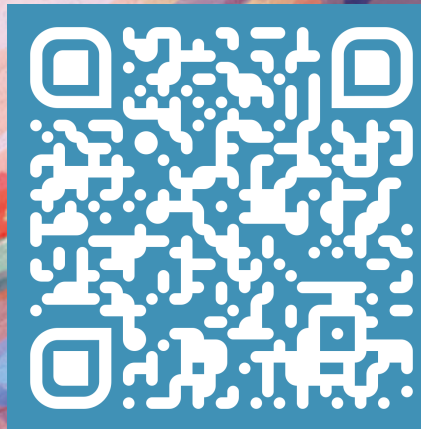


How to Beautify SQL and PL/SQL Code

Philipp



@phsalvisberg



<https://www.salvis.com/blog>

17.11.2020

Philipp

- Database centric development
- Model Driven Software Development
- Author of free SQL Developer Extensions
PL/SQL Unwrapper, PL/SQL Cop,
utPLSQL, plsscope-utils, oddgen and
Bitemp Remodeler



 @phsalvisberg

 <https://www.salvis.com/blog>

FOUNDED IN
1994

300 SLA's
(SERVICE LEVEL AGREEMENTS)

 **700**
EMPLOYEES

 **15 TRIVADIS WORKSPACES**
SWITZERLAND, GERMANY,
AUSTRIA, ROMANIA

4000 
TRAINING PARTICIPANTS PER YEAR

5 MILLION CHF 
BUDGET FOR SCIENCE
AND DEVELOPMENT PER YEAR

118 MILLION CHF
TURNOVER 

800 
CUSTOMERS

EXPERIENCE FROM 1900 PROJECTS PER YEAR 

Syntax Highlighting

```
BEGIN
  FOR rec IN (
    SELECT r.country_region AS region, p.prod_category, SUM(s.amount_sold) AS amount_sold
      FROM sales          s
      JOIN products      p    ON p.prod_id      = s.prod_id
      JOIN customers      cust ON cust.cust_id   = s.cust_id
      JOIN times          t    ON t.time_id     = s.time_id
      JOIN countries      r    ON r.country_id  = cust.country_id
    WHERE calendar_year = 2000
    GROUP BY r.country_region, p.prod_category
    ORDER BY r.country_region, p.prod_category
  ) LOOP
    IF rec.region = 'Asia' THEN
      IF rec.prod_category = 'Hardware' THEN
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      END IF;
    END IF;
  END LOOP;
END;
/
```

Step 1 - Highlight Keywords

```
BEGIN
  FOR rec IN (
    SELECT r.country_region AS region, p.prod_category, SUM(s.amount_sold) AS amount_sold
      FROM sales          s
     JOIN products        p   ON p.prod_id      = s.prod_id
     JOIN customers        cust ON cust.cust_id   = s.cust_id
     JOIN times            t   ON t.time_id      = s.time_id
     JOIN countries        r   ON r.country_id   = cust.country_id
    WHERE calendar_year = 2000
    GROUP BY r.country_region, p.prod_category
    ORDER BY r.country_region, p.prod_category
  ) LOOP
    IF rec.region = 'Asia' THEN
      IF rec.prod_category = 'Hardware' THEN
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      END IF;
    END IF;
  END LOOP;
END;
/
```


UPPERCASE?

trivadis



Source: <https://twitter.com/TomCostantino/status/1196257484370935808/photo/1>

Step 2 – lowercase Keywords

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
    from sales      s
    join products  p  on p.prod_id    = s.prod_id
    join customers cust on cust.cust_id = s.cust_id
    join times     t   on t.time_id    = s.time_id
    join countries r   on r.country_id = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```


Step 3 – Highlight Numbers

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
      from sales          s
    join products  p      on p.prod_id      = s.prod_id
    join customers cust on cust.cust_id     = s.cust_id
    join times     t      on t.time_id      = s.time_id
    join countries r      on r.country_id   = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

Step 3 – Highlight Strings

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
      from sales          s
      join products      p  on p.prod_id      = s.prod_id
      join customers      cust on cust.cust_id = s.cust_id
      join times          t   on t.time_id     = s.time_id
      join countries      r   on r.country_id  = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

