Typographie

Le travail de typographie pour le Web a connu une avancée intéressante ces dernières années. En effet, la typographie était un peu le parent pauvre du webdesign. Avec l'arrivée du CSS3, il a été possible de choisir beaucoup plus finement le type de police que l'on utilise. D'autre part, de même que la notion de grille de mise en page venue de l'imprimerie s'est répandue, les webdesigners se sont penchés sur ce que l'on nomme la grille verticale, qui permet de donner un rythme à l'agencement du texte sur la page.

I. Utilisation des polices de caractères

A. Récapitulatif des règles

Commençons par récapituler les règles CSS les plus courantes concernant les polices de caractères :

- **font-family** : famille de police ;
- **font-size** : taille ;
- **font-style** : gras, italique, oblique, etc. ;
- **font-variant** : variante ;
- **font-weight** : graisse ;
- **color** : couleur de police ;
- **text-align** : gauche, droite, centre ou justifié ;
- **text-decoration** : soulignement ;
- **text-transform** : minuscules, capitales, etc. ;
- **line-height** : interligne (hauteur de ligne).

B. Historique et format de polices

Avant CSS3, les webdesigners ne pouvaient utiliser que des polices qui étaient déjà installées sur l'ordinateur de l'utilisateur. Pour cette raison, les polices comme Arial, Helvetica, Verdana, Times et les génériques **serif** (police à empattements type Times) et **sans-serif** (police à bâton type Arial ou Helvetica) étaient les plus prisées. Observons la règle CSS suivante :

```
font-family: arial, helvetica, sans-serif;
```

Fig. 1 Vert1 © Skillandyou

Dans cette déclaration de polices, on demande à afficher en premier lieu Arial, présent sur de nombreux ordinateurs par défaut et en particulier les PC. Ensuite, on indique une seconde famille de polices, Helvetica, au cas où Arial ne serait pas présente sur la machine – ce qui est le cas de la majorité des ordinateurs d'Apple. Enfin, si la première ou la seconde famille de police n'est pas présente, on indique au navigateur qu'il doit utiliser la police sans-serif (police à bâtons) par défaut de la machine.

La seule solution pour utiliser des polices dites « exotiques » était de passer par des images. Elles n'étaient donc réservées qu'à des usages très spécifiques : menus, titres ou lettrines.

Avec CSS3, les concepteurs de sites web peuvent théoriquement utiliser n'importe quelle police. Il existe des sites spécialisés permettant de choisir des polices libres de droits ou payantes comme Dafont (https://www.dafont.com/fr/), FontSquirrel (https://www.fontsquirrel.com) ou encore des sites de fonderies (créateurs ou revendeurs de polices) comme Monotype (https://www.monotype.com/).

Lorsque vous avez trouvé ou acheté la police que vous souhaitez utiliser, il suffit de transférer le fichier de police sur votre serveur web (pour le cas d'un site en production) afin qu'il soit automatiquement téléchargé par l'utilisateur ; ou, si vous travaillez en local, dans le dossier du site.

Par contre, il vous faudra définir ces polices via CSS afin que le navigateur puisse retrouver les fichiers nécessaires à l'affichage du texte. De la même manière que les navigateurs ne gèrent pas nativement tous les formats vidéo, en particulier les codecs, il existe plusieurs formats de police web même si les navigateurs récents comprennent peu ou prou tous le même format.

Il faut donc constituer un jeu de police permettant de couvrir la plupart des navigateurs, on utilisera donc les formats suivants :

– ttf / otf : True Type Font / Open Type Font, fonctionne sur de nombreux navigateurs, on trouve le plus souvent un unique jeu de police .ttf ;

– woff / woff2 : Web Open Font Format, beaucoup plus léger que le précédent (woff2 étant encore plus léger que .woff mais réservé aux navigateurs les plus récents) ;

– .svg : version vectorisée, spécifique aux versions Apple, ce format n'est plus d'actualité ;

– .eot : format spécifique aux anciennes versions de IE, ce format n'est plus d'actualité.

