


CSS

IS

AWESOME!

Igor Laborie

Expert Java & Web, < **Monkey Patch** />

 @ilaborie

 igor@monkeypatch.io

 *Je ne suis pas un designer*

“

When designing computer systems, one is often faced with a choice between using a more or less powerful language for publishing information, for expressing constraints, or for solving some problem. This finding explores tradeoffs relating the choice of language to reusability of information. The "Rule of Least Power" suggests **choosing the least powerful language suitable** for a given purpose.

1. Texte
2. HTML (sémantique)
3. CSS (layout, style, animations simples)
4. SVG (formes et animations complexes)
5. JavaScript, WebAssembly (gestion d'états, appel backend, calculs)

!... mais il y a toujours de bonnes raisons pour ne pas suivre ces règles

- › Selectors
- › Box model
- › Float
- › Media Query
- › Animations
- › Gradients
- › Responsive Design
- › Media
- › Variables
- › Colors
- › Shapes
- › ...

- I. Utiliser un pré-processeur ?
- II. Unités
- III. Flexbox et Grid
- IV. Pseudo éléments
- V. Animations
- VI. Pseudo classes d'état
- VII. HTML
- VIII. Conclusion

UTILISER UN PRÉ-PROCESSEUR ?

```
button {  
  background: lightblue;  
  color: purple;  
  /*border: medium solid currentColor;*/  
  border: medium solid rgba(0,0,0,.42);  
}  
button.danger {  
  background: salmon;  
  color: rebeccapurple;  
}
```



Plop





Danger !

Alors utilise-t-on un pré-processeurs ?

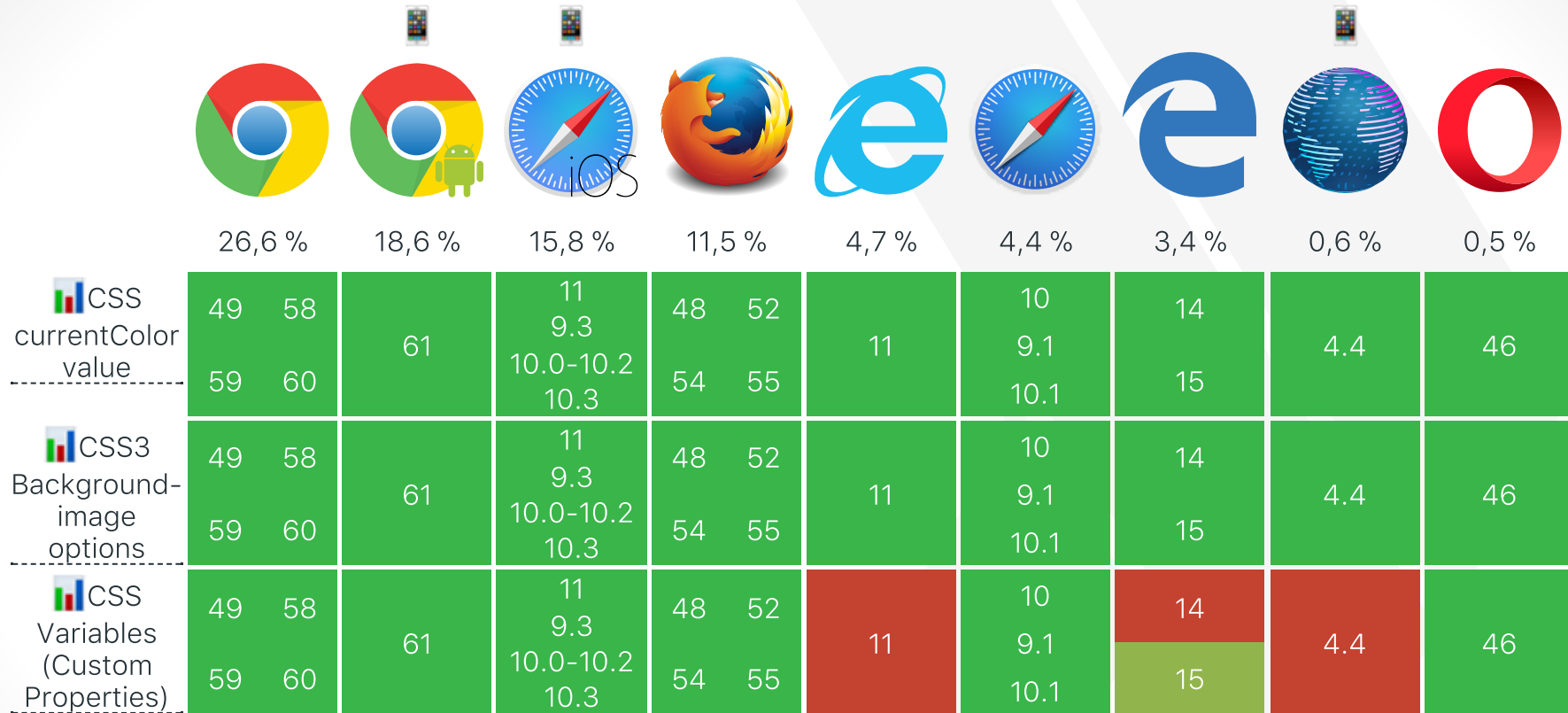


Oui, mais privilégiez:

