

**CSS
IS
AWESOME!**

\$ whoami

#1

Igor Laborie

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

 @ilaborie

 igor@monkeypatch.io

 *Je ne suis pas un designer*

@DevFestToulouse @ilaborie #CSS #

 DevFest
Toulouse 2017

“

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) & CSS (layout, style, animations simples)
3. SVG (formes et animations complexes)
4. JavaScripts



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

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

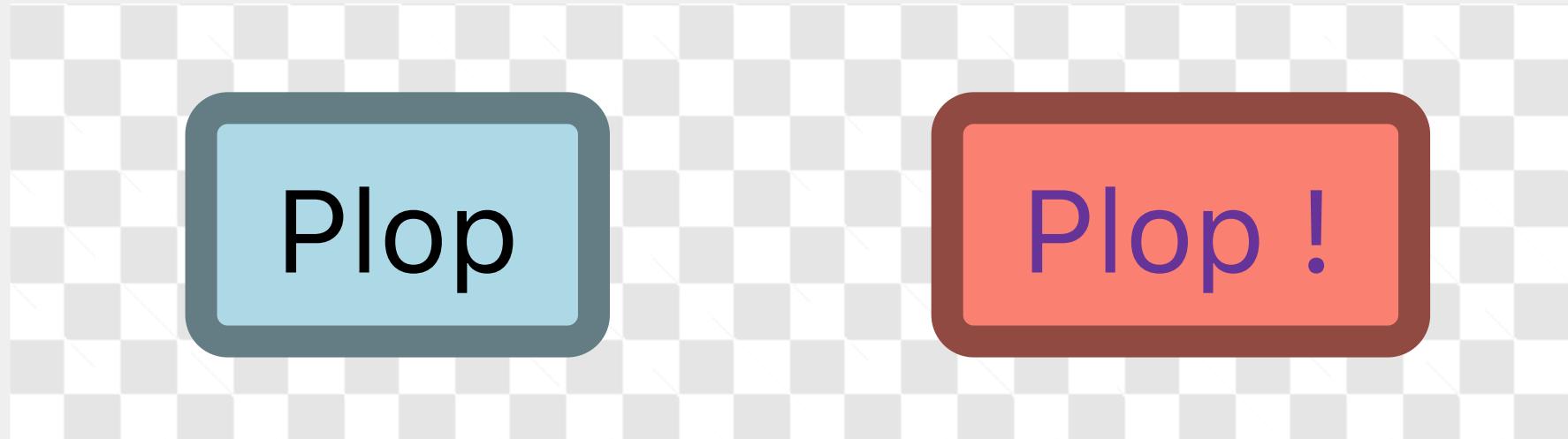
- I. Utiliser un pré-processeur ?
- II. Unités
- III. Flexbox et Grid
- IV. Pseudo éléments
- V. Animations
- VI. Pseudo classes d'état
- VII. HTML
- VIII. Conclusion

UTILISER UN PRÉ-PROCESSEUR ?

Bordure des boutons

#7

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



Alors utilise-t-on un pré-processeurs ?

#8

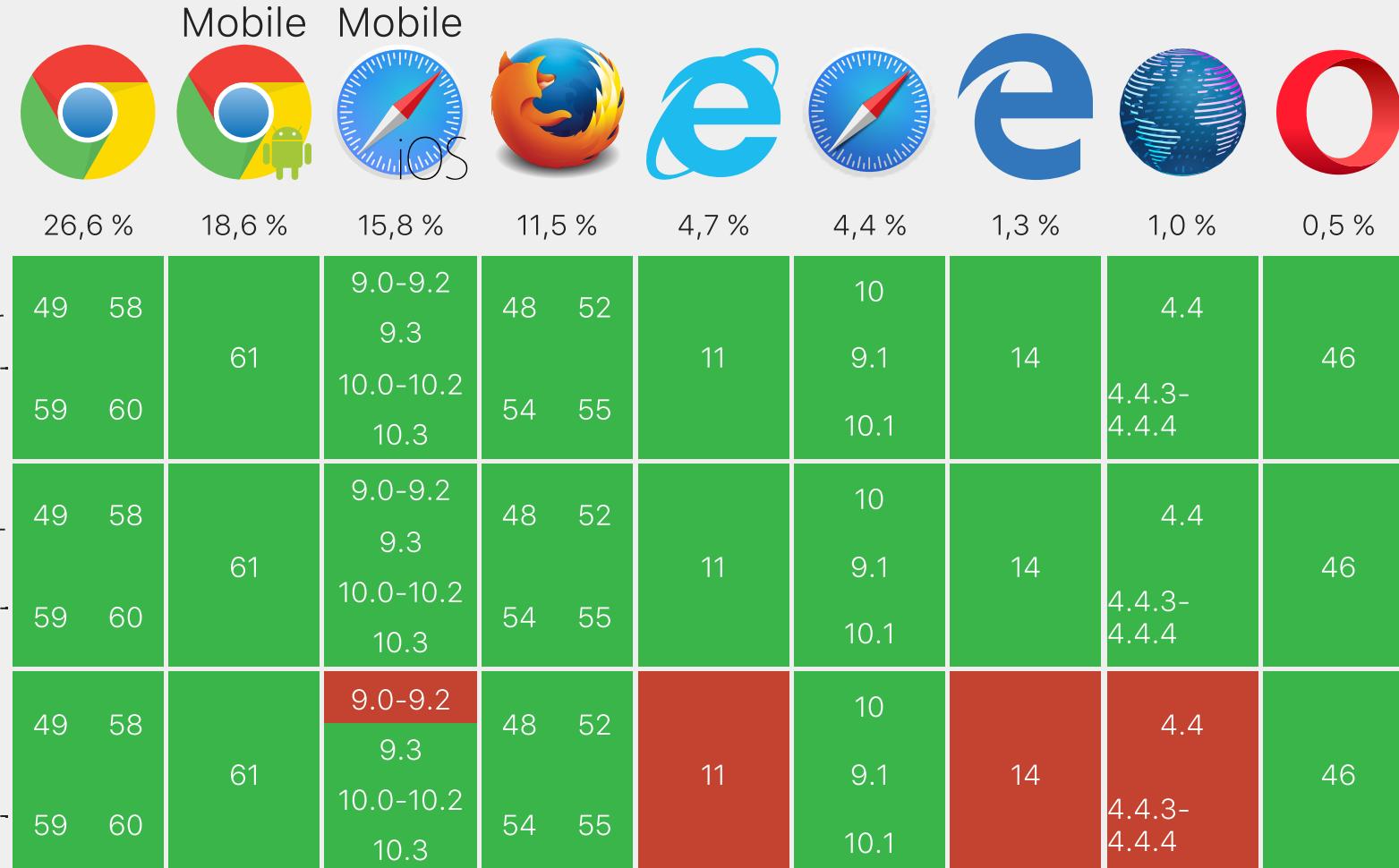
Oui, mais privilégez:

