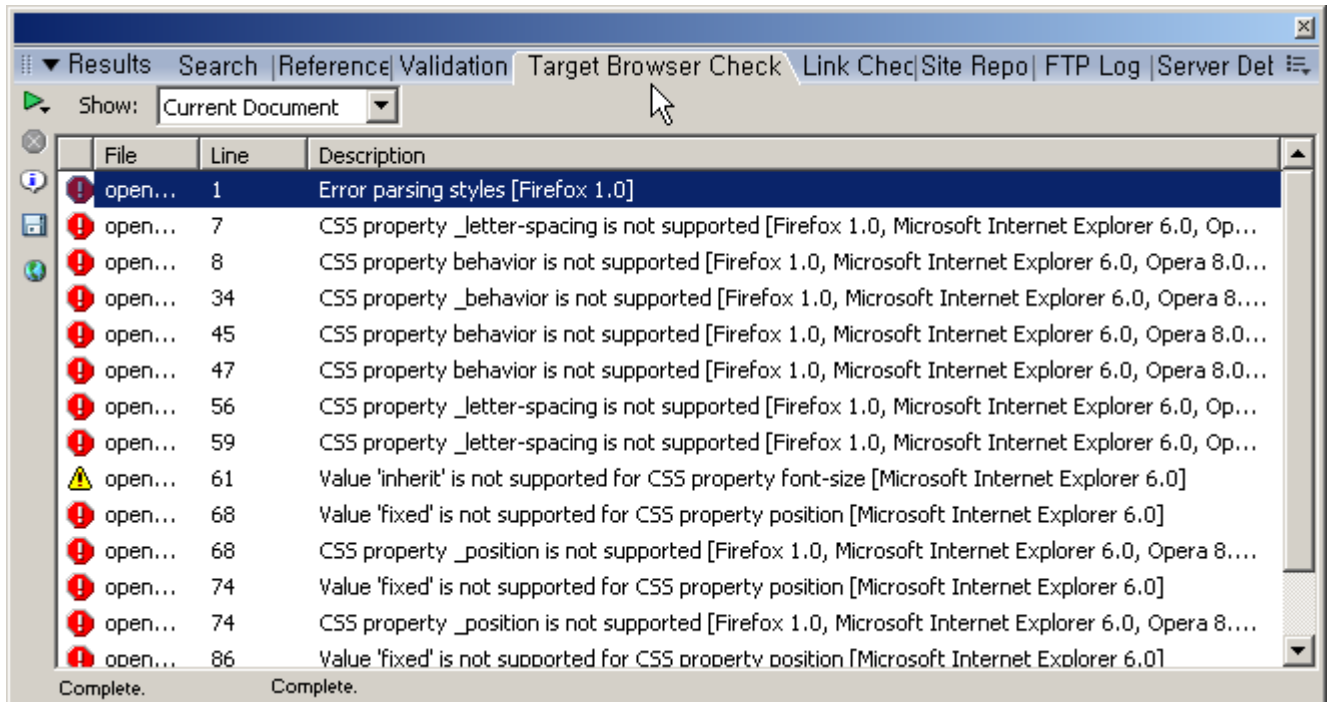


검사하고자 하는 CSS 문서를 열어놓고 Document Panel 의  Check Browser Support 기능을 실행합니다. 이것을 실행하더라도 화면상에는 아무 변화가 나타나지 않을 것입니다. 다음 과정을 진행 합니다.



다시한번  Check Browser Support 메뉴에서 Show All Errors 항목을 실행 합니다.





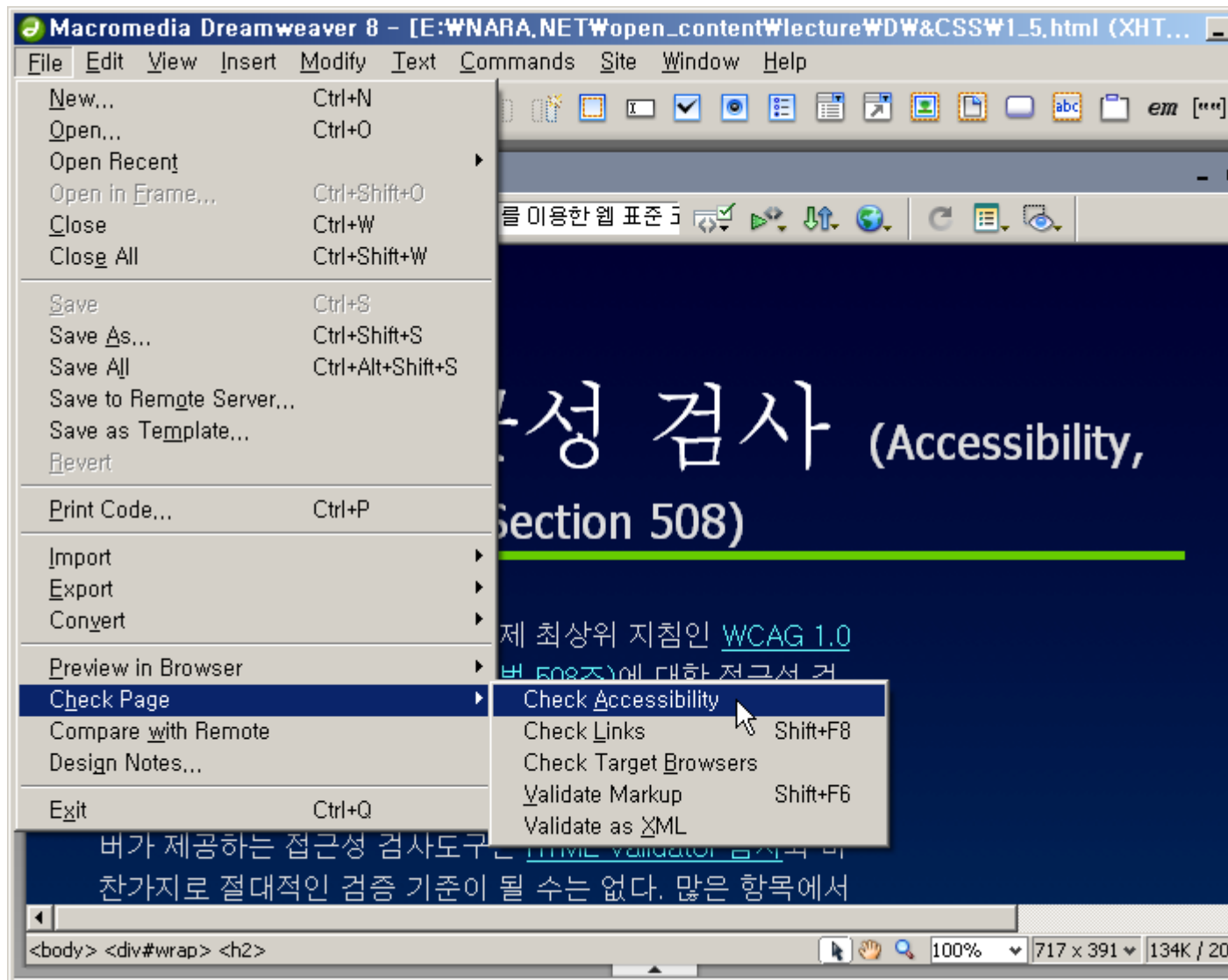
Result Panel 의 Target Browser Check 탭에 검사결과가 출력 됩니다. 사용자가 설정해 놓은 최소 지원 브라우저(Minimum Target Browser Versions)에서 지원되지 않는 CSS 항목들을 출력해 줍니다. 이곳에서 오류로 보고되는 항목들은 대부분 해당 브라우저에서 렌더링 오류를 유발하는 것들이기 때문에 이 검증방법은 Crowss Browsing 검사결과라고 판단해도 좋습니다. 한편 CSS Hack을 사용하였고 타 브라우저에서는 그것을 오류로 인식하지 않고 무시하더라도 드림위버는 Hack 자체를 오류라고 판단하고 검사결과에 반영합니다.

한 가지 아쉬운 점은 CSS 유효성 검사도구가 웹 브라우저의 버전에 대응하는 검사결과를 출력해주지만 W3C가 권고하는 현재의 활성 표준(CSS 2.0)에 대응하는 검사결과를 보고해 주지는 않는다는 점 입니다. 따라서 이 검사도구의 사용자는 브라우저에서 지원하지 않아 오류로 보고된 CSS 속성이 현재의 활성 표준인지 아닌지를 가늠할 수 없게 되는 문제가 있습니다.

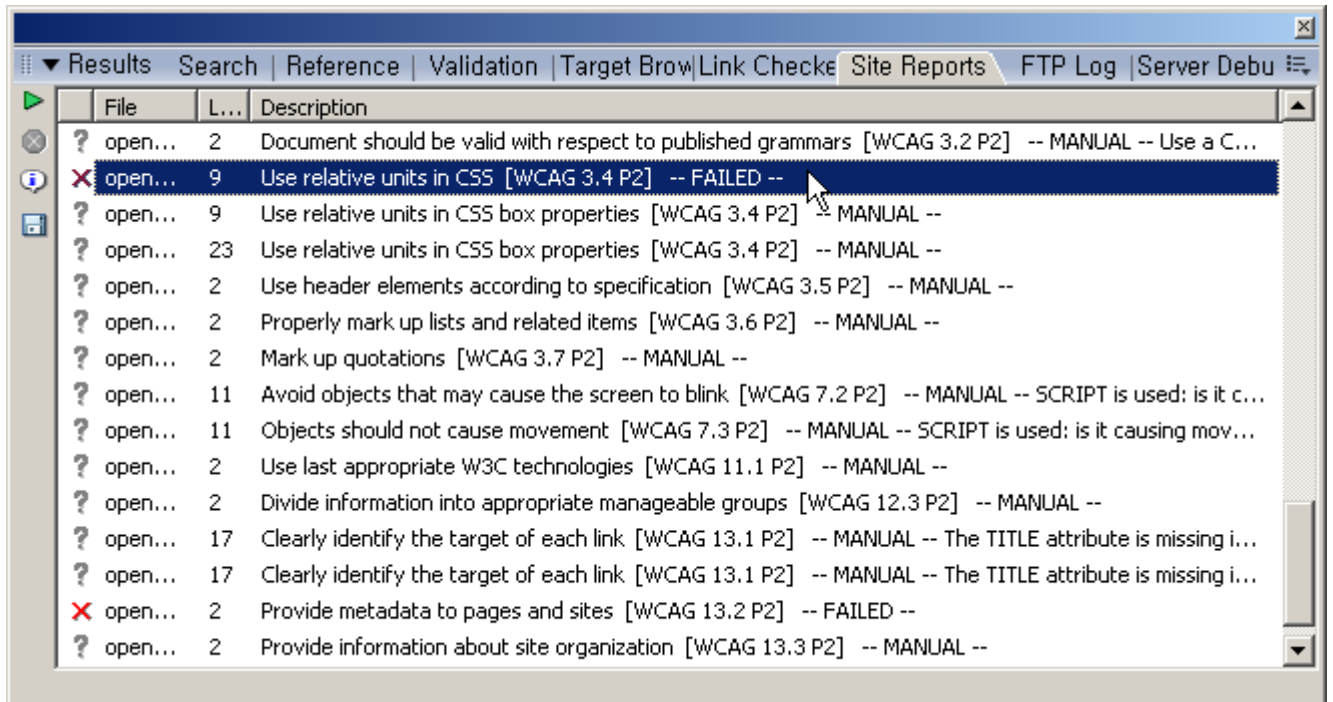
### 웹 접근성 검사 (Accessibility, WCAG 1.0 & Section 508)

드림위버는 웹 접근성에 대한 국제 최상위 지침인 WCAG 1.0 지침과 미국의 Section 508(재활법 508조)에 대한 접근성 검사를 지원하고 있습니다. 한국의 최상위 지침인 KWAG 1.0 지침은 WCAG 1.0 지침과 거의 흡사하기 때문에 KWAG 1.0 지침을 준수하기 위하여 다른 접근성 도구를 사용할 필요는 없습니다. 다만 드림위버가 제공하는 접근성 검사도구는 HTML Validator 검사와 마찬가지로 절대적인 검증 기준이 될 수는 없습니다. 많은 항목에서 사람에 의한 수동검사를 필요로 합니다. 한편 KADO-WAH라는 검증도구는 KWAG 1.0 지침을 기준으로 웹 접근성을 검증하고 이를 보고서로 제출하기 위한 용도로 적합합니다.

