Computed style
what's the haps?

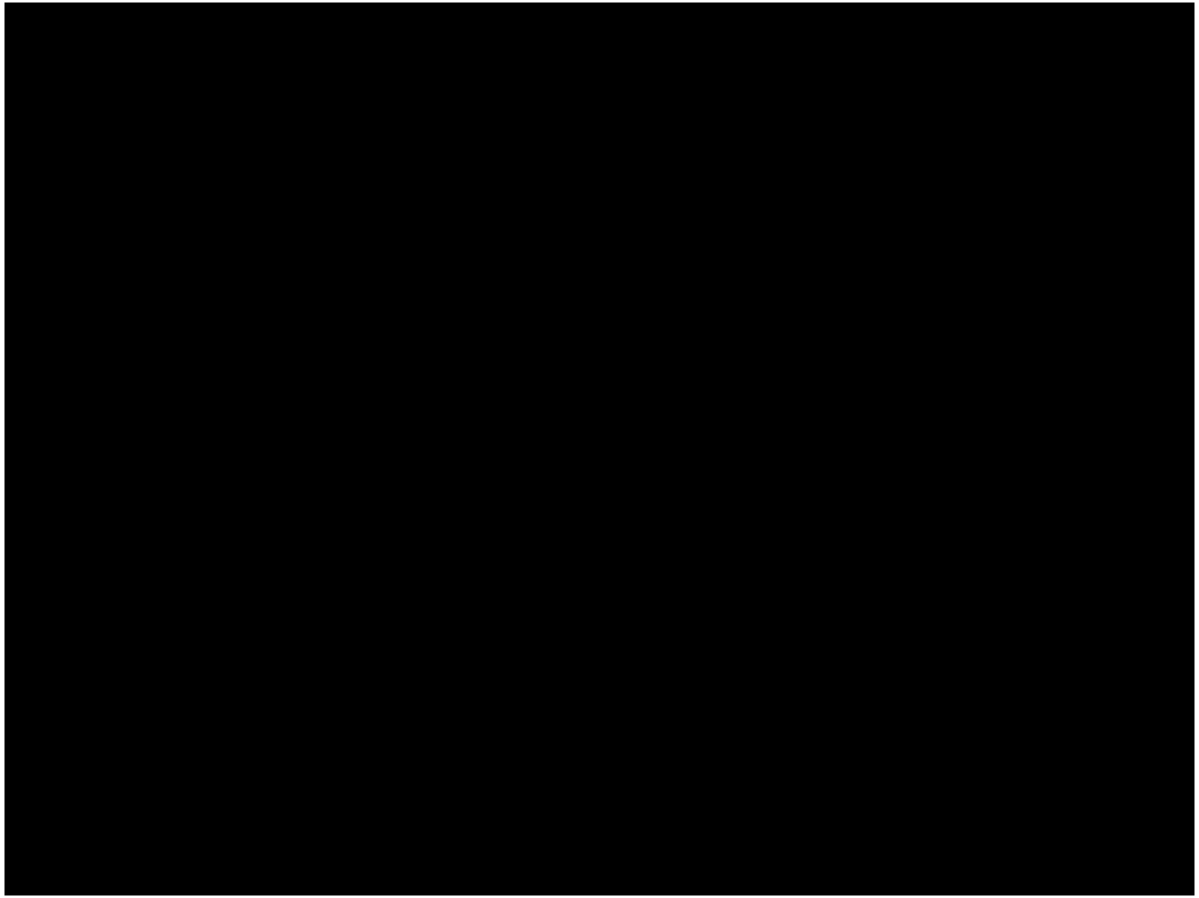
shend@ | nainar@
he/him | she/her

```
.fancy {
  color: red;
  text-decoration: underline;
}

<div class="fancy">amaze</div>
```