- le CSS
- les post-processeurs

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

Compatibilité

#9



UNITÉS

Une histoire d'unités CSS

#11



 CommitStrip

Les unités de longueur

#12

px, cm, pt, ... longueurs absolues (mesure physique)

em, rem fonction de la `font-size`

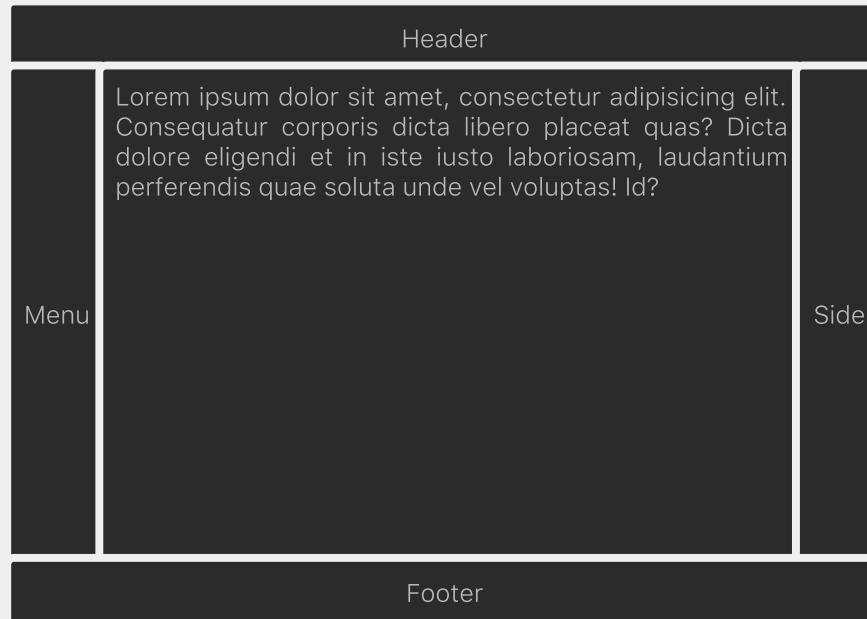
ex, ch hauteur d'un x, largeur d'un Ø

vh, vw $(100vh, 100vw) = (\text{hauteur}, \text{largeur})$ du
viewport

vmin, vmax $\min(1vh, 1vw), \max(1vh, 1vw)$

Holy Grail avec calc

#13



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



Unités



w3c

Truc et astuces



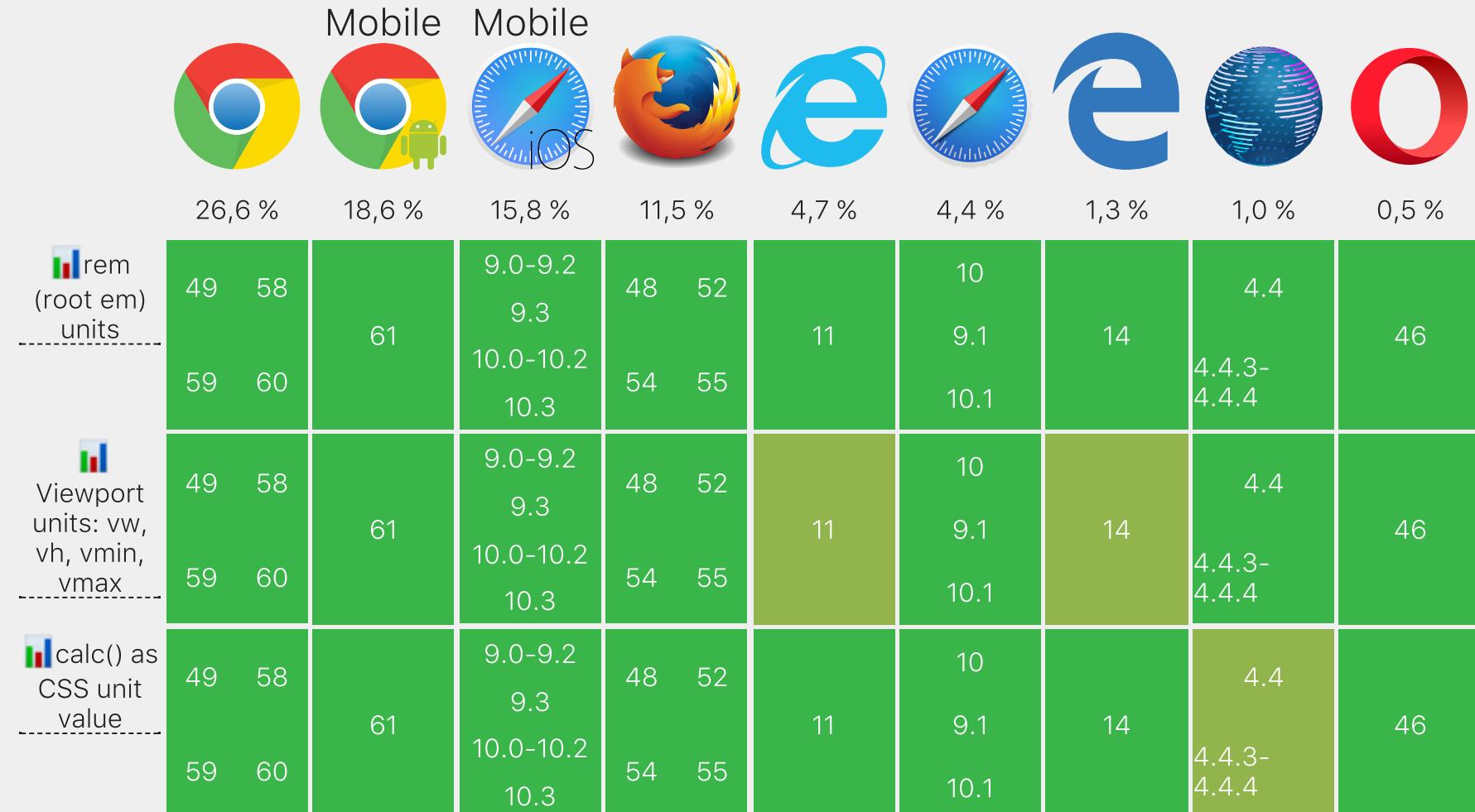