C. Déclaration de police @fontface

Pour utiliser une police spécifique, vous devez d'abord définir un nom pour la police (on utilise en général le nom de la police mais on peut tout à fait définir un autre nom, par exemple myGreatFont). Il faut déclarer la police dans le CSS via **@fontface** qui permet de pointer vers les fichiers de police.

D'abord on déclare la police exotique :

```
@fontface {
    font-family: "myGreatFont";
    src:url("lien-vers-police/police.ttf");
    src:url("lien-vers-police/police.woff2");
    src:url("lien-vers-police/police.woff");
}
```

Fig. 2 Vert2 © Skillandyou

Ensuite, il suffit de l'appeler dans le CSS comme on le fait d'habitude :

```
h2 {
    font-family: "myGreatFont", arial, sans-serif;
}
```

Fig. 3 Vert3 © Skillandyou

Notez qu'on indique également d'autres types de police pour plus de sécurité.

Faites bien attention aux liens que vous indiquez dans la source de la police. Celle-ci peut être stockée sur un serveur (cas d'un site en production) ou sur votre disque dur (cas plus probable lorsque l'on fait des essais). Cela veut dire que vos fichiers polices doivent être stockés à l'intérieur du fichier global du site.

Si nos polices sont dans un dossier nommé « fonts » à la racine du site et que le fichier CSS est également dans le dossier nommé « CSS » à la racine du site :

```
@fontface {
    font-family: "myGreatFont";
    src:url("../font/police.ttf");
    src:url("../font/police.woff2");
    src:url("../font/police.woff");
}
```

Fig. 4 Vert4 © Skillandyou

De plus, les jeux de polices sont souvent constitués des variantes : gras, italique, light, lightitalic, etc.

Exemple avec le jeu de la police Ubuntu :

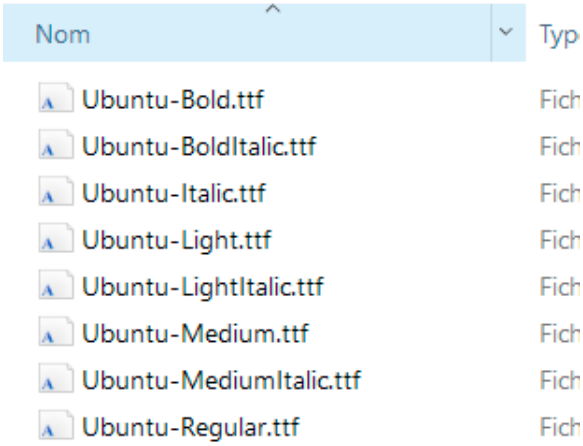

Fig. 5 Vert5 © Skillandyou

Dans ce cas, si l'on veut avoir un véritable jeu de police avec italique et gras, il faut le préciser en multipliant les importations pour expliquer au navigateur comment tout cela fonctionne.

```
@font-face {
  font-family: "Ubuntu";
  src:url("fonts/Ubuntu-Regular.ttf") format("truetype");
  font-style: normal;
  font-weight: normal;
 }
@font-face {
  font-family: "Ubuntu";
  src:url("fonts/Ubuntu-Italic.ttf") format("truetype");
  font-style: italic;
  font-weight: normal;
 }
```

Fig. 6 Vert6 © Skillandyou

On indique dans le premier appel (ici, on se contente d'un seul format de police pour faire plus simple) la version normale de notre police, puis dans un second appel la version en italique de la police.

Dès lors si on a :

```
body {
  font-family: serif;
}
.container {
  border:1px solid #222;
  text-align: center;
  font-family:"Ubuntu";
}
h2  {
  font-style: italic;
}
```

Fig. 7 Vert7 © Skillandyou

On déclare une police générique à empattements pour **body**, puis la police importée « Ubuntu » pour **.container**, enfin on demande à avoir un **h2** en italique.