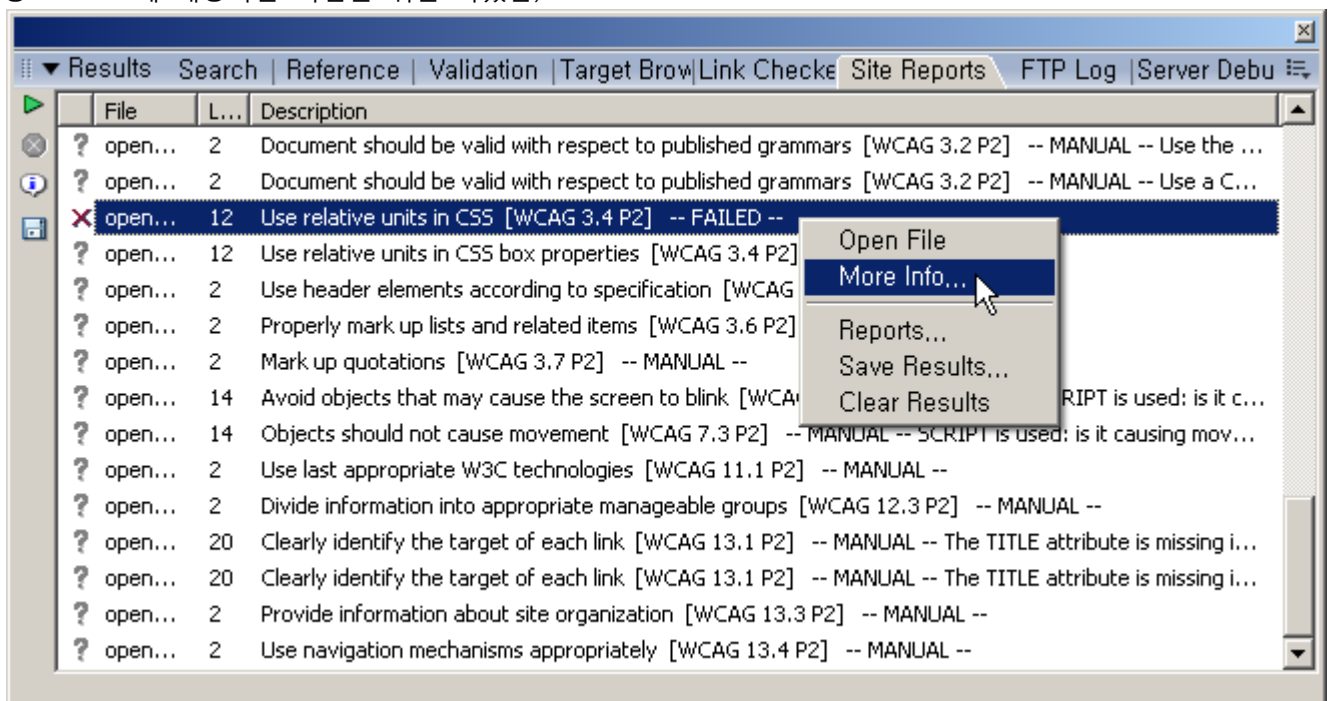
### WCAG 1.0 및 Section508 접근성 검사



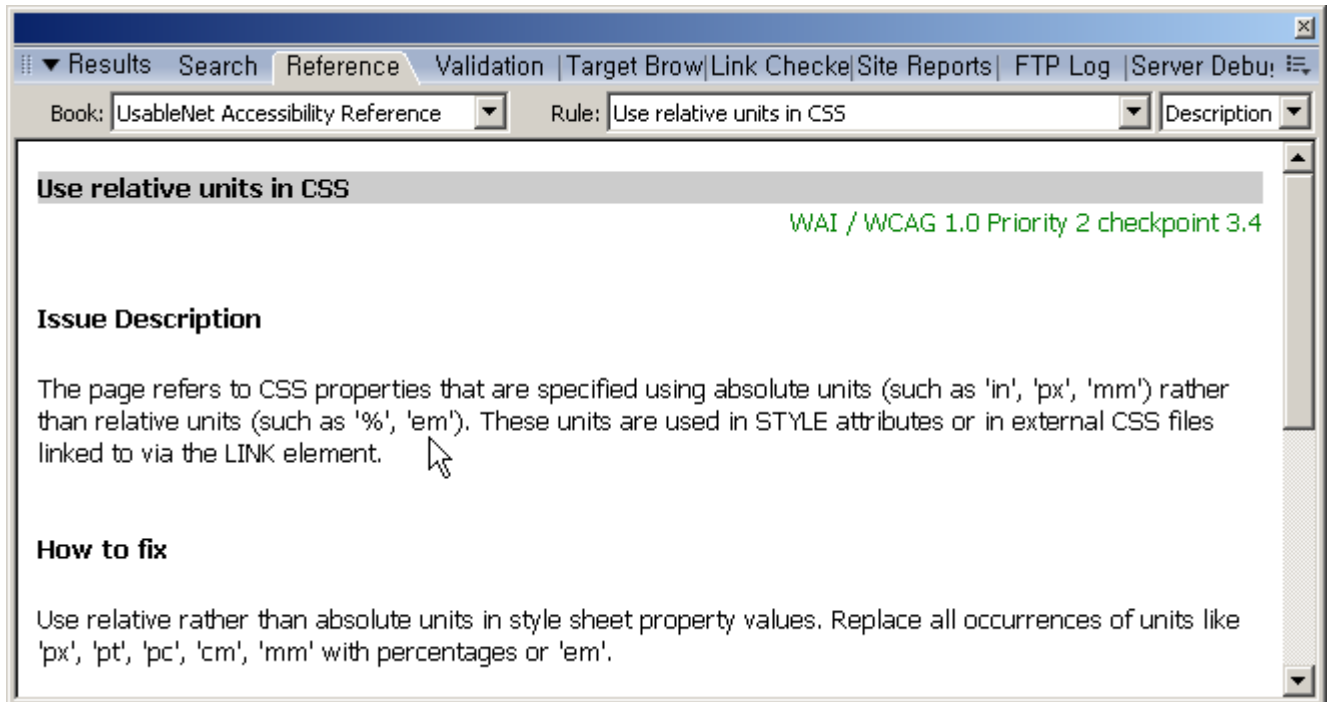
File > Check Page > Check Accessibility 메뉴를 선택하면 현재 열려있는 문서에 대한 웹 접근성 검사가 시작 됩니다.



현재문서 및 현재문서와 연결되어 있는 CSS 파일을 동시에 검사 합니다. 접근성 지침을 명백하게 위반한 코드가 발견되면 "X" 표시로 경고합니다. "?" 항목은 사람에 의한 수동검사가 필요한 항목으로서 --MANUAL-- 이라고 표시합니다. (이미지 설명 : WCAG Check Point 3.4 항목의 Priority 1 - 중요도 1- 에 해당하는 지침을 위반 하였음)



해당 항목 위에서 컨텍스트 메뉴(마우스 우측버튼)를 활성화 하고 More Info... 를 클릭하면 더욱 자세한 정보를 확인 할 수 있습니다.

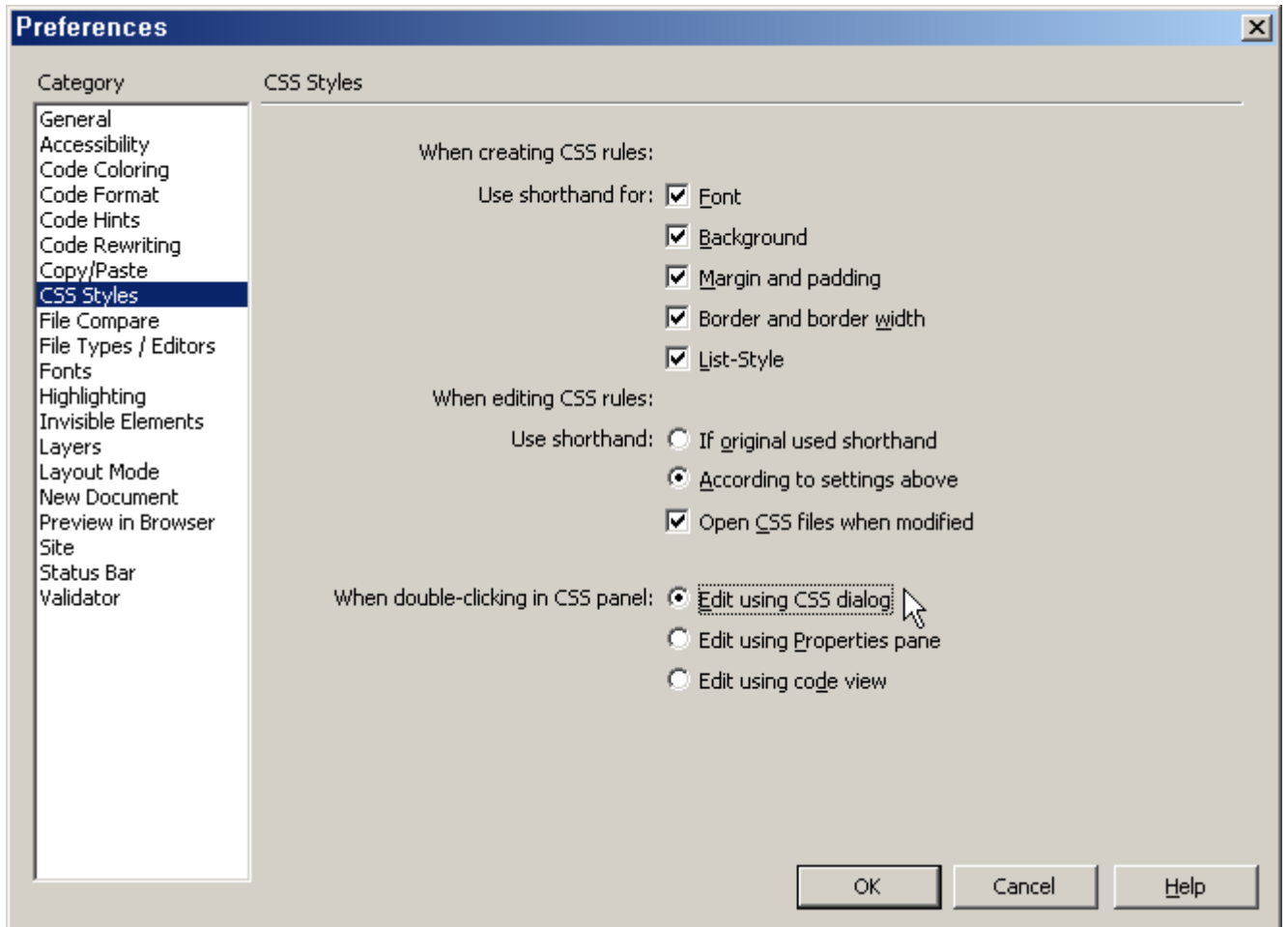


Results > Reference 탭으로 전환 되면서 자세한 설명을 보여줍니다. 드림위버 한국어 제품을 사용하더라도 이 항목은 영문으로 표기됩니다. 하지만 판독하기 쉬운 수준입니다.

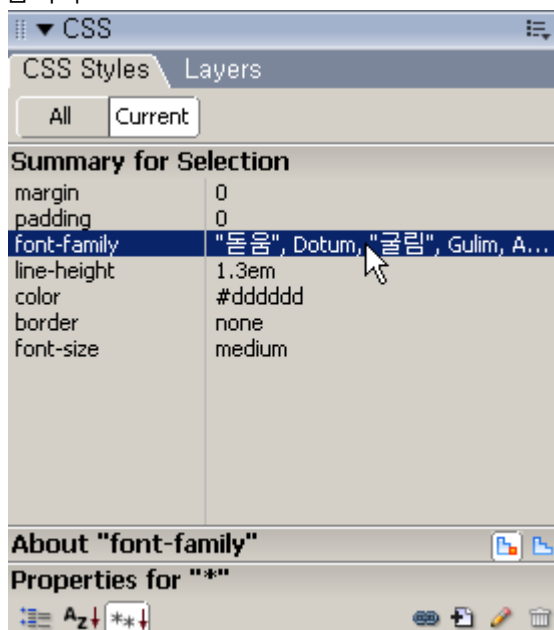
### CSS 대화상자 (CSS Dialog Box)

CSS 대화상자를 이용하여 코드를 관리하는 방법 입니다. CSS 숙련도가 비교적 낮은 초심자에게 권장되는 방법으로 CSS 코드의 순서를 제작자 의도대로 정리할 수 없는 단점이 존재합니다.

#### 대화상자를 이용한 CSS 관리

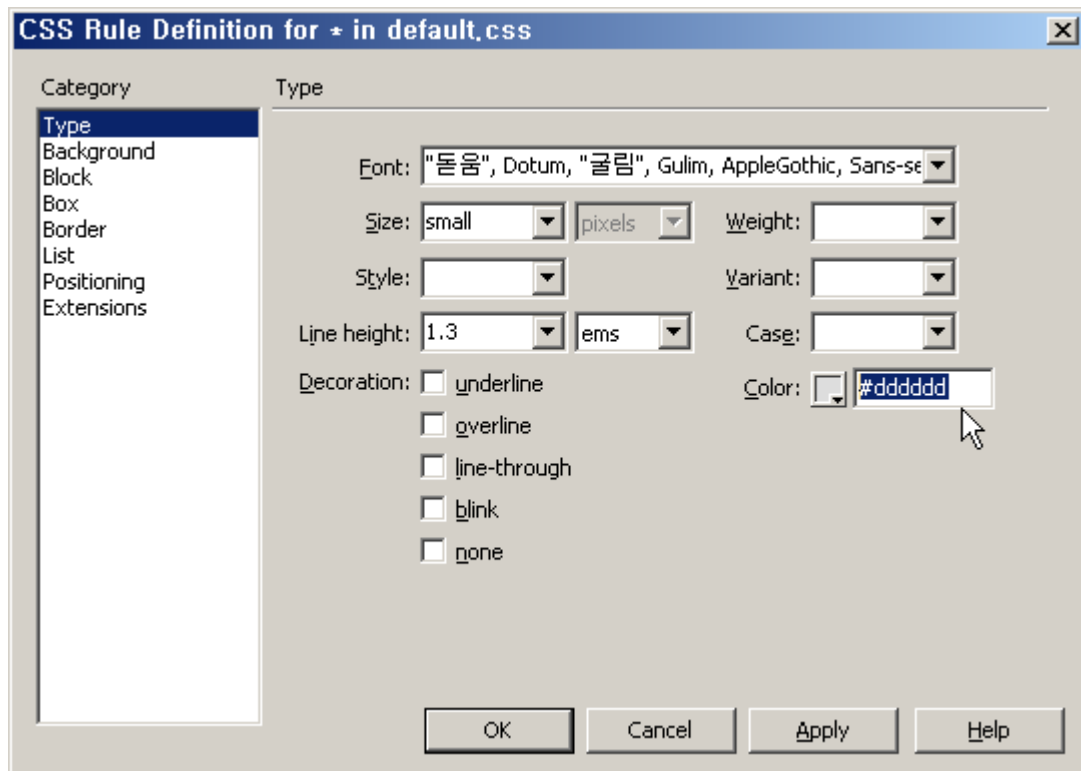


Preferences > CSS Styles 항목에서 When double-clicking in CSS panel(CSS 패널을 더블 클릭했을 때 다음 명령에 따름) 항목 가운데 Edit using CSS dialog(CSS 대화상자를 이용하여 편집)를 선택합니다.



현재 열려있는 페이지에서 편집하려고 하는 대상(엘리먼트, 이미지, 텍스트...)을 선택하면 CSS Styles 패널에는 위와같이 현재 선택물에 적용된 스타일만 정리해서 보여줍니다. 이것을 편집하려

면 수정하고자 하는 속성이나 값을 더블클릭 합니다.



