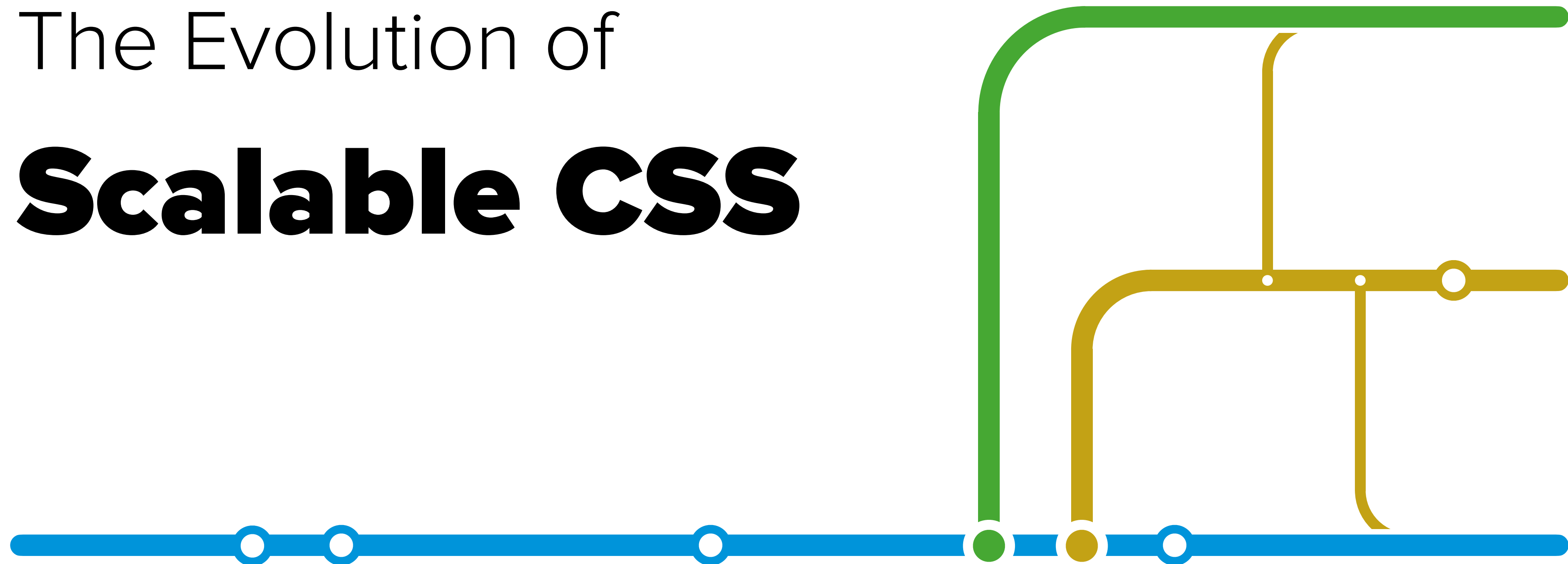



The Evolution of **Scalable CSS**






*We have to know the **past**
to understand the **present***

- Carl Sagan

*A well-researched **chronicle**,
describing how CSS **tools** and
techniques have evolved over time.*



A **chronicle** is a recording
of **significant historical events**
in the order of their occurrence,
as seen from the **chronicler's perspective**.

- Wikipedia



ANDREI PFEIFFER

Timișoara / RO



Code Designer



Co-Organizer



andreipfeiffer.dev

^sCALABLE

css

SCALABLE **CSS**

The most problematic

CSS Scalability Issues

- 💣 **Naming collisions**, due to global namespace
- ⚔️ **Specificity wars**, when overriding styles
- 🧟 **Zombie code**, non trivial to remove unused code



*It's time to talk about the elephant in the corner
of the room: **stylesheet maintainability.***