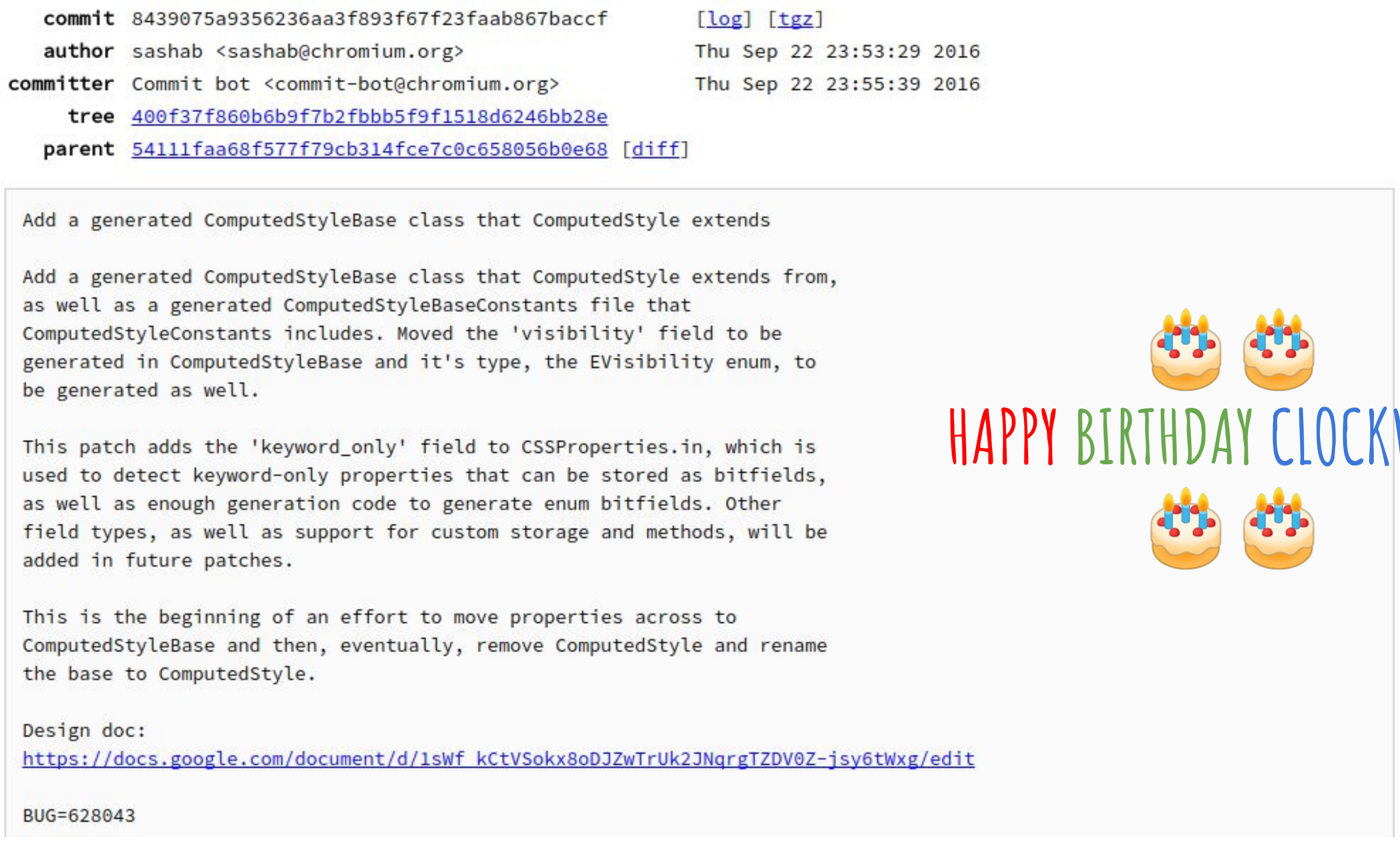
<u>amaze</u>

```
.box {
  color: red;
  text-decoration: underline;
}

<div class=“box”>amaze</div>
```

```
class ComputedStyle {
public:
  Color GetColor();
  void SetColor(Color);

  ETextDecoration GetTextDecoration();
  void SetTextDecoration(ETextDecoration);

  // ...
};
```



Quickly refactor w/o worrying about the guts of ComputedStyle



```
    commit 8439075a9356236aa3f893f67f23faab867baccf       [log] [tgz]
    author sashab <sashab@chromium.org>                   Thu Sep 22 23:53:29 2016
 committer Commit bot <commit-bot@chromium.org>           Thu Sep 22 23:55:39 2016
      tree 400f37f860b6b9f7b2fbbb5f9f1518d6246bb28e
    parent 54111faa68f577f79cb314fce7c0c658056b0e68 [diff]
```