CSS 대화상자 입니다. 원하는 항목을 찾아 수정하거나 새로운 속성을 추가할 수 있습니다.

```
* {  
margin:0;  
padding:0;  
color:#ddddd;  
font: small/1.3em "돋움", Dotum, "굴림", Gulim, AppleGothic, Sans-serif;  
}
```

한편 이렇게 대화상자를 이용하여 코딩하게 되면 CSS 코드의 줄바꿈을 마음대로 조정할 수 없게 됩니다. 대화상자를 이용하는 경우 드림위버 8 버전은 CSS의 속성 하나를 정의한 다음 개행(줄바꿈)을 시켜서 위와 같은 코드를 만들어 냅니다.

CSS 코드의 줄 바꿈 규칙은 개인의 선호 또는 조직의 규칙에 따르겠지만 만약 하나의 스타일을 한 줄에 코딩하는 것을 선호한다면 역시 대화상자를 이용한 CSS 편집은 불편한 방법이 됩니다.

## 환경 설정

범주

인라인 표시  
일괄 줄 바꿈  
레이아웃 모드  
보이지 않는 요소  
복사/붙여넣기  
브라우저에서 미리 보기  
사이트  
상태 표시줄  
새 문서  
액세스 가능성  
유효성 검사기  
코드 다시 작성  
코드 색상 표시  
**코드 포맷**  
코드 힌트  
파일 비교 / 편집기  
파일 유형  
AP 요소  
CSS 스타일

코드 포맷

들여쓰기(E): ☒ 사용: 1 탭

탭 크기(T): 4

자동 줄 바꿈(W): ☐ 다음 열마다: 76

행 분리 형식(L): CR LF(Windows)

기본 태그 대/소문자(C): <소문자>

기본 속성 대/소문자(A): 소문자="값"

대/소문자 무시: ☐ 태그(G) ☐ 속성(U)

TD 태그: ☐ TD 태그 내부에 행 분리를 포함하지 않음(B)

고급 서식: CSS(S)... 태그 라이브러리(R)...

도움말(H)

확인

취소

## CSS 소스 포맷 옵션

- ☒ 다음을 사용하여 속성 들여쓰기(E) 1 탭
- ☐ 각 속성을 별도의 행에(P)
- ☐ 여는 중괄호를 별도의 행에(B)
- ☐ 속성이 한 개 이상인 경우에만(M)
- ☐ 규칙의 모든 선택기를 동일한 행에(S)
- ☐ 규칙 사이의 빈 행(L)

미리 보기:

```
.myShortStyle { color: #000000; }
.myLongStyle1,
.myLongStyle2 { color: #FFFFFF; margin: 0px; padding: 0px; display: block; }
```

확인

취소

도움말

## CSS 소스 포맷 옵션

- ☒ 다음을 사용하여 속성 들여쓰기(E) 1 탭
- ☒ 각 속성을 별도의 행에(P)
- ☐ 여는 중괄호를 별도의 행에(B)
- ☐ 속성이 한 개 이상인 경우에만(M)
- ☐ 규칙의 모든 선택기를 동일한 행에(S)
- ☐ 규칙 사이의 빈 행(L)

미리 보기:

```
{.myShortStyle {
  color: #000000;
}
.myLongStyle1,
.myLongStyle2 {
  color: #FFFFFF;
  margin: 0px;
  padding: 0px;
  display: block;
}
```

확인

취소

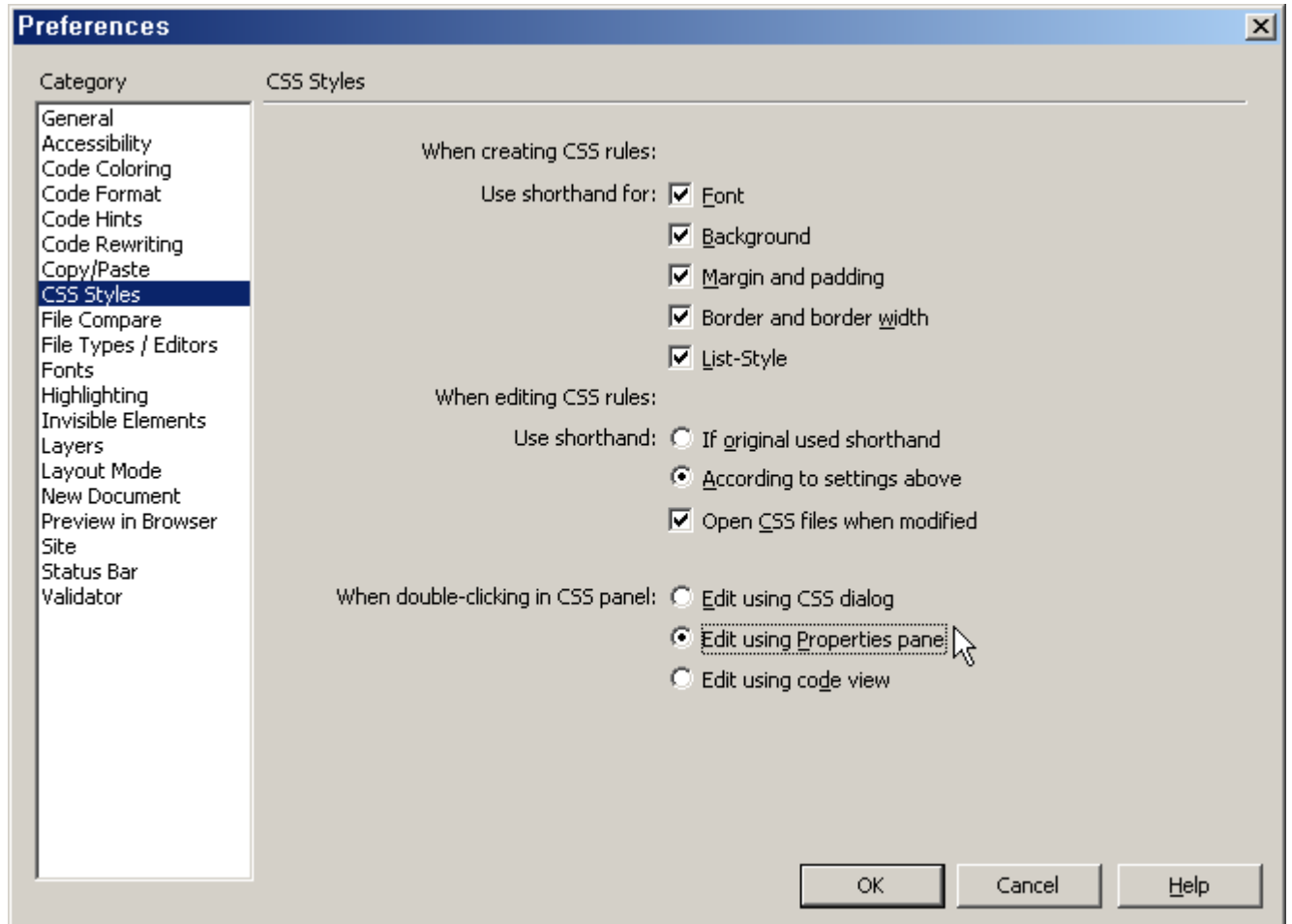
도움말



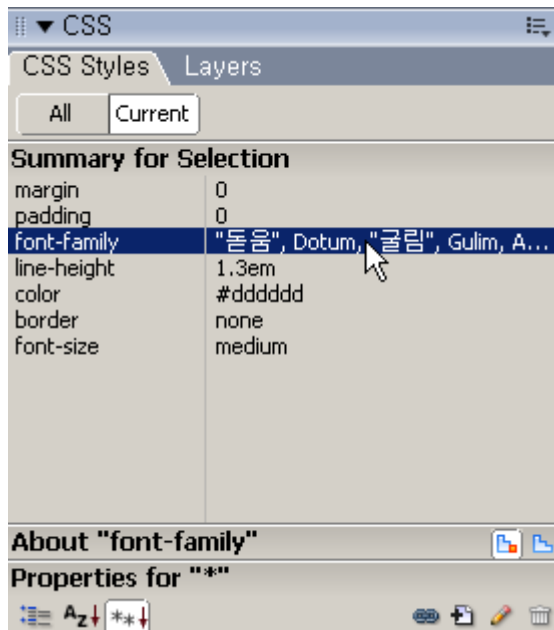
## CSS 속성패널 (CSS Properties Panel)

CSS 속성패널을 이용하여 코드를 관리하는 방법 입니다. CSS 숙련도가 비교적 중급에 해당하는 사용자에게 권장되는 방법으로서 이 방법 역시 CSS 코드의 순서를 제작자의 의도대로 정리할 수 없다

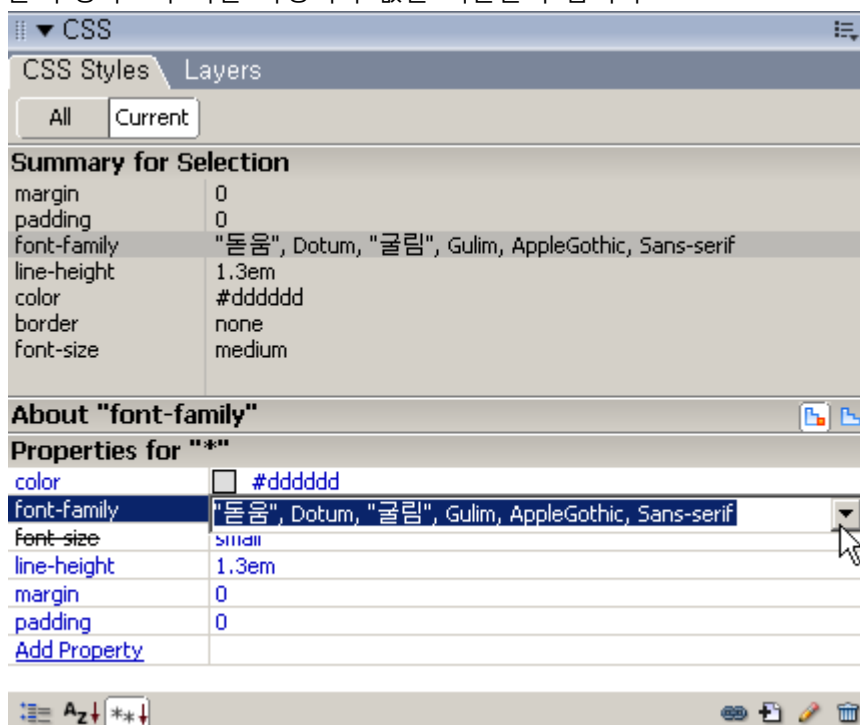
### 속성패널을 이용한 CSS 관리



Preferences > CSS Styles 항목에서 When double-clicking in CSS panel(CSS 패널을 더블 클릭했을 때 다음 명령에 따름) 항목 가운데 Edit using Properties pane (속성패널을 이용하여 편집)을 선택합니다.



현재 열려있는 페이지에서 편집하려고 하는 대상(엘리먼트, 이미지, 텍스트...)을 선택하면 CSS Styles 패널에는 위와같이 현재 선택물에 적용된 스타일만 정리해서 보여줍니다. 이것을 편집하려면 수정하고자 하는 속성이나 값을 더블클릭 합니다.



CSS Properties 패널 입니다. 원하는 항목을 수정하거나 새로운 속성을 추가할 수 있습니다.

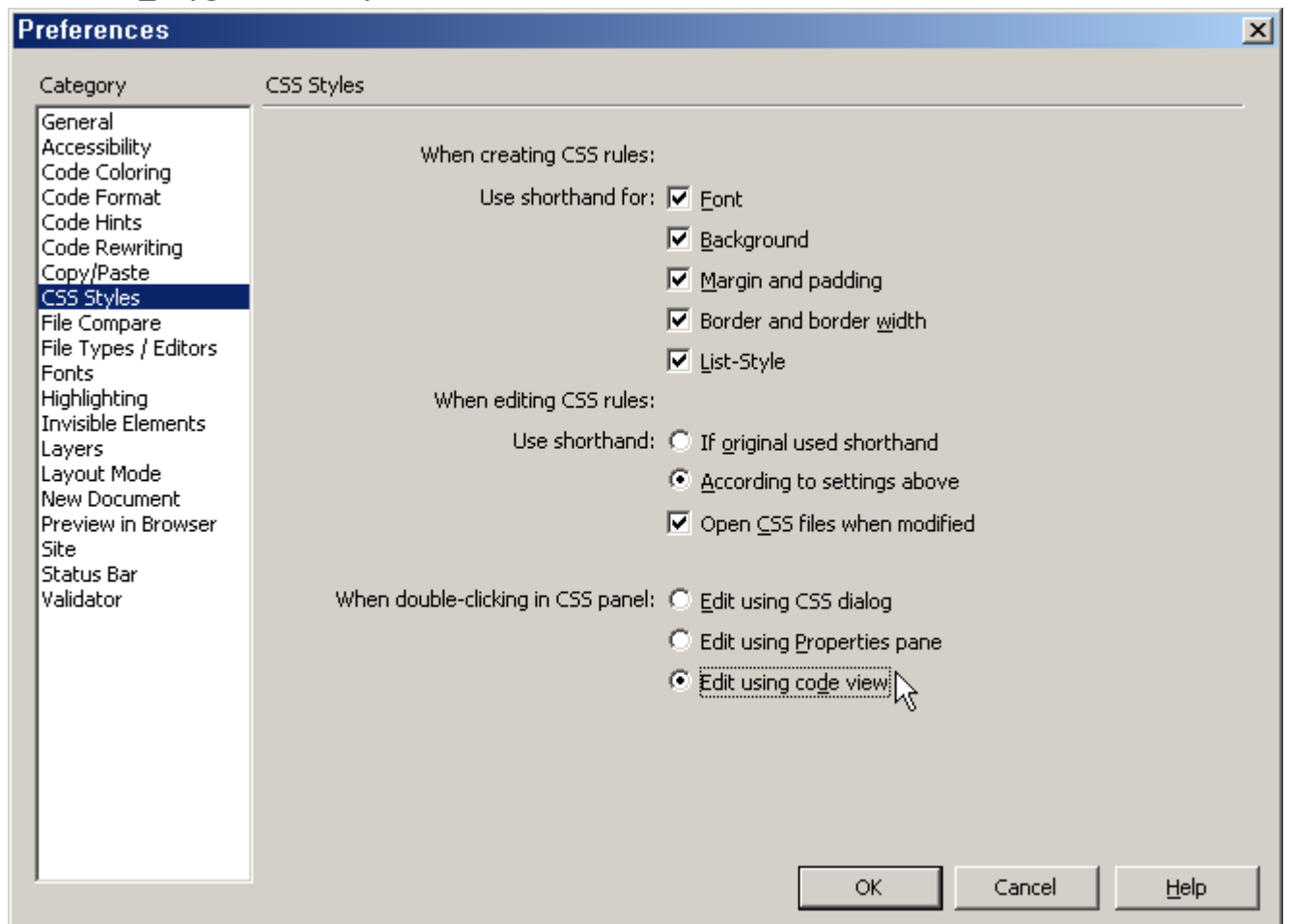
```
* {
margin:0;
padding:0;
color:#dddddd;
font: small/1.3em "돋움", Dotum, "굴림", Gulim, AppleGothic, Sans-serif;
}
```

한편 이렇게 Properties 패널을 이용하여 코딩하게 되면 CSS 코드의 줄바꿈을 마음대로 조정할 수 없게 됩니다. CSS 코드의 줄 바꿈 규칙은 개인의 선호 또는 조직의 규칙에 따르겠지만 만약 하나의 스타일을 한 줄에 코딩하는 것을 선호한다면 역시 Properties 패널을 이용한 CSS 편집은 불편한 방법이 됩니다.