Le HTML est le suivant :

```
    consectetur pellentesque, magna mi dignissim ipsum,</span> id
    varius turpis nisi vel arcu. Proin malesuada lacus faucibus
    tellus auctor, ac commodo dolor bibendum.
</p>
<div class="container">
    <h1>ça déchire  </h1>
    <h2>C'est de la balle</h2>
</div>
<p class="rem">
    Cras cursus eget libero id fermentum. Ut vel lobortis quam.
    Donec venenatis porttitor sem, <span class="petit-rem">quis
    egestas diam laoreet rutrum. Donec feugiat, sapien ut
```

Fig. 8 Vert8 © Skillandyou

On a placé **div.container** parent d'un **h1** et d'un **h2** entre des paragraphes de texte.

Logiquement, les polices en dehors de div.container vont être pilotées par **body**, on aura donc une police à empattements pour le texte correspondant.

Observons :

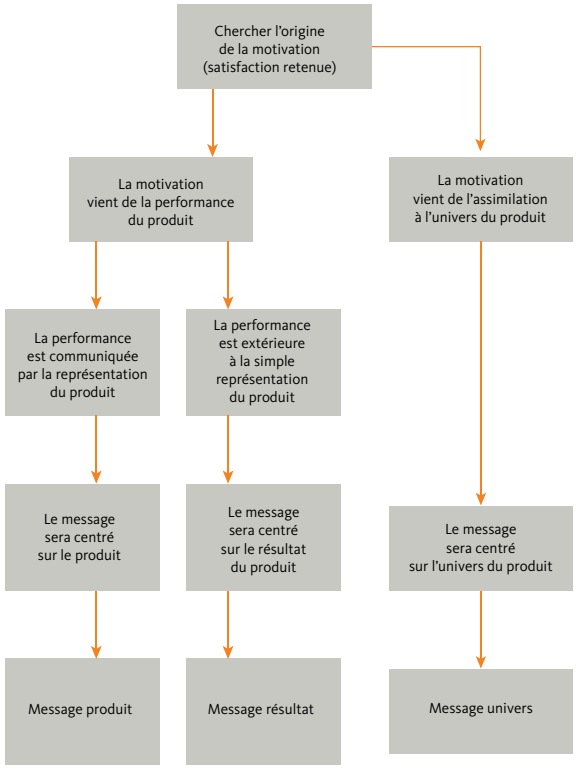

Fig. 9 Fig. 9 © DR

On constate que **h1** utilise la police « Ubuntu » mais **h2** aussi et que de plus le navigateur a compris que l'on demandait la variante « italisée » du jeu de police.

En revanche, si on met l'appel de la police en commentaires pour la désactiver :

```
@font-face {
  font-family: "Ubuntu";
  src:url("fonts/Ubuntu-Regular.ttf") format("truetype");
  font-style: normal;
  font-weight: normal;
 }
/* @font-face {
  font-family: "Ubuntu";
  src:url("fonts/Ubuntu-Italic.ttf") format("truetype");
  font-style: italic;
  font-weight: normal;
 }*/
```

Fig. 10 Vert10 © Skillandyou

Le résultat est le suivant :

ça déchire

C'est de la balle

Fig. 11 Vert11 © Skillandyou

Pas évident ? Comparons l'un et l'autre côte à côte en zoomant :

Italique du jeu de police	Italique du navigateur
ça déchire *C'est de la balle*	ça déchire *C'est de la balle*

Fig. 12 Vert12 © Skillandyou

Un regard attentif nous montre que la police de gauche est mieux dessinée : le « C », les « l », les « b » sont plus élégants, mais c'est surtout le « a » qui diffère. À partir de cette donnée, on peut constituer son propre jeu de police. Par exemple, on aurait pu choisir la version LightItalic du jeu de police Ubuntu pour définir nos italiques avec :

```
@font-face {
  font-family: "Ubuntu";
  src:url("fonts/Ubuntu-LightItalic.ttf") format("truetype");
  font-style: italic;
  font-weight: normal;
}
```