- 2005, Simon Willinson

PART 1/7

CSS Good Practices



Avoid ID selectors

//  avoid using ID selectors in CSS

```
#contact-form {}
```

//  prefer classes instead

```
.contact-form {}
```

Keep specificity low

//  avoid overly specific selectors

```
.main_menu ul li.item a.link {}
```

//  prefer low specificity

```
.main_menu .link {}
```

Practices don't really scale



No official comprehensive guide



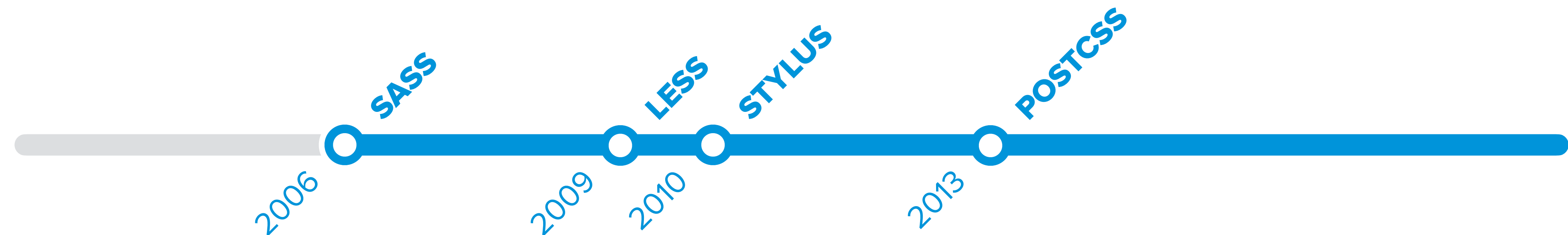
Cumbersome to learn and teach



Unscalable maintenance effort

PART 2/7

CSS Processors



Variables

Non-standard syntax, turned into standard feature

```
// SASS variable
```

```
$brand-color = #f7b228;
```

```
// CSS custom property
```

```
--brand-color: #f7b228;
```

Contextual styles

Nesting + Parent selector

// 🙌 single definition

```
.title {  
  &:hover {}  
  &::after {}  
}
```

// 🙌 verbose duplication

```
.title {}  
.title:hover {}  
.title::after {}
```


CSS Nesting Module

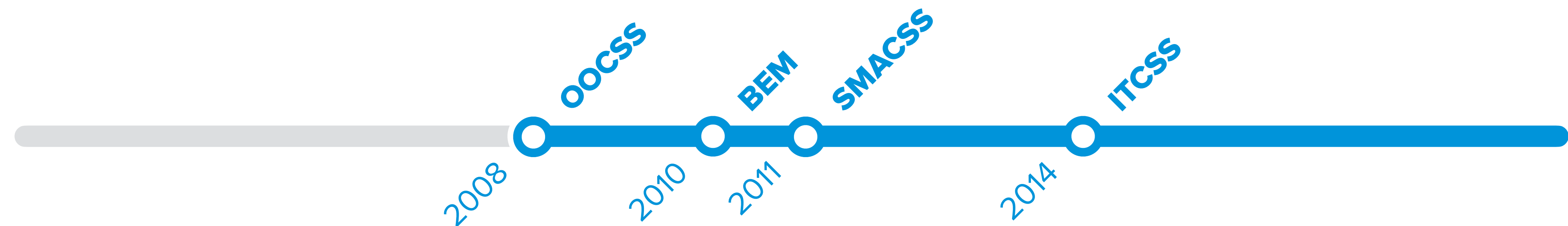
W3C First Public Working Draft, 31 August 2021