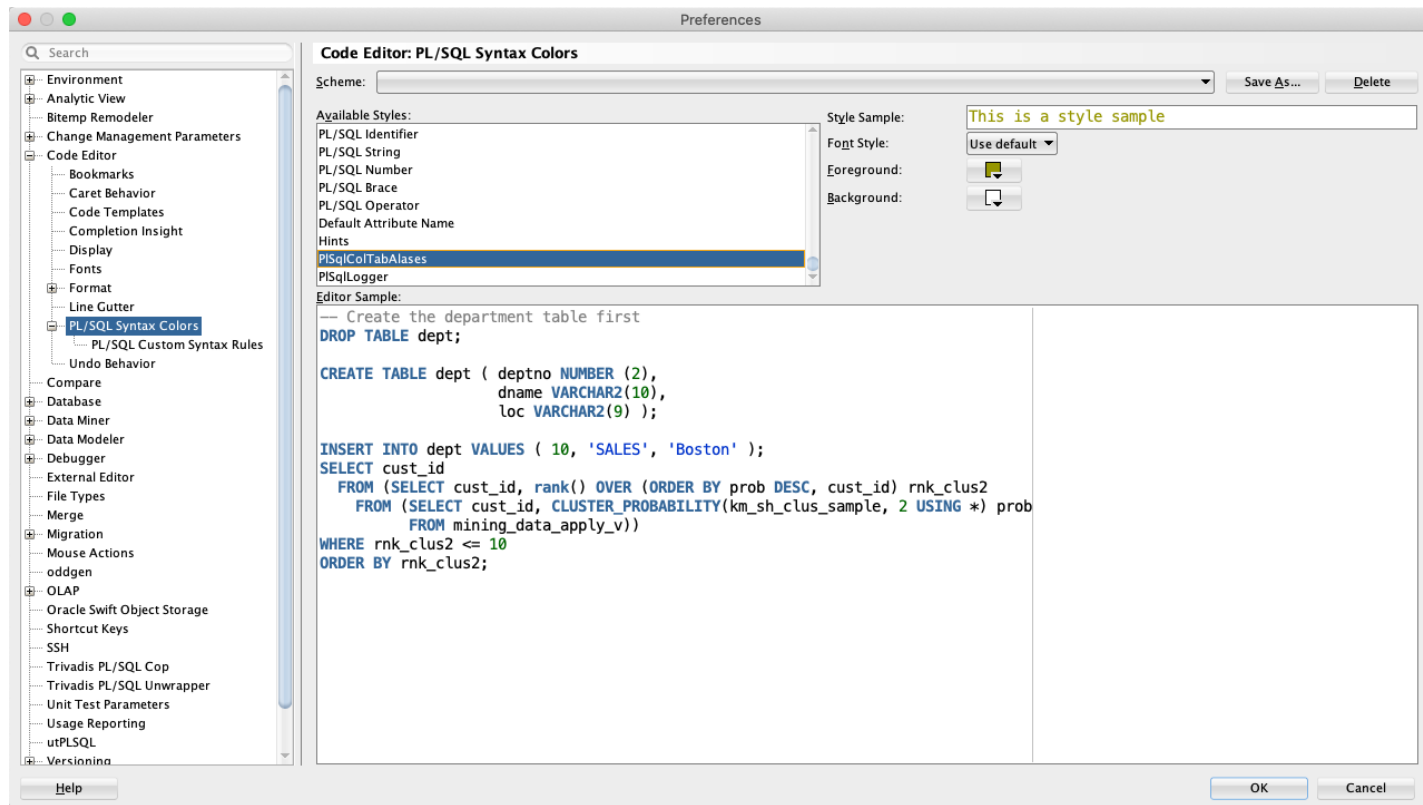
Step 4 – Highlight Comments

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
      from sales          s
    join products  p      on p.prod_id      = s.prod_id
    join customers cust on cust.cust_id     = s.cust_id
    join times     t      on t.time_id      = s.time_id
    join countries r      on r.country_id   = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

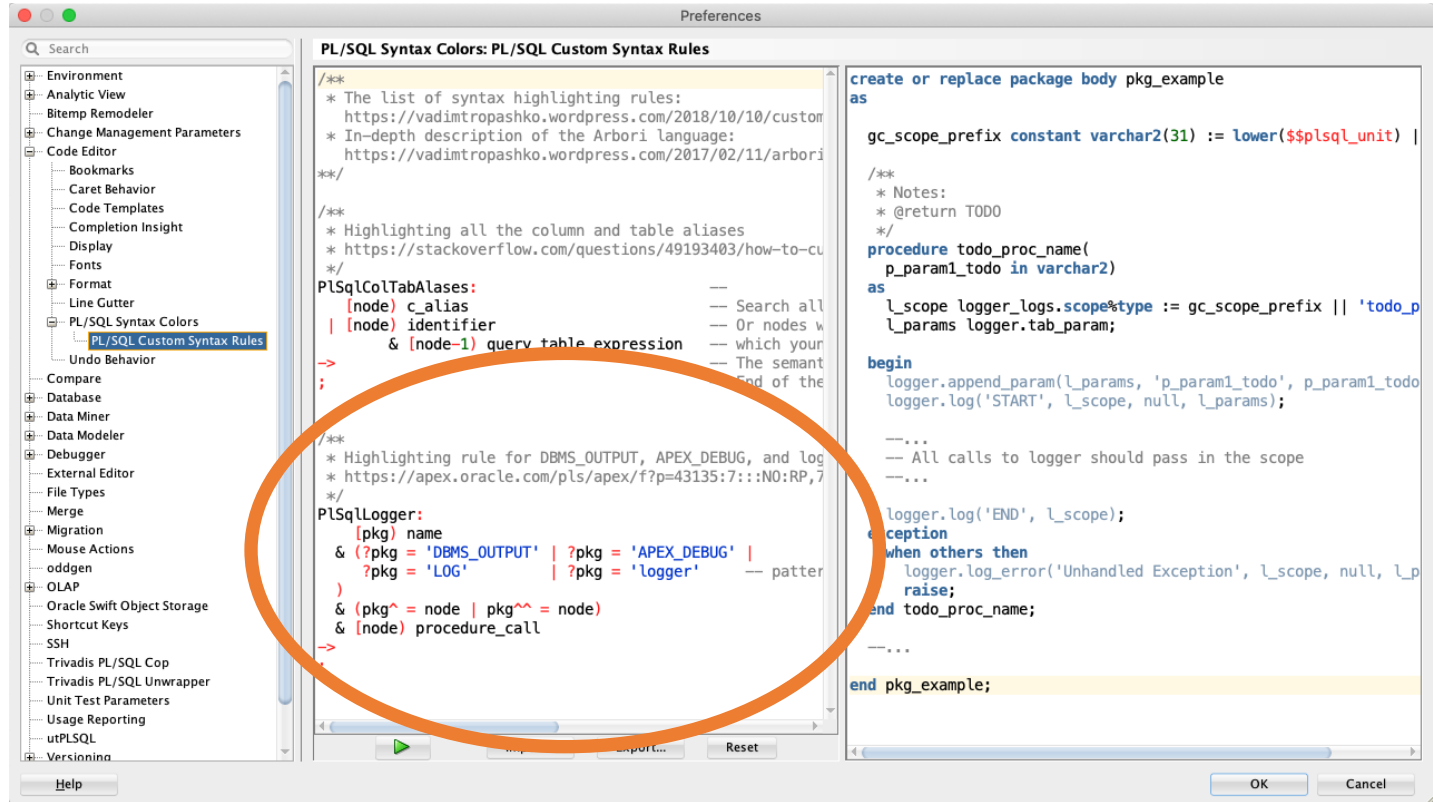
Step n – Highlight Aliases

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
      from sales          s
    join products  p      on p.prod_id    = s.prod_id
    join customers cust    on cust.cust_id = s.cust_id
    join times     t       on t.time_id   = s.time_id
    join countries r       on r.country_id = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

