

# CHAPTER 6

## FOSI Pseudo-Elements

---

A FOSI **pseudo-element** is an element that is not defined in the DTD, but which has an `e-i-c` in the FOSI. As with a DTD element, a pseudo-element `e-i-c` may have a context string and an occurrence setting. On the other hand, since a pseudo-element is not declared in the DTD, it has no attributes. However, a pseudo-element can test and use attribute values on ancestor elements and ancestor pseudo-elements.

Pseudo-elements are used for several purposes, and are termed accordingly with appropriate extensions, as shown in the following table.

**Table 10** Types of pseudo-elements

Type	Extension	Purpose	Example
formatting	.fmt	format the contents of <code>savetext</code> <code>conrule</code> and <code>usetext</code> source	<code>&lt;bold.fmt&gt;, \Practical FOSI\, &lt;/bold.fmt&gt;</code>
context	.ctx	provide context for <code>e-i-c</code> s	<code>&lt;e-i-c gi="para" context="para.ctx"&gt;</code>
programming	.psu	create programmatic processes	<code>&lt;do-it-once.psu&gt;, &lt;/do-it-once.psu&gt;</code>

**NOTE:** Formatting pseudo-elements can be nested, as shown in examples below.

### **FOSI Tip**

The name of a pseudo-element is limited to 71 characters.

### **FOSI Tip**

In the **All FOSI Components** panel of the style panels interface, select **Query**→**All Non-DTD (Pseudo-elements) Elements** to see a list of `e-i-c`'s for elements that are not defined in the DTD.

**FOSI TIP** 

For end notes, the `e-i-c` for footnote is coded in the `styldesc`, not the `ftndesc`.

## Formatting pseudo-elements

Formatting pseudo-elements format anything that is allowed in a `usetext` source or a `savetext` conrule (listed in **Table 77 Savetext conrule and usetext source syntax** on page 461), including `#CONTENT`, FOSI variables, and FOSI-generated material.

### Formatting pseudo-element examples

The first example shows nested formatting pseudo-elements.

**Figure 67** Formatting pseudo-elements in a `savetext` conrule

```
<stringdecl textid="abstract.txt" literal="">
<stringdecl textid="abstract-title.txt" literal="Abstract">
...
<e-i-c gi="abstract">
<charlist inherit="1" charsubsetref="SUPPRESS">
<savetext textid="abstract.txt"
conrule="<keep-together.fmt>,
<abstract-title.fmt>,\* \,abstract-title.txt,\* \,</abstract-title.fmt>,
<abstract.fmt>,#CONTENT,</abstract.fmt>,
</keep-together.fmt">
...
<e-i-c gi="abstract.fmt">
<charlist inherit="1" charsubsetref="block italic"></charlist>
</e-i-c>
<e-i-c gi="abstract-title.fmt">
<charlist inherit="1" charsubsetref="titlea postspace">.</charlist>
</e-i-c>
<e-i-c gi="keep-together.fmt">
<charlist inherit="1" charsubsetref="keep-together"></charlist>
</e-i-c>
```

The next example shows formatting pseudo-elements in a `usetext` source. `<Footnote>` elements throughout a `<chapter>` are suppressed and saved for output as end notes at the end of each `<chapter>`. Notice that formatting pseudo-elements `<chapter-endnotes.fmt>` and `<bold.fmt>` are nested. Also notice that `<chapter-endnotes.fmt>` provides context for `<bold.fmt>`, which has two `e-i-c`s with different context strings and different formatting for each context.

**Figure 68** Footnotes saved for output as end notes

```
<counter enumid="chapterct" style="arabic" initial="0">
<counter enumid="footnotect" style="arabic" initial="0">
<stringdecl textid="chapter-endnotes.app" literal="">
...
```

```

<e-i-c gi="chapter">
<charlist inherit="1">
<enumerat enumid="chapterct" incremnt="1">
<savetext placemnt="before" textid="chapter-endnotes.app" conrule="\">
<usetext placemnt="before"
source="\Chapter \,<bold.fmt>,chapterct,</bold.fmt>"></usetext>
<usetext placemnt="after" source="chapter-endnotes.app"></usetext>
...
<e-i-c gi="footnote" context="* chapter">
<charlist inherit="1" charsubsetref="block SUPPRESS">
<enumerat enumid="chapter-footnotect" incremnt="1">
<savetext textid="chapter-endnotes.app" append="1"
conrule="<chapter-endnote.fmt>,
<bold.fmt>,chapter-footnotect,</bold.fmt>,@1pi,#CONTENT,
</chapter-endnote.fmt>">
...
<e-i-c gi="chapter-endnote.fmt">
<charlist inherit="1" charsubsetref="block keep-together"></charlist>
...
<e-i-c gi="bold.fmt" context="chapter">
<charlist inherit="1" charsubsetref="bold"></charlist>
...
<e-i-c gi="bold.fmt" context="chapter-endnote.fmt">
<charlist inherit="1" charsubsetref="semi-bold"></charlist>
...

