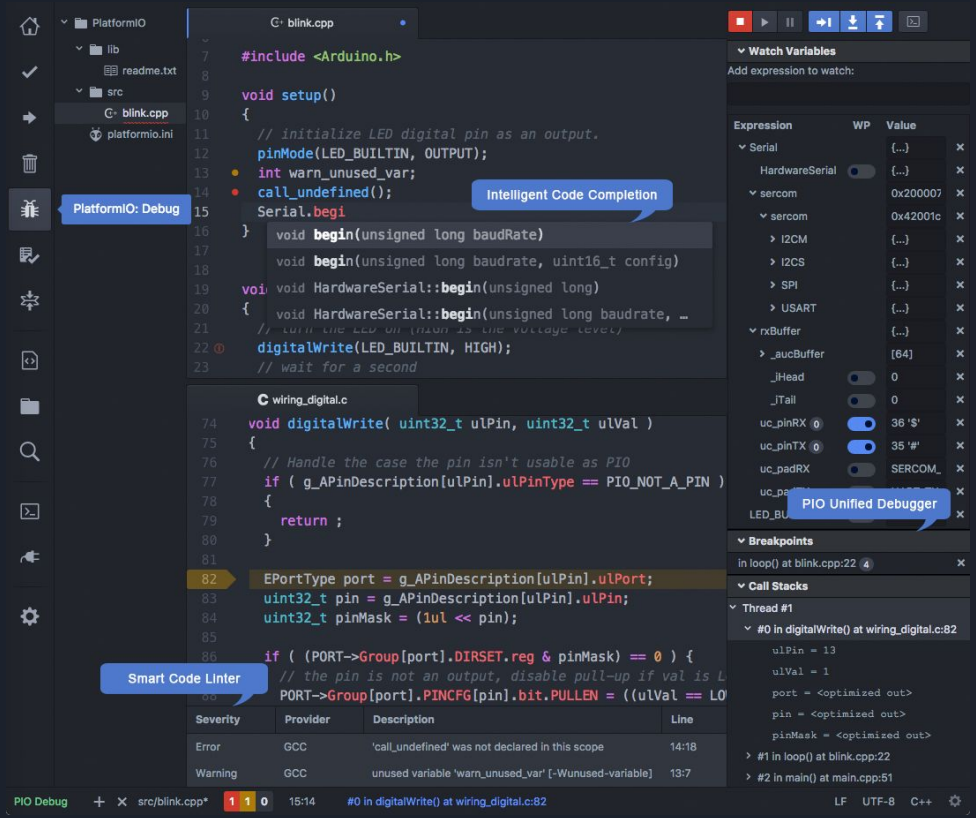
A Closer Look Into Text-Editor Bugs with Atom

By Jacob McCuddin

Atom

- Free, open-source software for text editing
- Cross-platform
- Built on Electron -a framework for building native desktop applications with web technologies such as Javascript & HTML
- Available on GitHub: https://github.com/atom/atom



Javascript

- Weakly & dynamically typed
- Object-oriented without distinction between types of objects
- Properties & methods can be added to objects dynamically
- Inheritance via “prototyping”
  - Every object points to ->a “prototype” -> null
  - Objects can extend their functionality by declaring new properties on the object, which in turn, points to the new property/object -> prototype -> null

```
function doSomething(){}
doSomething.prototype.foo = "bar";
console.log(doSomething.prototype);//doSomething { foo: 'bar' }
console.log(doSomething.prototype.prototype);//undefined (null)
```

The Gutter Bug -GitHub Issue #18086

Description: A soft-wrapped text editor's window renders WITHOUT accounting for custom gutter widths resulting in cropped text.

Only after clicking into the text editor window will the text be re-rendered

Soft-wrapping: wraps text to new line without altering line numbers -prevents need for horizontal scrolling.

Gutter: skinny column to denote useful information about the text -e.g. line number, Git insertions, etc.