calc



Fun with Viewport Units

Compatibilité

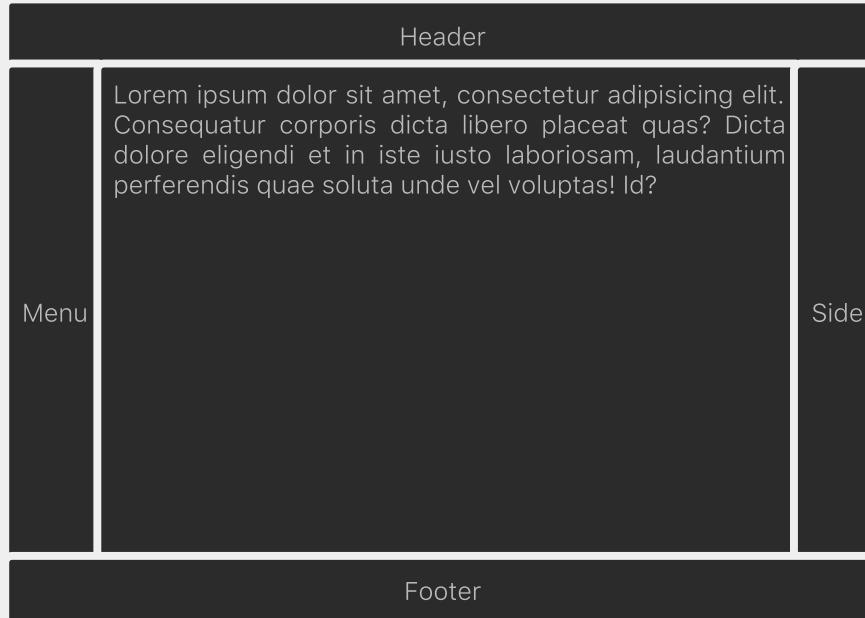
#15



FLEXBOX ET GRID

Holy Grail avec flexbox

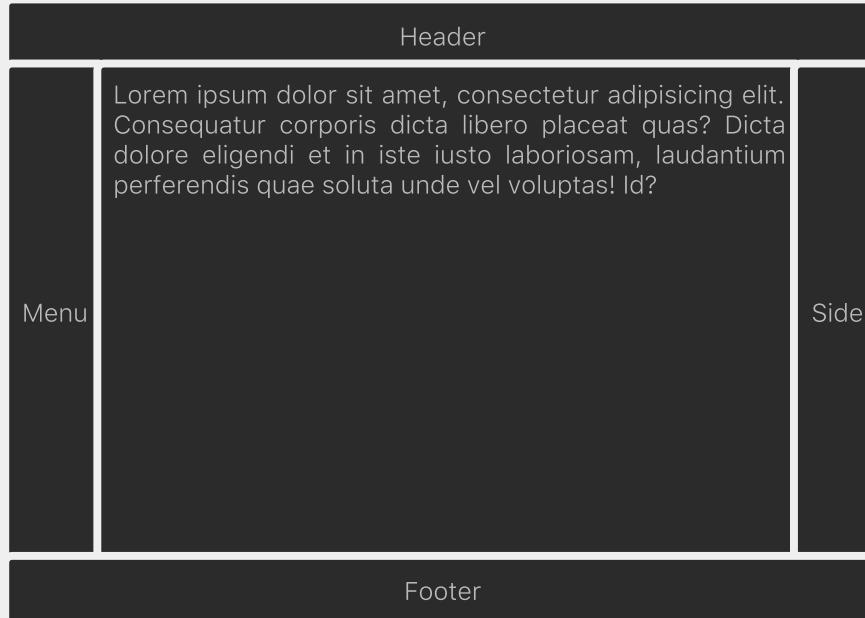
#17



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

Holy Grail avec grid

#18



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

Flexbox

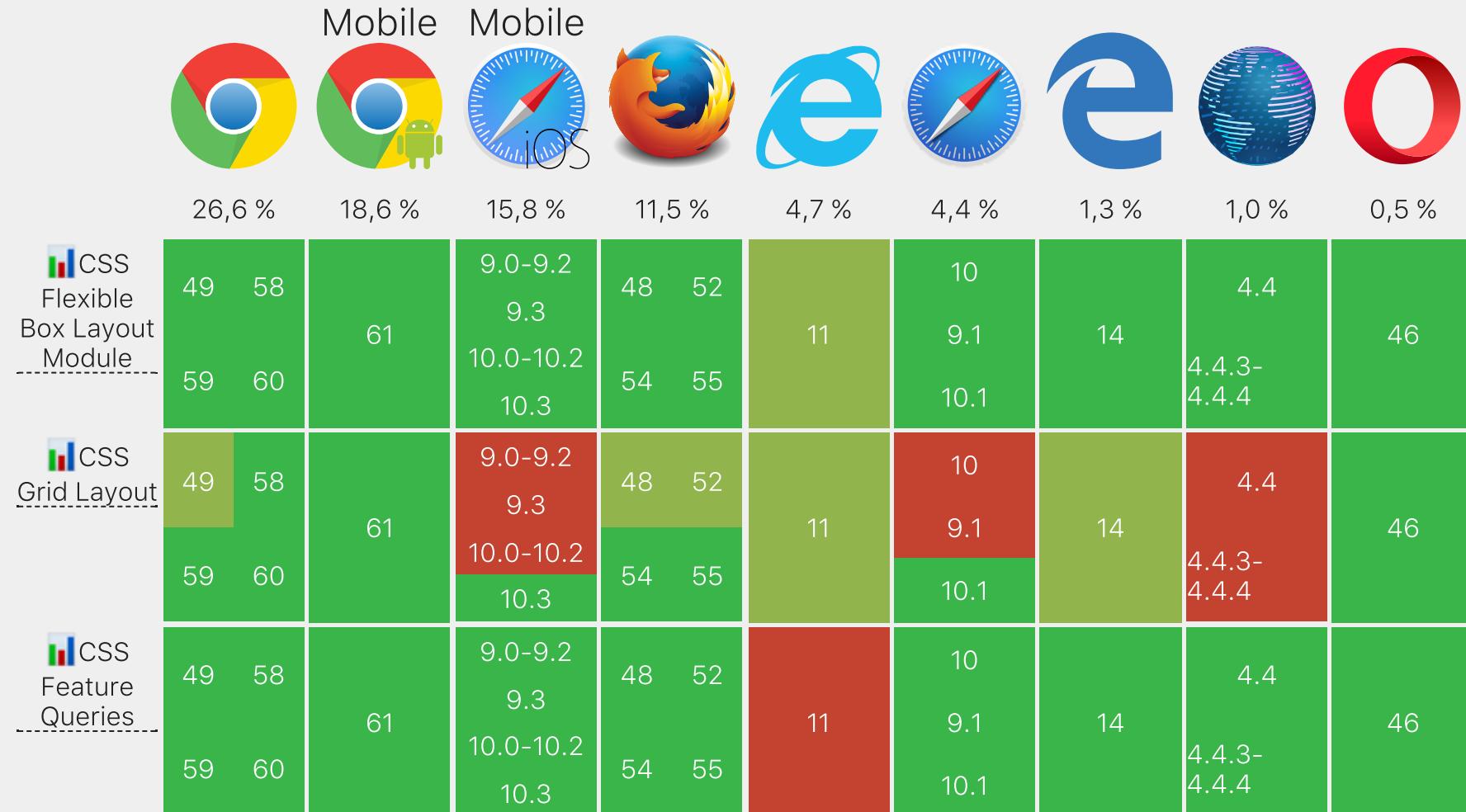
-  Flexbox, et le CSS redevient fun ! (Hubert SABLONNIÈRE)
-  Solved by Flexbox
-  Flexbox Froggy

Grid

-  @supports
-  # Grid by examples
-  CSS Grid Changes Everything (About Web Layouts)
by Morten Rand-Hendriksen
-  Grid Garden