```

Arbortext does not support a rule type of double for boxing. However, you can use pseudo-elements to output two boxes to mimic double lines. This concept is illustrated in the FOSI fragment below.

### Figure 69 Double-line boxing

```

<e-i-c gi="sidebar">
<charlist inherit="1" charsubsetref="block SUPPRESS">
<usetext source="<outer-box.fmt>,<inner-box.fmt>,#CONTENT,</inner-box.fmt>,
</outer-box.fmt>"></usetext>
...
<e-i-c gi="inner-box.fmt">
<charlist inherit="1" charsubsetref="block">
<boxing ...>
...
<e-i-c gi="outer-box.fmt">
<charlist inherit="1" charsubsetref="block">
<boxing ...>
...

```

Sometimes the boxed content includes gentext, as shown in the following figure.

**Figure 70 Double-line boxing with gentext**

```

<e-i-c gi="danger">
<charlist inherit="1" charsubsetref="block SUPPRESS">
<usetext source="<outer-box.fmt>,<inner-box.fmt>,<danger.fmt>,\DANGER!\,
</danger.fmt>,#CONTENT,</inner-box.fmt>,</outer-box.fmt></usetext>
...
<e-i-c gi="inner-box.fmt">
<charlist inherit="1" charsubsetref="block">
<boxing ...>
...
<e-i-c gi="outer-box.fmt">
<charlist inherit="1" charsubsetref="block">
<boxing ...>
...

```

**Figure 71** demonstrates why pseudo-elements should specify `inherit="1"` for `charlist` and category inheritance. Otherwise:

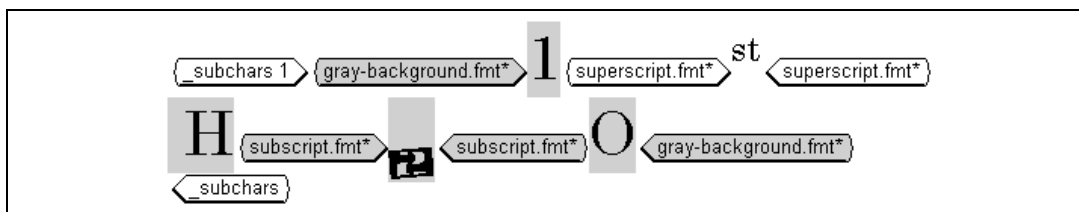
- formatting is not inherited from parent formatting pseudo-elements
- characteristics in inheriting categories coded in `subchars` have no effect

In this case, `<superscript.fmt>` does not inherit `highlt` characteristics from `<gray-background.fmt>` because `charlist` inheritance is disabled. The font category in `<superscript.fmt>`, however, enables inheritance, so the font famname specified in the `usetext` subchars is used.

On the other hand, `<subscript.fmt>` enables `charlist` inheritance and therefore inherits the gray background. However, the font category disables inheritance, so the `docdesc` font famname is used instead of the subchars font famname.

**NOTE:** The first graphic shows the output in Preview. The second graphic shows the Edit window display after `set gentexttagdisplay=full` (see ) is entered at the command line.

**Figure 71 Formatting problems when pseudo-element does not inherit**



### FOSI fragment

```

<docdesc>
<charlist>
<font famname="STOMP_Tinsnips" ...>
...
<e-i-c gi="test">
<charlist inherit="1" charsubsetref="block prespace">
<usetext source="<gray-background.fmt>,
\1\,<superscript.fmt>,\st\,</superscript.fmt>,
\ H\,<subscript.fmt>,\2\,</subscript.fmt>,\0\,
</gray-background.fmt">
<subchars charsubsetref="block">
<font inherit="1" famname="Century Schoolbook">
....
<e-i-c gi="gray-background.fmt">
<charlist inherit="1">
<highlt inherit="1" bckclr="gray">
...
<e-i-c gi="subscript.fmt">
<charlist inherit="1">
<font inherit="0" size="0.6em" offset="-0.3em">
...
<e-i-c gi="superscript.fmt">
<charlist inherit="0">
<font inherit="1" size="0.6em" offset="0.3em">
...