~/fixtures/patches

Project

- patches
  - .git
  - added-executable
  - executable-chang
  - few-patches.txt
  - gen-large-patch.rl
  - gen-many-patche
  - large-single-patch
  - many-patches.txt
  - original

The Gutter Bug -Github Issue #18086

Reproduction Steps:

1. Disable core packages that create decorations on TextEditor creation
2. Create a text file that will be soft-wrapped at your typical screen width.
3. Add the following to your init script: atom.workspace.observeTextEditors(editor => {editor.addGutter({name: 'repro', priority: 10})});
4. Add the following to your stylesheet: .gutter[gutter-name="repro"] { width: 100px; background-color: grey; }
5. Enable soft wrapping (not at preferred line length) in TextEditors in the Settings View
6. (Re-)open Atom in safe mode.
7. Open the file you prepared in step 2.
8. Interact with the opened editor pane in any way. A mouse click anywhere will trigger this.

The Gutter Bug -Github Issue #18086

5 src/text-editor-component.js

```diff
@@ -387,8 +387,9 @@ class TextEditorComponent {
387 387   }
388 388
389 389   measureContentDuringUpdateSync () {
    390 +   let gutterDimensionsChanged = false
390 391   if (this.remeasureGutterDimensions) {
391     -   this.measureGutterDimensions()
    392 +   gutterDimensionsChanged = this.measureGutterDimensions()
392 393   this.remeasureGutterDimensions = false
393 394   }
394 395   const wasHorizontalScrollbarVisible = (
@@ -419,7 +420,7 @@ class TextEditorComponent {
419 420   this.linesToMeasure.clear()
420 421   this.measuredContent = true
421 422
422     -   return wasHorizontalScrollbarVisible !== isHorizontalScrollbarVisible
    423 +   return gutterDimensionsChanged || wasHorizontalScrollbarVisible !== isHorizontalScrollbarVisible
423 424   }
424 425
```

measureContentDuringUpdateSync() -returns true if text-editor window needs to rerender

The Cut&Paste Fold Bug -GitHub Issue #16289

Description: Cutting and pasting (1 or several) folds in the text editor window results in overlapping text

Only reproduces approximately 10-20% of the time -tends to reproduce more frequently with several code folds

Code fold: collapsing (in most cases) a scope in the code to minimize irrelevant information

```
  it 'should tokenize a heredoc with embedded javascript correctly', ->
describe 'nowdocs', ->
  it 'should tokenize a heredoc with embedded XML correctly', ->

  it 'should tokenize a heredoc with embedded SQL correctly', ->
    waitsForPromise ->
      atom.packages.activePackage('language-css')
```

```
1578 ⌊⌋
1579 ··it 'should tokenize trait correctly', ->⏎
1588 ⌊⌋
1589 ··it 'should tokenize use const correctly', ->⏎
1602 ⌊⌋
1603 ··it 'should tokenize use function correctly', ->⏎
1620 ⌊⌋
1621 ··it 'should tokenize yield correctly', ->⏎
1639 ⌊⌋
1640 ··it 'should tokenize embedded SQL in a string', ->⏎
1664 ⌊⌋
1665 ··it 'should tokenize single quoted string regex escape characters correctly', ->⏎
1674 ⌊⌋
1675 ··it 'should tokenize single quoted string regex with escaped bracket', ->⏎
1681 ⌊⌋
1682 ··it 'should tokenize opening scope of a closure correctly', ->⏎
1697 ⌊⌋
1698 ··it 'should tokenize non-function-non-control operations correctly', ->⏎
1707 ⌊⌋
1708 ··it 'should tokenize a simple heredoc correctly', ->⏎
1743 ⌊⌋
1744 ··it 'does not match incorrect heredoc terminators', ->⏎
1758 ⌊⌋
1759 ··it 'should tokenize a longer heredoc correctly', ->⏎
1792 ⌊⌋
1793 ··it 'should tokenize a longer heredoc with interpolated values and escaped characters correctly', ->⏎
1833 ⌊⌋
1834 ··it 'should tokenize a nowdoc with interpolated values correctly', ->⏎
1871 ⌊⌋
1872 ··it 'should tokenize a heredoc with embedded HTML and interpolation correctly', ->⏎
1913 ⌊⌋
1914 ··it 'should tokenize a nowdoc with embedded HTML and interpolation correctly', ->⏎
1952 ⌊⌋
1953 ··it 'should tokenize a heredoc with illegal whitespace at the end of the line correctly', ->⏎
1971 ⌊⌋
1972 ··it 'should tokenize a heredoc with embedded XML correctly', ->⏎
1998 ⌊⌋
1999 ··it 'should tokenize a nowdoc with embedded XML correctly', ->⏎
2027 ⌊⌋
2028 ··it 'should tokenize a heredoc with embedded SQL correctly', ->⏎
2060 ⌊⌋
2061 ··it 'should tokenize a nowdoc with embedded SQL correctly', ->⏎
2095 ⌊⌋
2096 ··it 'should tokenize a heredoc with embedded javascript correctly', ->⏎
2154 ⌊⌋
2155 ··it 'should tokenize a nowdoc with embedded javascript correctly', ->⏎
2217 ⌊⌋
2218 ··it 'should tokenize a heredoc with embedded json correctly', ->⏎
2256 ⌊⌋
2257 ··it 'should tokenize a nowdoc with embedded json correctly', ->⏎
2297 ⌊⌋
2298 ··it 'should tokenize a heredoc with embedded css correctly', ->⏎
2324 ⌊⌋
2325 ··it 'should tokenize a nowdoc with embedded css correctly', ->⏎
2353 ⌊⌋
2354 ··it 'should tokenize a heredoc with embedded regex escaped bracket correctly', ->⏎
2373 ⌊⌋
2374 ··it 'should tokenize a nowdoc with embedded regex escape characters correctly', ->⏎
2398 ⌊⌋
2399 ··it 'should tokenize a nowdoc with embedded regex escaped bracket correctly', ->⏎
2420 ⌊⌋
2421 ··it 'should tokenize a heredoc with embedded regex escape characters correctly', ->⏎
2443 ⌊⌋
2444 ··it 'should tokenize a heredoc with embedded regex escaped bracket correctly', ->⏎
2463 ⌊⌋
2464 ··it 'should tokenize a nowdoc with embedded regex escape characters correctly', ->⏎
2488 ⌊⌋
2489 ··it 'should tokenize a nowdoc with embedded regex escaped bracket correctly', ->⏎
2510 ⌊⌋
2511 ··describe 'punctuation', ->=
```