Editors: Tab Atkins-Bittner, Adam Argyle

<https://www.w3.org/TR/css-nesting-1/>

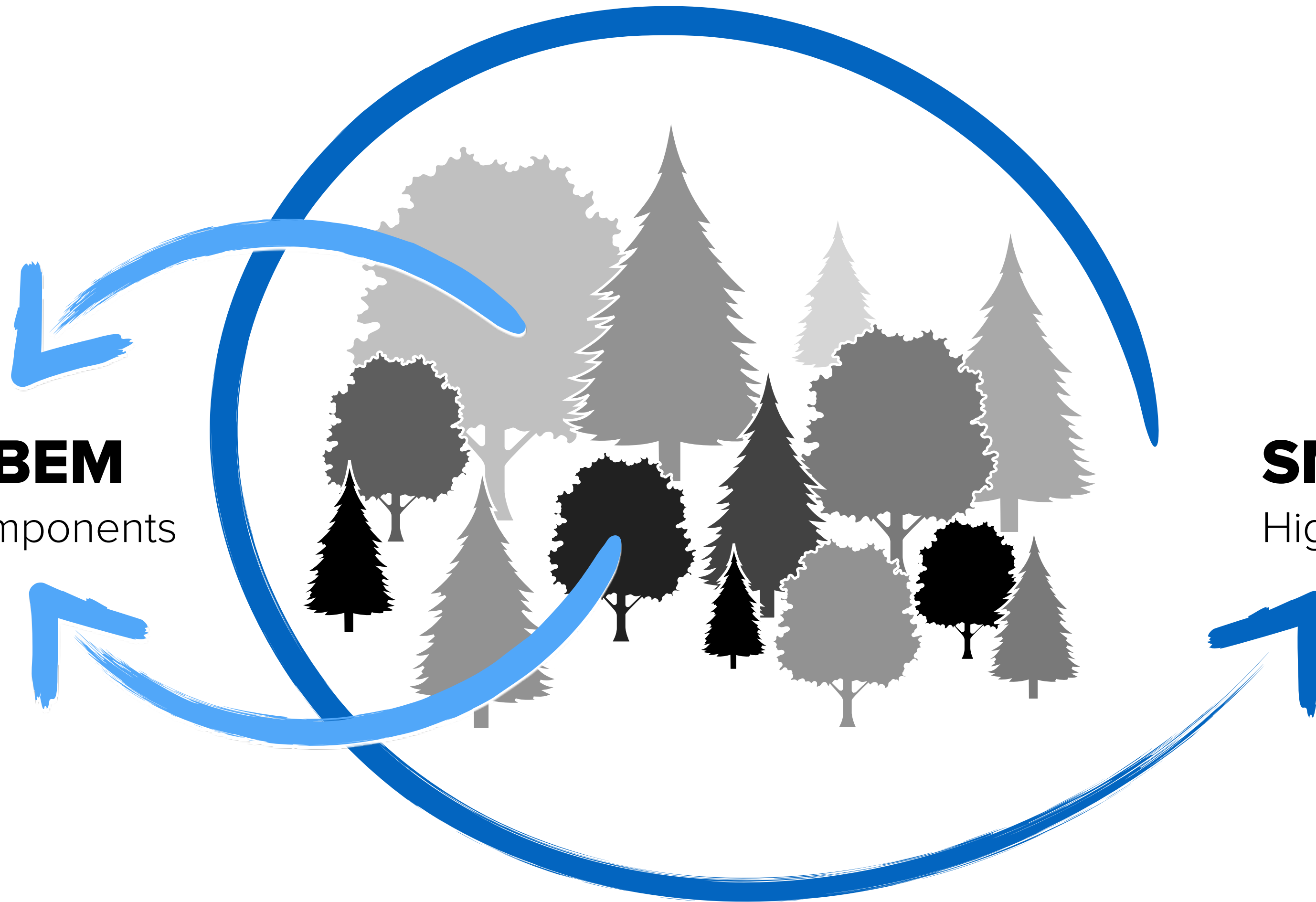
PART 3/7


CSS Methodologies



OOCSS, BEM
Individual Components

SMACSS, ITCSS
High-level Architecture





*[...] authors are encouraged to use values that describe **the nature of the content**, rather than values that describe the desired presentation.*

- HTML Living Standard

Semantic CSS

//  **Semantic** (conveys meaning)

```
<nav class="main-menu"></nav>
```

//  **Non-Semantic** (conveys implementation)

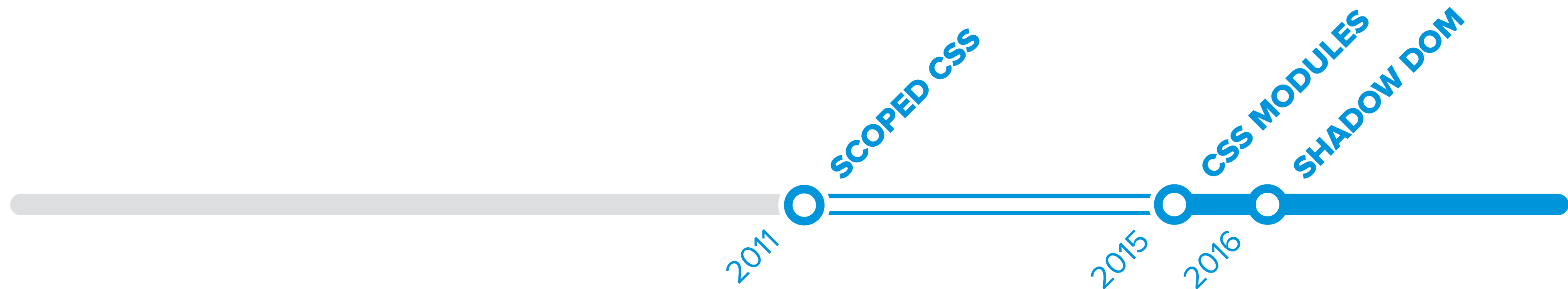
```
<nav class="flex bg-dark pad-small align-center"></nav>
```