Compatibilité

#20

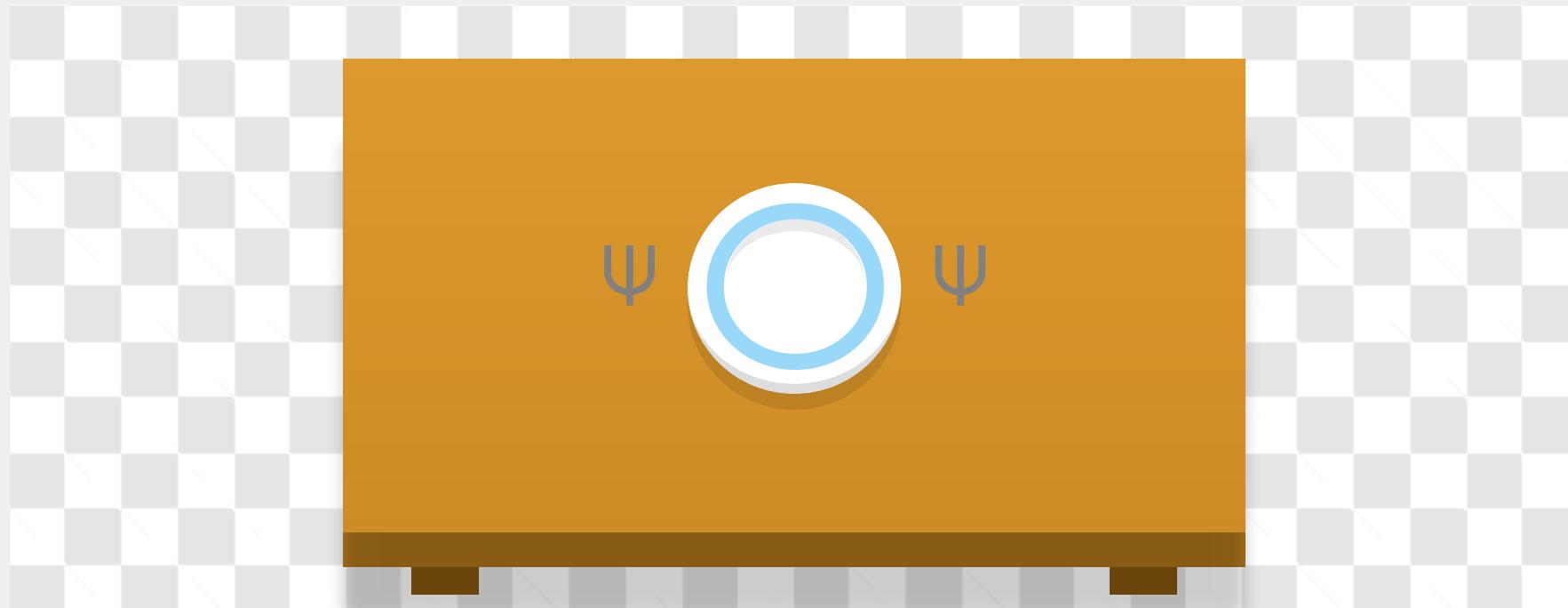


PSEUDO ÉLÉMENTS

Le dinner d'un philosophe

#22

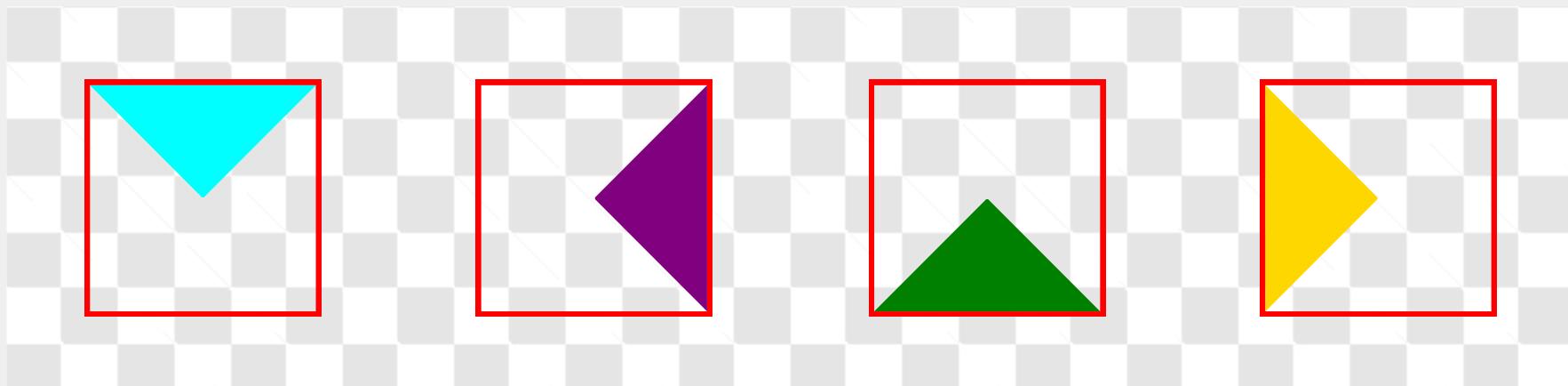
```
.table::before, .table::after {  
  color: gray;  
  font-size: 2rem;  
  content: '⠃';  
  transform: rotate(180deg);  
}
```



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](https://www.w3.org/TR/2011/REC-CSS2-20110607/generate.html#before-after-pseudo)



mais aussi :: first-letter, :: first-line,
:: selection, :: backdrop



[S An Ultimate Guide To CSS Pseudo-Classes And
Pseudo-Elements](https://css-tricks.com/ultimate-guide-pseudo-elements/)



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



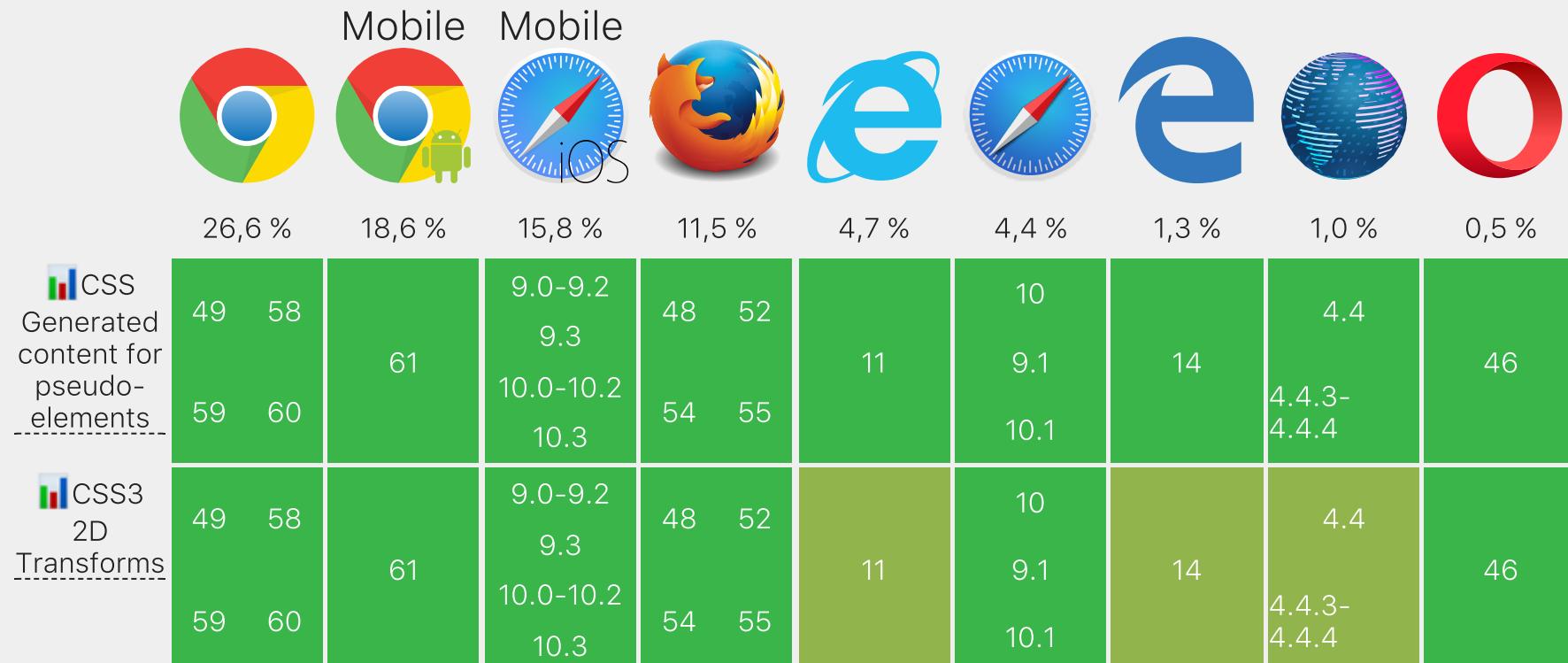
Table et assiette de  [CSS Diner](https://css-diner.com/)