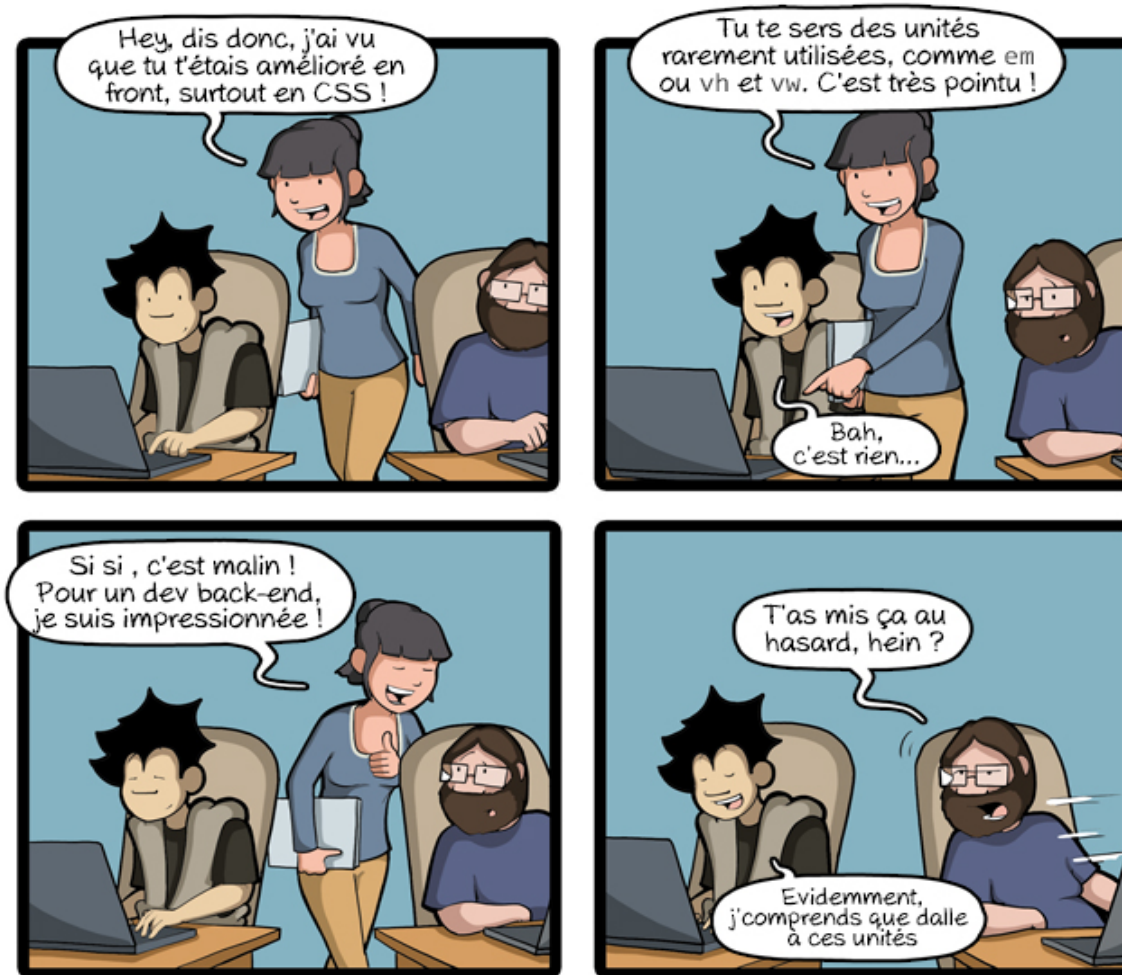
- › le CSS
- › les post-processeurs

- ›  currentColor
- ›  background-origin
- › **w3c** CSS Variables (aka Custom Properties)
- › **w3c** CSS Color Module Level 4


Navigateurs, usage ≥ 0,4% en France



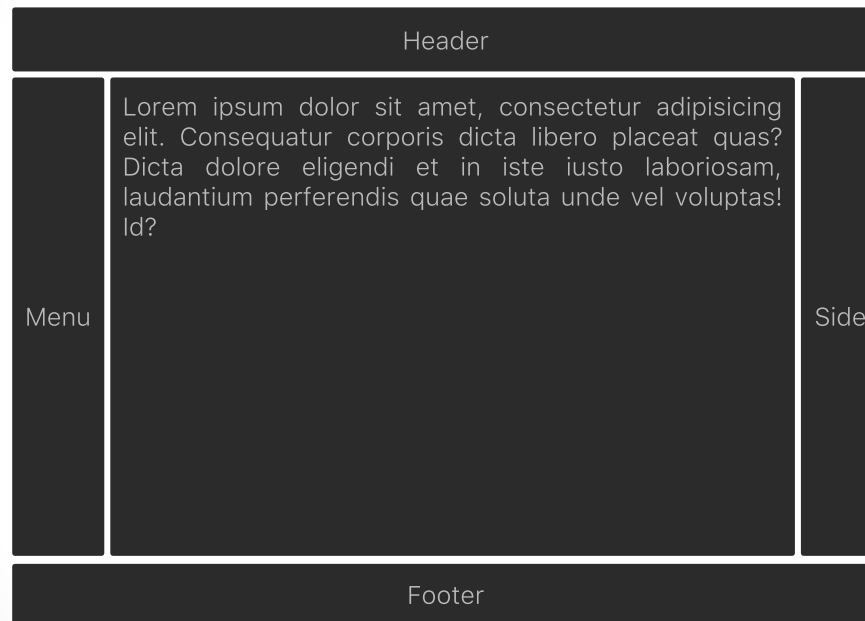
UNITÉS







CommitStrip.com

 CommitStrip

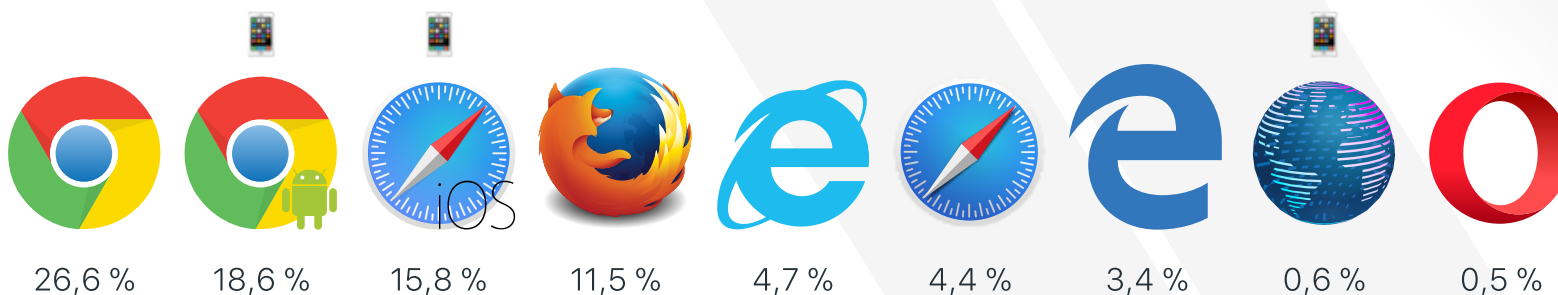
px, cm, pt, ...	longueurs absolues (mesure physique)
em, rem	fonction de la <code>font-size</code>
ex, ch	hauteur d'un x, largeur d'un 0
vh, vw	(100vh, 100vw) = (hauteur, largeur) du <i>viewport</i>
vmin, vmax	$\min(1vh, 1vw)$, $\max(1vh, 1vw)$



```
<body>
  <header>Header</header>
  <div>
    <nav>Menu</nav>
    <main>Content</main>
    <aside>Side</aside>
  </div>
  <footer>Footer</footer>
</body>
```

