(后) 现代前端 CSS 技术

喵喵

2020.10

喵喵的 (后) 现代 CSS

每个人的 CSS 写法都不一样

喵喵喜欢的功能和实践会着重于以下几点特征:

- 实现的性能 (60fps)
- 代码可维护性
- 最好存在 Fallback

What's in the box

- CSS Basics
- 一些新功能
  - CSSOM/Houdini, WAAPI, etc.
- 喵喵的 practice
  - SASS, BEM, etc.

What's in the box

- CSS Basics
- 一些新功能
  - CSSOM/Houdini, WAAPI, etc.
- 喵喵的 practice
  - SASS, BEM, etc.

例子：

- `https://meow.plus`
- `https://gust.construction`

CSS Optimization Basics


```
.jiege + .meow {
  opacity: 0;
  transition: opacity .2s ease;
}
```

CSS Optimization Basics


```
.jiege + .meow {
  opacity: 0;
  transition: opacity .2s ease;
}
```

```
<main>
  <h1 class="jiege">God himself</h1>
  <small class="meow">Meow</small>
</main>
```

CSS Optimization Basics (Cont.)[1]

- 找到 CSS 属性发生变化的元素: `main > small.meow`

[1] https://developers.google.com/web/fundamentals/performance/rendering

CSS Optimization Basics (Cont.)[1]

- 找到 CSS 属性发生变化的元素: `main > small.meow`
- 计算 Diff，并且观察是不是有 Transition / Animation

[1] https://developers.google.com/web/fundamentals/performance/rendering

CSS Optimization Basics (Cont.)[1]

- 找到 CSS 属性发生变化的元素: `main > small.meow`
- 计算 Diff，并且观察是不是有 Transition / Animation
- 本次重绘 Tick（如果是持续变化，那么在接下来的每一个 Tick）：

[1] https://developers.google.com/web/fundamentals/performance/rendering

CSS Optimization Basics (Cont.)[1]

- 找到 CSS 属性发生变化的元素: `main > small.meow`
- 计算 Diff，并且观察是不是有 Transition / Animation
- 本次重绘 Tick（如果是持续变化，那么在接下来的每一个 Tick）：
  - 计算 Style set
  - Layout: 重新计算元素的尺寸
  - Paint: 重新绘制元素
  - Composite: 将新绘制的元素叠加到图层上

[1] `https://developers.google.com/web/fundamentals/performance/rendering`

CSS Optimization Basics (Cont.)[1]

- 找到 CSS 属性发生变化的元素: `main > small.meow`
- 计算 Diff，并且观察是不是有 Transition / Animation
- 本次重绘 Tick（如果是持续变化，那么在接下来的每一个 Tick）：
  - 计算 Style set
  - Layout: 重新计算元素的尺寸
  - Paint: 重新绘制元素
  - Composite: 将新绘制的元素叠加到图层上

Layout 和 Paint 可以在不必须的情况下被省去（e.g. `opacity` 的变化）

[1] `https://developers.google.com/web/fundamentals/performance/rendering`

CSS Optimization Basics (Cont.)[1]

- 找到 CSS 属性发生变化的元素: `main > small.meow`
- 计算 Diff，并且观察是不是有 Transition / Animation
- 本次重绘 Tick（如果是持续变化，那么在接下来的每一个 Tick）：
  - 计算 Style set
  - Layout: 重新计算元素的尺寸
  - Paint: 重新绘制元素
  - Composite: 将新绘制的元素叠加到图层上

Layout 和 Paint 可以在不必须的情况下被省去（e.g. `opacity` 的变化）

在大量使用 Transition 的场景下，Layout 是非常慢的。Paint 会消耗更多的 GPU 带宽，相比 Composite 也更慢，最好也省去。

[1] `https://developers.google.com/web/fundamentals/performance/rendering`

选定过渡属性

稍稍过时的数据：https://csstriggers.com/

- transform
- opacity
- filter[2]
- clip-path[3]
- backdrop-filter??