Semantic CSS limitations

- Code repetition: `display: flex`, `font-weight: bold`, etc
- Ever growing stylesheets
- **Naming things is inherently difficult**

PART 4/7

Styles Encapsulation





Shadow DOM fixes CSS and DOM. It introduces scoped styles to the web platform, **without tools or naming conventions.**

- 2016, Eric Bidelman, [web.dev](https://web.dev/shadow-dom/)

The End of Global CSS

- Styles are scoped to their component
- No more naming collisions

Industry *de facto* standard



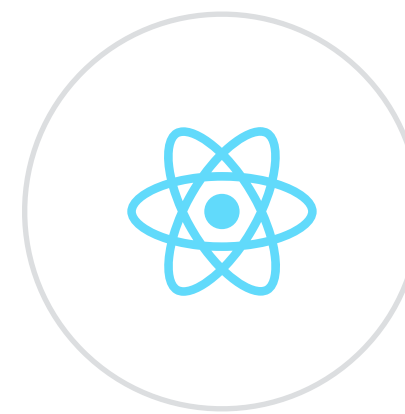
Vue

Scoped attribute
syntax



Angular

Shadow DOM
+ Emulated



React

CSS Modules
with CRA



Next

CSS Modules
out of the box

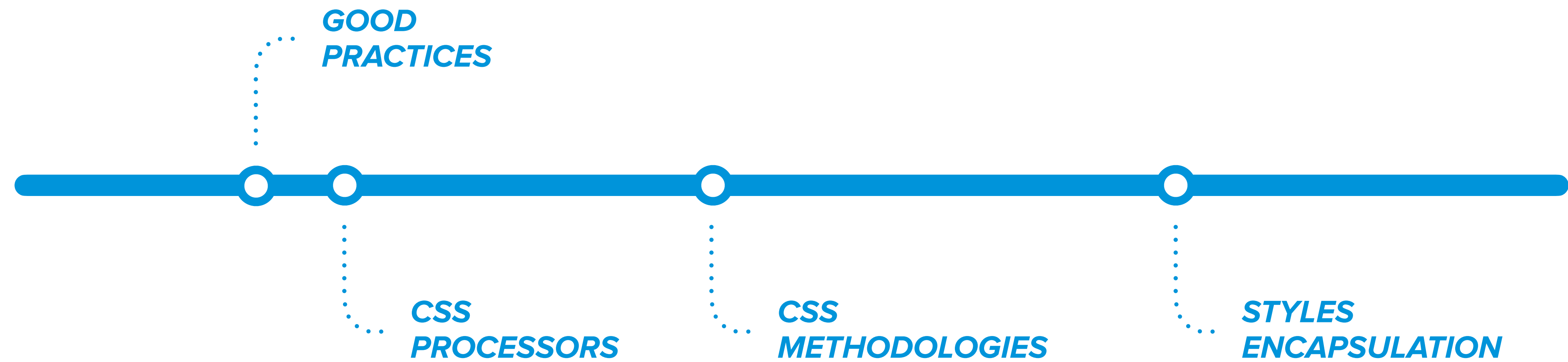


Gatsby

CSS Modules
out of the box

Semantic CSS

Traditional styling paradigm



PART 5/7

Atomic CSS



Non-Semantic classes

//  **Non-semantic** names (describing the implementation)

.text-red {}

//  **Semantic** names (describing the usage & purpose)

.text-error {}

Reusability

// 🎯 specific, less reusable

```
.title {  
  font-weight: bold;  
  color: green;  
}
```

// ♻️ highly reusable

```
.bold {  
  font-weight: bold;  
}  
.text-green {  
  color: green;  
}
```

Utility CSS classes

Also used by Semantic CSS frameworks



Colors: **.text-muted** or **.text-bg-info**



Typography: **.text-center** or **.h1** or **.bold**



Spacing: **.p-3** or **.mx-auto**

Usage

No specificity wars, no naming collisions, no duplication

```
<!-- Example from Tailwind CSS playground -->
```

```
<img class="
  relative flex min-h-screen flex-col justify-center overflow-hidden
  bg-gray-50 py-6 sm:py-12
" />
```