Fig. 13 Vert13 © Skillandyou

Cette fois-ci (nouvelle comparaison) :

Italique du jeu de police	Italique du navigateur
ça déchire *C'est de la balle*	ça déchire *C'est de la balle*

Fig. 14 Vert14 © Skillandyou

Ce qui dans un texte nous donnera :

Donec venenatis porttitor sem, quis egestas diam laoreet rutrum. Donec feugiat, sapien ut *consectetur pellentesque*, magna mi dignissim ipsum,id varius turpis nisi vel arcu.

Fig. 15 Vert15 © Skillandyou

Au lieu de la police « italisée » par le navigateur :

Donec venenatis porttitor sem, quis egestas diam laoreet rutrum. Donec feugiat, sapien ut *consectetur pellentesque*, magna mi dignissim ipsum,id varius turpis nisi vel arcu.

Fig. 16 Vert16 © Skillandyou

Il faut enfin préciser que plus on utilise de fichiers de police, plus on alourdit le chargement du site, chaque appel @fontface alourdit donc nos pages.

D. Les polices stockées

On peut également trouver une police stockée en ligne, par exemple les **Google Fonts**.

Première méthode

Il suffira de faire un lien directement dans le **head** du HTML :

```
<link href="https://fonts.googleapis.com/css?family=Kavivanar" rel="st
ylesheet">
```

Fig. 17 Vert17 © Skillandyou

Puis de faire appel à la nouvelle police dans le CSS :

```
font-family: 'Kavivanar', cursive;
```

Fig. 18 Vert18 © Skillandyou

Deuxième méthode

On importe la police dans le CSS :

```
@import url('https://fonts.googleapis.com/css?family=Kavivanar');
```

Fig. 19 Vert19 © Skillandyou

Puis on fait appel à la nouvelle police dans le CSS :

```
font-family: 'Kavivanar', cursive;
```

Fig. 20 Vert20 © Skillandyou

Pour chaque police choisie, Google Fonts récapitule ces deux possibilités.

La première méthode (standard) :