src/text-editor-component.js

```diff
@@ -266,9 +266,13 @@ class TextEditorComponent {
      if (useScheduler === true) {
        const scheduler = etch.getScheduler()
        scheduler.readDocument(() => {
-         this.measureContentDuringUpdateSync()
+         const restartFrame = this.measureContentDuringUpdateSync()
          scheduler.updateDocument(() => {
-           this.updateSyncAfterMeasuringContent()
+           if (restartFrame) {
+             this.updateSync(true)
+           } else {
+             this.updateSyncAfterMeasuringContent()
+           }
        })
      })
    } else {
      this.measureContentDuringUpdateSync()
      this.updateSyncAfterMeasuringContent()
    }

    this.updateScheduled = false
  }
```

void updateSync(useSchduler=false) {...}
-creates condition for recursive call by calling measureContentDuringUpdateSync() seen in previous slides

The Core Settings Restart Bug -GitHub Issue #19323

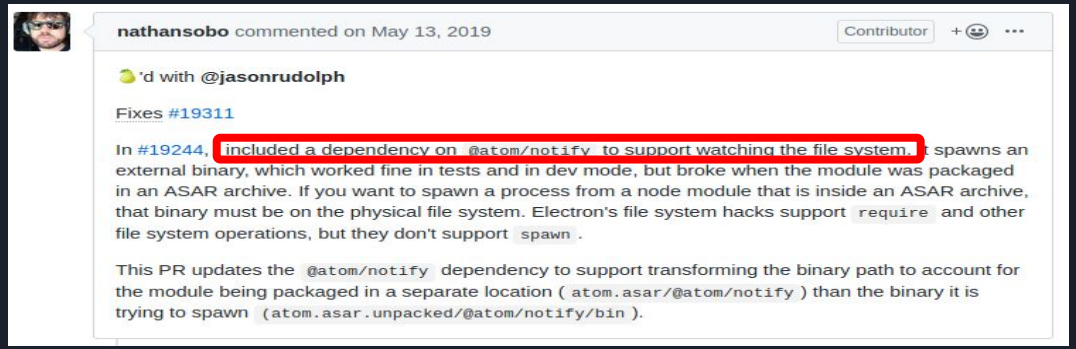
Description: Changing one of specific core settings in Atom results in a prompt to restart the application every time it launches