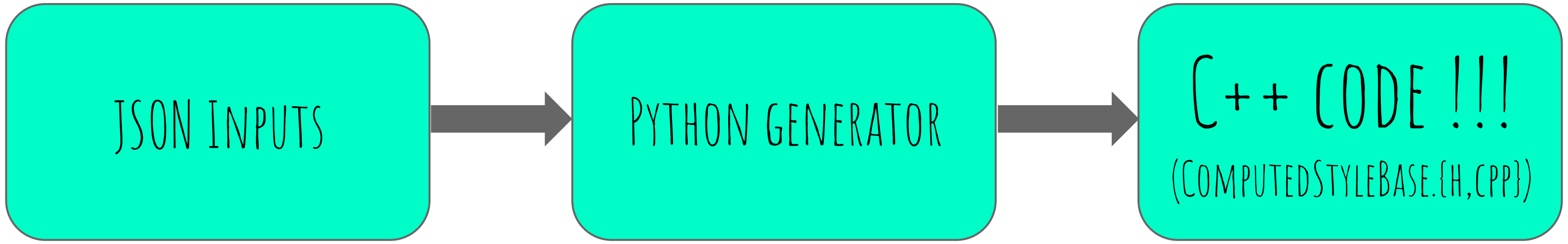
```
Add a generated ComputedStyleBase class that ComputedStyle extends

Add a generated ComputedStyleBase class that ComputedStyle extends from,
as well as a generated ComputedStyleBaseConstants file that
ComputedStyleConstants includes. Moved the 'visibility' field to be
generated in ComputedStyleBase and it's type, the EVisibility enum, to
be generated as well.

This patch adds the 'keyword_only' field to CSSProperties.in, which is
used to detect keyword-only properties that can be stored as bitfields,
as well as enough generation code to generate enum bitfields. Other
field types, as well as support for custom storage and methods, will be
added in future patches.

This is the beginning of an effort to move properties across to
ComputedStyleBase and then, eventually, remove ComputedStyle and rename
the base to ComputedStyle.

Design doc:
https://docs.google.com/document/d/1sWf_kCtVSokx8oDJZwTrUk2JNqrgTZDV0Z-jsy6tWxg/edit

BUG=628043
```

🎂 🎂

HAPPY BIRTHDAY CLOCKWORK!

🎂 🎂

What did we do?



```
{
    name: “color”,
    field_template: “external”,
    type_name: “Color”,
    include_paths: ["platform/graphics/Color.h"],
    default_value: “Color::kBlack”,
}
```

```cpp
static Color InitialColor() { return Color::kBlack; }

Color GetColor() const { return color_; }

void SetColor(Color v) { color_ = v; }



// later...
Color color_;
```

```
{
    name: “float”,
    field_template: “keyword”,
    keywords: [“none”, “left”, “right”],
    default_value: “none”
}
```

```
static EFloat InitialFloat() {
  return EFloat::kNone;
}

EFloat GetFloat() const {
  return static_cast<EFloat>(float_);
}

void SetFloat(EFloat v) {
  float_ = static_cast<unsigned>(v);
}

// later...
unsigned float_ : 2;
```

```
enum class EFloat {
    kNone,
    kLeft,
    kRight
};
```

Benefits

Less boilerplate

Prevent subtle bugs (e.g. typos)

Rapid prototyping

Challenges

Special cases

Shorthands are not properties

Is it a bug or not a bug?

About those experiments...

ComputedStyle Structure

Properties are **not all directly** stored on ComputedStyle

Groupings don’t ALWAYS make sense

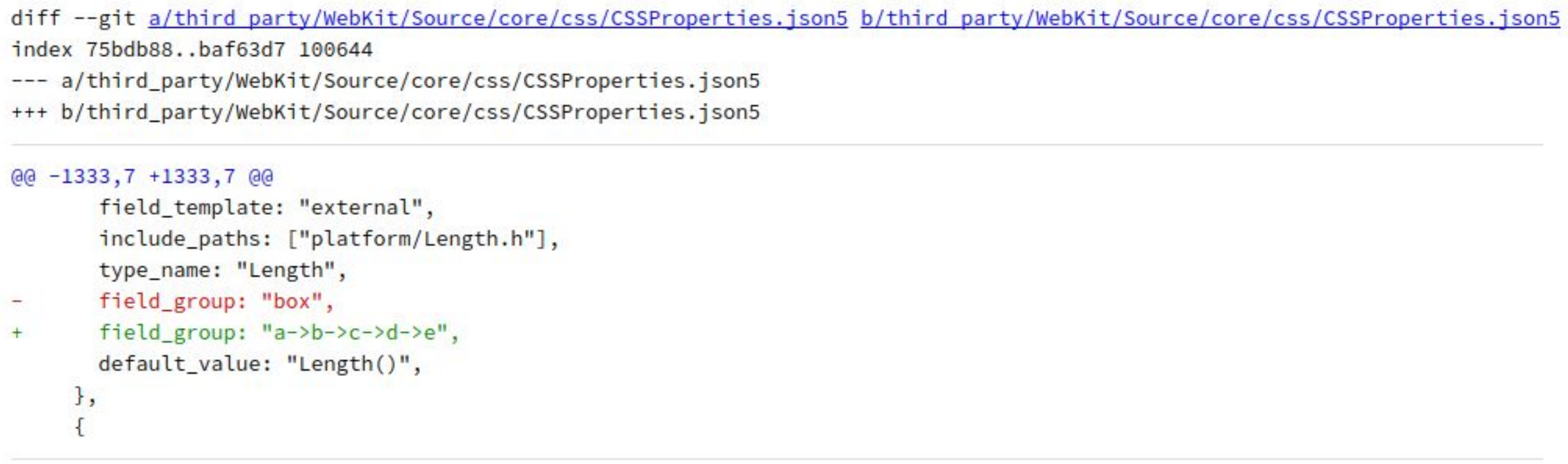
Some groups are ENORMOUS!