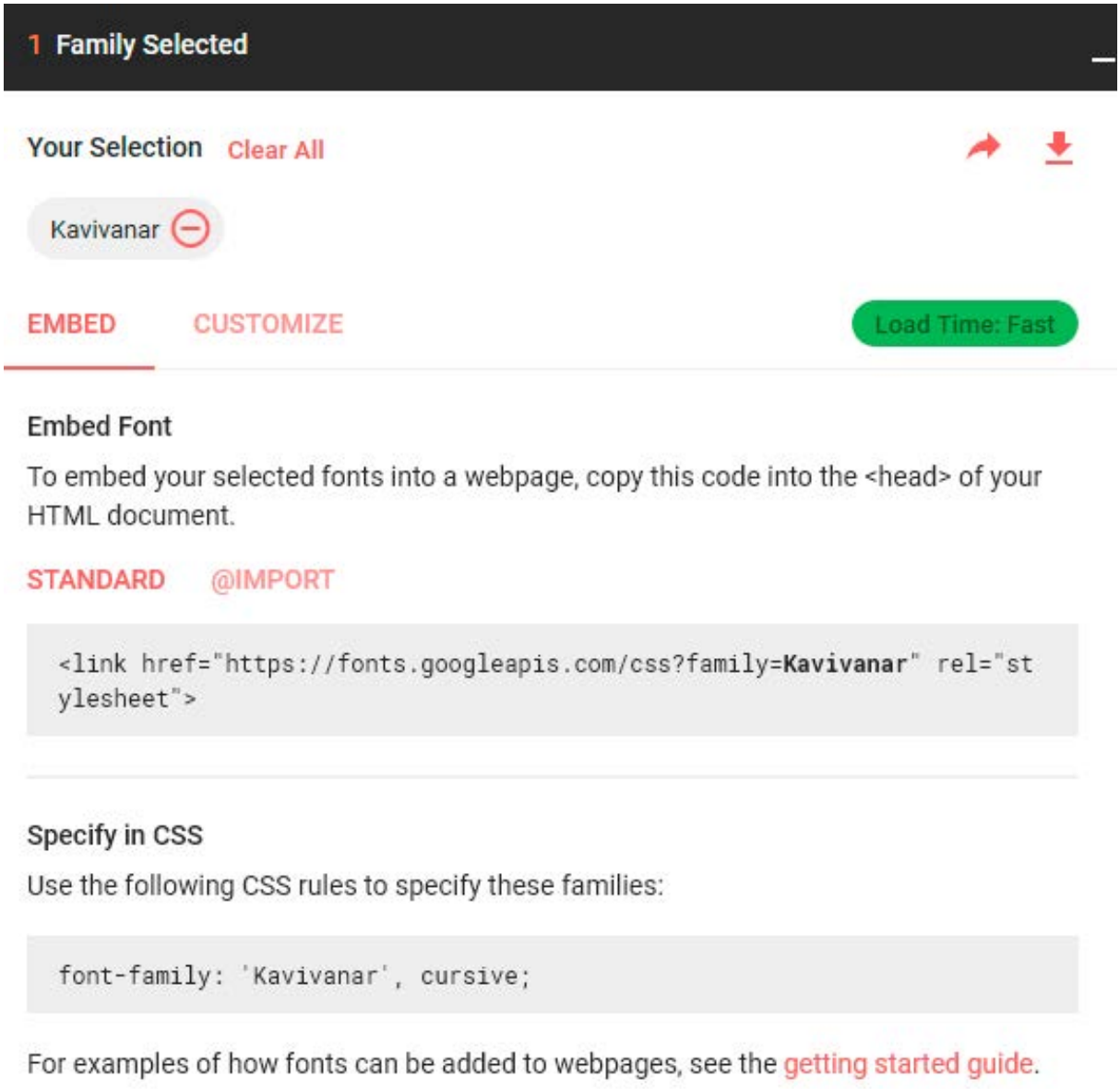

Fig.21 Vert21 © Skillandyou

La seconde méthode (@import) :