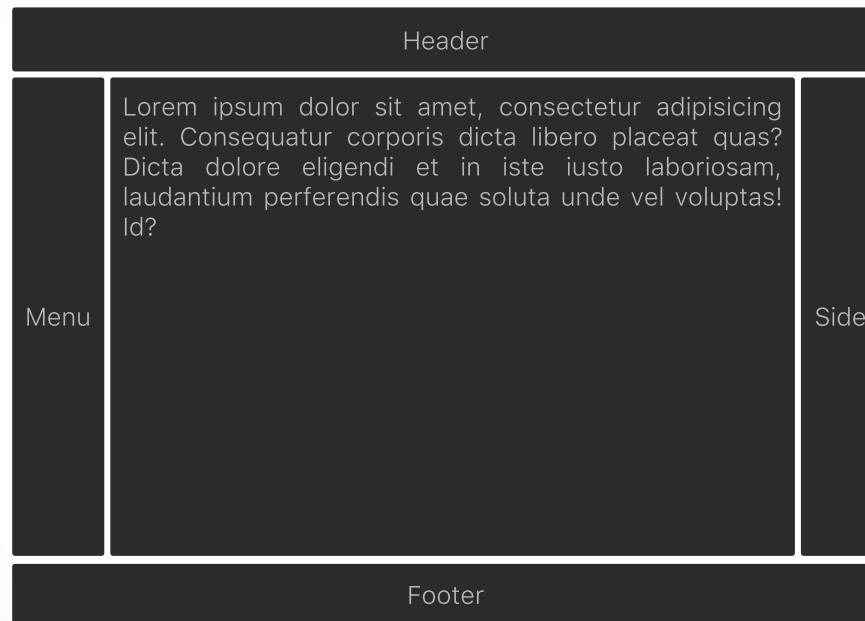
- ›  Unités
- ›  Truc et astuces
- ›  calc
- ›  Fun with Viewport Units

Navigateurs, usage ≥ 0,4% en France

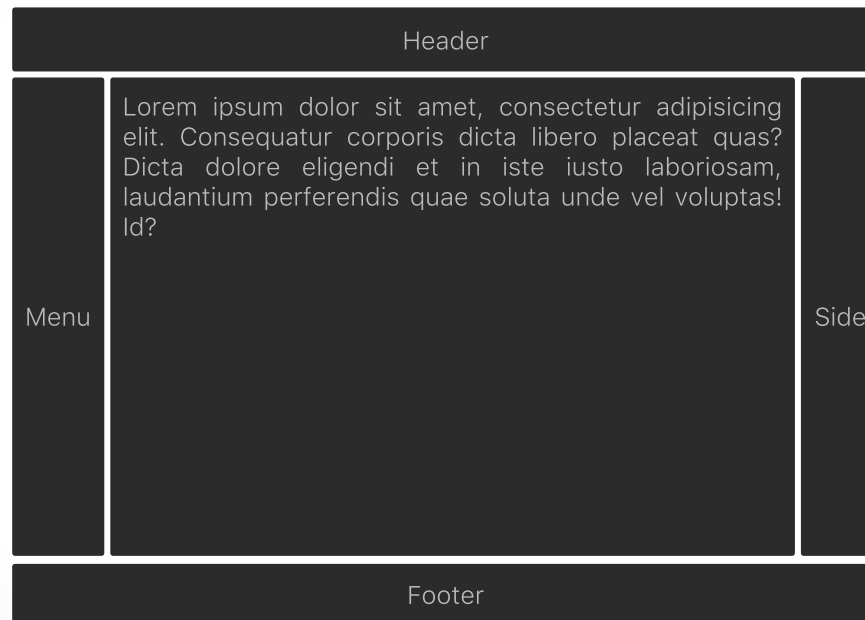


Unit	Chrome	Chrome Android	Safari iOS	Firefox	Opera	Safari	Edge	Opera Mini	Opera Desktop
rem (root em) units	49 58	61	11 9.3	48 52	11	10 9.1	14 15	4.4	46
	59 60		10.0-10.2 10.3	54 55		10.1			
Viewport units: vw, vh, vmin, vmax	49 58	61	11 9.3	48 52	11	10 9.1	14 15	4.4	46
	59 60		10.0-10.2 10.3	54 55		10.1			
calc() as CSS unit value	49 58	61	11 9.3	48 52	11	10 9.1	14 15	4.4	46
	59 60		10.0-10.2 10.3	54 55		10.1			

FLEXBOX ET GRID



```
<body>
  <header>Header</header>
  <div>
    <nav>Menu</nav>
    <main>Content</main>
    <aside>Side</aside>
  </div>
  <footer>Footer</footer>
</body>
```



```
<body>
  <header>Header</header>
  <div>
    <nav>Menu</nav>
    <main>Content</main>
    <aside>Side</aside>
  </div>
  <footer>Footer</footer>
</body>
```