PL/SQL Syntax Colors

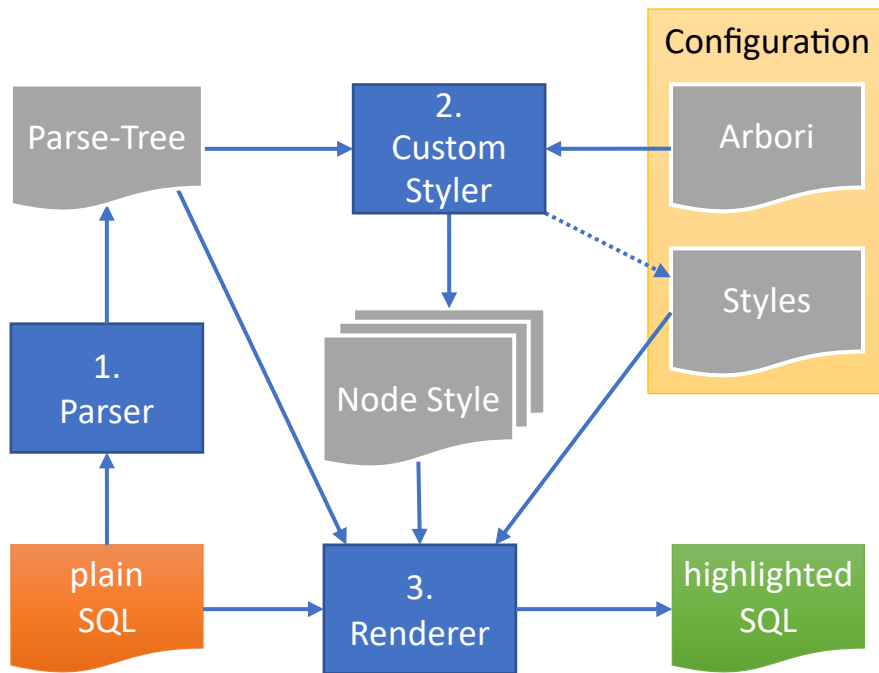


PL/SQL Custom Syntax Rules



Arbori – Part 1 (Styler)

How the Custom Styler (Highlighter) Works



Java Callback Functions (->)

```
public void PLSqlColTabAlases (Parsed target, Map<String, ParseNode> tuple) {  
    addStyle(target, tuple.get("node"), "PLSqlColTabAlases");  
}
```

```
public void PLSqlLogger (Parsed target, Map<String, ParseNode> tuple) {  
    addStyle(target, tuple.get("node"), "PLSqlLogger");  
}
```

JavaScript Callback Function – Example

Operator:

```
[node) ':' | [node) '=' | [node) '>' | [node) '<' | [node) '|' | [node) '!'
-> {
    struct.addStyle(target, tuple.get("node"), "Operator");
}
;
```

- Add Arbori query to "Custom Syntax Rules" (executed in the program's order)
- Restart SQL Developer
- "Operator" appears as new style under "PL/SQL Syntax Colors" (based on query name)
- Now you can change the font style (normal, bold, italic) and the foreground and background color

Code Outline in SQL Developer



The screenshot displays the SQL Developer interface with two main panes. The left pane, titled 'Code Outline', shows a hierarchical tree structure of the SQL code. The right pane, titled 'SQL Worksheet', shows the original SQL code. A blue thought bubble with the text 'This is a parse tree!' points from the Code Outline to the SQL code.

Code Outline (Left Pane):

```
sql_statement sql_statements
├── block_stmt labeled_block_stmt library_unit
│   ├── BEGIN
│   │   ├── labeled_nonblock_stmt loop_stmt nonblock_compound_stmt seq_of_stmts stmt unlabeled_nonblock_stmt
│   │   │   ├── iteration_scheme
│   │   │   │   ├── LOOP
│   │   │   │   │   ├── if_stmt labeled_nonblock_stmt nonblock_compound_stmt seq_of_stmts stmt unlabeled_nonblock_stmt
│   │   │   │   │   │   ├── and_expr boolean_primary pls_expr rel
│   │   │   │   │   │   ├── THEN
│   │   │   │   │   │   │   ├── if_stmt labeled_nonblock_stmt nonblock_compound_stmt seq_of_stmts stmt unlabeled_nonblock_stmt
│   │   │   │   │   │   │   │   ├── and_expr boolean_primary pls_expr rel
│   │   │   │   │   │   │   ├── THEN
│   │   │   │   │   │   │   │   ├── labeled_nonblock_stmt seq_of_stmts sim_stmt stmt unlabeled_nonblock_stmt
│   │   │   │   │   │   │   │   │   ├── function_call name procedure_call
│   │   │   │   │   │   │   │   │   │   ├── dotted_expr name name_wo_function_call
│   │   │   │   │   │   │   │   │   │   │   ├── dotted_expr name name_wo_function_call
│   │   │   │   │   │   │   │   │   │   │   ├── SYS identifier name name_wo_function_call
│   │   │   │   │   │   │   │   │   │   │   │   ├── DBMS_OUTPUT decl_id identifier
│   │   │   │   │   │   │   │   │   │   │   │   ├── PUT_LINE decl_id identifier
│   │   │   │   │   │   │   │   │   │   │   └── paren_expr_list
│   │   │   │   │   │   │   │   └── END
│   │   │   │   │   │   │   └── IF
│   │   │   │   │   │   └── END
│   │   │   │   │   └── IF
│   │   │   │   │   └── END
│   │   │   │   └── LOOP
│   │   │   │   └── END
│   │   └── END
└── /
```

SQL Worksheet (Right Pane):

```
1 begin
2   for rec in (
3     select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
4     from sales s
5     join products p on p.prod_id = s.prod_id
6     join customers cust on cust.cust_id = s.cust_id
7     join times t on t.time_id = s.time_id
8     join countries r on r.country_id = cust.country_id
9     where calendar_year = 2000
10    group by r.country_region, p.prod_category
11    order by r.country_region, p.prod_category
12  ) loop
13    if rec.region = 'Asia' then
14      if rec.prod_category = 'Hardware' then
15        /* print only one line for demo purposes */
16        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
17      end if;
18    end if;
19  end loop;
20 end;
21 /
22
```

Arbori Editor

trivadis

The screenshot displays the Arbori Editor interface. The main window shows a parse tree for a SQL statement. The tree structure includes nodes like `sql_statement`, `sql_statements`, `block_stmt`, `labeled_block_stmt`, `begin`, `labeled_nonblock_stmt`, `loop_stmt`, `nonblock_compound_stmt`, `seq_of_stmts`, `stmt`, `unlabeled_nonblock_stmt`, `iteration_scheme`, `loop`, `if_stmt`, `end`, and `end_loop`. A blue callout bubble points to the `sql_statement` node with the text "Action per result node (tuple)".