 [Dîner des philosophes](https://diner-des-philosophes.com/)

Compatibilité

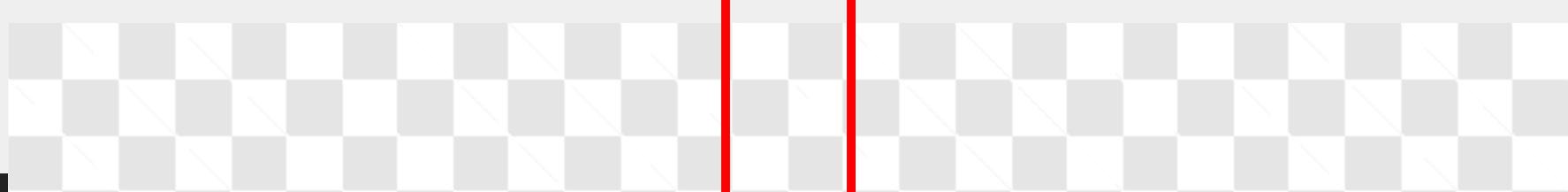
#26



ANIMATIONS

Texte de chargement

#28



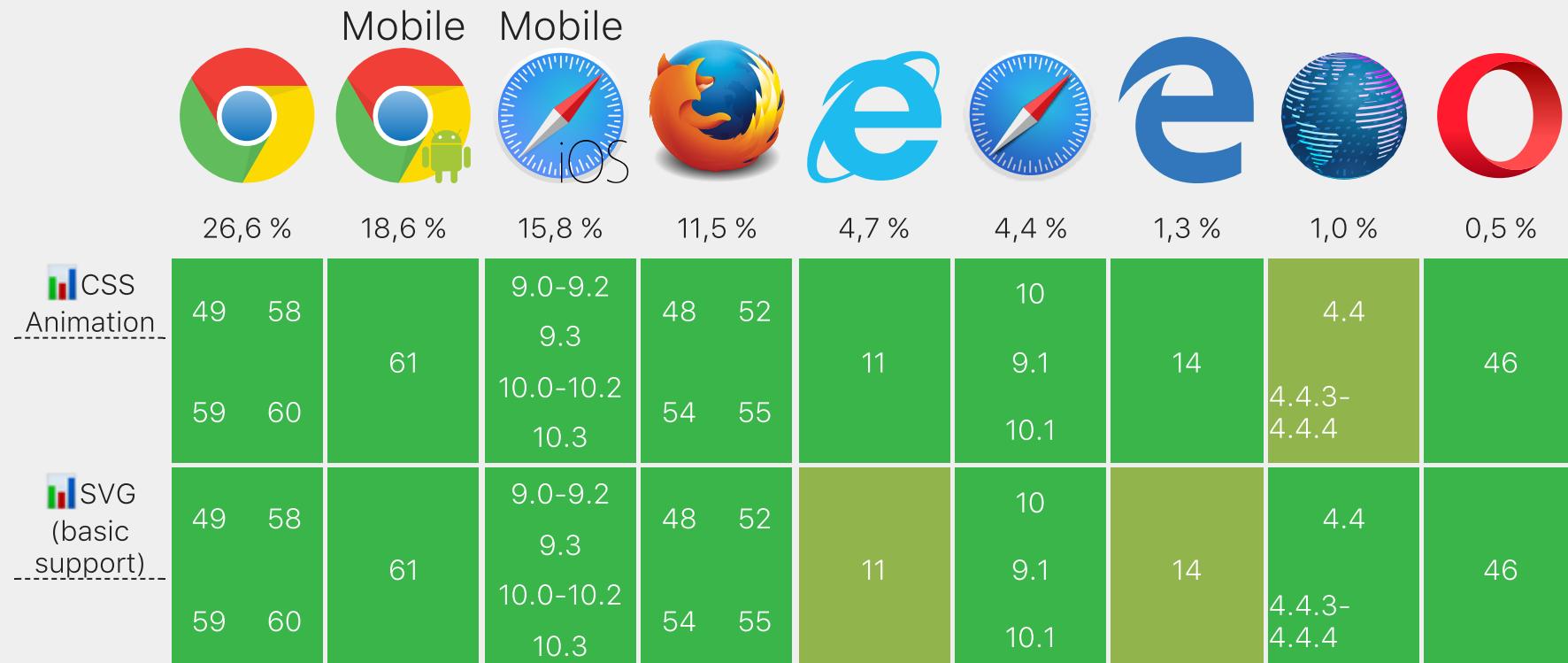
```
.editable svg path {  
  stroke: purple;  
  stroke-width: 1em;  
  fill: none;  
  stroke-dasharray: 4700;  
  stroke-dashoffset: 4700;  
  animation: draw 2s linear infinite;  
}  
@keyframes draw {  
  to { stroke-dashoffset: 0; }  
}
```