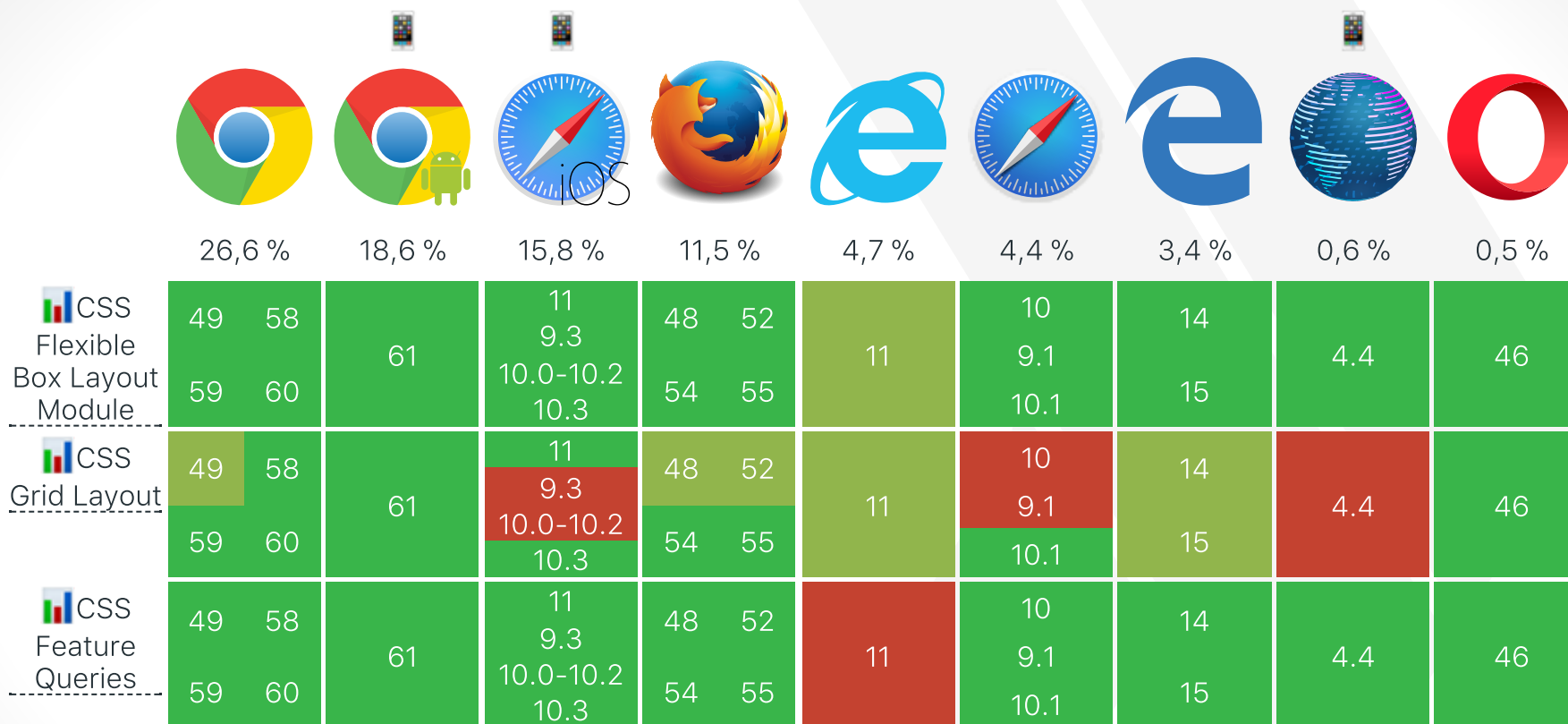
Flexbox

- › [📺 Flexbox, et le CSS redevient fun ! \(Hubert SABLONNIÈRE\)](#)
- › [👤 Solved by Flexbox](#)
- › [🐸 Flexbox Froggy](#)

Grid

- › [@supports](#)
- › [# Grid by examples](#)
- › [📺 CSS Grid Changes Everything \(About Web Layouts\) by Morten Rand-Hendriksen](#)
- › [🥕 Grid Garden](#)