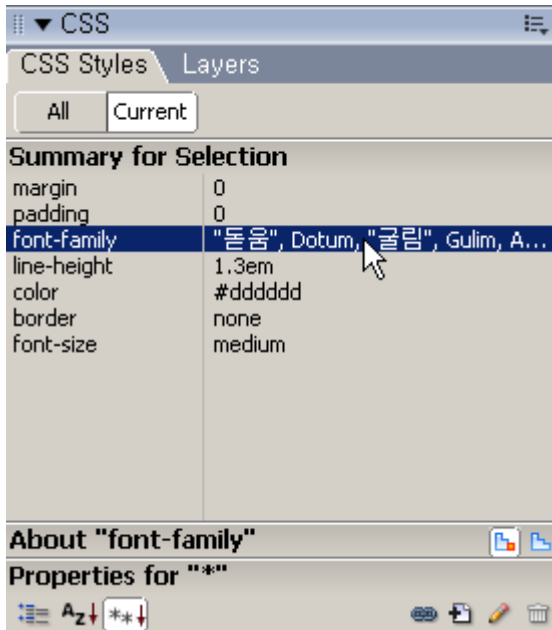
### CSS 코드뷰 (CSS Code View)

Code View를 이용하여 CSS 코드를 관리하는 방법 입니다. CSS 숙련도가 비교적 고급에 해당하는 사용자에게 권장되는 방법으로 이 방법은 CSS 코드의 순서와 줄바꿈 규칙을 사용자 의도대로 정리할 수 있습니다. 또한 복잡한 선택자 사용 규칙등 여러줄의 코드를 한 눈에 볼 수 있기 때문에 숙련자는 매우 빠르게 코드를 생성하거나 관리 할 수 있게 됩니다.

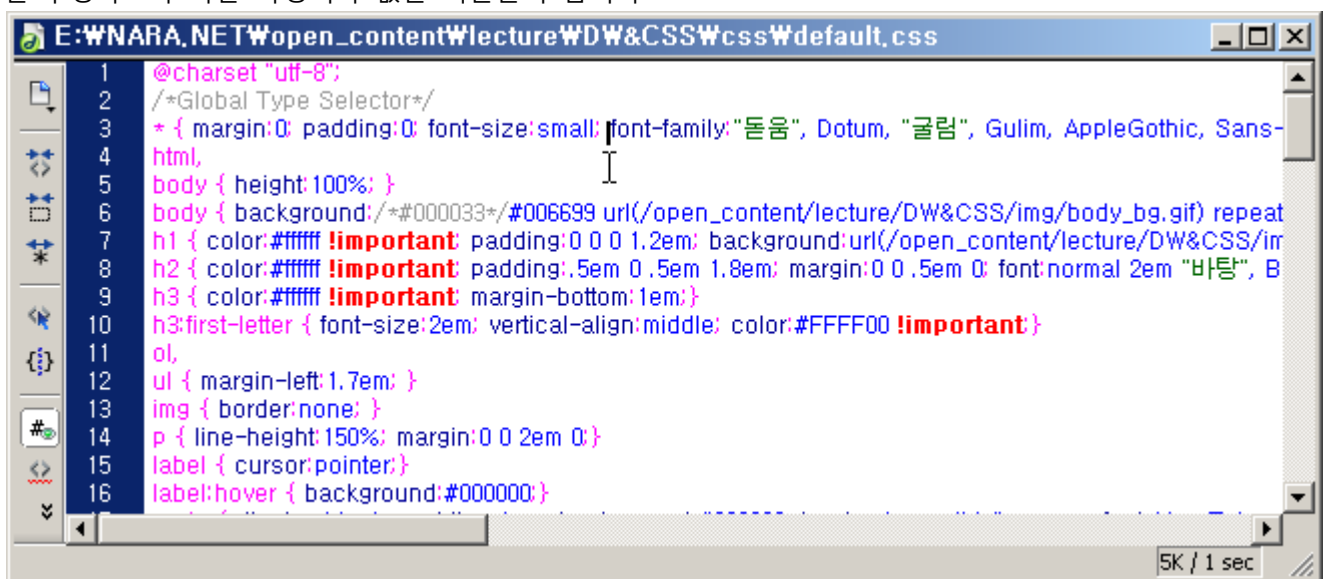
### Code View를 이용한 CSS 관리



Preferences > CSS Styles 항목에서 When double-clicking in CSS panel(CSS 패널을 더블 클릭했을 때 다음 명령에 따름) 항목 가운데 Edit using Code view(Code view를 이용하여 편집)을 선택 합니다.



현재 열려있는 페이지에서 편집하려고 하는 대상(엘리먼트, 이미지, 텍스트...)을 선택하면 CSS Styles 패널에는 위와같이 현재 선택물에 적용된 스타일만 정리해서 보여줍니다. 이것을 편집하려면 수정하고자 하는 속성이나 값을 더블클릭 합니다.

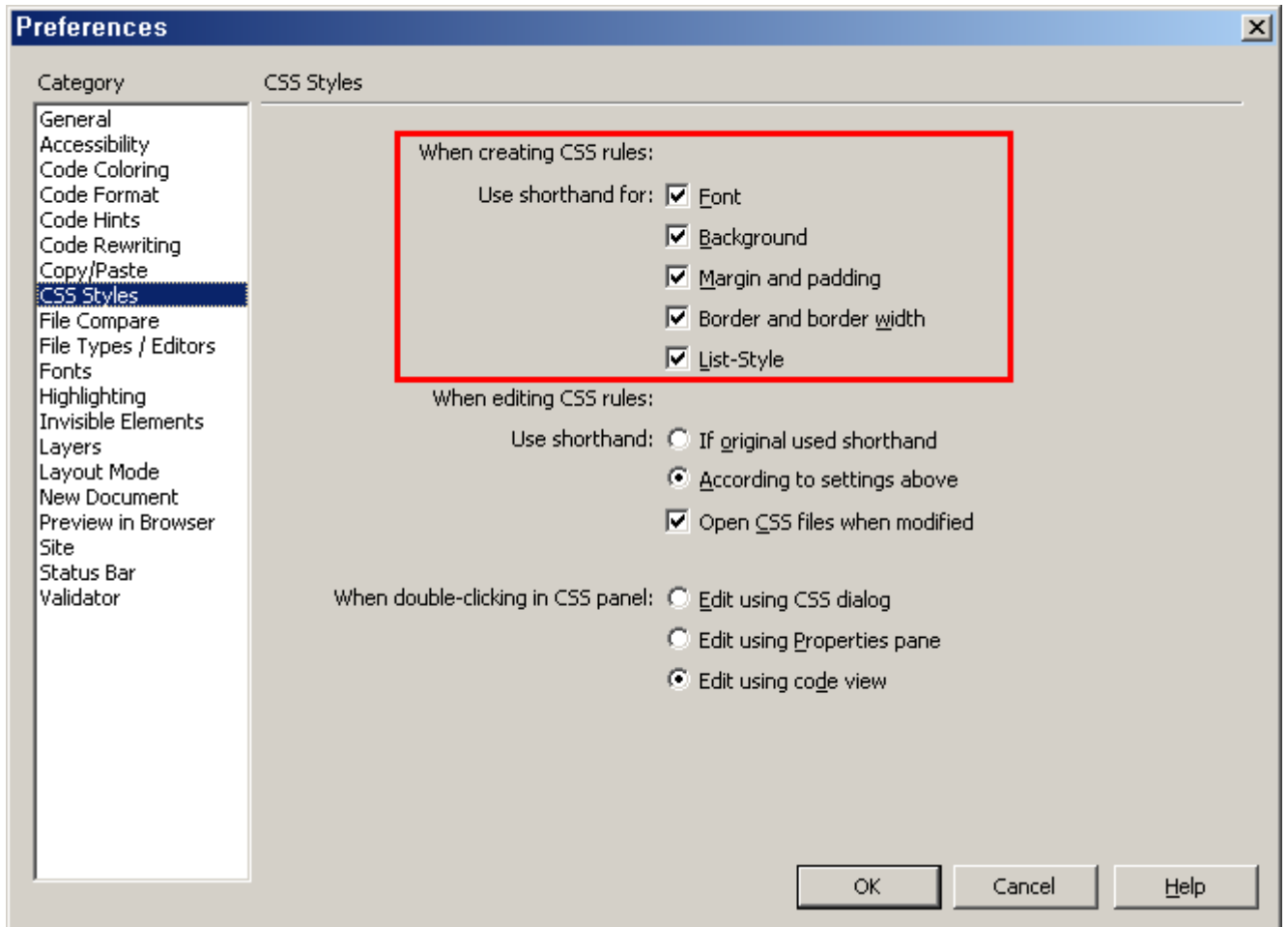


Code View 에 CSS 문서가 열리면서 해당 코드를 보여줍니다. CSS 코드를 직접 보면서 원하는 항목을 수정하거나 새로운 속성을 추가할 수 있습니다. 코드의 순서와 줄바꿈을 사용자가 직접 관리 합니다.

### CSS 대표속성 (CSS Shorthand Properties)

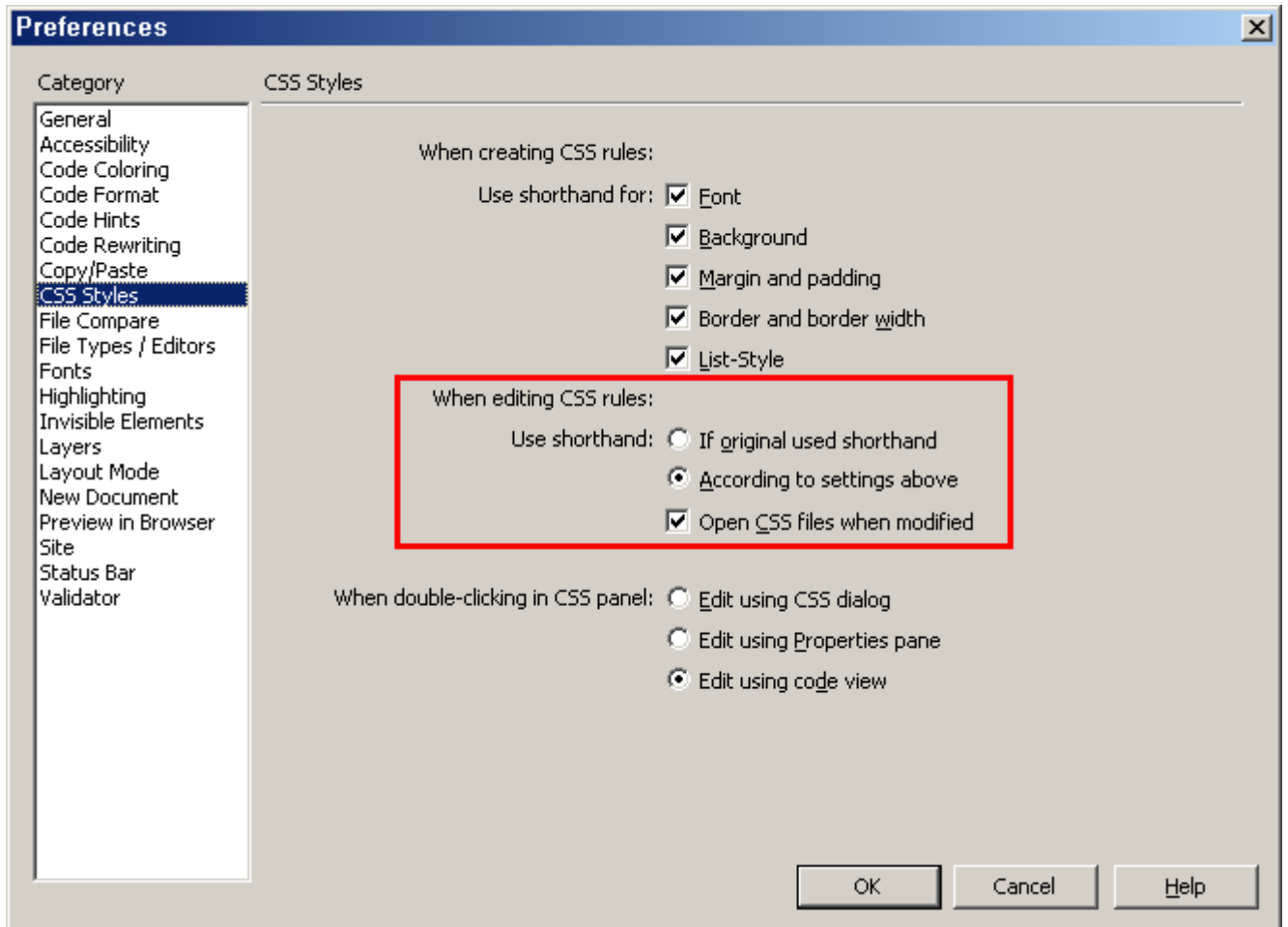
CSS 대표속성을 이용하면 CSS 코드의 양을 획기적으로 줄일 수 있습니다. 또한 코드가 간결해지기 때문에 코드를 생성할 때 뿐만 아니라 유지보수시에도 이롭습니다. 대표속성으로 코딩이 가능한 CSS 속성은 'font, background, margin, padding, border, list-style' 입니다.

