

CSS

Unités et sélecteurs modernes



unité "rem" sus aux pixels !



un pixel n'est pas accessible



Why you should never use px to set font-size in CSS

```
#never {  
  font-size: 32px;  
}  
  
#always {  
  font-size: 2rem;  
}
```

- Une taille en pixel est figée pour l'utilisateur (le zoom page n'est pas le zoom texte)
- Le pixel n'est pas responsive (1 pixel sur desktop ne vaut pas un pixel sur mobile)

la solution ? rem !



unité rem

l'unité fluide par excellence

1rem = 1x la taille de police de <html>

```
width:20rem
```

Largeur fixée à 20rem. Cette unité représente la font-size de l'élément racine (par exemple la taille de la police de l'élément html).



cette boîte a une largeur de 20rem

unité rem

mais quelle taille de police dans <div> ?



body

div

```
1 html {  
2   font-size: 15px;  
3 }  
4 body {  
5   width: 600px;  
6   padding: 10px;  
7   font-size: 20px;  
8 }  
9 div {  
10  font-size: 2rem;  
11 }
```

unité rem

et maintenant, quelle taille de police dans <div> ? 😊

body

section

div

```
1 html {  
2     font-size: 15px;  
3 }  
4 body {  
5     font-size: 200%;  
6 }  
7 section {  
8     font-size: 2em;  
9 }  
10 div {  
11     font-size: 2rem;  
12 }
```

astuce de pro !

variables CSS

```
1 :root {  
2   --text-14: 0.875rem;  
3   --text-16: 1rem;  
4   --text-18: 1.125rem;  
5   --text-20: 1.25rem;  
6 }
```



```
1 h1 {  
2   font-size: var(--text-20);  
3 }
```


unités de Viewport

leur référence ? la taille de la fenêtre

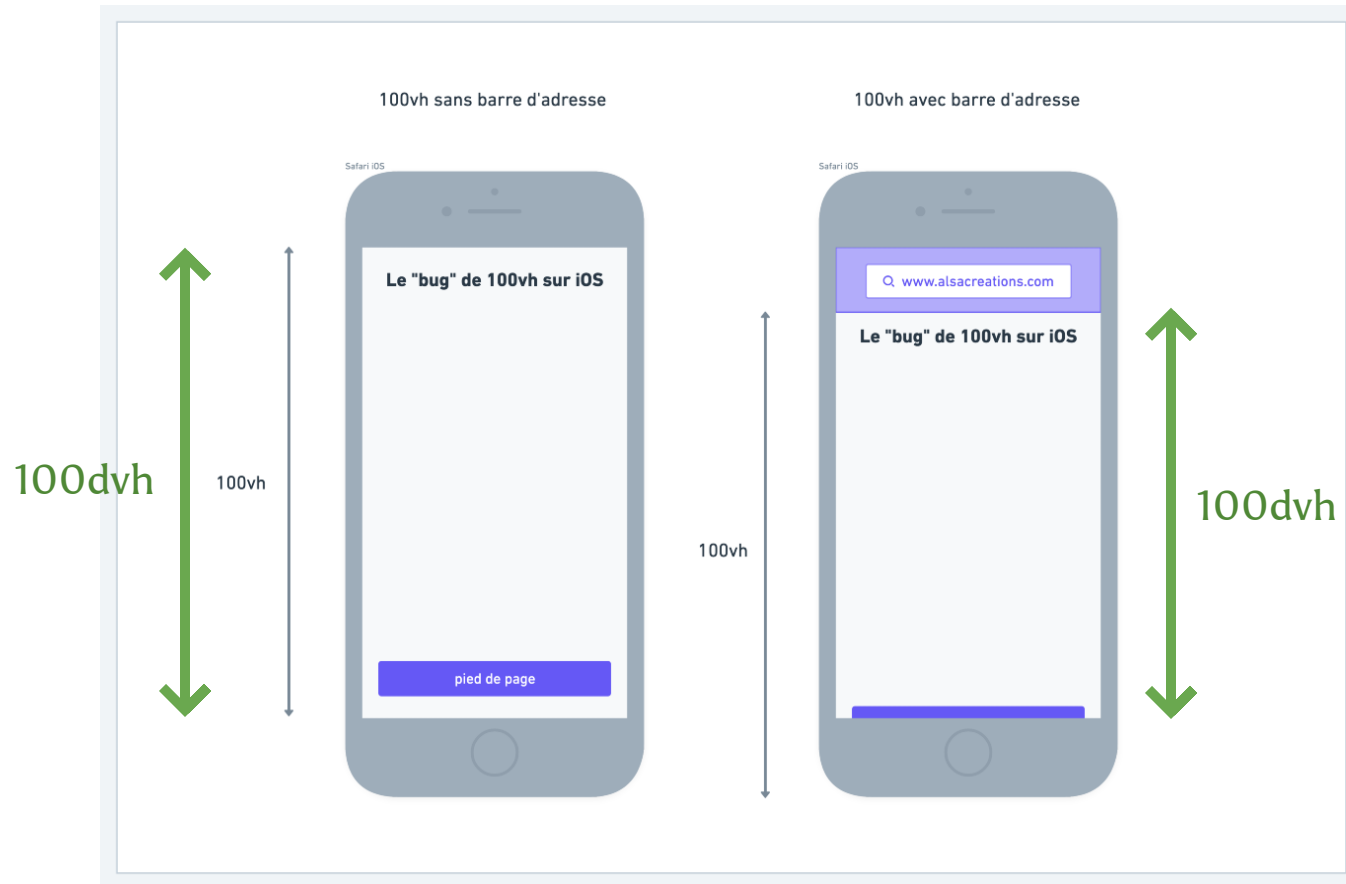
1vw = 1% de la largeur de la fenêtre

```
width:20vw
```