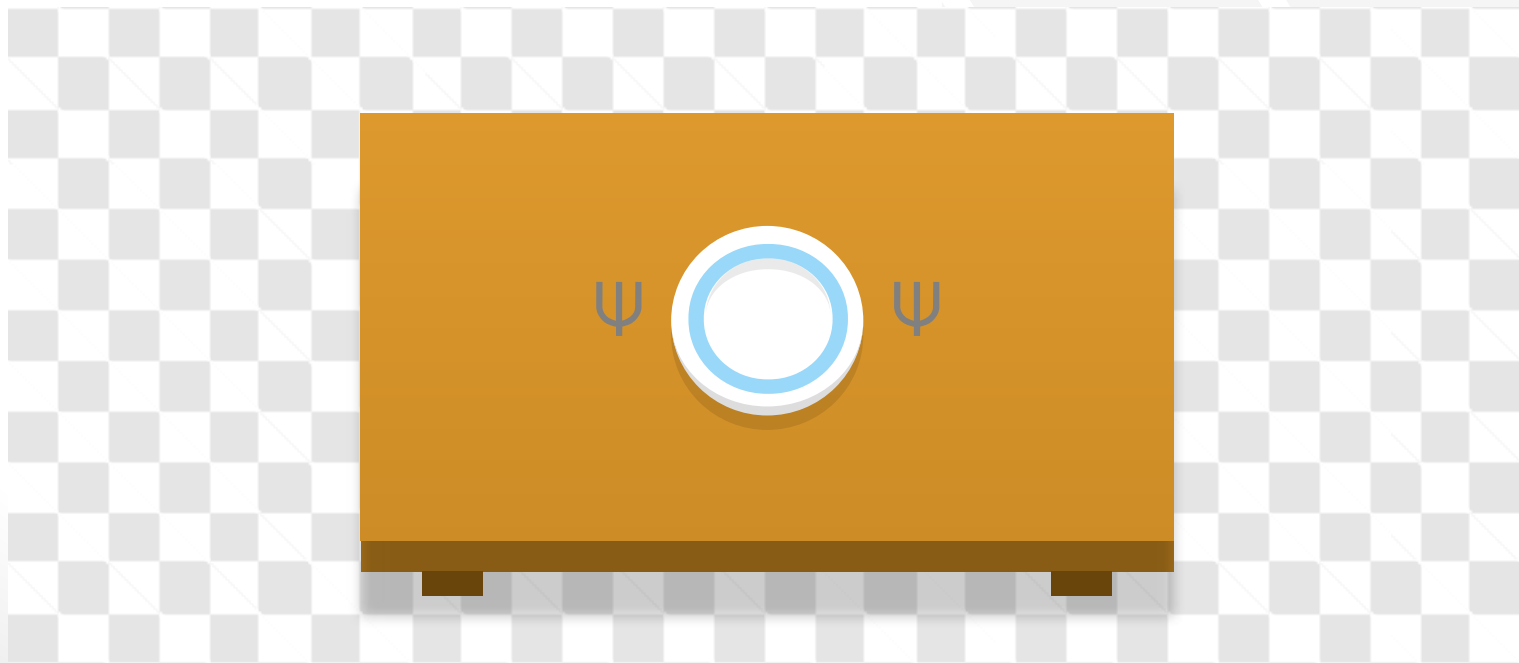
Navigateurs, usage ≥ 0,4% en France



PSEUDO ÉLÉMENTS

```
<div class="table">  
  <div class="plate"></div>  
</div>
```

```
.table::before, .table::after {  
  color: gray;  
  font-size: 2rem;  
  content: '⌘';  
  transform: rotate(180deg);  
}
```

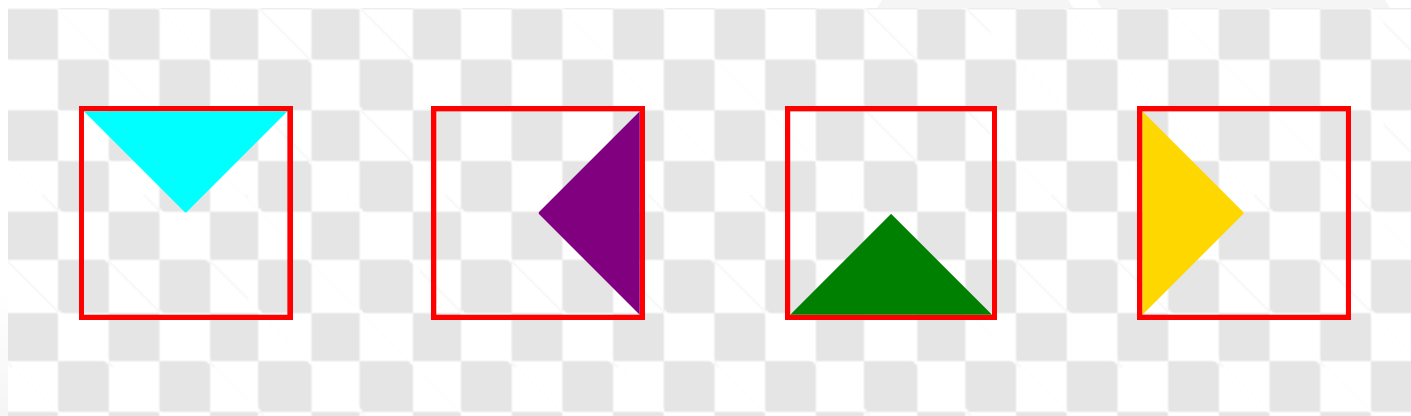


Triangle avec des bordures

#23

```
div.top, div.right, div.bottom, div.left {  
  border: 2em solid transparent;  
  display: inline-block;  
  box-shadow: 0 0 0 .1em red;  
}
```

```
div.top { border-top-color: cyan; }  
div.right { border-right-color: purple; }  
div.bottom { border-bottom-color: green; }  
div.left { border-left-color: gold; }
```




```
.popover {  
  position: relative;  
  background: teal;  
}  
.popover::before {  
  position: absolute;  
  z-index: -1;  
  content: '';  
  top: 1.25em; left: 1em;  
  border: .8em solid transparent;  
  border-top-color: teal;  
  transform: skew(-30deg);  
}
```



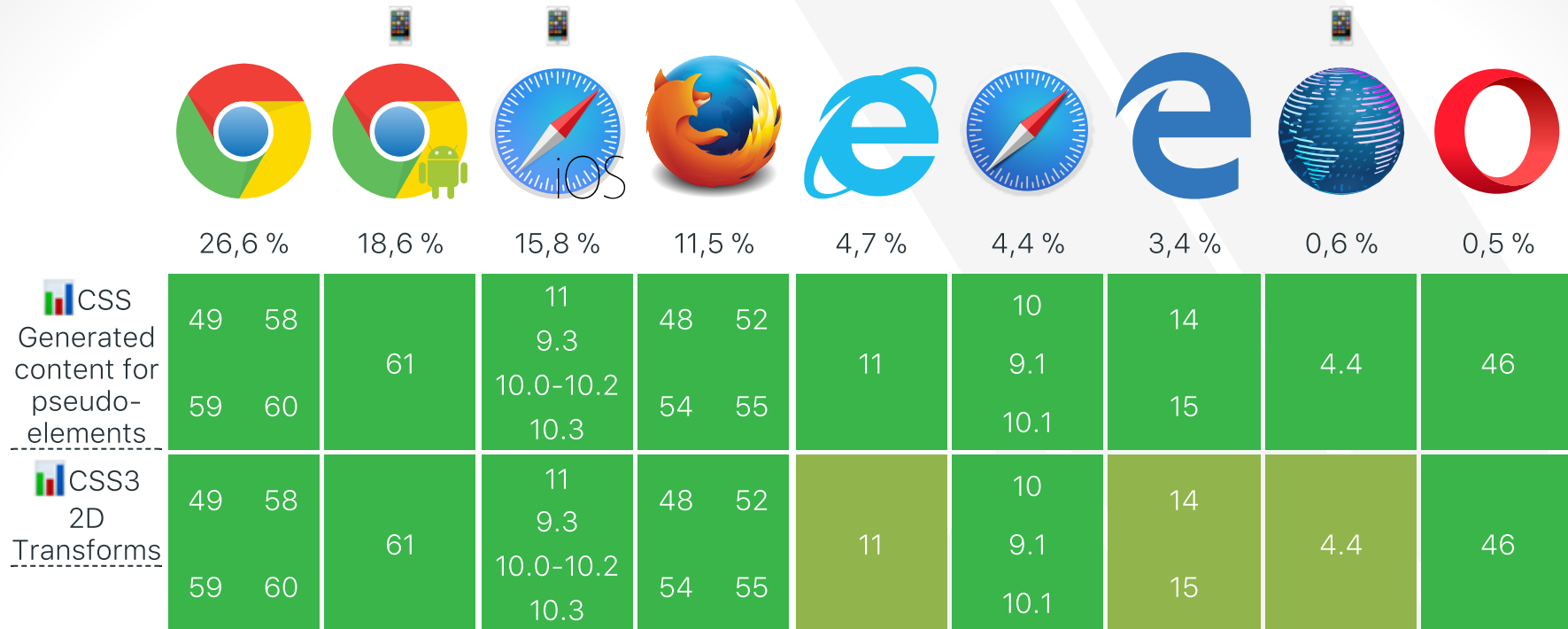
Hello DevFest Toulouse !