### CSS 대표속성 사용을 위한 환경설정(Ctrl+U)



Preferences > CSS Styles 항목에서 When creating CSS rules(새 CSS 규칙을 생성할 때) 항목 가운데 Use shorthand for(대표속성 사용)의 하위 항목(Font, Background, Margin and Padding, Border and border width, List-Style)을 모두 선택하면 CSS 대화상자 또는 CSS 속성패널을 이용할 때 대표속성으로 코딩 됩니다. Code View를 이용하여 CSS를 관리하는 사용자에게는 이 설정이 무의미합니다.

- When creating CSS rules: (CSS 규칙을 생성할 때)
  - Use shorthand for: (~에 대하여 대표속성을 사용함)
    - Font (폰트)
    - Background (배경)
    - Margin and Padding (마진 & 패딩)
    - Border and border width (보더 & 보더 두께)
    - List-Style (목록)



Preferences > CSS Styles 항목에서 When editing CSS rules(CSS 규칙을 편집 할 때) 항목 가운데 Use shorthand(대표속성 사용)의 하위 항목을 선택하는 방법에 따라 사용자 정의 설정에 따라 것인지 기존 코드의 규칙에 따라 것인지 설정할 수 있습니다. 이 규칙 역시 Code View를 이용하여 CSS를 관리하는 사용자에게는 무의미 합니다.

- When editing CSS rules: (CSS 규칙을 편집할 때)
  - Use shorthand: (대표속성을 사용함)
    - If original used shorthand (기존의 코드가 대표속성으로 코딩되어 있다면 대표속성으로 코딩)
    - According to setting above (사용자의 CSS 생성규칙에 따라 기존의 코드는 대표속성으로 덮어쓰기 됨)
    - Open CSS files when modified (코드가 수정될 때 CSS 파일 열기)

#### » 보충설명 : CSS 대표속성 규칙과 예제 코드

##### Font의 대표속성 사용규칙

Font의 대표속성 사용규칙은 다른 속성에 비하여 조금 복잡한 규칙을 지니고 있기에 조금 자세히 설명하겠습니다.

1. 'font-size 와 font-family' 는 반드시 정의 되어야 한다.
2. 'font-size 와 font-family' 는 반드시 그 순서를 지켜야 하며 'font-size' 가 먼저 정의되어야 한다. (잘된 예: 'small dotum', 잘못된 예:'dotum small')

3. 'font-size 와 font-family' 는 다른 속성보다 항상 나중에 정의되어야 한다.(잘된 예:'bold small dotum', 잘못된 예: 'small dotum bold')
4. 'font-size 와 font-family' 이외의 속성은 없어도 무방하며'font-size 와 font-family' 보다 먼저 정의되기만 한다면 나머지 속성들은 더이상 순서에 구애받지 않는다.
5. 'font-style, font-weight' 따위를 정의하지 않으면 마크업에서 기본적으로'italic, bold' 체로 표현되는 서체에'normal' 스타일이 적용된다. (예:'font-style' 을 정의하지 않으면'em, address' 따위의'italic' 속성이 제거되고'font-weight' 를 정의하지 않으면'h1~h6, th' 따위가 기본적으로 지니고 있는 'bold' 속성이 제거됨.)

#ex1 {font: small dotum;}

#ex2 {font: italic small-caps bold small dotum;}

### **Background**

#ex {background: #CCC url(dot.png) no-repeat 50% 50% fixed;}

### **Margin**

#ex1 {margin: 1em 2em 1em 2em;} = #ex1 {margin: 1em 2em;}

### **Padding**

#ex2 {padding: 1em 2em 1em 2em;} = #ex2 {padding: 1em 2em;}

### **Border**

#ex1 {border: 1px solid #CCC;}

#ex2 {border: 1px solid #CCC; border-bottom:none;}

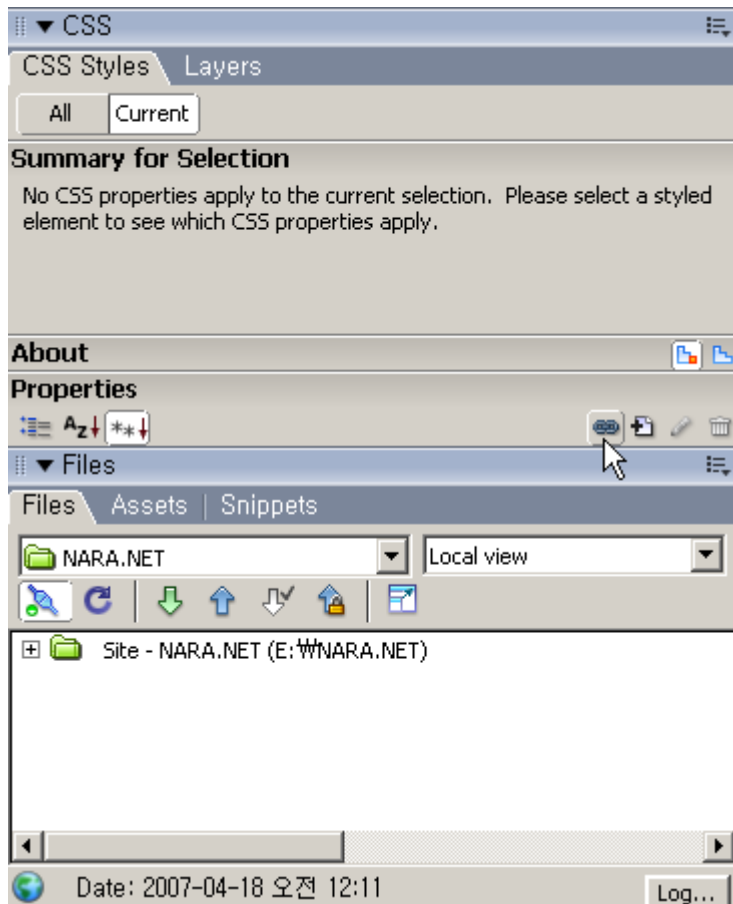
### **List-Style**

#ex {list-style:url(bullet.png) inside;}

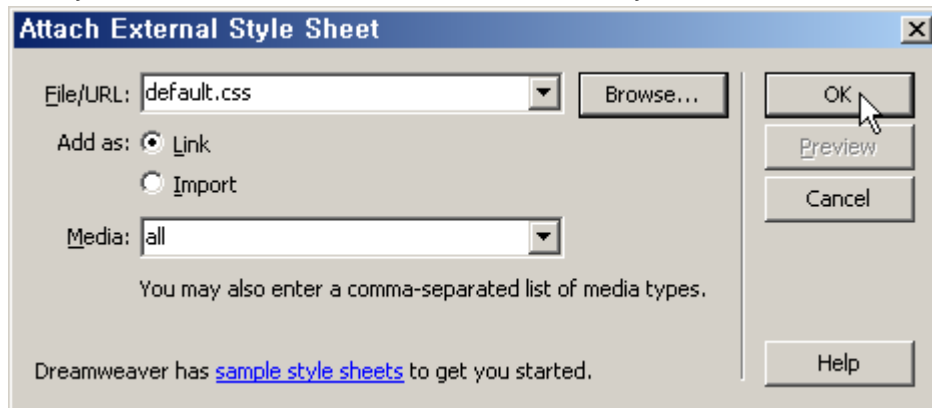
## **CSS 링크 & 가져오기, 미디어 타입 설정 (CSS Link & Import, Media Type)**

Link 방식과 Import 방식은 CSS를 불러온다는 점에서 같지만 Link 방식은 HTML 문서에서만 사용할 수 있고 Import 방식은 HTML 문서와 CSS 문서에서 모두 사용이 가능하다는 점이 다릅니다.

### **HTML 문서에서 CSS Link 하기**



Css Styles 패널의 우측 하단에 위치한  Attach Style Sheet 버튼을 누릅니다.



default.css 라는 파일을 External CSS Link 방식으로 불러옵니다. Add as 항목에서 Link 를 선택합니다. Media 항목을 선택하지 않는 경우 기본값은 all 이 되는데 현재는 all 이라는 Media Type 을 일부러 지정하였습니다. Media Type의 종류에는 다음과 같은 것들이 있습니다.

- all (모든 출력 장치)
- aural (음성 출력)
- braille (점자 출력)
- handheld (포켓, 모바일)
- print (인쇄)
- projection (투사 장치)
- screen (스크린, 모니터)



- tty (전신 타자기, Tele Type Writer)
- tv (텔레비전, Television)

위와 같이 대화상자를 이용하여 외부 CSS를 링크하게 되면 드림위버는 아래와 같은 코드를 생성하게 됩니다.

```
<link href="default.css" rel="stylesheet" type="text/css" media="all" />
```

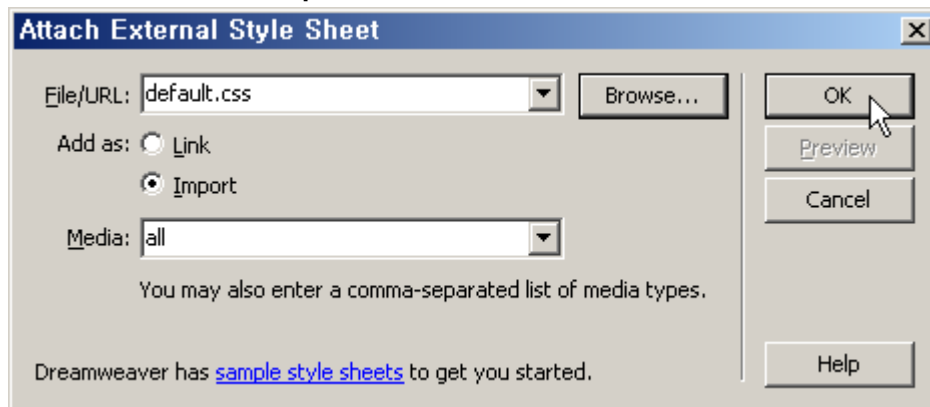
만약 Media Type 으로서 print 를 지정하였다면 해당 CSS 규칙은 인쇄할 때에만 적용 됩니다. 지금 여러분이 보고 계신 이 문서는 HTML 문서이지만 모니터 화면(screen)에 출력되는 스타일과 인쇄 장치(print)를 이용하여 출력할 때의 스타일이 다르게 표현 됩니다. 현재의 문서를 HTML 버전으로 보고 계시다면 인쇄 미리보기 버튼을 눌러서 출력 결과를 확인해 보시기 바랍니다. 웹 브라우징 장치에서 출력되는 스타일과 인쇄장치를 이용하여 출력되는 스타일이 다르게 보일 것입니다. 현재의 문서에는 다음과 같은 두 개의 CSS 파일이 Link 방식으로 연결되어 있으며 하나의 파일은 모든 장치를 위한 CSS이고 다른 하나는 인쇄장치를 위한 CSS 입니다.

```
<link href="default.css" rel="stylesheet" type="text/css" media="all" />
```

```
<link href="print.css" rel="stylesheet" type="text/css" media="print" />
```

인쇄를 위하여 별도의 HTML 파일을 제작하는 것은 낭비 입니다.

#### HTML 문서에서 CSS Import 하기



HTML 문서에서 외부 CSS를 Import 하는 방법은 Link 하는 방법과 동일합니다. Add as 항목에서 Import 를 선택하면 됩니다.

```
<style type="text/css" media="all">
```

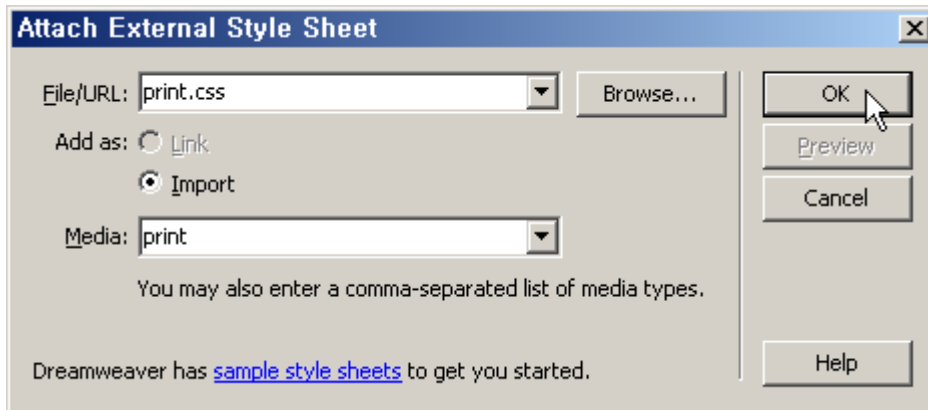
```
<!--
```

```
@import url("default.css");
```

```
-->
```

```
</style>
```

#### CSS 문서에서 CSS Import 하기



CSS 문서에서 또 다른 외부 CSS를 Import 하는 방법 또한 Link 하는 방법과 동일합니다. Add as 항목에서 Import 를 선택하면 됩니다. CSS 문서에서 또 다른 외부 CSS를 Import 하고자 할 때 Add as: Link 항목은 비활성화 됩니다.

```
@charset "utf-8";
```

```
@import url("print.css") print;
```

```
* { ... }
```

```
body { ... }
```