Before we can experiment

Everything depends on how properties are grouped!

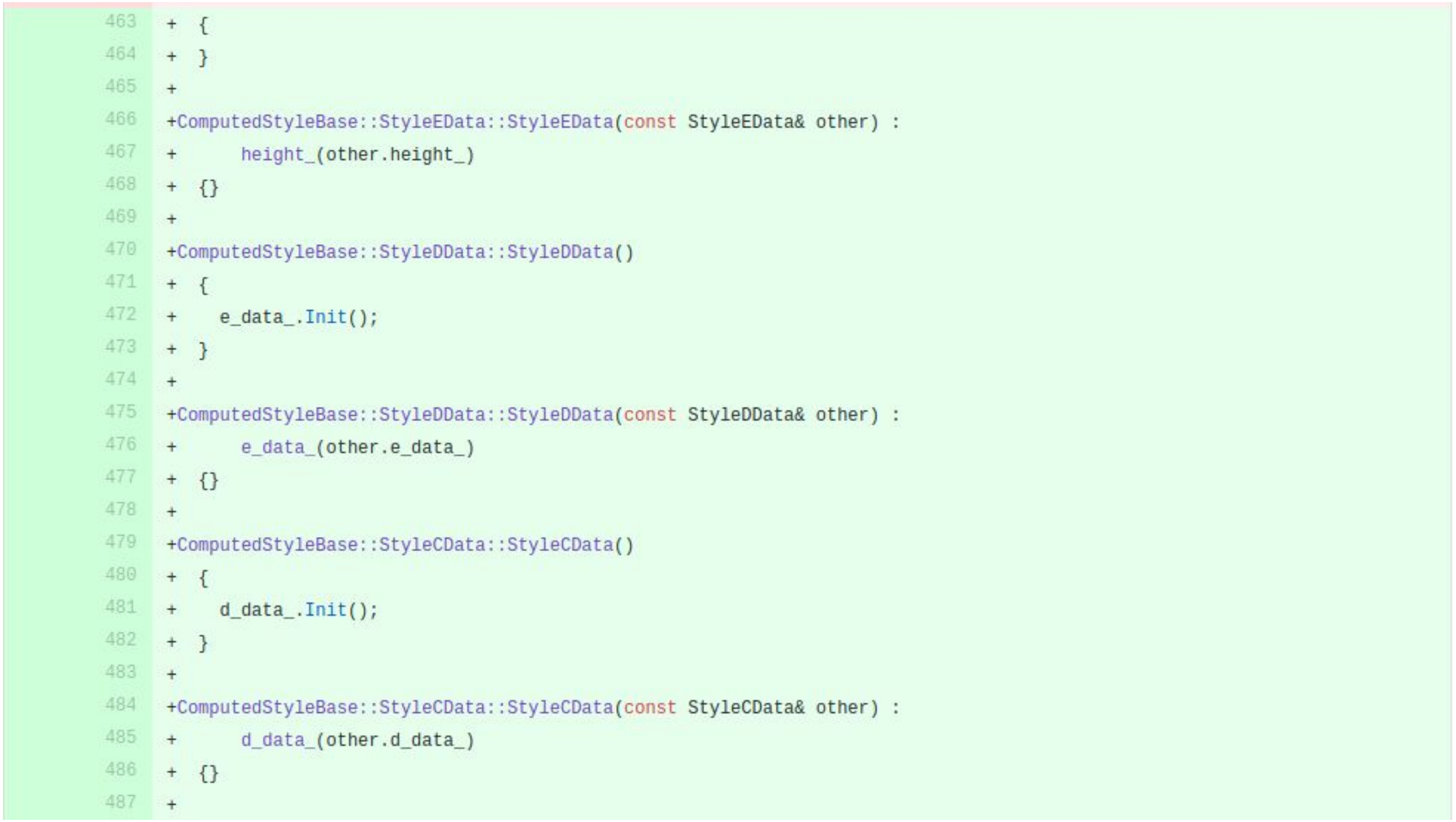
ComputedStyle diffing functions

Change where "height" is stored

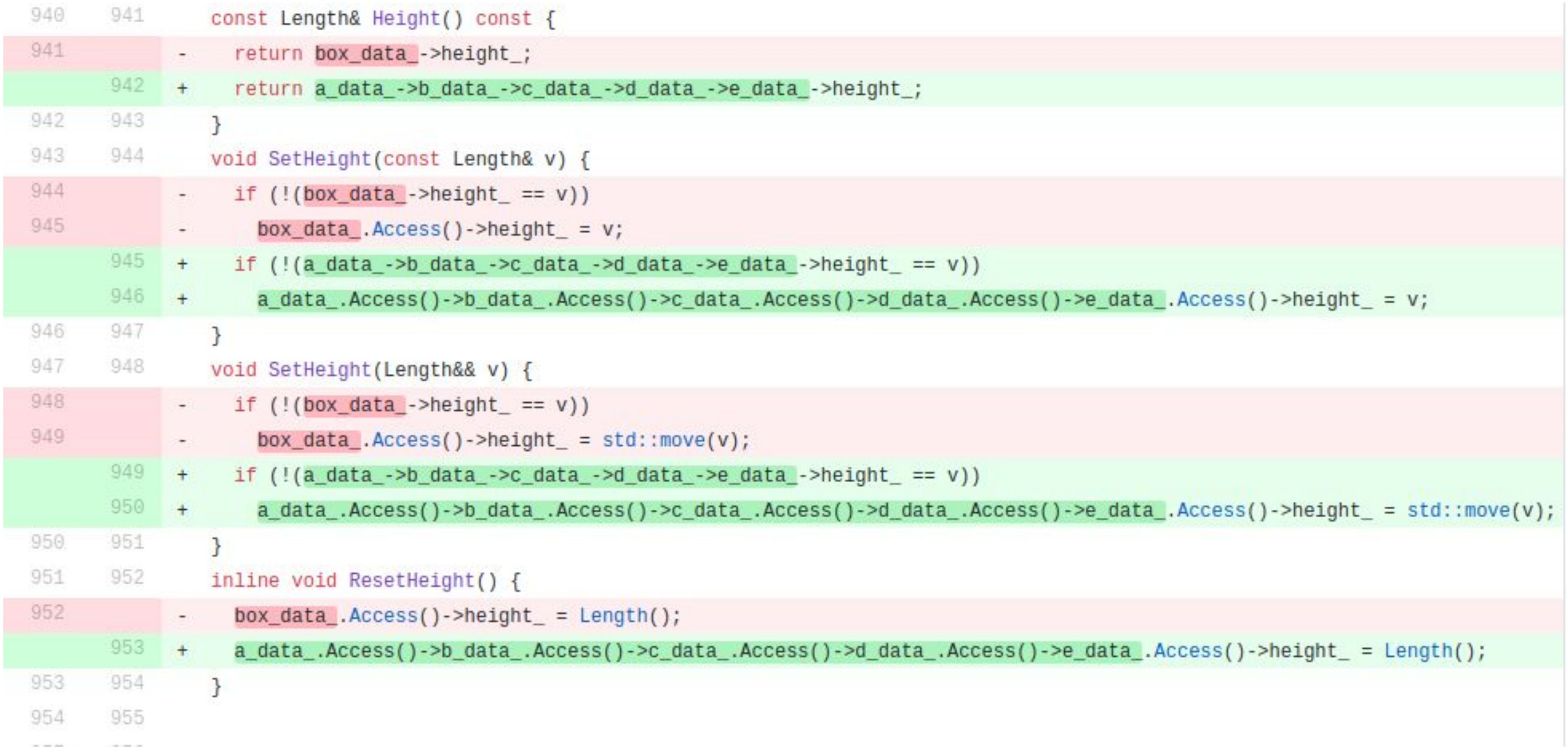


```
diff --git a/third_party/WebKit/Source/core/css/CSSProperties.json5 b/third_party/WebKit/Source/core/css/CSSProperties.json5
index 75bdb88..baf63d7 100644
--- a/third_party/WebKit/Source/core/css/CSSProperties.json5
+++ b/third_party/WebKit/Source/core/css/CSSProperties.json5
```