- › [w3c The :before and :after pseudo-elements](#)
- › mais aussi `::first-letter`, `::first-line`,
`::selection`, `::backdrop`
- › [📖 An Ultimate Guide To CSS Pseudo-Classes And Pseudo-Elements](#)

- ⚠ `::before` et `::after` ne marchent pas sur `input`, `img`, `iframe` (pas encore spécifié)

- › Table et assiette de [👩‍🍳 CSS Diner](#)
- › [© Dîner des philosophes](#)








Navigateurs, usage $\geq 0,4\%$ en France



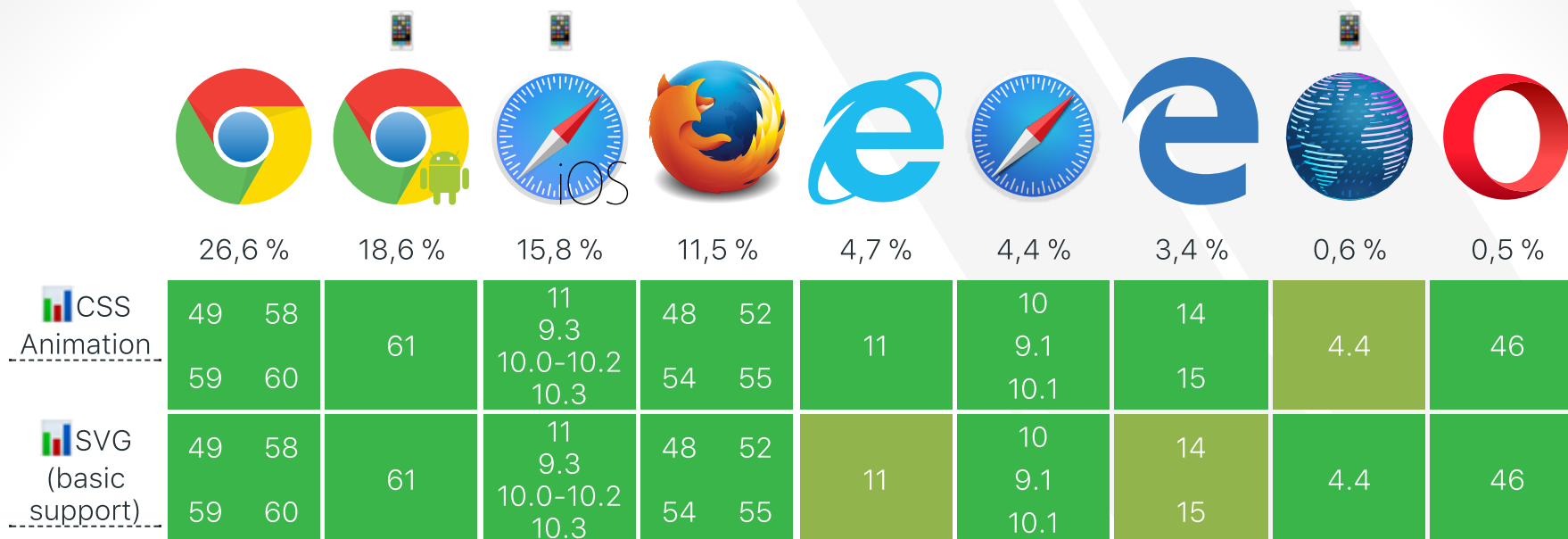
ANIMATIONS


```
.editable svg path {  
  stroke: purple;  
  stroke-width: 1em;  
  fill: none;  
  stroke-dasharray: 4700;  
  stroke-dashoffset: 4700;  
  animation: draw 2s linear infinite;  
}  
@keyframes draw {  
  to { stroke-dashoffset: 0; }  
}
```



- ›  Utiliser les animations CSS
- ›  Text spinners
- ›  CSS only loaders
- ›  Animate.css
- ›  How SVG Line Animation Works
- ›  Animated line drawing in SVG
- ›  CSS triggers

Navigateurs, usage ≥ 0,4% en France



PSEUDO CLASSES D'ÉTAT



➡ hover me

Hello DevFest Toulouse

- › :hover
- › :focus
- › :visited
- › :checked
- › :valid
- › :invalid
- › :empty
- › :active
- › :target
- › ...

```
.editable input[type=checkbox] + label::before {  
  content: 'Click if you like it';  
}  
  
.editable input[type=checkbox]:checked + label::before {  
  content: '💖💖💖';  
}  
  
fieldset input[type=checkbox] { opacity: 0; }
```