Reproduction:

1. Change a setting from the following list to a non-default value
    - core.titleBar
    - core.colorProfile
    - core.fileSystemWatcher
2. Restart Atom

The Core Settings Restart Bug -GitHub Issue #19323

High hopes…



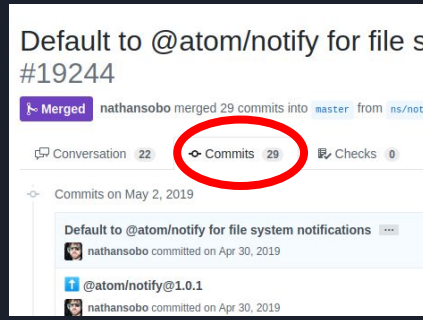
nathansobo commented on May 13, 2019 Contributor

'd with **@jasonrudolph**

Fixes #19311

In #19244, included a dependency on `@atom/notify` to support watching the file system. t spawns an external binary, which worked fine in tests and in dev mode, but broke when the module was packaged in an ASAR archive. If you want to spawn a process from a node module that is inside an ASAR archive, that binary must be on the physical file system. Electron's file system hacks support `require` and other file system operations, but they don't support `spawn`.

This PR updates the `@atom/notify` dependency to support transforming the binary path to account for the module being packaged in a separate location (`atom.asar/@atom/notify`) than the binary it is trying to spawn (`atom.asar.unpacked/@atom/notify/bin`).



Default to @atom/notify for file s
#19244

Merged nathansobo merged 29 commits into `master` from `ns/not`

Conversation 22 Commits 29 Checks 0

Commits on May 2, 2019

Default to @atom/notify for file system notifications
nathansobo committed on Apr 30, 2019

@atom/notify@1.0.1
nathansobo committed on Apr 30, 2019

However, the issue persisted…

The Core Settings Restart Bug -GitHub Issue #19323

After further debugging, a discovery was made...

Arcanemagus commented on May 16, 2019 • edited ▾ Member Author ···

I added some logging to the `onDidChange` binding and it looks like the value is being initialized to the default value (`native`) and *then* changed to the user's custom value (`atom` in my case), which is triggering the `onDidChange` event.

Edit: Confirmed that if I change the setting to `native` build based on `b79d908` .

```
272   async launch (options) {
273     if (!this.configFilePromise) {
274       this.configFilePromise = this.configFile.watch()
275
276       // TodoElectronIssue: In electron v2 awaiting the watcher causes some delay
277       // in Windows machines, which affects directly the startup time.
278       if (process.platform === 'win32') {
279         this.configFilePromise.then(disposable => this.disposable.add(disposable))
280       } else {
281         this.disposable.add(await this.configFilePromise)
282       }
283       this.config.onDidChange('core.titleBar', () => this.promptForRestart())
284       this.config.onDidChange('core.colorProfile', () => this.promptForRestart())
285       this.config.onDidChange('core.fileSystemWatcher', () => this.promptForRestart())
```

nathansobo commented on May 17, 2019 Contributor

Fixes #19323.

In #19246, we stopped awaiting the watching of the config file on Windows for performance reasons. However, directly after the `await` of the promise returned by `watch`, we call `this.config.onDidChange` with the assumption that it has already been loaded. This PR moves those observations into a `then` callback chained onto the original promise so that we only observe changes once the file is loaded, regardless of whether we call `await`. It also adds the returned disposable in the `then` callback on all platforms just to unify the logic as much as possible.

```
271 271
272 272     async launch (options) {
273 273       if (!this.configFilePromise) {
274     -       this.configFilePromise = this.configFile.watch()
    274 +       this.configFilePromise = this.configFile.watch().then(disposable => {
    275 +         this.disposable.add(disposable)
    276 +         this.config.onDidChange('core.titleBar', () => this.promptForRestart())
    277 +         this.config.onDidChange('core.colorProfile', () => this.promptForRestart())
    278 +       })
275 279
276 280         // TodoElectronIssue: In electron v2 awaiting the watcher causes some delay
277 281         // in Windows machines, which affects directly the startup time.
278     -       if (process.platform === 'win32') {
279     -         this.configFilePromise.then(disposable => this.disposable.add(disposable))
280     -       } else {
281     -         this.disposable.add(await this.configFilePromise)
    282 +       if (process.platform !== 'win32') {
    283 +         await this.configFilePromise
282 284         }
283     -       this.config.onDidChange('core.titleBar', () => this.promptForRestart())
284     -       this.config.onDidChange('core.colorProfile', () => this.promptForRestart())
285 285       }
286 286
287 287       let optionsForWindowsToOpen = []
```