-  Utiliser les animations CSS
-  Text Spinner
-  CSS only loader
-  Animate.css
-  How SVG Line Animation Works
-  Animated line drawing in SVG

Compatibilité

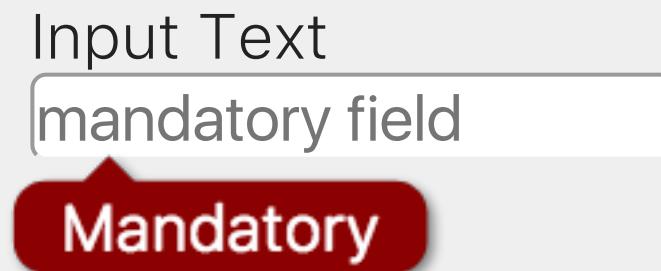
#31



PSEUDO CLASSES D'ÉTAT

Usage des info-bulles

#33



➡ hover me

Hello DevFest Toulouse

- :hover ➤ :valid
- :focus ➤ :invalid
- :visited ➤ :empty
- :checked ➤ :target
- ...

Checkbox Hack

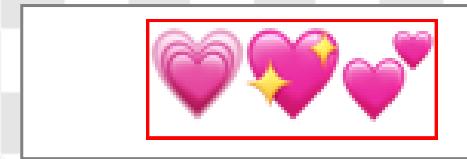
#35

```
.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

#36