“ The science of operations, as derived from mathematics more especially, is a science of itself, and has its own abstract truth and value.
~~~~~ Ada Lovelace



```
.switch + label {  
  display: block;  
  position: relative;  
  padding: .1em;  
  width: 2em;  
  height: 1em;  
  background-color: #ccc;  
  border-radius: 1em;  
  border: medium solid #444;  
  transition: 0.4s;  
}  
  
.switch:checked + label {  
  background-color: green;  
}
```

```
.switch + label::before {  
  display: block;  
  position: absolute;  
  content: '';  
  top: 0.1em;  
  left: 0.1em;  
  height: 1em;  
  width: 1em;  
  background-color: #fff;  
  border-radius: 50%;  
  transition: all 0.25s;  
}  
  
.switch:checked + label::before {  
  transform: translateX(1em);  
}
```

Switch  

## ➔ Hiding Content for Accessibility

```
.panel input[type=checkbox] {  
  position: fixed;  
  left: -100vmax;  
}
```

✓

▼ Apollo 11

“ The computer (or rather the software in it) was smart enough to recognize that it was being asked to perform more tasks than it should be performing. It then sent out an alarm, which meant to the astronaut, I'm overloaded with more tasks than I should be doing at this time and I'm going to keep only the more important tasks; i.e., the ones needed for landing ... Actually, the computer was programmed to do more than recognize error conditions. A complete set of recovery programs was incorporated into the software. The software's action, in this case, was to eliminate lower priority tasks and re-establish the more important ones ... If the computer hadn't recognized this problem and taken recovery action, I doubt if Apollo 11 would have been the successful moon landing it was.[26]

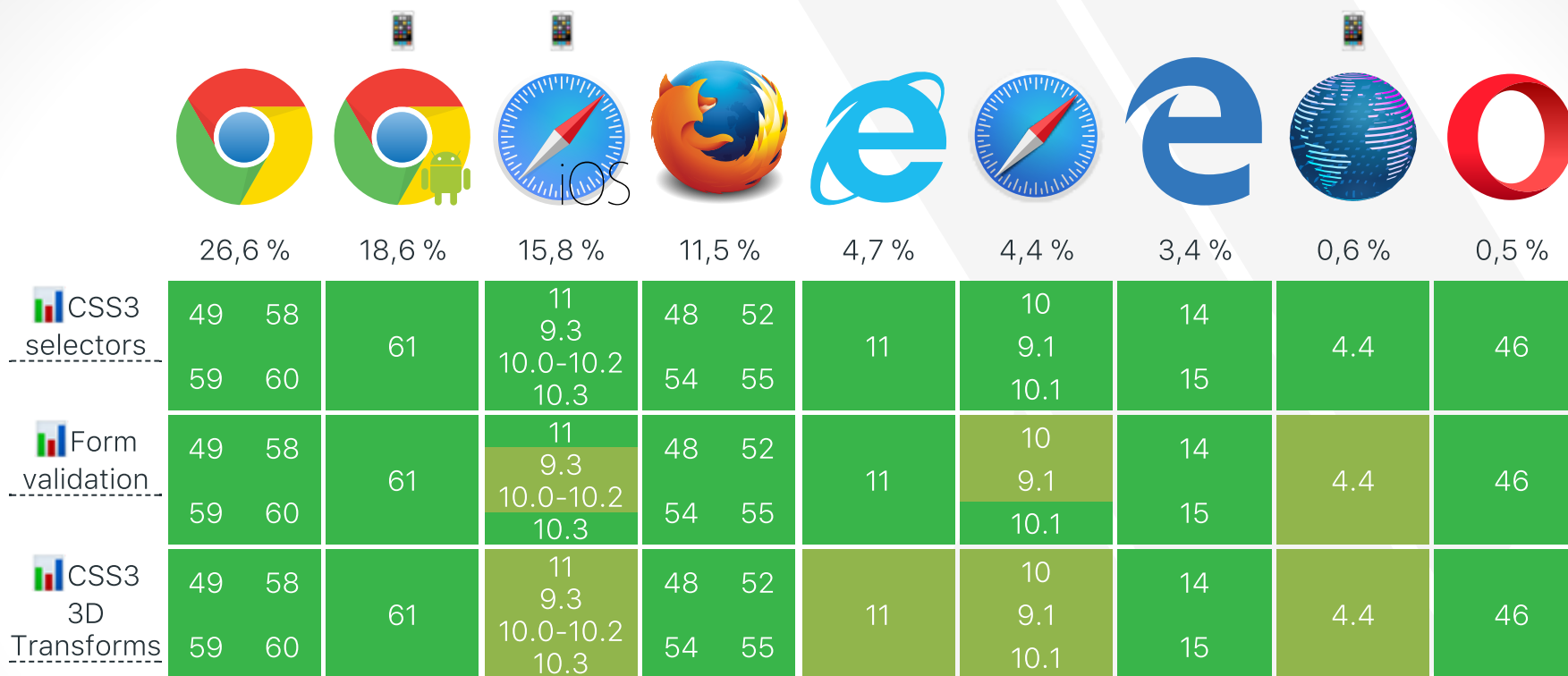
*Letter from Margaret H. Hamilton, Director of Apollo Flight Computer Programming MIT Draper Laboratory, Cambridge, Massachusetts[27], titled "Computer Got Loaded", published in Datamation, March 1, 1971*