On the right side, there is a console window titled "temp_worksheet10" and "0395016857951.arbori". It shows the output of the `PrintParseTreeToConsole` function. The output is a tuple: `[0,135) sql_statement sql_statements`. A blue callout bubble points to this output with the text "Arbori Query".

An orange arrow points from the `sql_statement` node in the parse tree to the tuple in the console output. A blue callout bubble points to the tuple with the text "Result tuples in parse tree".

Demo 1

Code Formatting (Beautifying)

Tokenized

```
begin
for
rec
in
(
select
r
.
country_region
as
region
,
p
.
prod_category
,
sum
(
s
.
amount_sold
)
as
amount_sold
from
sales
s
join
products
p
on
p
.
prod_id
=
s
.
prod_id
join
customers
cust
on
cust
.
cust_id
=
s
.
cust_id
join
times
t
on
t
.
time_id
=
s
.
time_id
join
countries
r
on
r
.
country_id
=
cust
.
country_id
where
calendar_year
=
2000
group
by
r
.
country_region
,
p
.
prod_category
order
by
r
.
country_region
,
p
.
prod_category
)
loop
if
rec.region
=
'Asia'
then
if
rec
.
prod_category
=
'Hardware'
then
/* print only
one line for
demo purposes
*/
sys
.
dbms_output
.
put_line
(
'Amount: '
||
rec
.
amount_sold
)
;
end
if
;
end
if
;
end
loop;
end;
```


Minified – No Wrap Edition



```
begin for rec in select r.country_region as region,p.prod_category_name as amt_sold as amount_sold from sales s join products p on p.prod_id=s.prod_id join customers cust on cust.cust_id=s.cust_id join times t on t.time_id=s.time_id join countries c on c.country_id=cust.country_id where calendar_year=2009 group by r.country_region,p.prod_category order by r.country_region,p.prod_category loop if rec.region='Asia' then if rec.prod_category='Hardware' then v/print only one line for demo purposes v/kyv.dbm_output_put_line('Amount: '||rec.amount_sold||pad if not if not loop end;
```

Minified – Word Wrap Edition

```
begin for rec in(select r.country_region as
region,p.prod_category,sum(s.amount_sold)as amount_sold from sales s join
products p on p.prod_id=s.prod_id join customers cust on cust.cust_id=s.cust_id
join times t on t.time_id=s.time_id join countries r on
r.country_id=cust.country_id where calendar_year=2000group by
r.country_region,p.prod_category order by r.country_region,p.prod_category)loop
if rec.region='Asia'then if rec.prod_category='Hardware'then/* print only one
line for demo purposes */sys.dbms_output.put_line('Amount: '||rec.amount_sold
);end if;end if;end loop;end;
/
```

```
begin for rec in (select r.country_region as region, p.prod_category,
sum (s.amount_sold) as amount_sold from sales s join products p on p.
prod_id = s .prod_id join customers cust on cust .cust_id = s.cust_id
join times t on t.time_id= s.time_id join countries r on r.country_id
= cust .country_id where calendar_year=2000 group by r.country_region
, p . prod_category order by r . country_region , p . prod_category )
loop if rec.region = 'Asia' then if rec . prod_category = 'Hardware'
then /* print only one line for demo purposes */ sys . dbms_output .
put_line('Amount: '||rec.amount_sold );end if; end if; end loop; end;
/
```

```
begin
  in(select r
    as region ,p .
    s.amount_sold ) as
s join products p on
join customers cust on
times t on t . time_id =s.    time_id join countries r on
  r.country_id = cust.country_id where calendar_year =
  2000 group by r.country_region , p.prod_category
  order by r .country_region, p.prod_category
) loop if rec . region = 'Asia' then if
  rec.prod_category = 'Hardware' then
    /* print only one line for demo
    purposes */sys.dbms_output
      . put_line ( 'Amount: '
        ||rec.amount_sold);
      end if;end if;
    end loop;
  end;
/
```

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
    from sales s join products p on p.prod_id = s.prod_id
    join customers cust on cust.cust_id = s.cust_id
    join times t on t.time_id = s.time_id
    join countries r
      on r.country_id = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category) loop
    if rec.region = 'Asia' then
if rec.prod_category = 'Hardware' then
  /* print only one line for demo purposes */
  sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
end if;
    end if;
  end loop;
end;
/
```

But the Machine Can Run This

```
begin
  in(select r
    as region ,p .
    s.amount_sold ) as
s join products p on
join customers cust on
times t on t . time_id =s.      time_id join countries r on
  r.country_id = cust.country_id where  calendar_year =
  2000 group by r.country_region , p.prod_category
  order by r .country_region, p.prod_category
) loop if rec . region = 'Asia' then if
  rec.prod_category = 'Hardware' then
    /* print only one line for demo
    purposes */sys.dbms_output
      . put_line ( 'Amount: '
        ||rec.amount_sold);
      end if;end if;
    end loop;
  end;
/
```

So What?

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

-- Martin Fowler



Source: https://en.wikiquote.org/wiki/Martin_Fowler

"It is a matter of fact, that code is more often read than written – therefore we should take efforts to ease the work of the reader, which is not necessarily the author."

-- Roger Troller



Source: <https://trivadis.github.io/plsql-and-sql-coding-guidelines/v3.6/>

Readable – Style 1 of ∞

```
begin
  for rec in (
    select r.country_region as region, p.prod_category, sum(s.amount_sold) as amount_sold
      from sales            s
    join products          p  on p.prod_id      = s.prod_id
    join customers          cust on cust.cust_id = s.cust_id
    join times              t   on t.time_id     = s.time_id
    join countries          r    on r.country_id = cust.country_id
    where calendar_year = 2000
    group by r.country_region, p.prod_category
    order by r.country_region, p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```