```
.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*

# Principe pour les onglets

#38

```
<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>
```

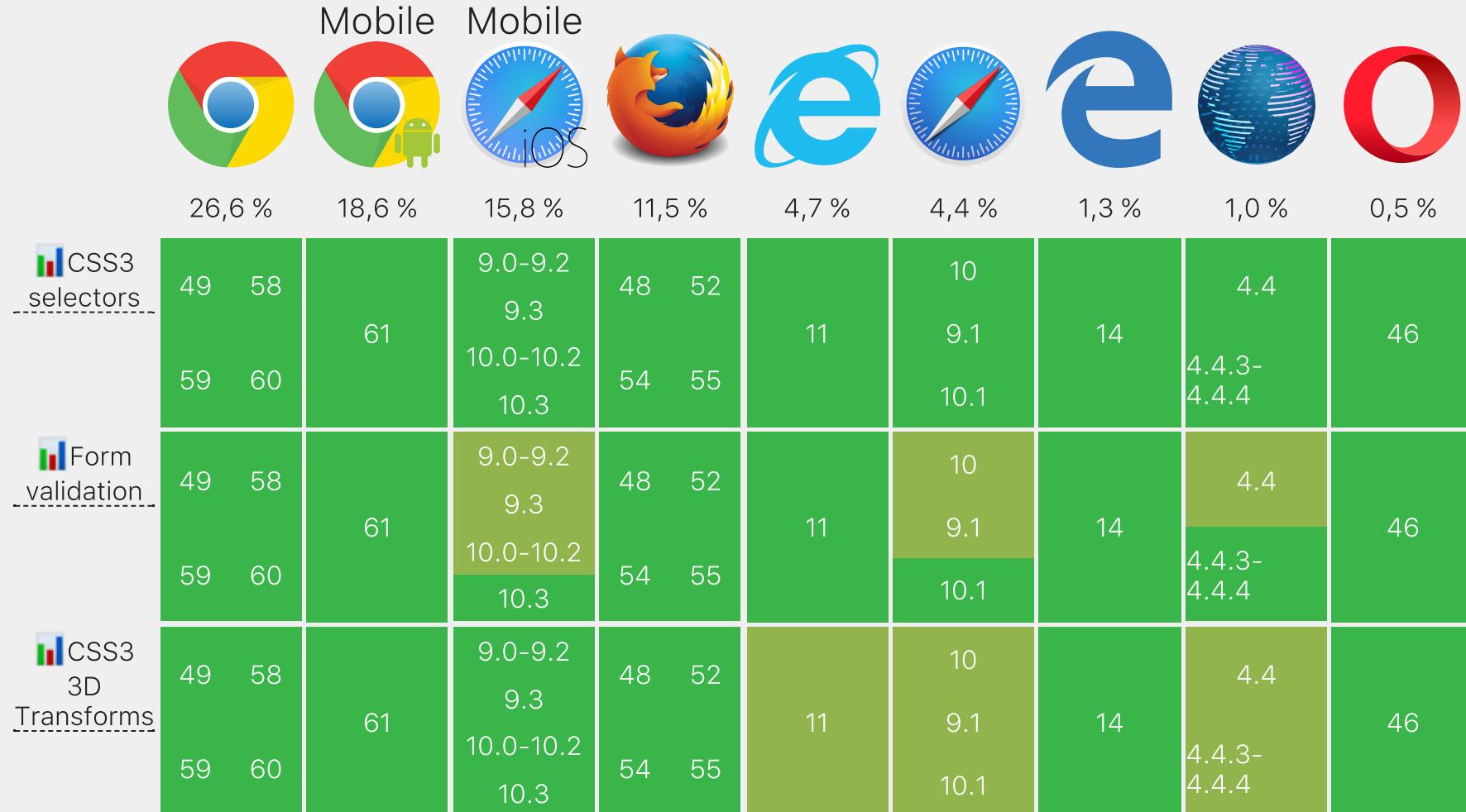
# Démo des onglets

#39



# Compatibilité

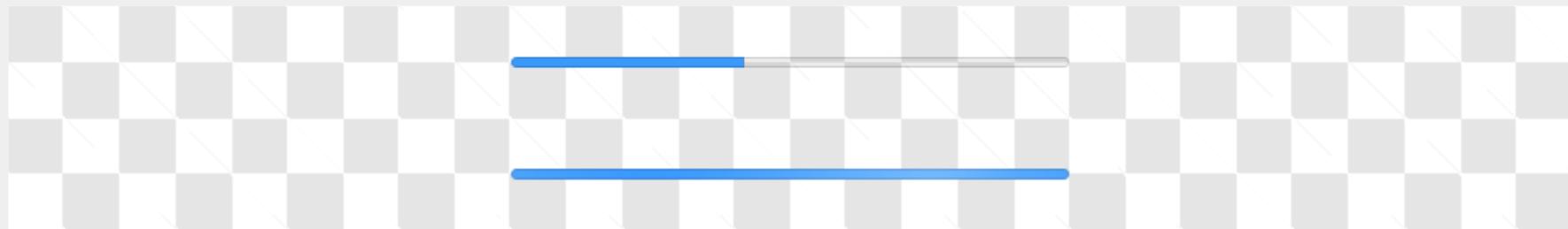
#40



# HTML

## \* The HTML5 progress Element

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



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

```
.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);
}
```

des détails

# Dialog

#44

```
.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);  