The Font Family Bug -GitHub Issue #13663

In Atom, the “style.less” file allows you to customize virtually all visual asthetics of the text editor.

```
atom-workspace {
    font-family: Mononoki;
}
```

Bug Description: Syntactically incorrect CSS or a quotation mark triggers a “CssSyntaxError” when attempting to use Atom.

While appearing to be a trivial error, the bug did not appear in versions prior to Atom 1.13.0 -indicating a deeper problem.

The Font Family Bug -GitHub Issue #13663

"PostCSS is a tool used to help automate routine CSS operations"
-good Wikipedia summary

as-cii commented on Jan 20, 2017 Contributor

Fixes #13663.

This pull request fixes an error that was thrown by `postcss` when trying to upgrade deprecated selectors of a file containing malformed CSS. With these changes all the CSS files that contain an invalid syntax will be skipped and the automatic transformation won't be applied.

/cc: @atom/core

109 src/style-manager.js

```diff
@@ -250,59 +250,70 @@ module.exports = class StyleManager {
 250  250
 251  251   function transformDeprecatedShadowDOMSelectors (css, context) {
 252  252     const transformedSelectors = []
 253       -   const transformedSource = postcss.parse(css)
 254       -   transformedSource.walkRules((rule) => {
 255       -     const transformedSelector = selectorParser((selectors) => {
 256       -       selectors.each((selector) => {
 257       -         const firstNode = selector.nodes[0]
 258       -         if (context === 'atom-text-editor' && firstNode.type === 'pseudo' && firstNode.value === ':host') {
 259       -           const atomTextEditorElementNode = selectorParser.tag({value: 'atom-text-editor'})
 260       -           firstNode.replaceWith(atomTextEditorElementNode)
 261       -         }
       253  +   let transformedSource
       254  +   try {
```

as-cii on Jan 20, 2017 Author Contributor

We could wrap the whole function inside a try/catch but that would prevent v8 from optimizing it out. Given that we run this code during startup, I think that narrowing the scope of the unoptimized code that gets executed is a good idea.

Reply...

```diff
       255  +     transformedSource = postcss.parse(css)
       256  +   } catch (e) {
       257  +     transformedSource = null
       258  +   }
```