Largeur fixée à 20/100e de la largeur du viewport.

- **vw** : largeur du viewport
- **vh** : hauteur du viewport
- **vmin** : côté le plus petit du viewport
- **vmax** : côté le plus grand du viewport

unités de Viewport étendues



- **svw / svh** : small viewport width / height
- **lvw / lvh** : large viewport width / height
- **dvw / dvh** : dynamic viewport width / height

tailles intrinsèques

`width: min-content`

"Taille du contenu" minimum

Lorem Elsass
Ipsum mitt
picon bière
munster
Bibbeleskaas
un
Flammekueche!

`width: max-content`

Lorem Elsass Ipsum mitt picon bière munster Bibbeleskaas un Flammekueche!

"Taille du contenu" maximum, insécable

`width: fit-content`

Lorem Elsass Ipsum mitt picon bière munster Bibbeleskaas un Flammekueche!

"Taille du contenu" maximum, sécable

`width: auto`

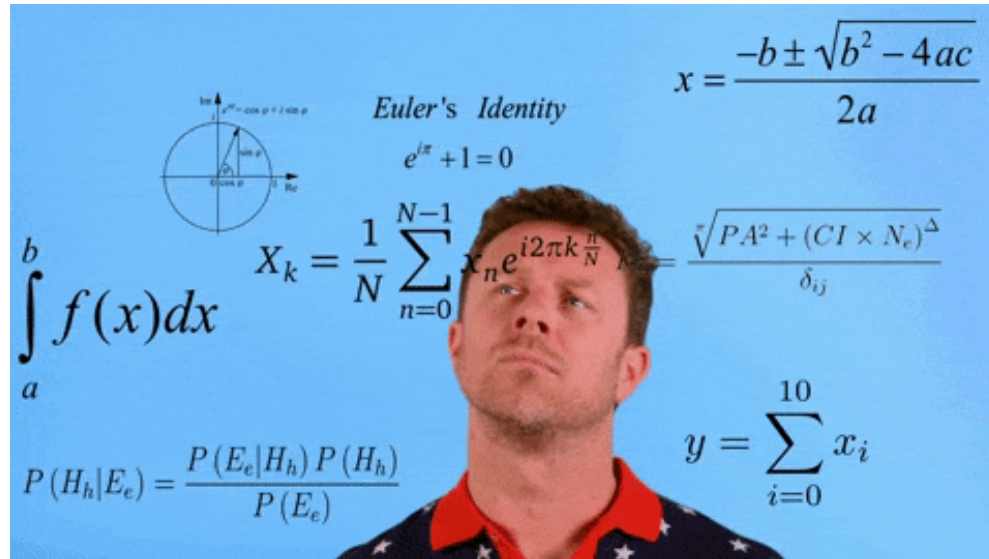
Lorem Elsass Ipsum mitt picon bière munster Bibbeleskaas un Flammekueche!

"Taille du contenu"... mais dépend du type
d'élément (block, inline, flex item, grid item, ...)