Readable – Style 2 of ∞

```
begin
  for rec in (select r.country_region    as region,
                    p.prod_category,
                    sum(s.amount_sold) as amount_sold
              from sales s join products p on p.prod_id    = s.prod_id
                        join customers cust on cust.cust_id = s.cust_id
                        join times      t   on t.time_id    = s.time_id
                        join countries r    on r.country_id = cust.country_id
              where calendar_year = 2000
              group by r.country_region, p.prod_category
              order by r.country_region, p.prod_category)

  loop
    if rec.region = 'Asia'
    then
      if rec.prod_category = 'Hardware'
      then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

Readable – Style 3 of ∞

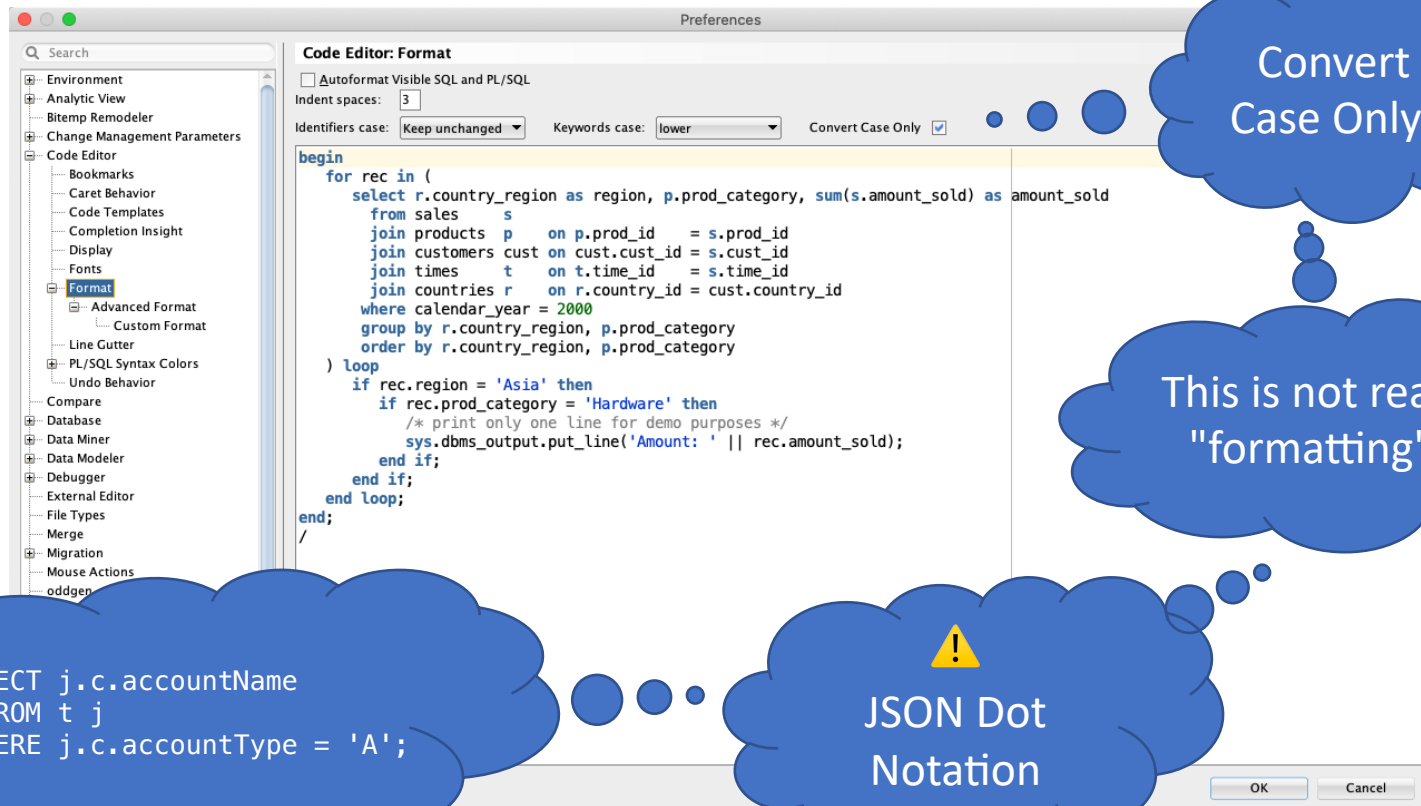
```
begin
  for rec in (
    select r.country_region as region,
           p.prod_category,
           sum(s.amount_sold) as amount_sold
    from   sales s
    join   products p
    on     p.prod_id = s.prod_id
    join   customers cust
    on     cust.cust_id = s.cust_id
    join   times t
    on     t.time_id = s.time_id
    join   countries r
    on     r.country_id = cust.country_id
    where  calendar_year = 2000
    group by r.country_region,
             p.prod_category
    order by r.country_region,
             p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