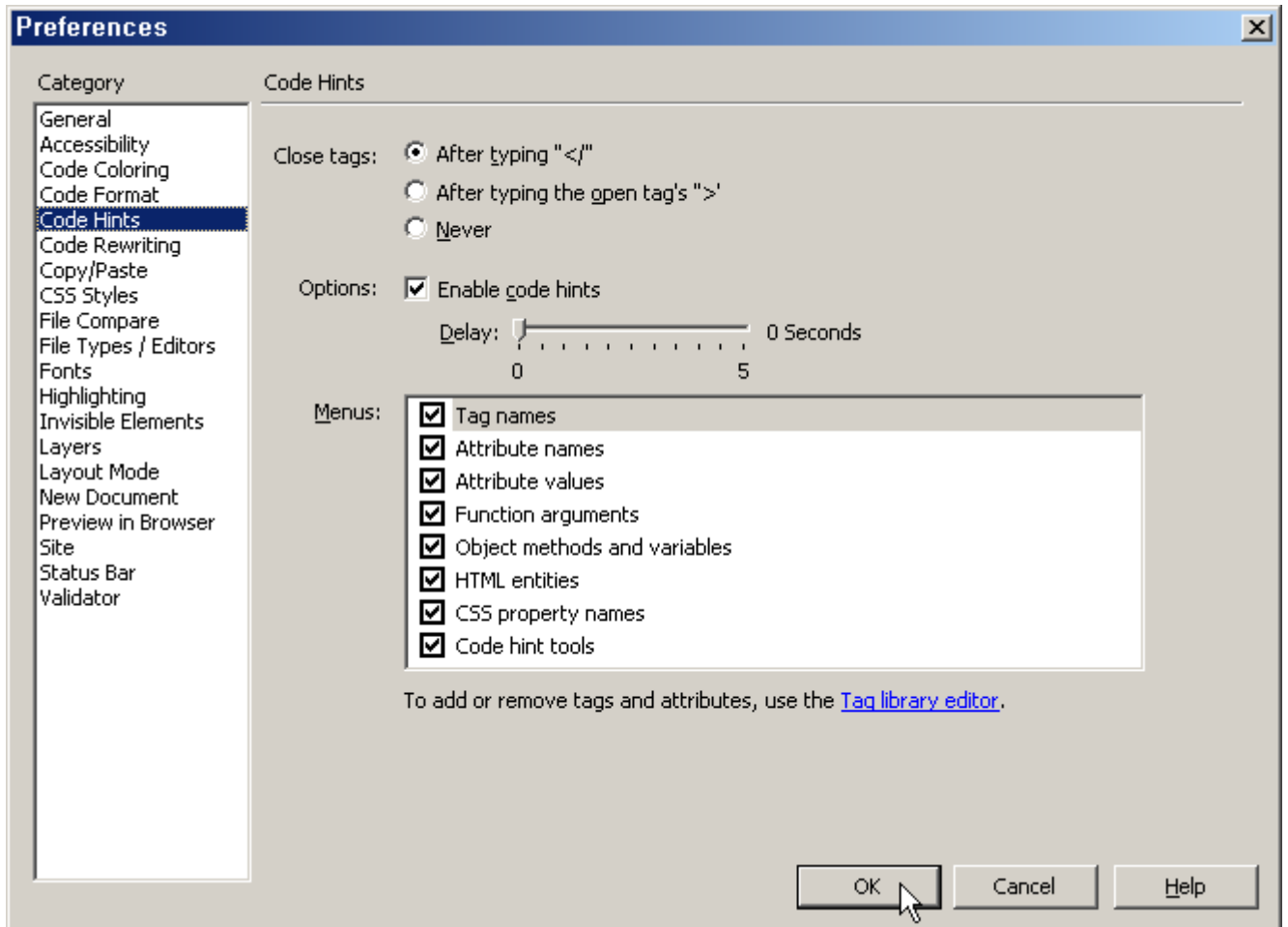
단, CSS 문서에서 또 다른 CSS를 Import 하는데에는 한 가지 규칙이 있습니다. 다른 어떤 CSS 정의 규칙보다 먼저 나와야 한다는 점 입니다. 따라서 드림위버는 Import 규칙을 적용하면서 CSS 문서의 가장 첫 줄에 Import 규칙을 선언하게 됩니다. 만약 코드 첫 줄에 문자셋(@charset "utf-8";) 규칙이 포함되어 있다면 Import 규칙은 두 번째 줄에 선언 됩니다.

Internet Explorer는 현재 활성화된 버전(7)에 이르기까지 Import 규칙에 적용된 Media Type을 인식하지 못하는 버그가 있습니다. 따라서 Media Type을 지정하려면 Link 규칙을 사용하는 것이 좋고 Import 규칙을 사용할 때에는 Media Type을 지정하지 않아야 한다는 것을 염두해 두어야 합니다. Link 규칙에 적용된 Media Type은 대부분의 브라우저에서 문제가 없습니다.

### 코드 자동 완성 (Code Hints)

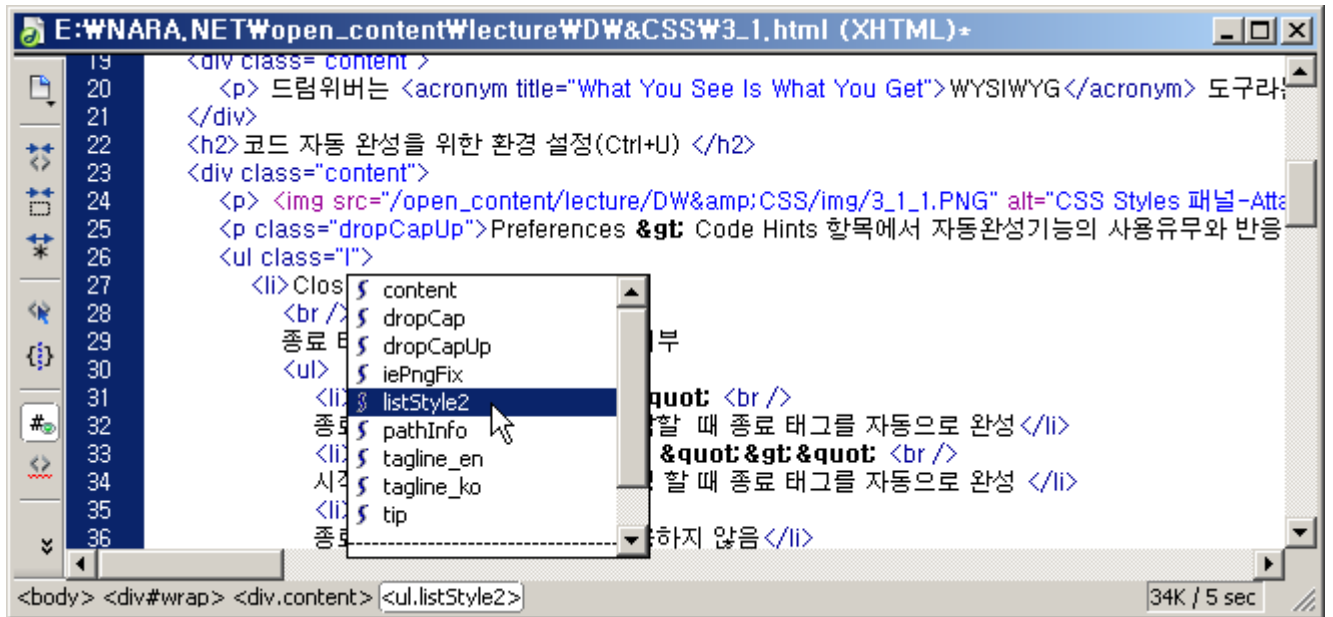
드림위버는 WYSIWYG 도구라는 장점 이외에도 Hand Coding 사용자를 위하여 다양한 편의를 제공하고 있으며 Code Hints 도구는 HTML, CSS 코드의 자동완성기능을 지원하여 코드의 정확도와 생산성을 향상시켜 줍니다.

### 코드 자동 완성을 위한 환경 설정(Ctrl+U)



Preferences > Code Hints 항목에서 자동완성기능의 사용유무와 반응시간 및 세부 지원 항목들을 선택할 수 있습니다. 드림위버 설치 직후 기본값은 모든항목에 대하여 사용함으로 설정되어 있습니다.

- Close tags:
  - 종료 태그의 자동완성 기능 사용 여부
    - After typing "</"
      - 종료 태그 꺾쇠의 타이핑을 시작할 때 종료 태그를 자동으로 완성
    - After typing the open tag's ">"
      - 시작 태그의 닫기 꺾쇠를 타이핑 할 때 종료 태그를 자동으로 완성
    - Never
      - 종료 태그 자동완성 기능을 사용하지 않음
- Options:
  - 자동완성 기능 사용여부 설정
    - Enable code hints
      - 체크하면 코드 힌트 도구를 사용함
- Menus:
  - 어떤 항목들을 자동완성 시킬 것인지 체크함(모든 항목의 체크가 권장됨)



사용자가 타이핑 하는 글자에 따라 관련 '엘리먼트, 에트리뷰트, 에트리뷰트 값, ID, Class' 등이 나타나게 되며 사용자는 그것을 마우스로 클릭하거나 Enter 키를 입력하여 완성할 수 있습니다. 상기 이미지는 사용자가 Class 이름의 값을 자동으로 완성시키는 장면 입니다.

사용자가 자동완성기능을 사용하는 방법은 다음과 같습니다.

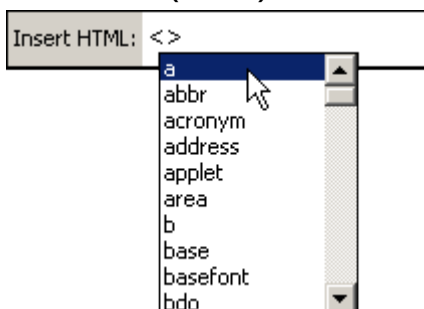
- 괄호 또는 꺾쇠 입력 : 처음으로 코드를 생성할 때.
- Space : 코드를 추가하기 위하여 띄어쓰기를 시도 할 때.
- Ctrl+Space : Code Hints 도구가 나타나지 않을 때 강제로 이것을 띄움.

Internet Explorer 전용 태그와 속성에 대하여는 자동완성 기능을 지원하지 않으며 W3C 권장 표준 코드만을 지원하고 있습니다.

## 빠른 태그 편집 (Quick Tag Editor)

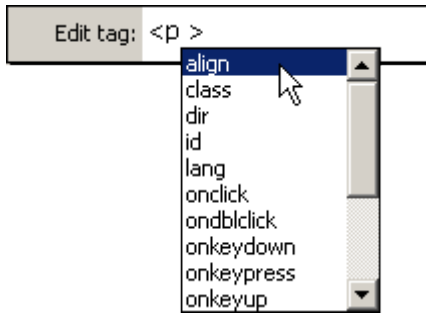
Quick Tag Editor 는 Design View 와 Code View 에서 '삽입(Insert HTML), 편집(Edit Tag), 감싸기(Wrap Tag)' 할 때 매우 유용한 도구 입니다. 빠른 태그 편집기는 단축키 'Ctrl+T' 를 이용하며 이것을 연거푸 눌러서 입력모드를 전환(삽입-편집-감싸기) 시킵니다.

### Insert HTML (Ctrl+T)



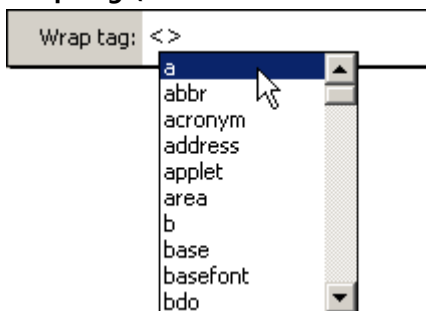
아무 엘리먼트도 선택되지 않은 상태에서 단축키 'Ctrl+T' 를 한 번 누르면 Quick Tag Editor 는 Insert HTML 모드로 열립니다. 새로운 HTML 엘리먼트를 삽입할 수 있습니다.

### Edit Tag (Ctrl+T & Ctrl+T)



아무 엘리먼트도 선택되지 않은 상태에서 단축키 'Ctrl+T' 를 연거푸 두 번 누르면 Quick Tag Editor 는 Edit HTML 모드로 열립니다. 현재 작업 커서가 위치한 곳의 부모 엘리먼트를 편집할 수 있습니다.

#### Wrap Tag (Ctrl+T & Ctrl+T & Ctrl+T)



아무 엘리먼트도 선택되지 않은 상태에서 단축키 'Ctrl+T' 를 연거푸 세 번 누르면 Quick Tag Editor 는 Wrap Tag 모드로 열립니다. 현재 작업 커서가 위치한 곳의 부모 엘리먼트를 다른 엘리먼트로 다시 감싸기 할 수 있습니다.

이 기능은 Design View 에서 보다 유용하며 Code View 에서는 태그 감싸기(Wrap Tag) 작업을 시도할 때 매우 효과적 입니다. 코드 뷰에서 이 기능을 사용하려면 해당 엘리먼트를 선택한 상태여야 합니다.

Wrap Tag 기능은 Code View 에서 부모/자식 엘리먼트를 선택할 때 사용하는 단축키 'Ctrl+[, Ctrl+]' 와 함께 사용하면 궁합이 잘 맞습니다.

- Ctrl+[  
부모 엘리먼트를 선택
- Ctrl+]  
자식 엘리먼트를 선택

#### 단축키의 확장 (Keyboard Shortcuts)

코드를 생성하고 편집할 때 단축키를 많이 사용할 수록 작업속도는 획기적으로 향상됩니다. 자주 사용되는 유용한 단축키를 소개하고 자신만의 단축키를 만들어 사용하는 방법을 설명합니다.