PART 6/7

CSS in JS



CSS in JS goals

Supported by all libraries

- Generate unique CSS classes, like CSS Modules
- Contextual styles, like CSS Processors
- Dead code elimination
- Variable sharing

CSS in JS

Synergies





style9



Emotion

Styled JSX



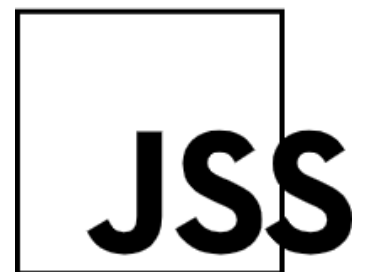
Styled Components



Stitches



Compiled



TypeStyle

Aphrodite



vanilla-extract





style9



Styled JSX



Styled Components



Emotion



A thorough analysis of CSS in JS

github.com/andreipeiffer/css-in-js



Compiled



vanilla-extract



TypeStyle

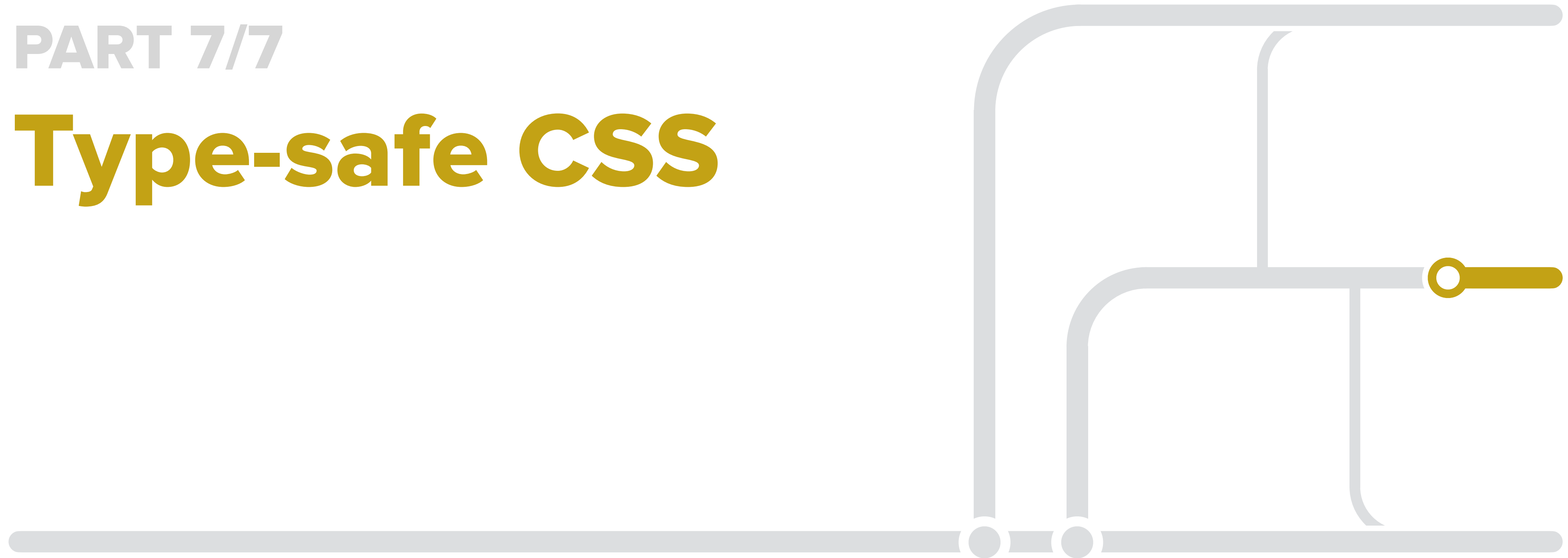
Aphrodite


Scalable CSS Paradigms



PART 7/7

Type-safe CSS





Type safety is the extent to which a programming language ***discourages or prevents type errors.***

- Wikipedia

Lack of type safety

With standard web technologies

```
.notificationStyle {  
  color: #f7b228;  
}
```

```
// 🚫 incorrect CSS class names are ignored
```

```
<div class="notification"></div>
```

Lack of type safety

With standard web technologies

```
.notificationStyle {  
  --color-warn: #f7b228;  
  
  // 🚫 misspelled CSS variables fail silently  
  color: var(--color-warning);  
}
```

Static type checkers

Non-standard tools for detecting compile time errors

- **TypeScript**, in 2012
- **Flow**, in 2015

Type checking

Without executing code

```
const colors = { warn: "#00875a", ... };
```

`colors.warning`



👉 Property 'warning' does not exist on type '{...}'