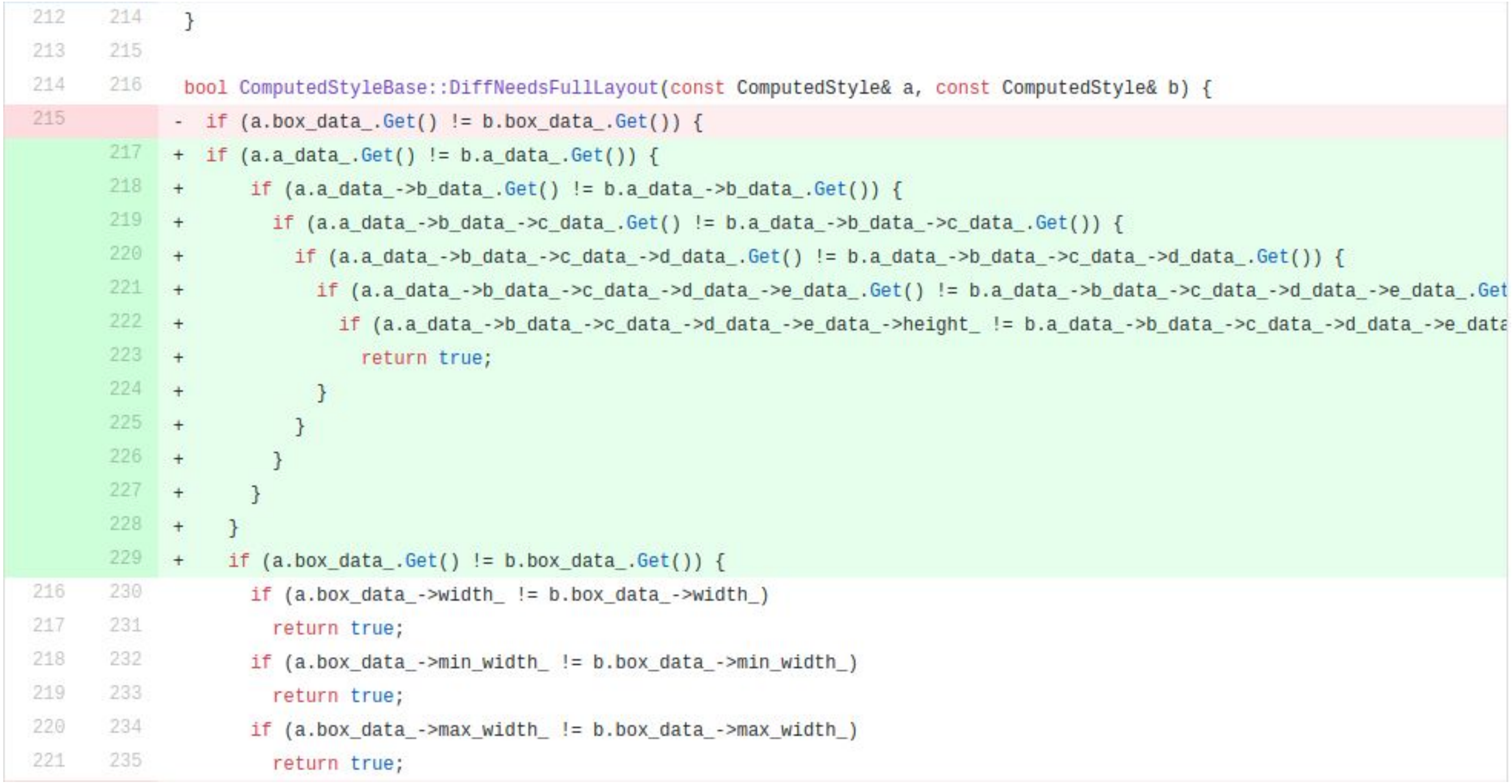
```
@@ -1333,7 +1333,7 @@
       field_template: "external",
       include_paths: ["platform/Length.h"],
       type_name: "Length",
-      field_group: "box",
+      field_group: "a->b->c->d->e",
       default_value: "Length()",
     },
     {
```



```
+  {
+  }
+
+ComputedStyleBase::StyleEData::StyleEData(const StyleEData& other) :
+      height_(other.height_)
+  {}
+
+ComputedStyleBase::StyleDData::StyleDData()
+  {
+    e_data_.Init();
+  }
+
+ComputedStyleBase::StyleDData::StyleDData(const StyleDData& other) :
+      e_data_(other.e_data_)
+  {}
+
+ComputedStyleBase::StyleCData::StyleCData()
+  {
+    d_data_.Init();
+  }
+
+ComputedStyleBase::StyleCData::StyleCData(const StyleCData& other) :
+      d_data_(other.d_data_)
+  {}
+
```



```diff
   const Length& Height() const {
-    return box_data_->height_;
+    return a_data_->b_data_->c_data_->d_data_->e_data_->height_;
   }
   void SetHeight(const Length& v) {
-    if (!(box_data_->height_ == v))
-      box_data_.Access()->height_ = v;
+    if (!(a_data_->b_data_->c_data_->d_data_->e_data_->height_ == v))
+      a_data_.Access()->b_data_.Access()->c_data_.Access()->d_data_.Access()->e_data_.Access()->height_ = v;
   }
   void SetHeight(Length&& v) {
-    if (!(box_data_->height_ == v))
-      box_data_.Access()->height_ = std::move(v);