#### 유용한 단축키 목록

##### 문서

##### 새 문서 열기 :

Ctrl+N

##### 문서 닫기 :

Ctrl+W

**다른이름으로 저장 :**

Ctrl+Shift+S

**검사**

**마크업 유효성 검사 :**

Shift+F6

**링크 유효성 검사 :**

Shift+F8

**편집**

**부모 엘리먼트 선택 :**

Ctrl+[

**자식 엘리먼트 선택 :**

Ctrl+]

**코드뷰 행 이동 :**

Ctrl+G

**코드 힌트 보기 :**

Ctrl+Space

**코드 짝 맞추어 접어두기 :**

Ctrl+Shift+J

**접힌 코드 펼치기 :**

Ctrl+Shift+E

**환경설정 :**

Ctrl+U

**보기**

**코드뷰/디자인뷰 전환 :**

Ctrl+`

**모든 시각도구 숨김 :**

Ctrl+Shift+I

**패널 숨김 :**

F4

**삽입**

**태그 :**

Ctrl+E

**이미지<img> :**

Ctrl+Alt+I

**플래시<object> :**

Ctrl+Alt+F

**테이블<table> :**

Ctrl+Alt+T

**앵커<a> :**

Ctrl+Alt+A

**줄바꿈<br> :**

Shift+Enter

**자간 띄움(&nbsp;):**

Ctrl+Shift+Space

**변경**

**페이지 속성 :**

Ctrl+J

**빠른 태그 편집 :**

Ctrl+T

**셀 합치기 :**

Ctrl+Alt+M

**셀 나누기 :**

Ctrl+Alt+S

**행 삽입 :**

Ctrl+M

**행 삭제 :**

Ctrl+Shift+M

**열 삽입 :**

Ctrl+Shift+A

**문맥**

**문장 :**

Ctrl+Shift+P

**제목 :**

Ctrl+(1~6)

**strong :**

Ctrl+B

**em :**

Ctrl+I

**영문 철자 검사 :**

Shift+F7

**사이트**

**내려받기 :**

Ctrl+Shift+D

**올리기 :**

Ctrl+Shift+U

**링크 유효성 검사 :**

Ctrl+F8

패널

결과 패널 :

F7

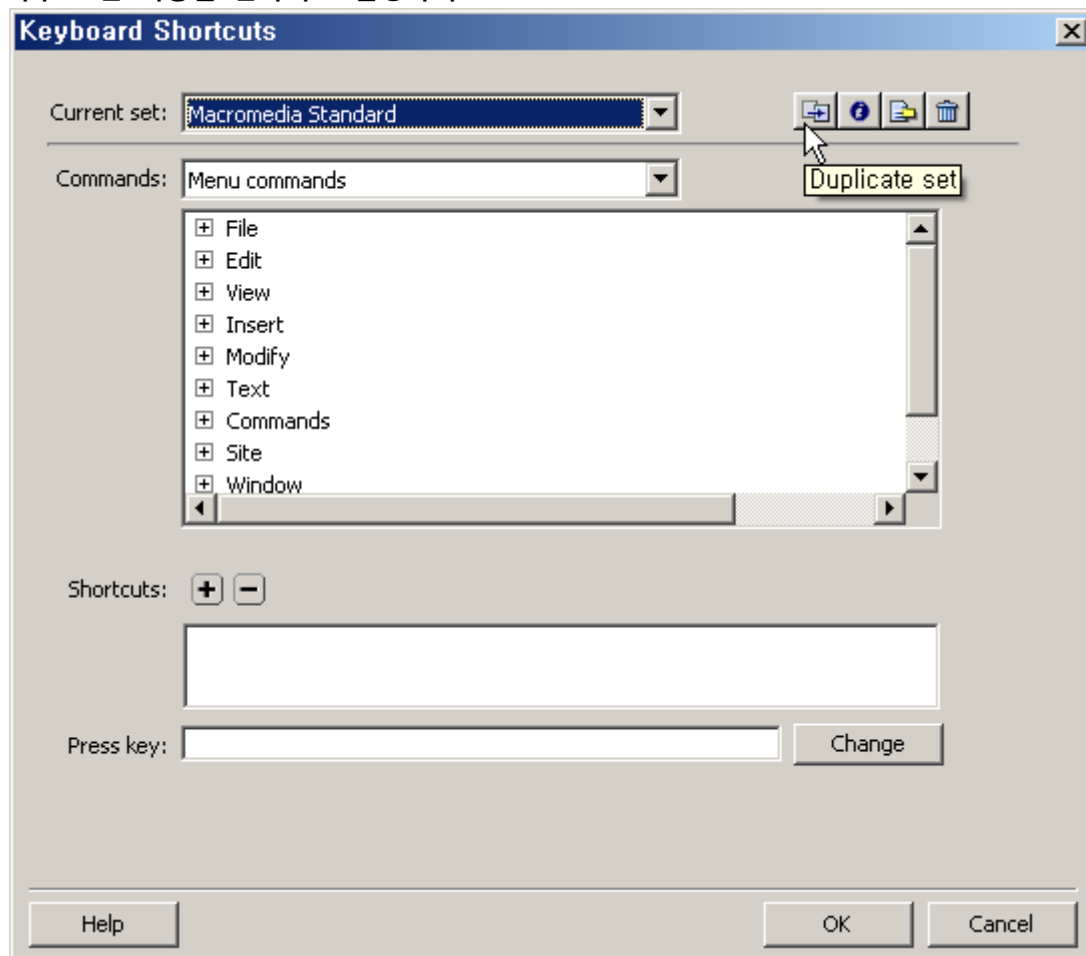
코드 창 :

F10

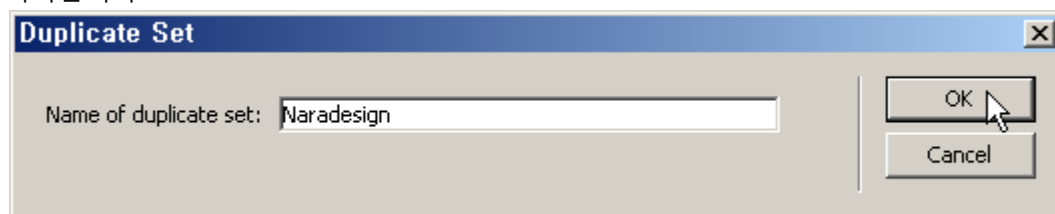
패널 모두 숨김 :

F4

자주 쓰는 기능을 단축키로 설정하기

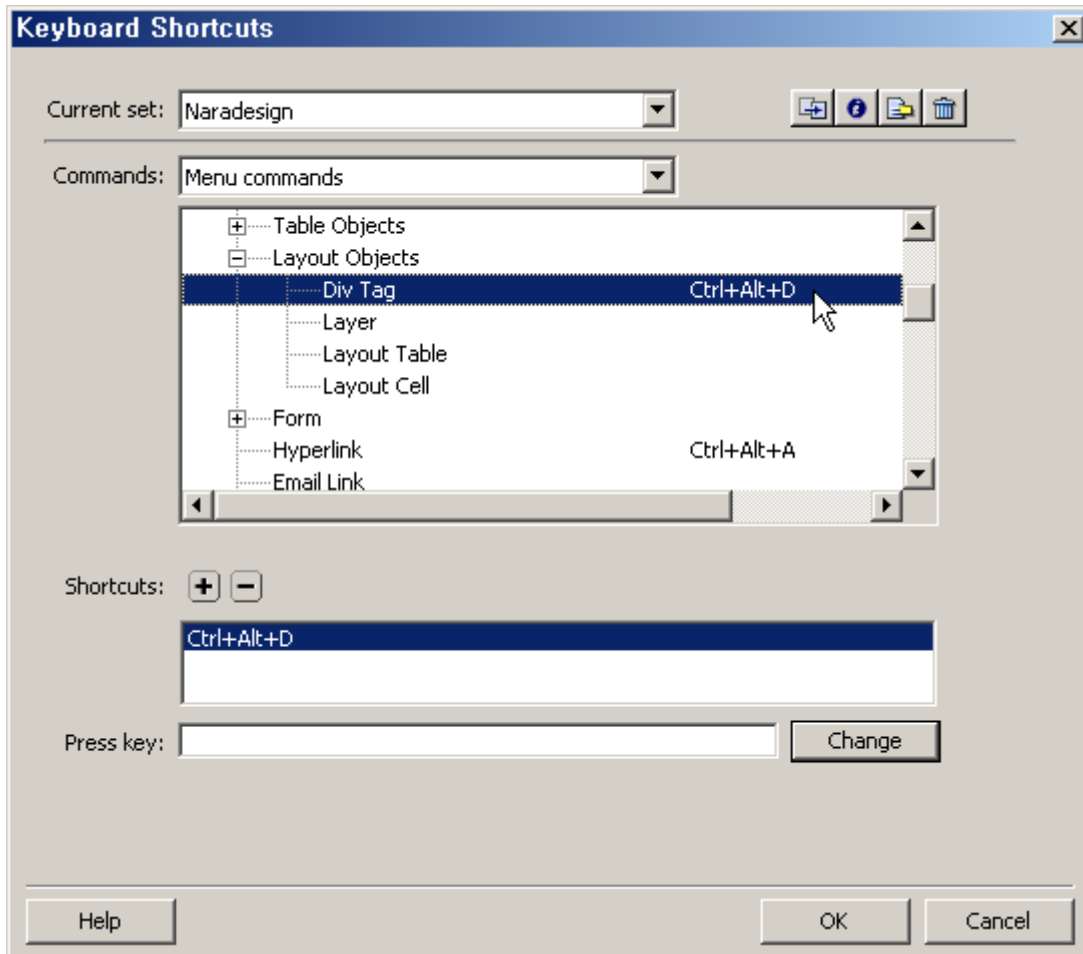


Edit > Keyboard Shortcuts 메뉴를 실행한 다음 현재의 Macromedia Standard 단축키 셋을 한 번 복사합니다.



새로 만들어질 단축키 셋의 이름을 입력합니다.





자주 사용되지만 단축키가 없는 항목을 찾아내어 나만의 단축키를 설정합니다. 예제에서는 div 태그 삽입에 대한 단축키로 Ctrl+Alt+D, a 태그 삽입에 대한 단축키로 Ctrl+Alt+A 를 설정하였습니다.

### 삽입 메뉴 관리 (Insert Panel Favorites)

코드 자동 완성(Code Hints), 빠른 태그 편집(Quick Tag Editor), 단축키(Keyboard Shortcuts) 보다 때로는 삽입메뉴(Insert Menu)를 이용하는 것이 편할 때가 있습니다. 삽입메뉴는 사용자의 선호에 따라 두 가지 형태(Tab&Menu)로 존재할 수 있는데 Menu 형태로 두는 것이 공간절약 측면에서 효율적 입니다. 자주 쓰는 명령을 즐겨찾기에 추가해 놓고 즐겨찾기 메뉴를 활성화 시켜 놓으면 더욱 편리하게 사용할 수 있습니다.

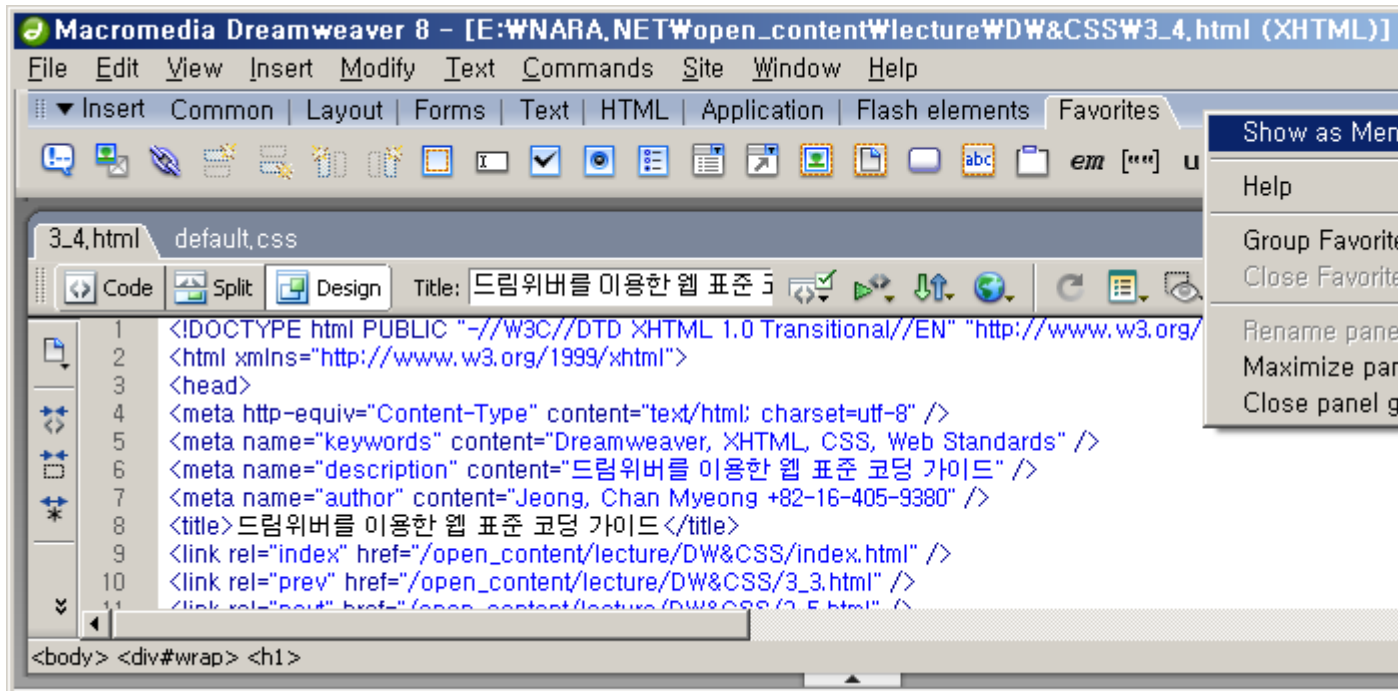
### 삽입메뉴의 보기 옵션



Tab 형태로 보기. (Show as Tab)

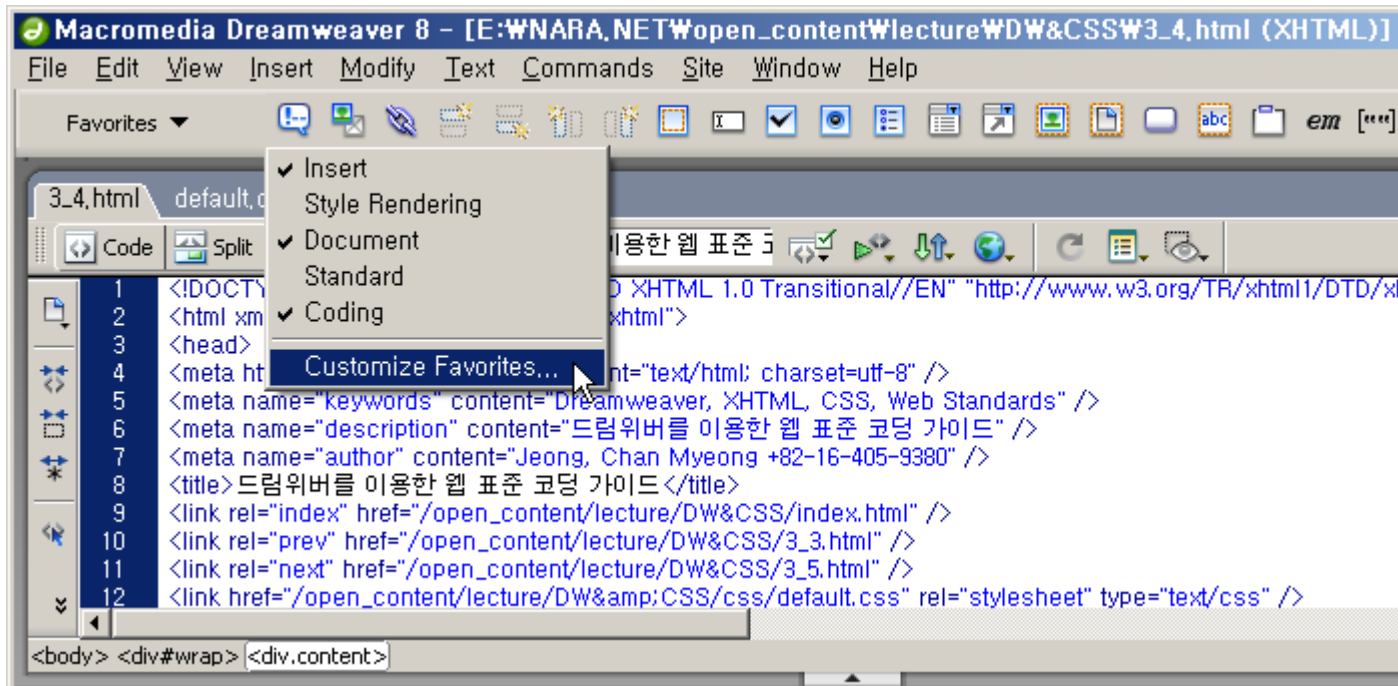


Menu 형태로 보기. (Show as Menu)