[2] https://www.chromium.org/developers/design-documents/image-filters

[3] https://groups.google.com/a/chromium.org/g/paint-dev/c/3bXUo0X3C5I

产生了一些问题...

目标效果：一个容器的宽度连续变化。

产生了一些问题...

目标效果：一个容器的宽度连续变化。

- 可以用 `width`，但是会很慢。

产生了一些问题...

目标效果：一个容器的宽度连续变化。

- 可以用 `width`，但是会很慢。
- `scale`？最开始会有一次突变

产生了一些问题...

目标效果：一个容器的宽度连续变化。

- 可以用 `width`，但是会很慢。
- `scale`? 最开始会有一次突变
- 在父亲元素上用 `clip-path`，在孩子元素上用 `translate`

产生了一些问题...

目标效果：一个容器的宽度连续变化。

- 可以用 `width`，但是会很慢。
- `scale`? 最开始会有一次突变
- 在父亲元素上用 `clip-path`，在孩子元素上用 `translate`

怎么让“宽度”这个变量变化？

产生了一些问题... (Cont.)

朴素的方案：可以每个用到的地方分别写一个 class

```
.parent { transition: clip-path .2s ease; }
.left-child { transition: transform .2s ease; }

.parent.shrink {
  clip-path: polygon(
    50vw 0, 50vw 100vh, 100vw 100vh, 100vw 0);
}
.parent.shrink .left-child {
  transform: translateX(50vw);
}
```

产生了一些问题... (Cont.)

朴素的方案：可以每个用到的地方分别写一个 class

```
.parent { transition: clip-path .2s ease; }
.left-child { transition: transform .2s ease; }

.parent.shrink {
  clip-path: polygon(
    50vw 0, 50vw 100vh, 100vw 100vh, 100vw 0);
}
.parent.shrink .left-child {
  transform: translateX(50vw);
}
```

如果共享的状态涉及大量元素呢，或者有大量不同的状态呢？
e.g. 暗色模式，多种不同的宽度，etc...

CSS Variable / Custom Property


```
.parent {
  --slice: 0vw;
  clip-path: polygon(
    var(--slice) 0, var(--slice) 100vh,
    100vw 100vh, 100vw 0,
  );
  transition: clip-path .2s ease;
}
.parent.shirnk { --slice: 50vw; }
.left-child {
  transition: transform .2s ease;
  transform: translateX(var(--slice));
}
```

CSS Variable / Custom Property (Cont.)


```
--var-name: "Anything here"
```

- 可以继承
- **无类型**
- 未设置时无初始值，因此在使用的时候相当于非法属性值。

CSS Variable / Custom Property (Cont.)


```
--var-name: "Anything here"
```

- 可以继承
- **无类型**
- 未设置时无初始值，因此在使用的时候相当于非法属性值。

```
transition: display 10s ease;
```

CSS Variable / Custom Property (Cont.)


```
--var-name: "Anything here"
```

- 可以继承
- **无类型**
- 未设置时无初始值，因此在使用的时候相当于非法属性值。

```
transition: display 10s ease;
transition: --var-name 10s ease;
```

Houdini: @property


```
@property --shift-width {
  syntax: '<length>';
  inherits: true;
  initial-value: 0;
}
```

Houdini: @property


```
@property --shift-width {
  syntax: '<length>';
  inherits: true;
  initial-value: 0;
}
```

--shift-width 现在是有类型的了，因此以下过渡可以正常工作：

```
transition: --shift-width .2s ease;
```

Houdini: @property


```
@property --shift-width {
  syntax: '<length>';
  inherits: true;
  initial-value: 0;
}
```

`--shift-width` 现在是有类型的了，因此以下过渡可以正常工作：

```
transition: --shift-width .2s ease;
```

同时能保证在同一 tick，所有用到这个变量的 Transition 都是同步的。

Houdini: @property (Cont.)