```

## Context pseudo-elements

Sometimes a pseudo-element is needed to provide different contexts for formatting pseudo-elements, as shown in the following FOSI fragment.

### FOSI fragment

```

<e-i-c gi="bold.fmt" context="para.ctx">
<charlist inherit="1" charsubsetref="semi-bold">
...
<e-i-c gi="bold.fmt" context="title.ctx">
<charlist inherit="1" charsubsetref="bold">
...

```

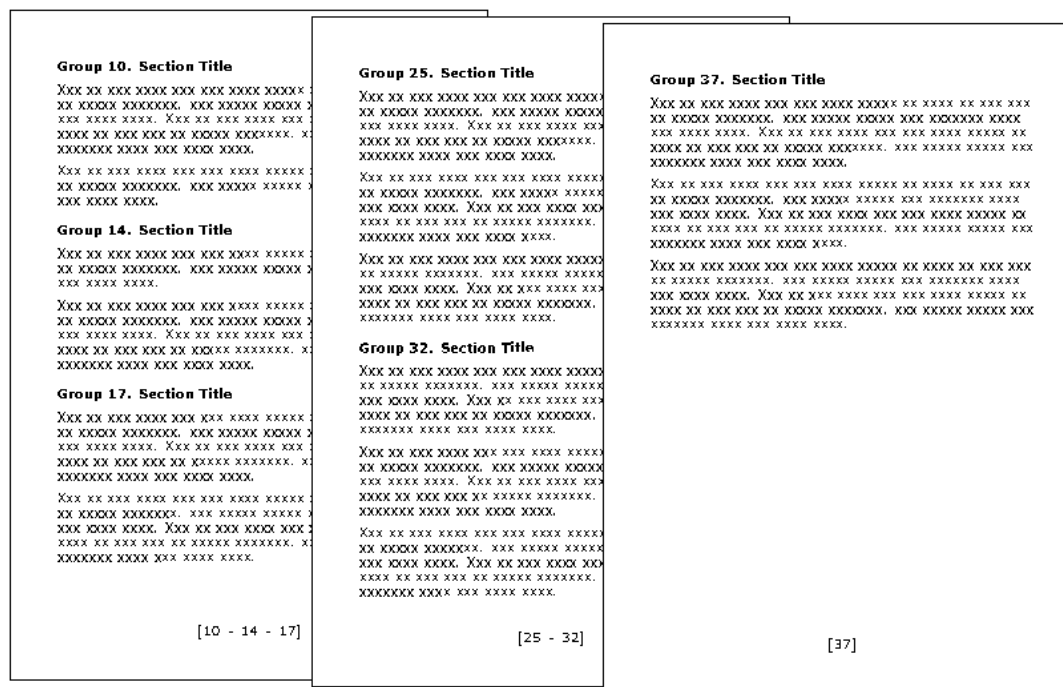
**NOTE:** A pseudo-element whose only purpose is to provide context does not need to be coded in the FOSI.

## Context pseudo-element example

This example uses the [FB] string modifier (see page 463) to output in the footer the *group* attributes for the <section> elements on each page. Square brackets surround the number(s). Multiple numbers are separated from each other by a dash surrounded by spaces. The characters generated depends on the occurrence setting for the *group.fmt e-i-cs*.

**NOTE:** Occurrence applies to the position within the parent tag, which in this case is the context pseudo-element *group.ctx*.

**Figure 72 Occurrence with pseudo-elements**



### DTD fragment

```
<!ELEMENT doc (section)+ >
<!ELEMENT section (title,para+) >
<!ATTLIST section group NUMBER #REQUIRED >
```

```
<!ELEMENT (title|para) (#PCDATA) >
```

### XML fragment

```
<section group="10">
<title>...</title>
<para>...</para>
</section>
<section group="14">
<title>...</title>
<para>...</para>
</section>
<section group="17">
<title>...</title>
<para>...</para>
</section>
<section group="25">
<title>...</title>
<para>...</para>
</section>
<section group="32">
<title>...</title>
<para>...</para>
</section>
<section group="37">
<title>...</title>
<para>...</para>
</section>
```