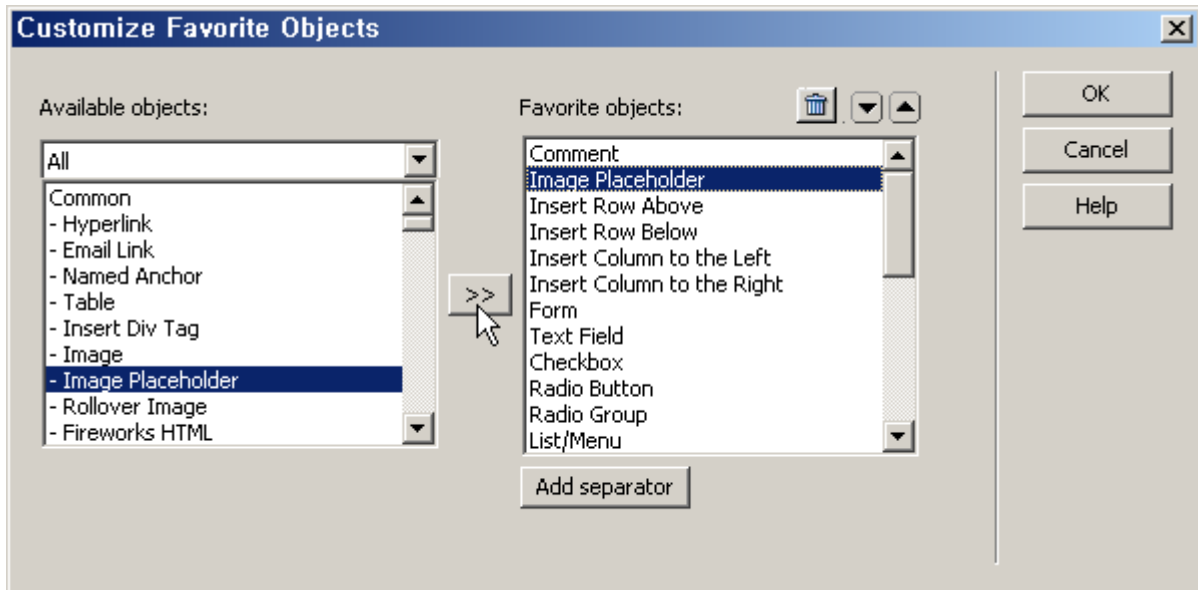


삽입메뉴 영역에서 컨텍스트 메뉴(마우스 우측 버튼 클릭)를 이용하면 Tab 형태와 Menu 형태 보기를 전환 할 수 있습니다.

#### 삽입메뉴의 즐겨찾기(Favorites) 설정



삽입메뉴 영역에서 컨텍스트 메뉴(마우스 우측버튼 클릭)를 활성화 시킨 다음 Customize Favorites 명령을 실행합니다.

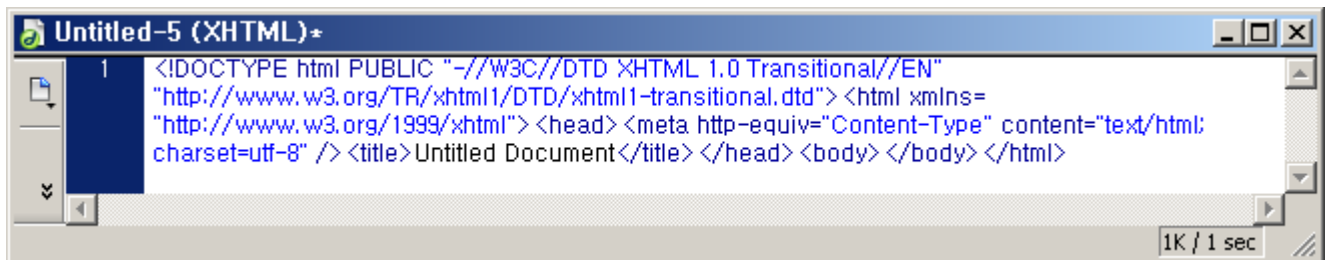


좌측에 있는 메뉴 명령 중 자주쓰는 명령 객체들을 우측으로 옮겨 놓으면 해당 항목이 즐겨찾기에 등록됩니다.

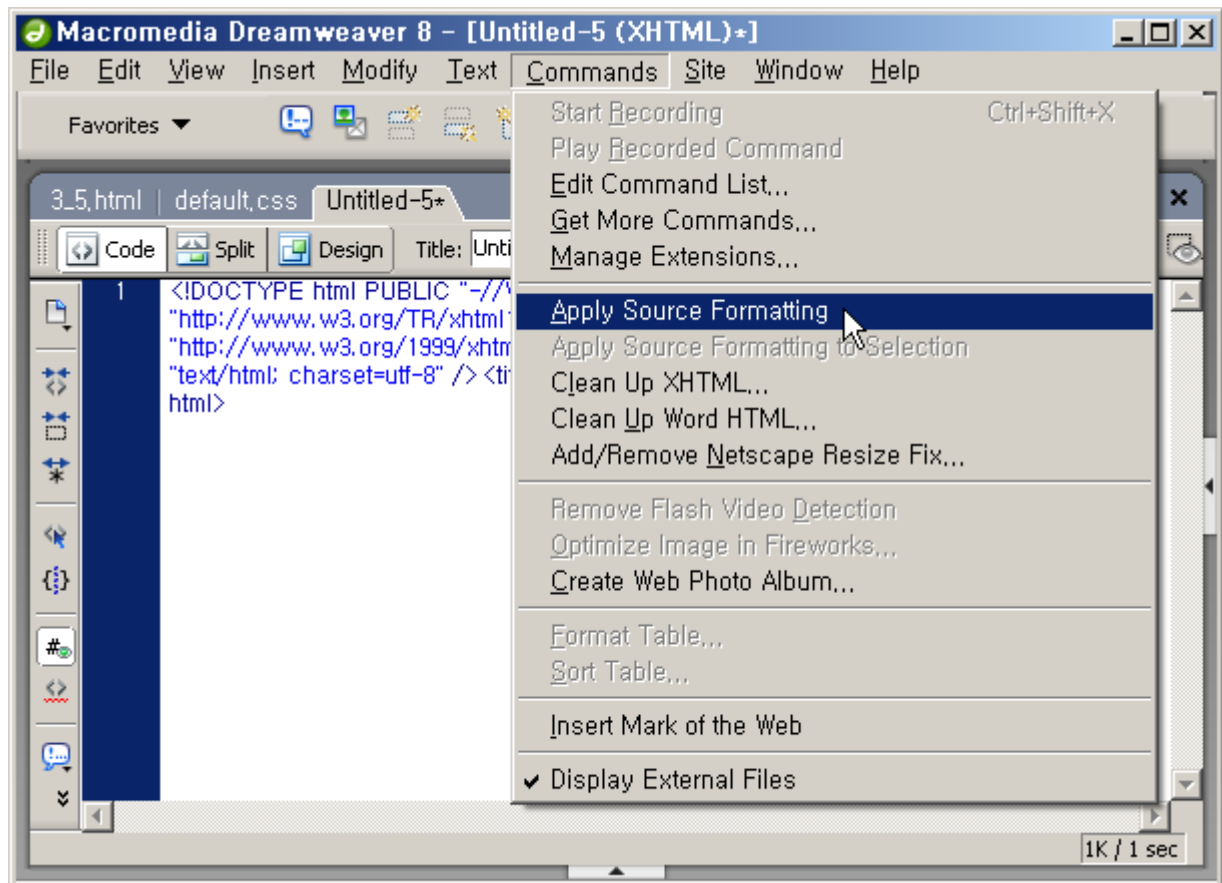
### 코드 관리 (Code Management)

웹사이트 구축 작업을 위하여 HTML 코드를 공유해야 하는 사람들이 많아질수록 코드관리의 중요성은 높아집니다. 드림위버에서 생성하는 코드는 정해진 규칙에 따라서 자동으로 들여쓰기와 줄바꿈을 하게 되지만 Hand Coding 을 하거나 또는 그 비율이 높아지면 자연스럽게 코드의 포맷도 흐트러지게 됩니다. 드림위버는 이것을 다시 자동으로 정렬 시켜주며 코드의 들여쓰기 간격과 엘리먼트에 따른 줄바꿈 설정을 사용자가 원하는 포맷으로 변경할 수 있도록 지원하고 있습니다.

### 코드 서식의 재 정렬

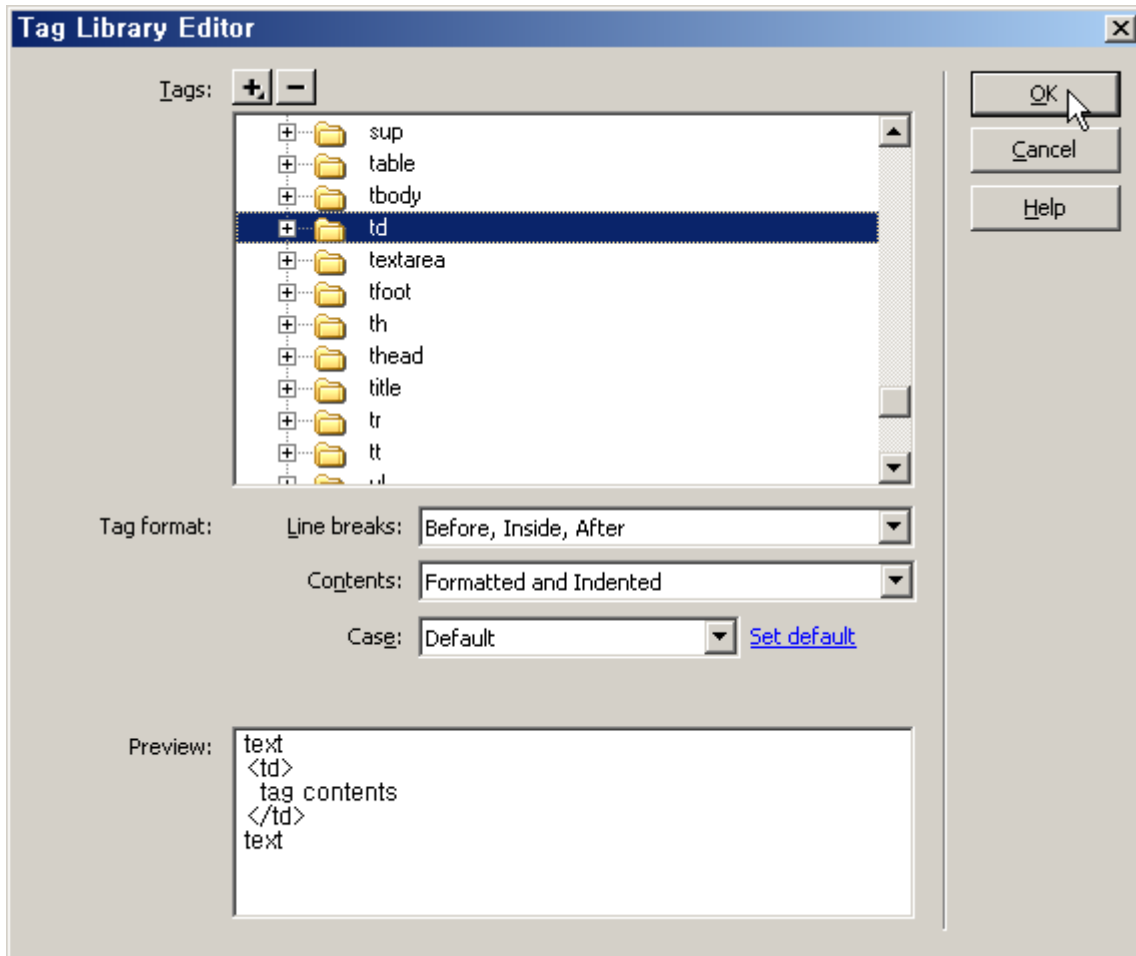


코드의 줄바꿈 규칙이 깨진 상태 입니다.



Commands > Apply Source Formatting 명령으로 다시 복구할 수 있습니다. 이 명령에 대한 단축키를 따로 설정하는 것도 좋은 방법이며 단축키를 설정하지 않고 빠르게 실행하려면 Alt+(C-A)를 누릅니다. (Alt 키를 누른 상태로 C와 A키를 순차적으로 함께 누르는 방법) 이것은 키보드를 이용하여 기본 메뉴명령을 제어하는 형식입니다.

**코드 줄 바꿈 사용자 정의**



Edit > Tag Libraries 항목을 실행하면 각각의 HTML 엘리먼트에 대하여 '줄 바꿈 설정, 들여쓰기 설정, 대소문자 설정'을 사용자 정의 할 수 있습니다. (XHTML 버전의 문서에서는 대문자 엘리먼트를 허용하지 않기 때문에 대소문자 규칙이 적용되지 않습니다)

출처:웹접근성본부