能不能再给力一些？

Houdini: @property (Cont.)

能不能再给力一些？

```
.foobar {
  transition:
    --first-stage .2s 0s ease,
    --second-stage .2s .1s ease;

  transform: translateX(calc(
    var(--first-stage) + var(--second-stage)
  ));
}
```

WAAPI

Web Animation API

[WAAPI] defines a model and an API for interacting with it, for synchronization and timing of changes to the presentation of a Web page.[4]

[4] https://drafts.csswg.org/web-animations-1/

WAAPI

Web Animation API

[WAAPI] defines a model and an API for interacting with it, for synchronization and timing of changes to the presentation of a Web page.[4]

```
const animation = el.animate(
  [{ transform: 'translateY(0)' }],
  [{ transform: 'translateY(100px)' }],
  {
    duration: 1000, delay: 1000,
    easing: 'ease', fill: 'both'
  },
);
await animation.finished;
```

[4] https://drafts.csswg.org/web-animations-1/

WAAPI (Cont.)

- 减少 CSS 的 parsing 时间
- 方便浏览器优化

WAAPI (Cont.)

- 减少 CSS 的 parsing 时间
- 方便浏览器优化
- 可以并行生成多个不同属性的 Transition

WAAPI (Cont.)

- 减少 CSS 的 parsing 时间
- 方便浏览器优化
- 可以并行生成多个不同属性的 Transition
- **可以和 Houdini 一起用**!

WAAPI (Cont.)

- 减少 CSS 的 parsing 时间
- 方便浏览器优化
- 可以并行生成多个不同属性的 Transition
- **可以和 Houdini 一起用**!
  - 可以并行生成多个相同属性的 Transition

WAAPI (Cont.)

- 减少 CSS 的 parsing 时间
- 方便浏览器优化
- 可以并行生成多个不同属性的 Transition
- **可以和 Houdini 一起用！**
  - 可以并行生成多个相同属性的 Transition

非常适合用来写：FLIP，进入/离开过渡

ResizeObserver & IntersectionObserver

ResizeObserver & IntersectionObserver

检测元素的尺寸、相对于一个父元素的位移变化

ResizeObserver & IntersectionObserver

检测元素的尺寸、相对于一个父元素的位移变化

- 和 Canvas 配合使用

ResizeObserver & IntersectionObserver

检测元素的尺寸、相对于一个父元素的位移变化

- 和 Canvas 配合使用
- 和 `position: sticky` 配合使用
- ...

ResizeObserver & IntersectionObserver

检测元素的尺寸、相对于一个父元素的位移变化

- 和 Canvas 配合使用
- 和 `position: sticky` 配合使用
- ...

相比于 `resize` 和 `scroll` 事件，减少 polling。

ResizeObserver & IntersectionObserver

检测元素的尺寸、相对于一个父元素的位移变化

- 和 Canvas 配合使用
- 和 `position: sticky` 配合使用
- ...

相比于 `resize` 和 `scroll` 事件，减少 polling。

BTW: `MutationObserver`

Also worth reading:

- CSSOM
  - 可以提供 Houdini 的一个 Fallback
  - 可以在不强制 Repaint 的情况下从 CSS 得到部分属性
  - CSS Paint API[5]
- Grid Layout