Readable – Style 4 of ∞

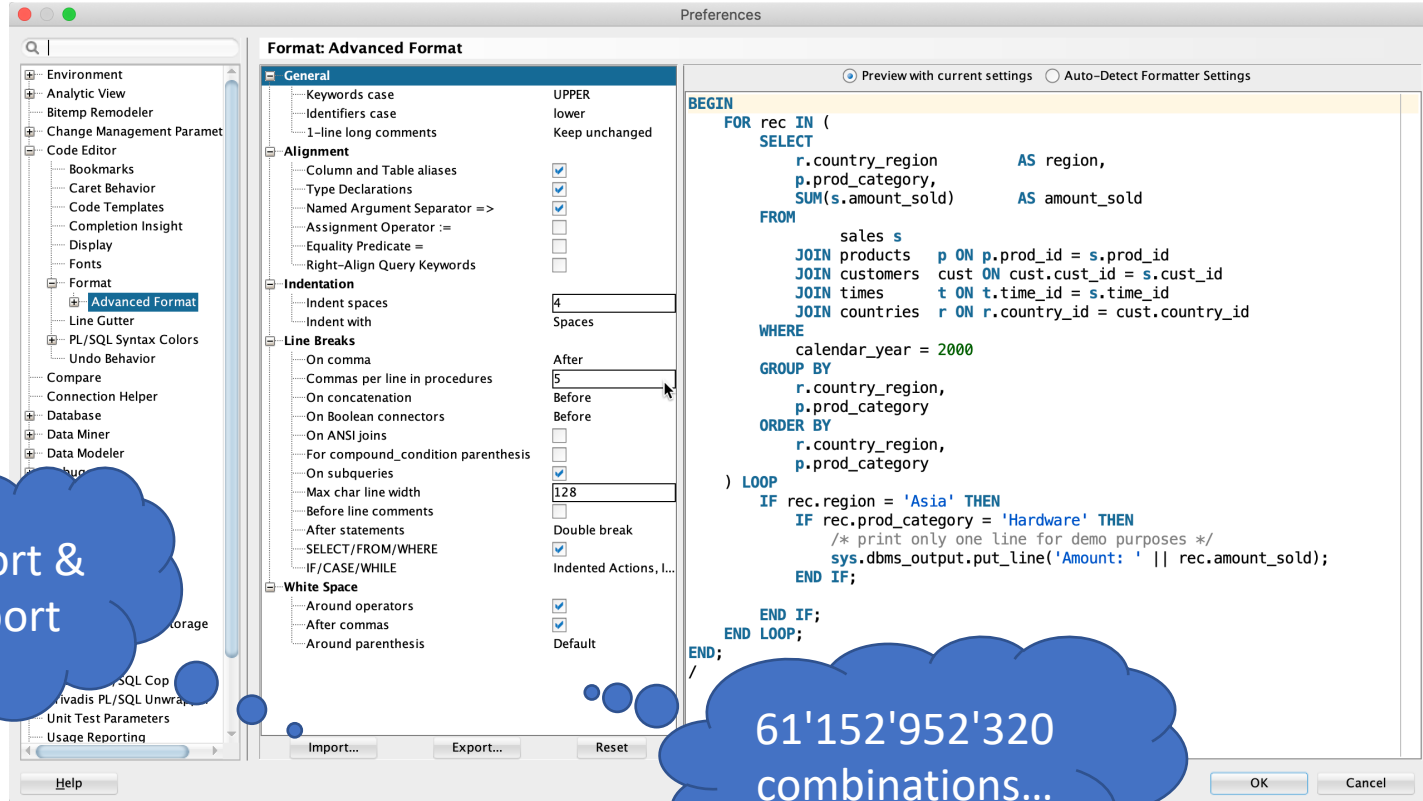
```
begin
  for rec in (
    select r.country_region  as region,
           p.prod_category,
           sum(s.amount_sold) as amount_sold
    from   sales             s
    join   products          p
    on     p.prod_id = s.prod_id
    join   customers         cust
    on     cust.cust_id = s.cust_id
    join   times             t
    on     t.time_id = s.time_id
    join   countries         r
    on     r.country_id = cust.country_id
    where  calendar_year = 2000
    group by r.country_region,
             p.prod_category
    order by r.country_region,
             p.prod_category
  ) loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then
        /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

Format

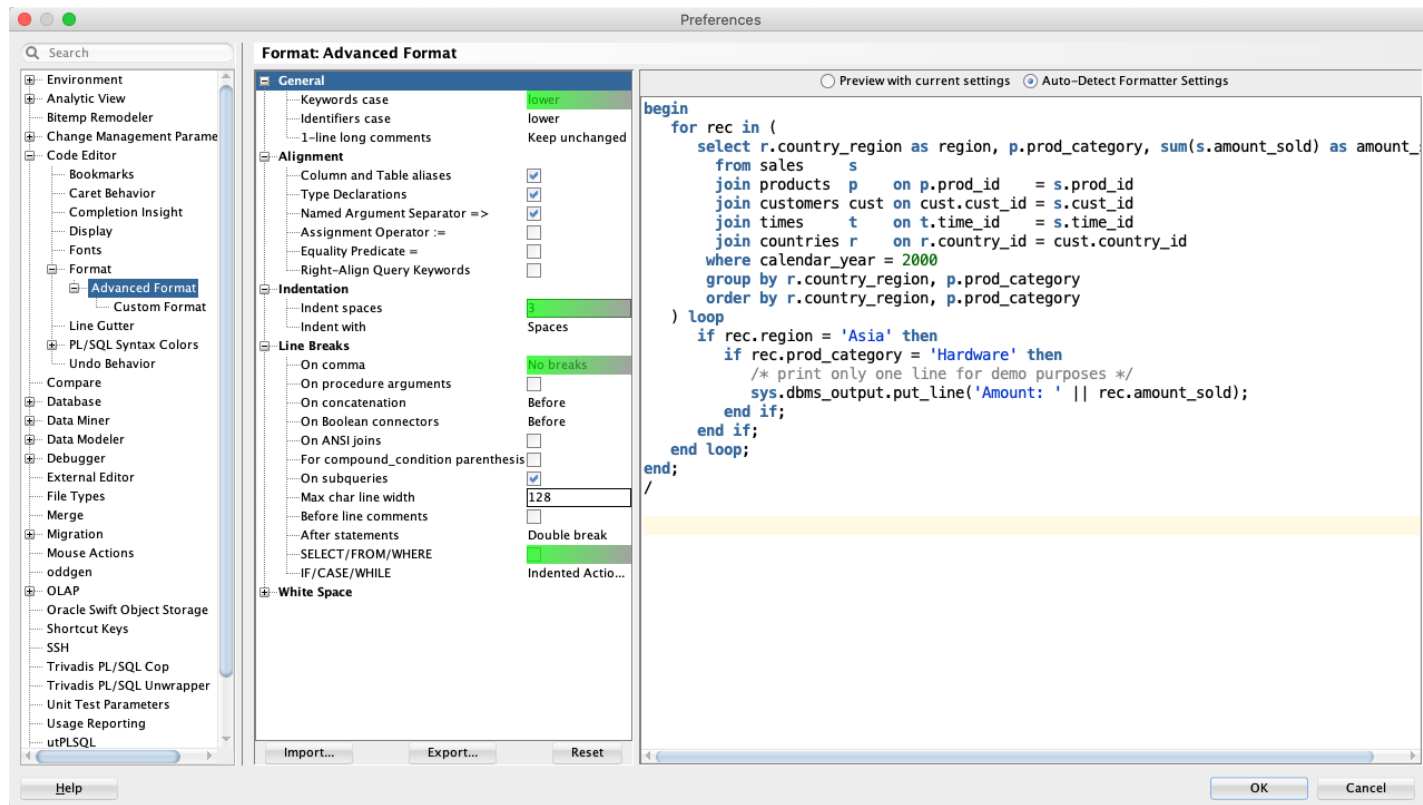
trivadis



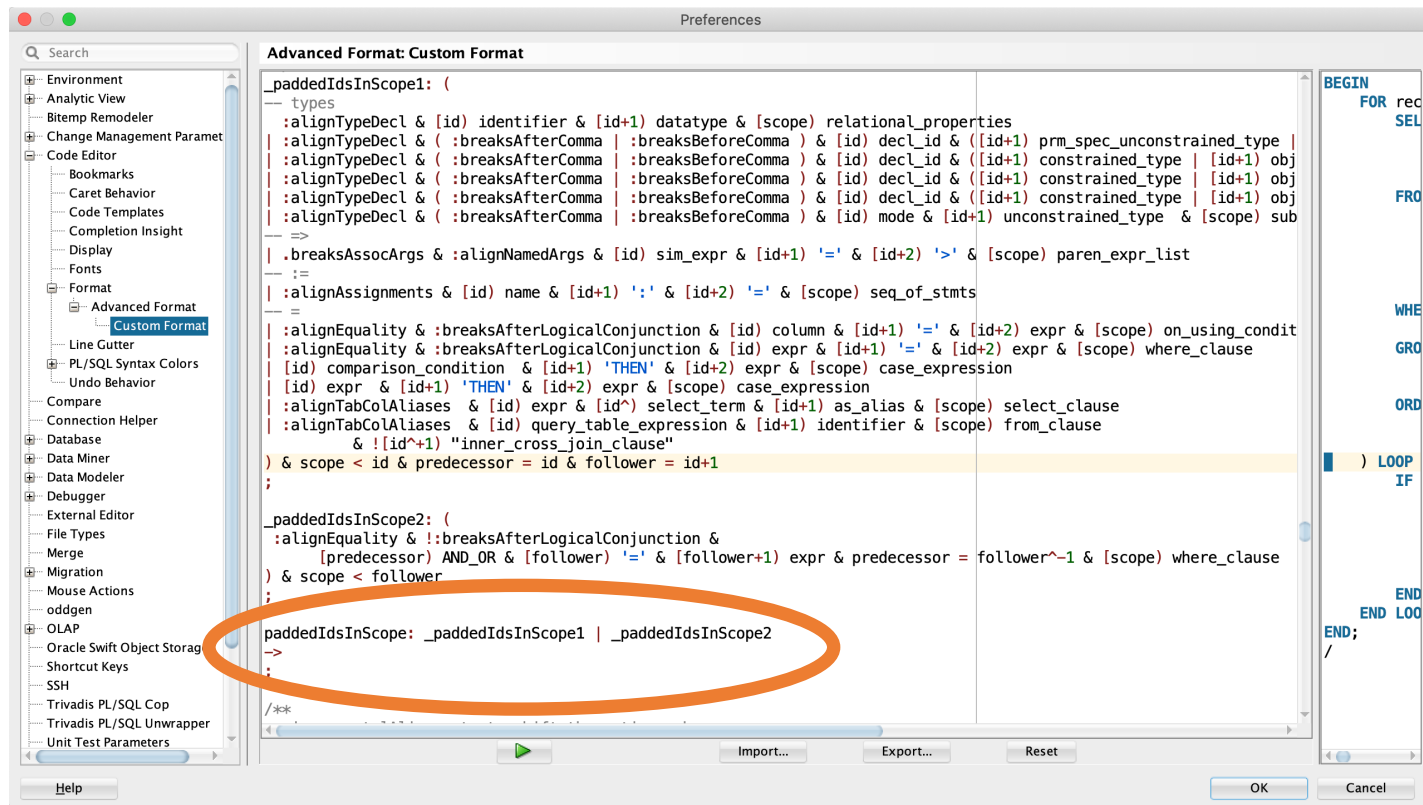
Advanced Format – Default Settings



Auto-Detect Formatter Settings

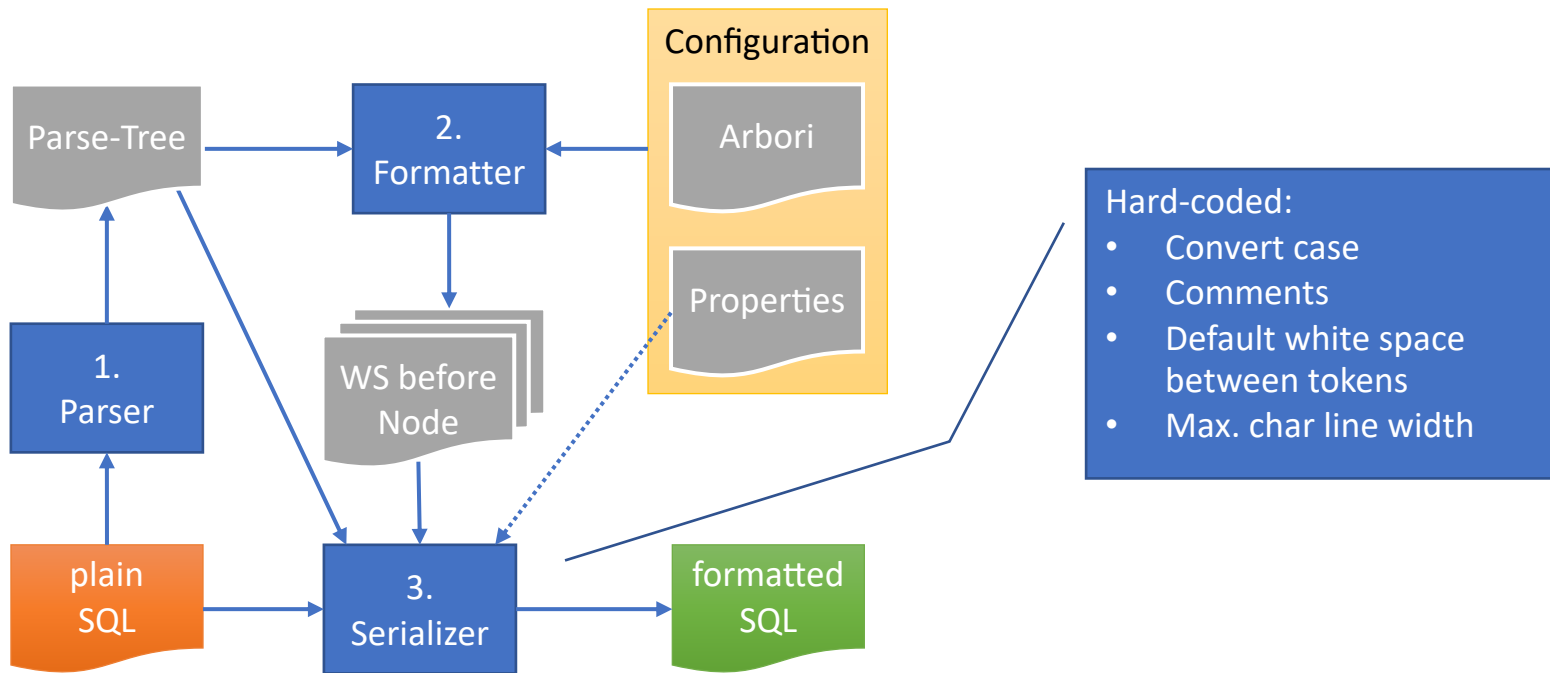


Custom Format



Arbori – Part 2 (Formatter)

How the Formatter Works



Java Callback Functions (->)