}
```





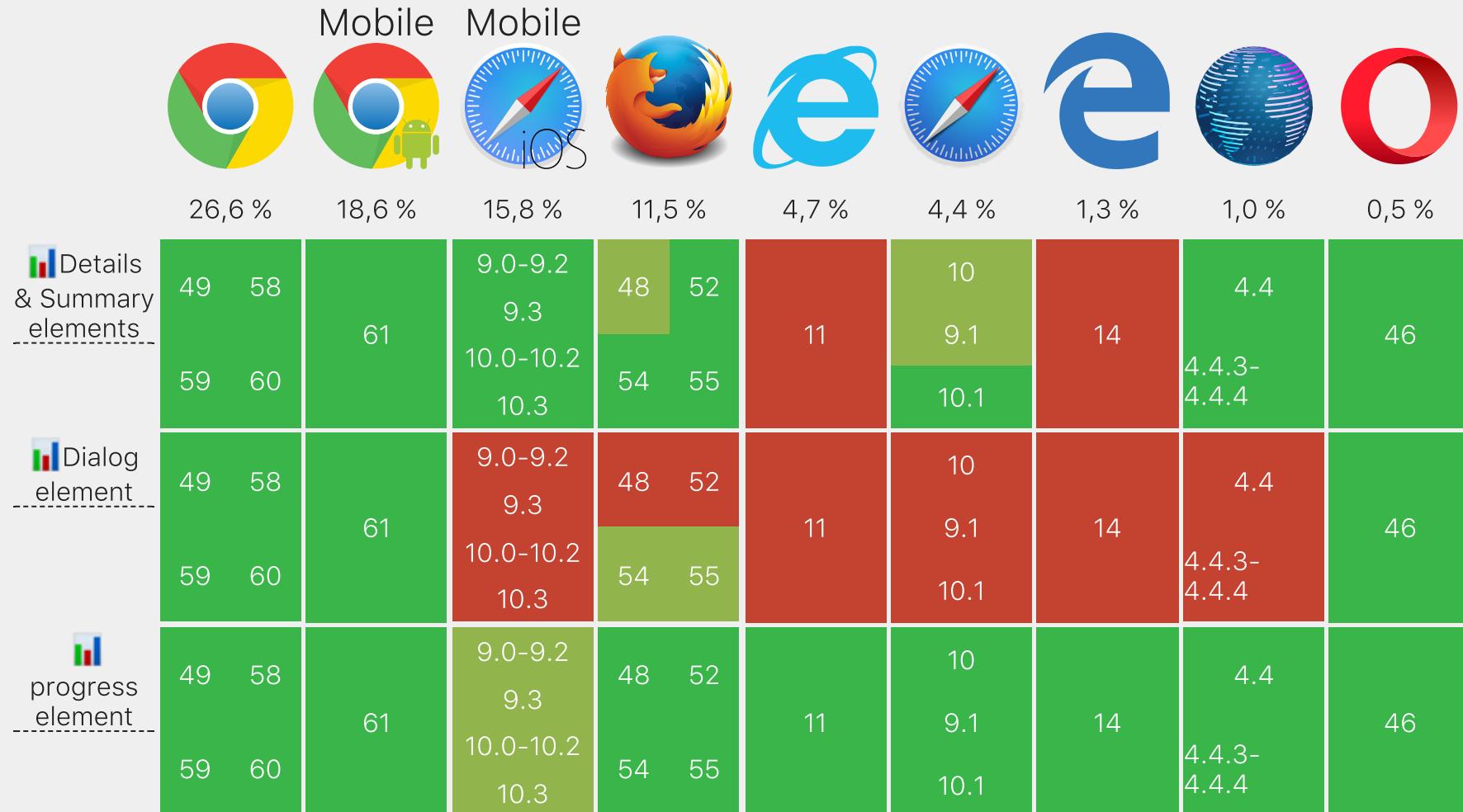
[Collapsible Panel Polyfill](#)



[Dialog Polyfill](#)

# Compatibilité

#46



# CONCLUSION

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

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

-   [les slides en HTML](#)
-   [les slides en PDF](#)
-   [le code](#)
-   [Making Of](#)



(Ctrl|⌘) + Shift + i



➤ CSS Secret by Lea Verou



➤ CSS sur MDN ➤ The A11Y Project



➤ CodePen , ➤ JSFiddle , ➤ Dabblet ,...



➤ CSS Tricks , ➤ Smashing Magazine



➤ CSS Flags

CSS  
is  
Awesome!