```
<div class="tabs">
  <input type="radio" name="tab" id="home" checked>
  <input type="radio" name="tab" id="projects">
  <input type="radio" name="tab" id="about">
  <nav>
    <label for="home">Home</label>
    <label for="projects">Projects</label>
    <label for="about">About</label>
  </nav>
  <div data-for="home">Home page</div>
  <div data-for="projects">Projects page</div>
  <div data-for="about">About page</div>
</div>
```





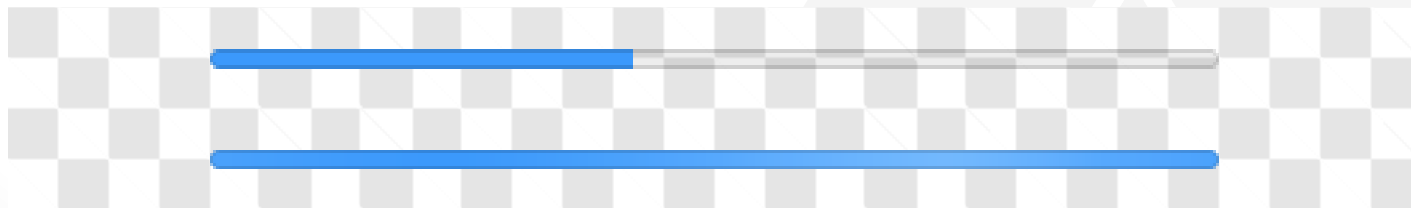
Navigateurs, usage ≥ 0,4% en France



# HTML

## \* The HTML5 progress Element

```
<progress value="42" max="100">42 %</progress>  
<progress></progress>
```



```
<details>
  <summary>Des détails</summary>
  <p>Plus d'infos
    à propos des détails.</p>
</details>
```

```
details {
  border: medium solid currentcolor;
  border-radius: .25em;
}



details summary {
  background: #888; color: #eee;
}
```



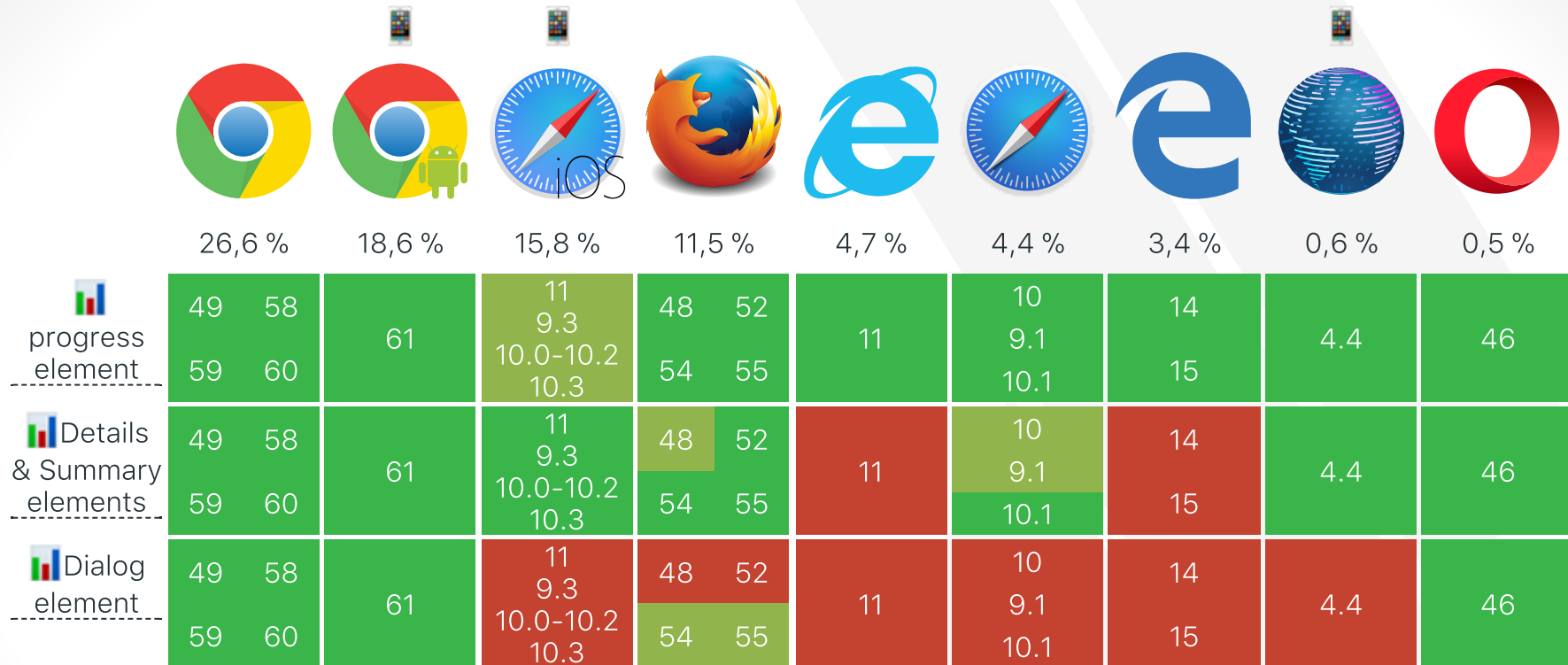
▶ Des détails

```
.editable dialog {  
  box-shadow : .25em .25em .125em rgba(0, 0, 0, 0.42);  
}  
  
.editable dialog::backdrop {  
  position      : fixed;  
  top           : 0; right : 0; bottom : 0; left : 0;  
  background-color : rgba(0, 0, 0, 0.8);  
}
```



- ›  [Better details polyfill](#)
- ›  [Dialog Polyfill](#)

Navigateurs, usage ≥ 0,4% en France






# CONCLUSION



1. Utilisez du CSS pour simplifier le code
2. Utilisez intelligemment les pre/post-processeurs
3. HTML, SVG are Awesome !
4. JavaScript, TypeScript can be Awesome !

👉 Traitez le CSS comme du code

1. Revue de code
2. DRY
3. Clean Code
4. Single Responsibility Principle
5. ...

- >  [les slides en HTML](#)
- >  [les slides en PDF](#)
- >  [le code](#)
- > [Blog: 'Making Of'](#)

- › (Ctrl|⌘) + Shift + i
- › ➡ [CSS Secrets by Lea Verou](#)
- › [CSS sur MDN](#) , ➡ [The A11Y Project](#)
- › ➡ [CodePen](#) , ➡ [JSFiddle](#) , ➡ [Dabblet](#) , ...
- › \* [CSS Tricks](#) , [Smashing Magazine](#)
- › [CSS Flags](#) , ➡ [A Single Div](#)

CSS  
is  
Awesome!