```
public void ... (Parsed target, Map<String, ParseNode> tuple) {...}
```

- [indentedNodes1](#) (node)
- [indentedNodes2](#) (node)
- [skipWhiteSpaceBeforeNode](#) (node)
- skipWhiteSpaceAfterNode (node)
- identifiers (identifier)
- [extraBrkBefore](#) (node)
- extraBrkAfter (node)
- [brkX2](#) (node)
- [rightAlignments](#) (node)
- [paddedIdsInScope](#) (scope, predecessor, follower)
- incrementalAlignments (node)
- [pairwiseAlignments](#) (node, predecessor)
- [ignoreLineBreaksBeforeNode](#) (node)
- [ignoreLineBreaksAfterNode](#) (node)
- dontFormatNode (node)

Demo 2

Java Callback Function Call – Example

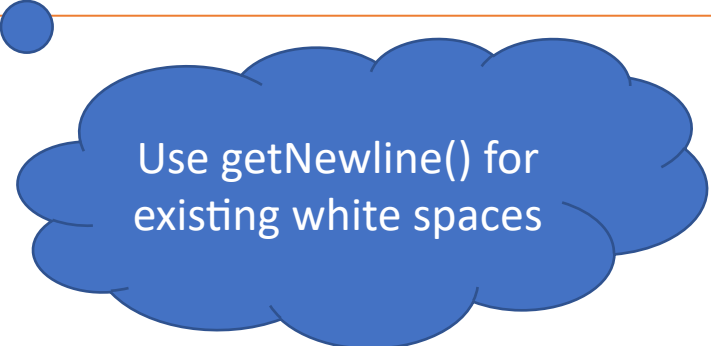
```
dontFormatNode:  
  [node) numeric_literal | [node) select_term  
  ->  
  ;
```

calls
struct.dontFormatNode
(target, tuple);

expects "node"
in tuple

JavaScript Callback Function – Example

