Introduction

Jan 20, 2021

TablesNG is an effort to reimplement tables in Chrome. It is part of LayoutNG, a multi-year Chrome layout engine rewrite. Progress of the new implementation is tracked in Chrome's bug tracker.

The old table implementation architecture had limitations that made fixing many important bugs impossible without the rewrite. The new implementation emphasizes correctness. What does correctness mean for tables?

The <table> tag is old. It is so old that it predates standard bodies. Its layout has never been fully formally specified, and is still being worked on in the latest spec draft.

TablesNG implements the specification as much as possible. If the feature is not specified, it tries to follow what the existing browsers do. If browsers disagreed, we made an educated guess, documented it, wrote wpt tests, and will follow up with the CSS working group to define a standard for all browsers. The tests are currently being reviewed and released.

TablesNG fixes many bugs (77 new web platform tests pass), and increases predictability. Some bug fixes add capabilities that were never implemented before.

TablesNG will be turned on gradually. For testing, you can turn it on in chrome://flags. You can also use `--enable-blink-features=LayoutNGTable` and `--disable-blink-features=LayoutNGTable` on the command line.

TablesNG is not used for printing yet. Bug 1155197

Developer Visible Changes

This is a list of major differences. While reading this document, you can play with the examples on https://jsbin.com/ramojom/edit?html,output (archived on github)
Interoperability: Firefox and Safari were tested. Safari and Chrome's old implementation often failed in the same way, because they both evolved from WebKit.

Subpixel geometry

Legacy engine rounded table geometry to whole pixels. This could cause alignment issues when zooming.
Example:

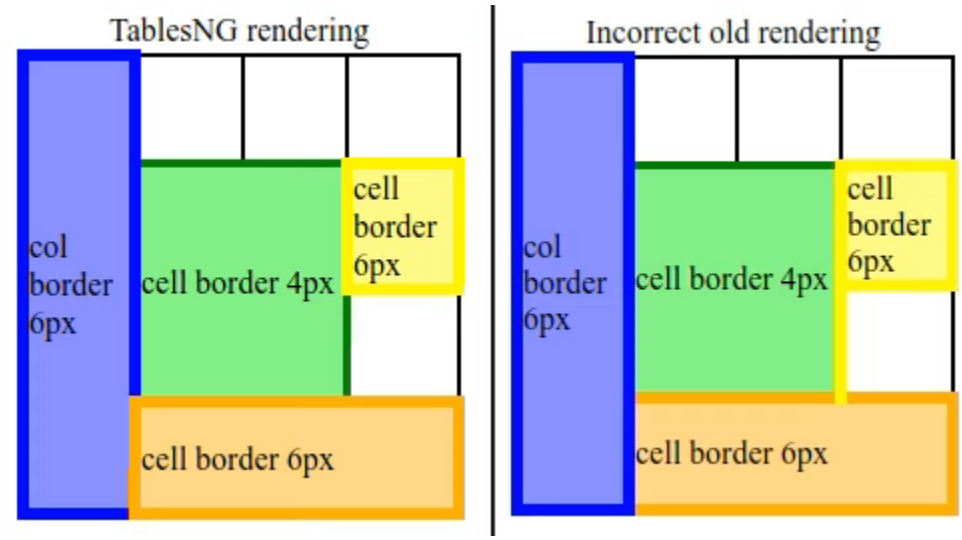
```
<table style="width:100px">
  <td>x</td> <td>x</td> <td>x</td>
</table>
```

TablesNG TD widths are: 33.33px 33.33px 33.34px
Legacy TD widths are: 33px 33px 34px
Interoperability: full. Firefox/Safari match Chrome.

Improved collapsed border painting for cells that span multiple rows/columns

Collapsed borders for large cells are now painted correctly. Bug: 2902

An example:



The former rendering had multiple problems. Green cell's right border has 2 segments, yellow and green. The old code incorrectly painted the entire tall cell border as a single segment. Orange cell's left border is defined by cell, and has higher precedence than blue cell's right border that is defined by column.
Interoperability: matches Firefox. Safari looks like Chrome's old rendering.