### FOSI fragment

```
<stringdecl textid="groups-on-page.txt" literal="">
...
<footer>
<usetext source="<group.ctx>,groups-on-page.txt [FB],</group.ctx>">
<subchars charsubsetref="block center">
...
<e-i-c gi="group.fmt" context="group.ctx" occur="only">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\[" placemnt="before"></usetext>
<usetext source="]\\" placemnt="after"></usetext>
...
<e-i-c gi="group.fmt" context="group.ctx" occur="first">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\[" placemnt="before"></usetext>
...
<e-i-c gi="group.fmt" context="group.ctx" occur="middle">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\ - \" placemnt="before"></usetext>
...
<e-i-c gi="group.fmt" context="group.ctx" occur="last">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\ - \" placemnt="before"></usetext>
```

```

<usetext source="\]" placemnt="after"></usetext>
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="block keep-together">
</charlist>
<att>
<fillval attname="group" attloc="section" fillcat="savetext"
fillchar="conrule">
<charsubset>
<savetext textid="group.tmp">
<savetext textid="groups-on-page.txt"
conrule="<group.fmt>,group.tmp,</group.fmt>" append="0">
...

```

## Programming pseudo-elements

Programming pseudo-elements are needed for the following two reasons:

- When an attribute specification needs to specify `placemnt="after"`, which is not possible. Instead, a `usetext` with `placement="after"` is coded with a pseudo-element as its source. The `e-i-c` for the pseudo-element is coded with the attribute rule.
- In an `e-i-c`, when a counter or string is assigned a value with `enumerat` or `savetext`, and `#FOSI` (discussed on page 92) is coded in the same `e-i-c` in order to test the value of the counter or string, the result of the test is not correct. `#FOSI` must be coded in a pseudo-element for the assignment to be recognized. A `usetext` source is coded in the `e-i-c` with a pseudo-element as its source. The `e-i-c` for the pseudo-element is coded with the `specval` that contains `#FOSI`.

### Programming pseudo-element example

**NOTE:** This example contains code that does not achieve the desired result. It is intended for purposes of illustration not production use.

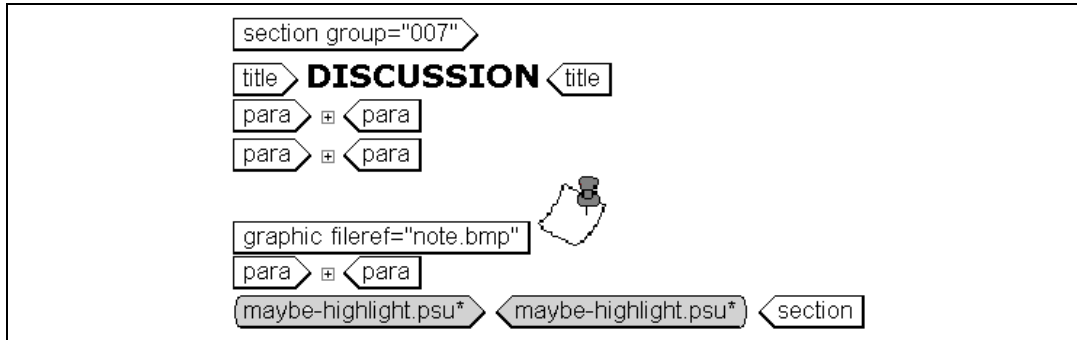
This example demonstrates the two reasons programming pseudo-elements are used.

- the presence of a `<graphic>` in a `<section>` must be detected at the end of the `<section>`
- the `specval` with `#FOSI` that tests `graphic.yn` cannot be coded in the `e-i-c` for `<section>`

Notice that a pseudo-element is a child of the `e-i-c` that calls it and can test the attributes on the parent or other ancestor elements.

Also notice the result when an attspec in a pseudo-element specifies formatting to be applied to element content. As the Edit window view shows, only the pseudo-element is affected, not the content of the `<section>`.

**Figure 73 Pseudo-element attribute formats pseudo-element**



#### FOSI fragment

```
<stringdecl textid="graphic.yn" literal="n">
...
<e-i-c gi="graphic" context="section">
<charlist inherit="1" charsubsetref="graphic">
<savetext textid="graphic.yn" conrule="\y\">
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="division">
<savetext textid="graphic.yn" conrule="\n\">
<usetext source="<maybe-highlight.psu>,</maybe-highlight.psu>"
placemnt="after"></usetext>
...
<e-i-c gi="maybe-highlight.psu">
<charlist inherit="1"></charlist>
<att logic="and">
<specval attname="group" attloc="section" attval="#EQ#007">
<specval attname="graphic.yn" attloc="#FOSI" attval="y">
<charsubset>
<hight inherit="1" bckclr="gray">
...

```