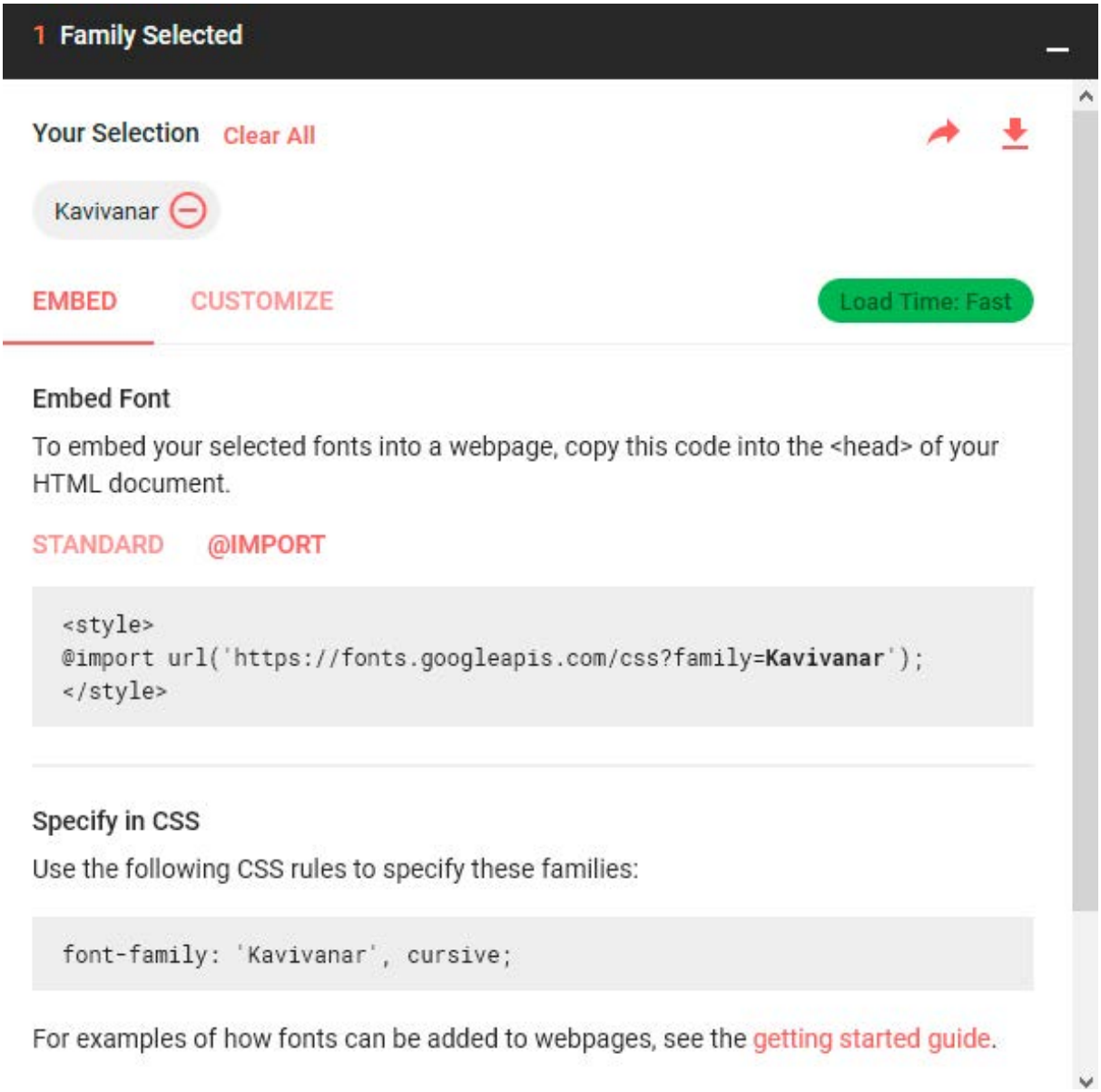

Fig. 22 Vert22 © Skillandyou

Il est également possible de télécharger les fichiers de police (l'icône cerclée de jaune de l'écran suivant) :