Exercice !

unités et valeurs modernes



Les sélecteurs CSS

pour cibler les bons éléments



élément, classe et id

```
p {...}
```

← cible tous les éléments <p> de la page

```
.banane {...}
```

← cible tous les éléments de class="banane"

```
#kiwi {...}
```

← cible l'élément avec un id="kiwi"



enfants et descendants

```
.kiwi span {...}
```

← cible tous les éléments descendants d'un élément de classe "kiwi"

descendants

```
.kiwi > span {...}
```

← cible tous les éléments enfants d'un élément de classe "kiwi"

enfants

combinaison de sélecteurs

```
p.kiwi {...}
```

← cible tous les éléments <p> ayant aussi une class="kiwi"

ET

```
p , .kiwi {...}
```

← cible tous les éléments <p> ou ceux ayant une class="kiwi"

OU

siblings (fraternité)

```
h1 + p {...}
```



cible tous les éléments <p> dont le frère immédiatement précédent est <h1>

frères

```
h1 ~ p {...}
```



cible tous les éléments <p> dont l'un des frères précédent est <h1>

frères

place dans la fratrie

```
.kiwi:first-child {...;}
```



cible les éléments .kiwi et qui sont premiers enfants

```
.kiwi:nth-child(3) {...;}
```



cible les éléments .kiwi et qui sont 3e enfants

```
.kiwi:nth-child(odd) {...;}
```



cible les éléments .kiwi et qui sont impairs

pseudo-classes (= tout ce qui commence par ":")

```
.element:hover {...}
```



cible un élément lors du survol (souris)

```
.element:focus {...}
```



cible un élément lors du focus (clavier, ou clic)

```
.element:hover,  
.element:focus {...}
```



cible un élément au survol ou au focus

```
.element:only-child {...}
```



cible un élément qui n'a pas de frère

```
.element:empty {...}
```



cible un élément qui n'a pas d'enfant

pseudo-éléments (= tout ce qui commence par "::")

```
.element::before {  
  content: "kiwi";  
  background: hotpink;  
}
```



génère un pseudo-élément au début du contenu de .element

Attribut

```
[type] {...;}
```

```
[type="number"] {...;}
```



cible les éléments dont l'attribut HTML "type" vaut "number"

```
[href^="https"] {...;}
```



cible les éléments dont l'attribut HTML "href" commence par "https"

```
[href$=".pdf"] {...;}
```



cible les éléments dont l'attribut HTML "href" termine par ".pdf"