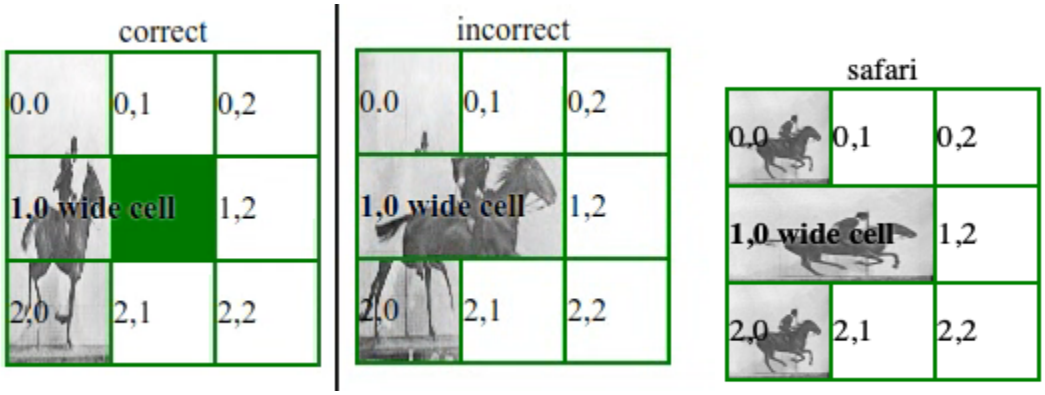
Improved background painting for cells that span multiple rows/columns

Cell's background image can be defined by a col/section/row (background source). TablesNG will render col/section/row defined backgrounds inside large cells correctly.
The geometry of the background is defined by the source, not by the cell. The former rendering defined background geometry of large cells by the cell.
You can see the difference in the example below:

```
<table>
  <col style="background-image:horse.jpg;
              background-size: 100% 100%;
              background-repeat: no-repeat;
              background-color: green;">
   <tr>
     <td>0.0</td> <td>0,1</td> <td>0,2</td>
   </tr>
   <tr>
     <td colspan=2>1,0 wide cell</td>
     <td>1,2</td>
   </tr>
   <tr>
     <td>2,0</td> <td>2,1</td> <td>2,2</td>
   </tr>
```

Wide cell's background is defined by the <COL> element. Correct rendering uses <COL> geometry to compute background image size. Incorrect rendering uses a blend of <COL> and <TD> geometry.

The same issues apply to <TR> backgrounds.



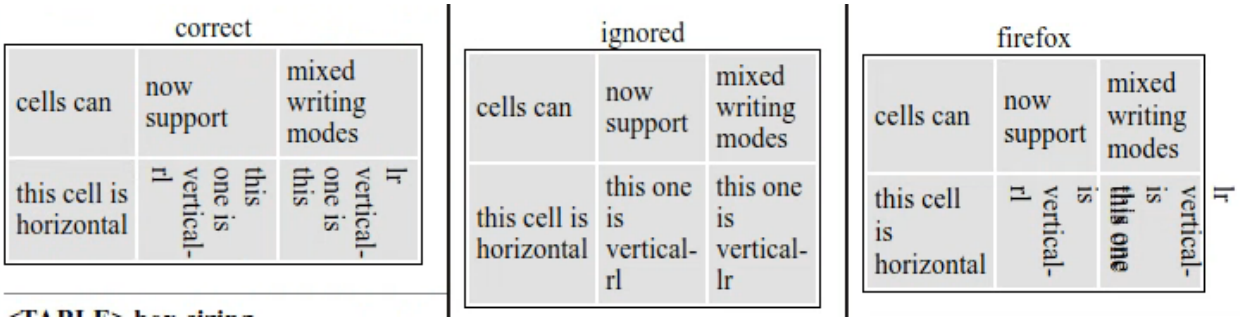
Interoperability:
- Matches Firefox, with the exception of background-repeat property. Chrome defaults to background-repeat: repeat, while Firefox defaults to background-repeat:no-repeat. For maximum compatibility, set background-repeat explicitly.
- Safari incorrectly uses <TD> geometry to render <COL> background.