CSS in TypeScript

Prevents incorrect or misspelled styles

```
const colors = { warn: "#00875a", ... };
```

```
const notificationStyle = css({  
  color: colors.warning  
});
```

```
<div class={`${notification}`}></div>
```

👉 Cannot find name 'notification'

Typed Interfaces

Code completion and self-documented

```
const colors = { warn: "#00875a", ... };
```

```
interface NotificationProps {  
  // ✅ restricted to predefined colors  
  color: keyof typeof colors;  
}
```

Safe refactoring

Manual or automatic

- Eliminates the fear of changing existing code
- Refactoring tools: *Rename*, *Extract*, etc

Towards a type-safe future

CSS Typed Object Model

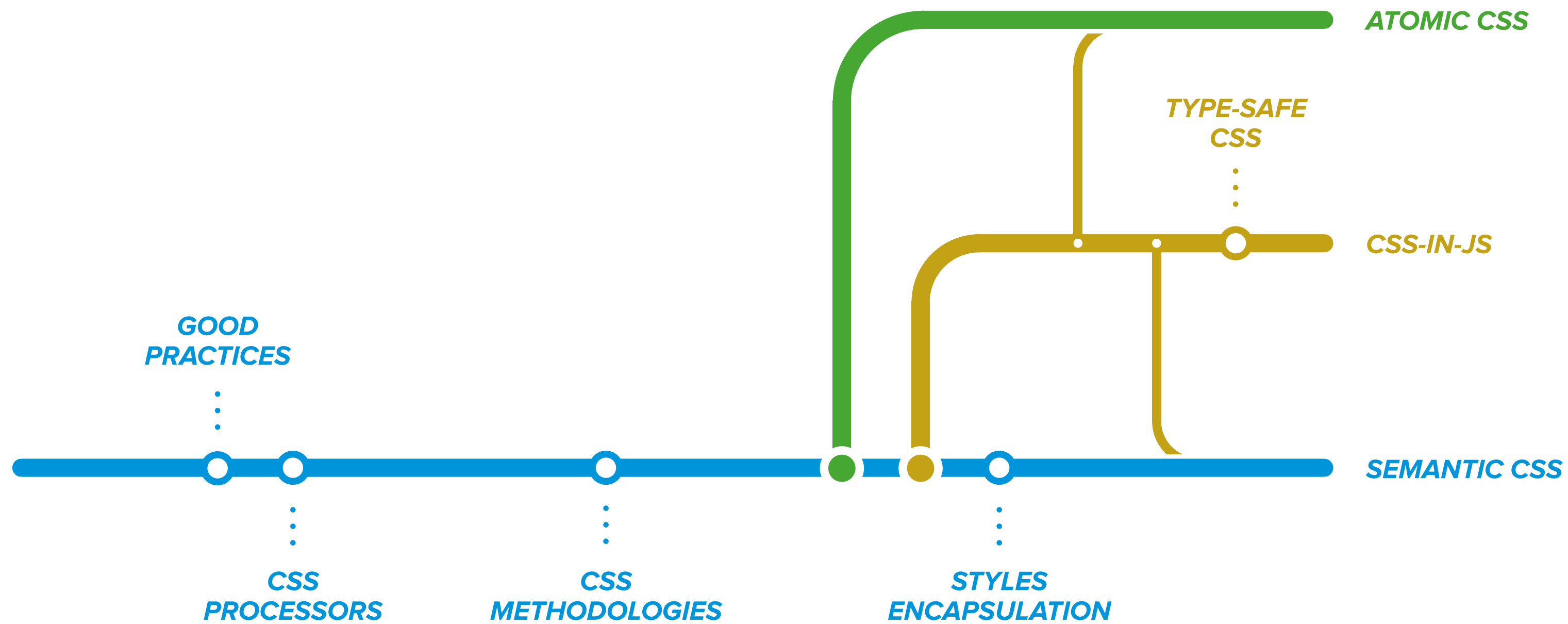
Introduced in 2018

```
element.attributeStyleMap.set("font-size", CSS.em(3));
```

ECMAScript Type Annotations

Proposed in 2022

```
function sort(list: Array<Item>, order: "asc" | "desc") {}
```

The Evolution of Scalable CSS

andreipfeiffer.dev/blog

THANK YOU



andreipfeiffer.dev