[5] https://developer.mozilla.org/en-US/docs/Web/API/CSS_Painting_API/Guide

喵喵's practice

喵喵喜欢的写 CSS 的方式：

- BEM 命名方式（喵喵裁剪版）
- SCSS
- 使用 `@import`

BEM

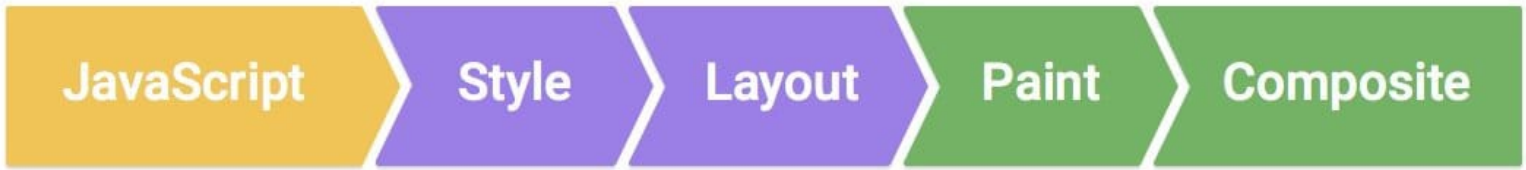
Block, Element, Modifier

```
.button--primary__text--hint { }
/* .button.primary .text.hint */
```

BEM

Block, Element, Modifier

```
.button--primary__text--hint { }
/* .button.primary .text.hint */
```



减少检查 Selector 的时间：不需要检查父子、兄弟关系。

BEM (Cont.)

例外：需要使用 Pseudo-element 的时候。

```
.container:hover .title { opacity: 1; }
.paragraph::first-leter { font-weight: 900; }
.timestamp:before { content: "TIMESTAMP >"; }
```

BEM (Cont.)

例外：需要使用 Pseudo-element 的时候。

```
.container:hover .title { opacity: 1; }
.paragraph::first-leter { font-weight: 900; }
.timestamp:before { content: "TIMESTAMP >"; }
```

减少 JS 的使用，“样式就写在样式表里，和 JS 无关”
而且还快。

SCSS

- 和 BEM 相性很好
- 有好多内置的函数（生成色板，etc.）
- @for

SCSS

- 和 BEM 相性很好
- 有好多内置的函数（生成色板，etc.）
- @for

```
@mixin with-title($active) {
  &__title {
    font-size: 1.4em;
    &--#{$active} {
      font-size: 1.8em;
    }
  }
}
.post { @include with-title("foo"); }
.list { @include with-title("bar"); }
```

Alternatives

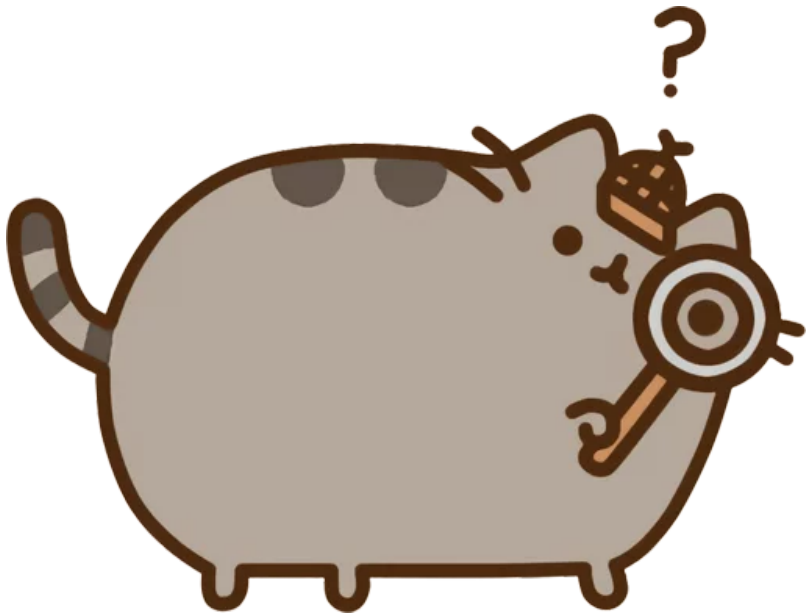
- CSS-in-JS
- styled-components

Alternatives

- CSS-in-JS
- styled-components

喵喵不用的原因：
样式变化的时候可能需要 parse 注入的 CSS，编辑器支持比较差，浏览器难以优化，难以调试，etc.

That's All!



Question time!

https://meow.c-3.moe/sth-about-jielabs

https://meow.c-3.moe/writing-meow-plus