+    if (!(a_data_->b_data_->c_data_->d_data_->e_data_->height_ == v))
+      a_data_.Access()->b_data_.Access()->c_data_.Access()->d_data_.Access()->e_data_.Access()->height_ = std::move(v);
   }
   inline void ResetHeight() {
-    box_data_.Access()->height_ = Length();
+    a_data_.Access()->b_data_.Access()->c_data_.Access()->d_data_.Access()->e_data_.Access()->height_ = Length();
   }
```



```
 }

 bool ComputedStyleBase::DiffNeedsFullLayout(const ComputedStyle& a, const ComputedStyle& b) {
- if (a.box_data_.Get() != b.box_data_.Get()) {
+ if (a.a_data_.Get() != b.a_data_.Get()) {
+   if (a.a_data_->b_data_.Get() != b.a_data_->b_data_.Get()) {
+    if (a.a_data_->b_data_->c_data_.Get() != b.a_data_->b_data_->c_data_.Get()) {
+     if (a.a_data_->b_data_->c_data_->d_data_.Get() != b.a_data_->b_data_->c_data_->d_data_.Get()) {
+      if (a.a_data_->b_data_->c_data_->d_data_->e_data_.Get() != b.a_data_->b_data_->c_data_->d_data_->e_data_.Get
+       if (a.a_data_->b_data_->c_data_->d_data_->e_data_->height_ != b.a_data_->b_data_->c_data_->d_data_->e_data
+        return true;
+      }
+     }
+    }
+   }
+  }
+  if (a.box_data_.Get() != b.box_data_.Get()) {
    if (a.box_data_->width_ != b.box_data_->width_)
     return true;
    if (a.box_data_->min_width_ != b.box_data_->min_width_)
     return true;
    if (a.box_data_->max_width_ != b.box_data_->max_width_)
     return true;
```