```
aliasFix:  
  [node) as_alias  
  -> {  
    var node = tuple.get("node");  
    struct.putNewline(node.from, " ");  
  }  
;
```



Use getNewline() for
existing white spaces

Java Callback Function Call – Example

```
_paddedIdsInScope1: (  
...  
-- fix: do not align as_alias for multiple select_term on the same line  
| (:breaksBeforeComma | :breaksAfterComma) & :alignTabColAliases  
  & [id) expr & [id^) select_term & [id+1) as_alias & [scope) select_clause  
...  
) & scope < id & predecessor = id & follower = id+1  
;  
  
paddedIdsInScope: _paddedIdsInScope1 | _paddedIdsInScope2  
->  
;
```

calls
struct.paddedIdsInScope
(target, tuple);

expects "scope",
"predecessor",
"follower" in tuple

SQLcl

Built-in Format Command

help format

FORMAT

FORMAT BUFFER - formats the script in the SQLcl Buffer

FORMAT RULES <filename> - Loads SQLDeveloper Formatter rules file to formatter.

FORMAT FILE <input_file> <output_file>

Format used is default or for SQLcl can be chosen by setting an environmental variable pointing to a SQLDeveloper export (.xml) of formatter options.

The variable is called SQLFORMATPATH

In SQLDeveloper the format options are the default chosen in the preferences.

Custom Formatter Command / JavaScript

tvdfORMAT

missing mandatory <rootPath> argument.

usage: tvdfORMAT <rootPath> [options]

mandatory arguments:

<rootPath> file or path to directory containing files to format (content will be replaced!)
use * to format the SQLcl buffer

options:

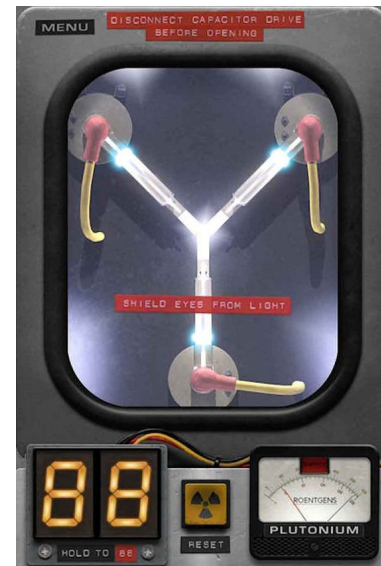
ext=<ext> comma separated list of file extensions to process, e.g. ext=sql,pks,pkb
mext=<ext> comma separated list of markdown file extensions to process, e.g. ext=md,mdown
xml=<file> path to the file containing the xml file for advanced format settings
xml=default uses default advanced settings included in sqlcl
xml=embedded uses advanced settings defined in format.js
arbori=<file> path to the file containing the Arbori program for custom format settings
arbori=default uses default Arbori program included in sqlcl

See: <https://github.com/Trivadis/plsql-formatter-settings/tree/main/sqlcl>

Core Messages

With Arbori You Can

- Arbori is what makes highly customized code highlighting and formatting possible.
- Code Outline and the Arbori editor are helpful tools.
- Want to learn more?
 - <https://vadimtropashko.wordpress.com/>
 - <https://www.salvis.com/blog/tag/arbori/>
 - <https://github.com/Trivadis/plsql-formatter-settings>



You are welcome to join us at the Expo area.
We're looking forward to meeting you.



Link to the Expo area:

<https://www.vinivia-event-manager.io/e/DOAG/portal/expo/29731>

trivadis