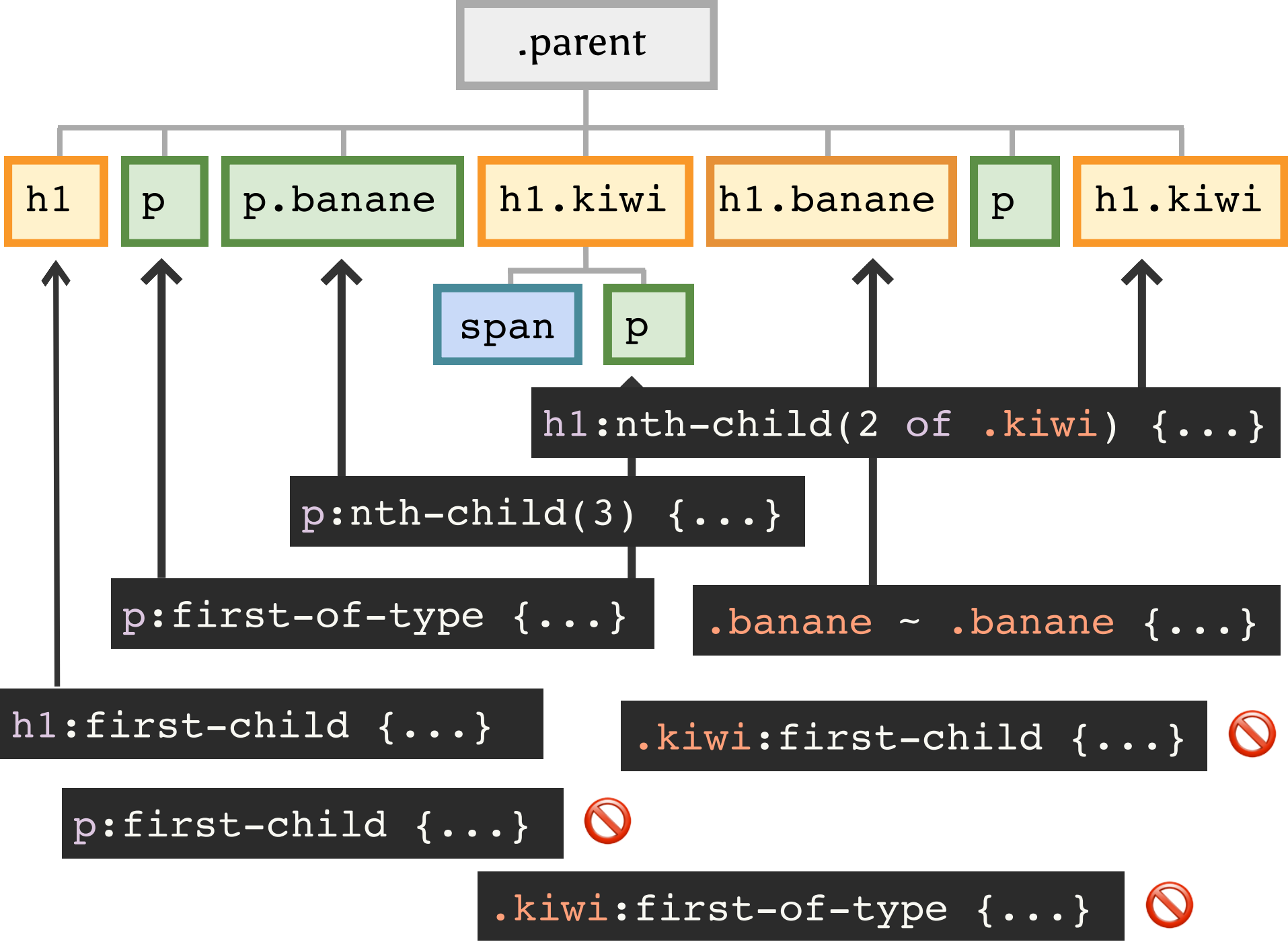
```
[href*="alsacreation"] {...;}
```



cible les éléments dont l'attribut HTML "href" contient "alsacreation"

QUIZ!





les sélecteurs CSS modernes



:not()

pseudo-classe de négation

```
div:not(.kiwi) {  
  /* div qui n'a pas la classe .kiwi */  
}
```

```
div:not(:first-child) {  
  /* div qui n'est pas première enfant */  
}
```

```
div:not(:hover, :focus) {  
  /* div qui n'est pas survolée ni focusée */  
}
```

```
div:not(.container > div) {  
  /* div qui n'est pas enfant direct de .container */  
}
```

```
div:not(div + div) {  
  /* div qui n'est pas précédée d'une div */  
}
```

```
div:not(.kiwi):not(:first-child) {  
  /* div qui n'a pas la classe .kiwi et qui n'est pas lère */  
}
```

:has()

quand tu veux cibler le parent (mais pas que...)



:has()

```
a:has(img) {  
  color: hotpink;  
}
```

→ cible les liens qui contiennent une image

```
a:has(> img) {  
  color: hotpink;  
}
```

→ cible le lien parent direct d'une image

```
button:not(:has(> .icon)) {  
  color: hotpink;  
}
```

→ cible un bouton qui n'est pas parent direct d'un .icon

```
label:has(+ input) {  
  color: hotpink;  
}
```

→ cible le label précédant un input

je veux mettre en avant les produits soldés

```
1 .product:has(.discount) {  
2   order: -1;  
3 }
```



je veux une gouttière conditionnelle

```
1 .card {  
2   display: grid;  
3   grid-template-columns: auto 1fr;  
4 }  
5 .card:has(>img) {  
6   gap: 20px;  
7 }
```



Je veux une gouttière ici, mais uniquement quand il y a une image

<https://codepen.io/raphaelgoetter/pen/vYdVQPX>

je veux une couleur de fond quand un enfant est coché

```
1 .form-check:has(input:checked) {  
2   background: #ddd;  
3 }
```

C'est votre choix

- ☐ Oh oui, spammez-moi !
- ☐ Vous pouvez répéter la question ?



Sélecteurs de focus

clavier / clic / touch

:focus

Indicateur de focus global quelle que soit la méthode (clavier, souris, touch, JavaScript, etc.)

:focus-visible

Indicateur de focus uniquement là où il est pertinent pour l'utilisateur (principalement les utilisateurs de clavier)

:focus-within

Indicateur de focus pour un élément ou l'un de ses descendants ayant le :focus

Sélecteurs de formulaire

pour cibler les états d'un formulaire

:required

Required input

*Required

:disabled

Disabled input

:valid

Valid input

*Required

:invalid

Invalid input

You must enter a valid email

:checked

Checked input

☒ Option 1 ☐ Option 2 ☐ Option 3 ☐ Option 4

:in-range/:out-of-range

Out-of-range input

Out of range (value must be between 10 and 10)

Sélecteurs de formulaire

pour cibler les états d'un formulaire

~~:valid~~



:user-valid

~~:invalid~~



:user-invalid

Versions plus récentes et plus "intelligentes" (uniquement après interaction avec l'utilisateur)

<https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/:user-valid>

Exercice !

sélecteurs de formulaire

Formulaires et sélecteurs

Nom :

Helmut



Site web :

dans ton



E-mail :

qds

☒ Oh oui, je veux être spammé !

Envoyer

C'était top !

c'est quoi la suite ?