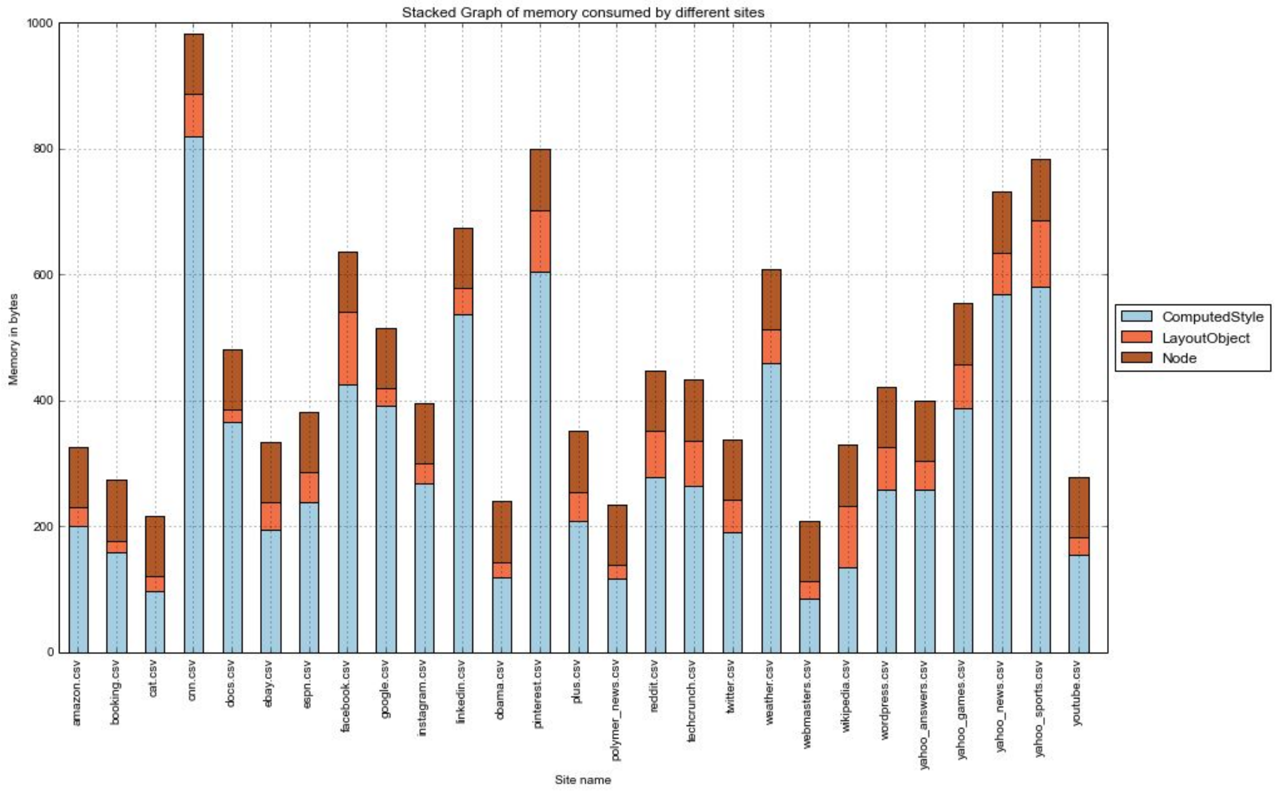
Understanding memory

Memory consumption: ComputedStyle vs Node vs LayoutObject

Just generate print statements - **easy peasy!**

But… write Python scripts that analyze the deluge of prints

Computed Style vs Node vs LayoutObject

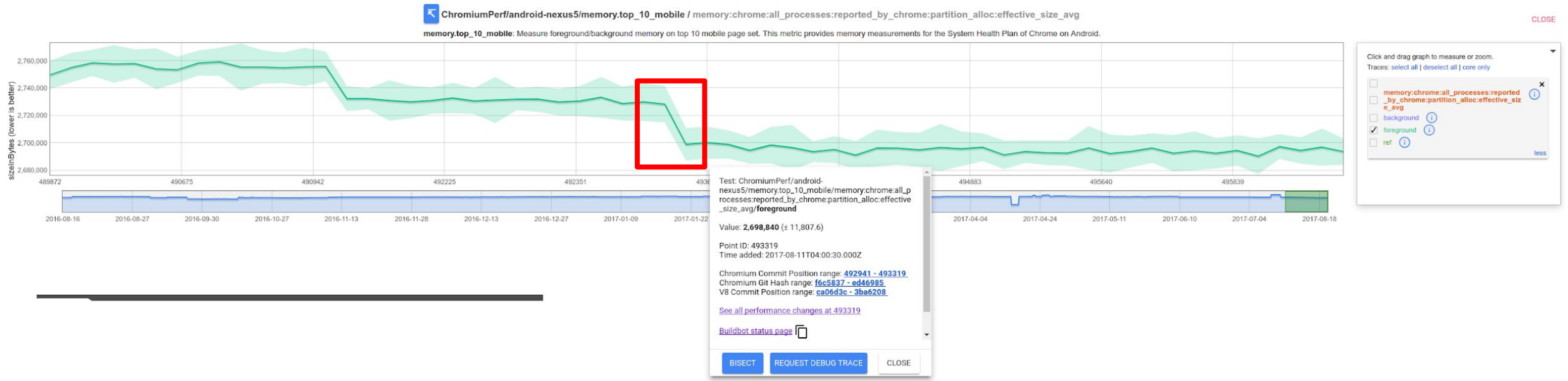


Groups on ComputedStyle

Amazing intern over the past 3 months - wave to Minh Duc everyone!

Worked on finding an optimum grouping for ComputedStyle

Aim: A ComputedStyle that adapts to the kind of pages that developers are writing



ChromiumPerf/android-nexus5/memory.top_10_mobile / memory:chrome:all_processes:reported_by_chrome:partition_alloc:effective_size_avg

memory.top_10_mobile: Measure foreground/background memory on top 10 mobile page set. This metric provides memory measurements for the System Health Plan of Chrome on Android.

Test: ChromiumPerf/android-nexus5/memory.top_10_mobile/memory:chrome:all_processes:reported_by_chrome:partition_alloc:effective_size_avg/**foreground**

Value: **2,698,840** (± 11,807.6)

Point ID: 493319
Time added: 2017-08-11T04:00:30.000Z

Chromium Commit Position range: 492941 - 493319
Chromium Git Hash range: f6c5837 - ed46985
V8 Commit Position range: ca06d3c - 3ba6208

See all performance changes at 493319

Buildbot status page

Click and drag graph to measure or zoom.
Traces: select all | deselect all | core only

- [ ] memory:chrome:all_processes:reported_by_chrome:partition_alloc:effective_size_avg
- [ ] background
- [✓] foreground
- [ ] ref

Conclusion

Before: ComputedStyle was a handwritten class > 7000 lines

Now: Modifiable via JSON, feel free to experiment away!

You're welcome

Image courtesy Disney 2016

Different types of properties