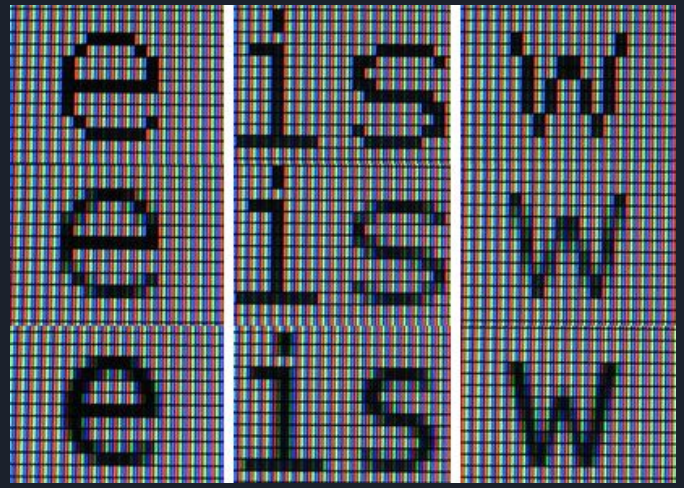
```
-        let previousNodeIsAtomTextEditor = false
-        let targetsAtomTextEditorShadow = context === 'atom-text-editor'
-        let previousNode
-        selector.each((node) => {
-          if (targetsAtomTextEditorShadow && node.type === 'class') {
-            if (DEPRECATED_SYNTAX_SELECTORS.has(node.value)) {
-              node.value = `syntax--${node.value}`
-            }
-          } else {
-            if (previousNodeIsAtomTextEditor && node.type === 'pseudo' && node.value === '::shadow') {
-              node.type = 'className'
-              node.value = '.editor'
-              targetsAtomTextEditorShadow = true
-            }
+  if (transformedSource) {
+    transformedSource.walkRules((rule) => {
+      const transformedSelector = selectorParser((selectors) => {
+        selectors.each((selector) => {
+          const firstNode = selector.nodes[0]
+          if (context === 'atom-text-editor' && firstNode.type === 'pseudo' && firstNode.value === ':host') {
+            const atomTextEditorElementNode = selectorParser.tag({value: 'atom-text-editor'})
+            firstNode.replaceWith(atomTextEditorElementNode)
          }

-          previousNode = node
-          if (node.type === 'combinator') {
-            previousNodeIsAtomTextEditor = false
-          } else if (previousNode.type === 'tag' && previousNode.value === 'atom-text-editor') {
-            previousNodeIsAtomTextEditor = true
-          }
```

The Subpixel Aliasing Bug -GitHub Issue #12652

Description: After the release of Atom version 1.3.5, the application lost all ability to produce subpixel aliasing

Subpixel aliasing: Also known as Subpixel Rendering, subpixel aliasing is the act of utilizing the physical structure of the computer's display to produce a seemingly sharper image. (works best with LCDs)

Right: photo of 3 different characters rendered in monochrome, traditional antialiasing, & subpixel rendering from top to bottom (respectively)



The Subpixel Aliasing Bug -GitHub Issue #12652

Chromium (utilized by ElectronJS) only enables subpixel aliasing when it knows the content it will be rendering will be aligned to physical pixel boundary. In other words, if a unique display resolution or scaling causes the pixel ratio's (height/width) to be a non-integer value, Chromium will not render with subpixel aliasing due to how it calculates which extra pixels to be colored.

32 src/text-editor-component.js

```
3191 3191          }
@@ -3548,10 +3548,10 @@ class LinesTileComponent {
3548 3548          style: {
3549 3549            contain: 'strict',
3550 3550            position: 'absolute',
3551      -          height: height + 'px',
3552      -          width: width + 'px',
     3551 +          height: ceilToPhysicalPixelBoundary(height) + 'px',
     3552 +          width: ceilToPhysicalPixelBoundary(width) + 'px',
3553 3553            willChange: 'transform',
3554      -          transform: `translateY(${top}px)`,
     3554 +          transform: `translateY(${roundToPhysicalPixelBoundary(top)}px)`,
3555 3555            backgroundColor: 'inherit'
3556 3556          }
3557 3557        },
@@ -4303,3 +4303,13 @@ class NodePool {
4303 4303      }
4304 4304    }
4305 4305  }
     4306 +
     4307 + function roundToPhysicalPixelBoundary (virtualPixelPosition) {
     4308 +   const virtualPixelsPerPhysicalPixel = (1 / window.devicePixelRatio)
     4309 +   return Math.round(virtualPixelPosition / virtualPixelsPerPhysicalPixel) * virtualPixelsPerPhysicalPixel
     4310 + }
     4311 +
     4312 + function ceilToPhysicalPixelBoundary (virtualPixelPosition) {
     4313 +   const virtualPixelsPerPhysicalPixel = (1 / window.devicePixelRatio)
     4314 +   return Math.ceil(virtualPixelPosition / virtualPixelsPerPhysicalPixel) * virtualPixelsPerPhysicalPixel
     4315 + }
```

Sources

Bug1: https://github.com/atom/atom/issues/19323

Bug2: https://github.com/atom/atom/issues/18086

Bug3: https://github.com/atom/atom/issues/16289

Bug4: https://github.com/atom/atom/pull/13668

Bug5: https://github.com/atom/atom/issues/12652

Questions