CSS: <TD> supports orthogonal writing modes

Cells should support writing-modes that differ from TABLE's writing mode. TablesNG implements this CSS feature in Chrome. Previously, a cell's writing mode was always forced to match table's.

An example:

```
<style> td { width: 50px; height: 50px;} </style>
<table class="vertical">
  <tr>
    <td>cells can</td>
    <td>now support</td>
    <td>mixed writing modes</td>
  </tr>
  <tr >
    <td>this cell is horizontal</td>
    <td style="writing-mode:vertical-rl">this one is vertical-rl</td>
    <td style="writing-mode:vertical-lr">this one is vertical-lr</td>
  </tr>
</table>
```



Without specified height, vertical cells will grow vertically to their max inline size. But specifying height causes trouble for Firefox.

Interoperability:
- Firefox orthogonal cells can overflow if height is specified.
- Safari and old Chrome force cells to match table's writing mode.

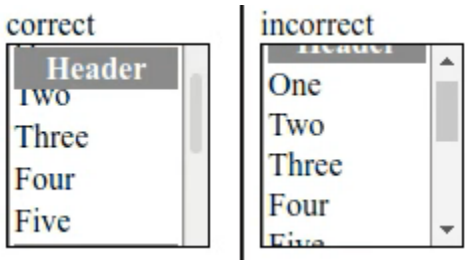
CSS: visibility:collapse for table columns

TablesNG implements visibility:collapse for table columns. It works just like collapse does for rows. The cells in the collapsed column are hidden. Other column/row sizes do not change. Table's width will shrink.
One area of incompatibility is cells that span multiple columns, when some columns get hidden. The details of this are discussed in bug 1157626.
Interoperability:
Firefox supports it. Wide cells can cause trouble.
Safari does not support visibility:collapse on columns.

Sections/rows can have position: that is not static

Sections/rows now support position:relative|fixed|sticky.
position:relative can be used to contain absolutely positioned children of TDs.
position:sticky can be used to ensure header rows stay visible while scrolling.
Example:
<TR> with position:sticky will stick to <TBODY>

```
<div style="width:100px;height:100px;overflow:auto">
<table>
  <tbody style="position:relative">
    <tr style="position:sticky;top:1px;left:0><th>Header</th></tr>
    <tr><td>One</td></tr>
    <tr><td>Two</td></tr>
    etc....
    </tbody>
</table>
</div>
```



Interoperability:
position:sticky: works in Firefox and Safari.
position:relative: rows/sections do not become containing blocks for absolutely positioned elements in Safari

And many other tweaks to make our code more standards compliant

- <TABLE> now supports box-sizing:content-box.
- <COL><COLGROUP> supports min-width.
- <TBODY> supports height.
- <CAPTION> height:percent is treated as auto height when computing its intrinsic height.
- <CAPTION> margins are used when computing the table's minimum width.
- <TABLE> height is distributed over all <TBODY>ies. Legacy code only distributed over the 1st TBODY.
- Section/Row DOM offsetWidth/Height property interaction with border-spacing. They now do not include empty space around the row/section.
- nowrap vs <TD> width: if column width is constrained by nowrap minimum width, and <TD> width, Legacy ignored nowrap width. TablesNG does not.

- Track merging algorithm is will not merge <TD> defined columns. Legacy used to merge TD-defined columns. Spec compliant.
- Tables inside flexbox sizing has changed to be spec compliant.
- many small fixes to table sizing algorithms, especially regarding percentage handling.

Interesting bugs that were not fixed

1086927 position:sticky and collapsed borders. All collapsed borders are painted on the table, not on the cell. Therefore, sticky cell will "lose" its borders, and appear to not stick to the top with top:0px.

Post-release bugs

The worst bug reported so far is table height redistribution: Legacy code used to redistribute table height to table bodies even when their height was 0. New code does not. Workaround: give tbody { height:1px } Height redistribution over empty tbody's

Several sites reported bad layouts because they were accidentally using min-width on COL elements. Bug: min-width and table columns backward compatibility. Workaround: remove min-width from COL elements, or { col { min-width: auto !important; } to your CSS.