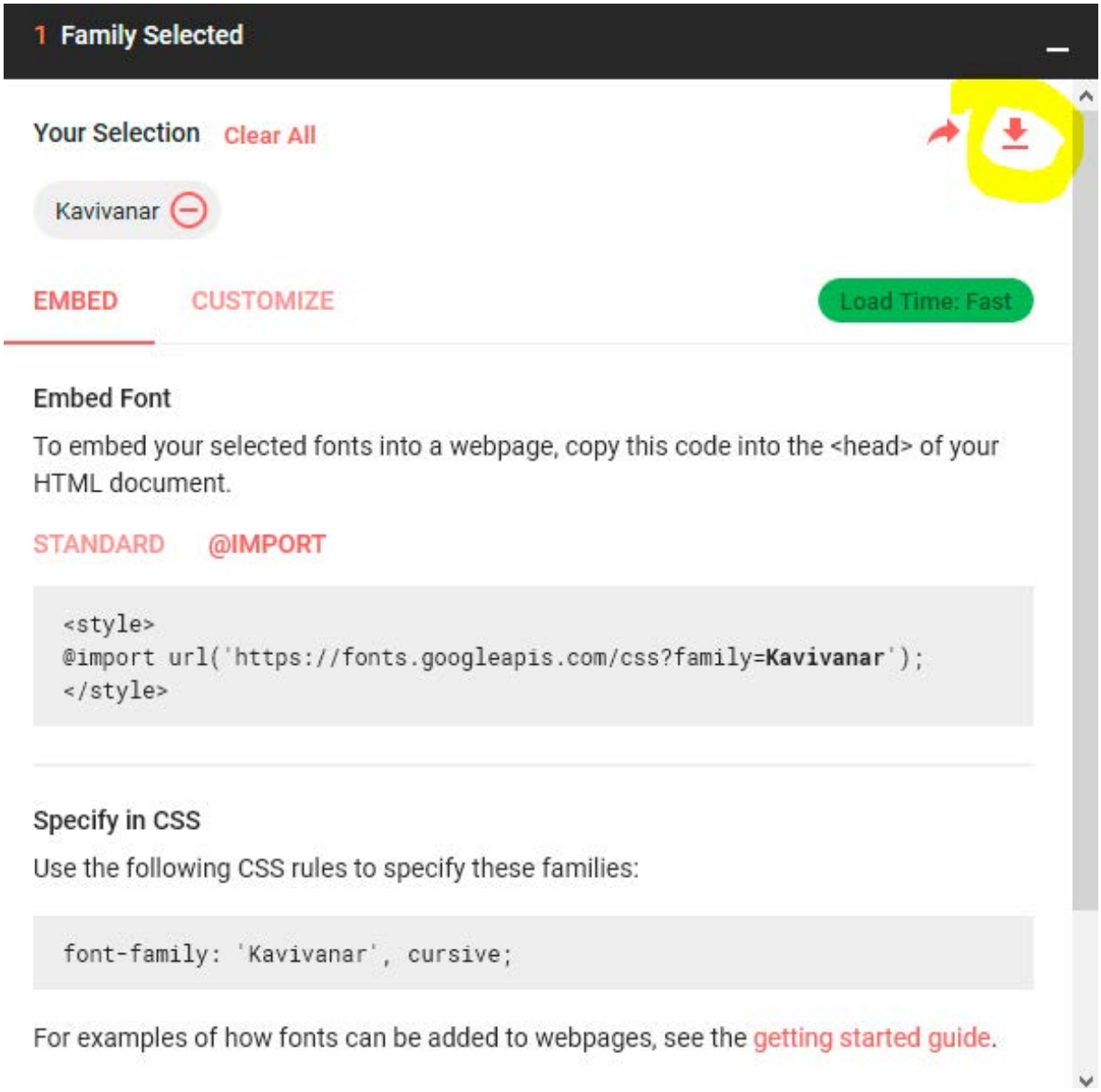

Fig. 23 Vert23 © Skillandyou

II. Les unités de police

A. Unités relatives et fixes

Il faut d'abord savoir que la taille du texte par défaut d'une page web est de **16px**. Mais cette taille peut être modifiée par l'utilisateur dans les paramètres du navigateur. D'autre part, l'internaute peut à tout moment zoomer sur sa page et augmenter soit la taille de l'image et des polices, soit uniquement la taille du texte.

Observons le comportement de deux unités différentes : les pixels et les em. Les premiers correspondent à une unité fixe, les seconds à une unité relative ou proportionnelle.

Que se produit-il dans les cas de figure suivants ?

Soit le HTML suivant :

```html
<div class="container">
    <p class="pixel">
        Cras cursus eget libero id fermentum. Ut vel lobortis quam. Donec
        venenatis porttitor sem, <span class="petit-px"> quis egestas diam
        laoreet rutrum. Donec feugiat, sapien ut consectetur pellentesque,
        magna mi dignissim ipsum,</span> id varius turpis nisi vel arcu. Proin
        malesuada lacus faucibus tellus auctor, ac commodo dolor bibendum.
    </p>
    <p class="em">
        Cras cursus eget libero id fermentum. Ut vel lobortis quam. Donec
        venenatis porttitor sem, <span class="petit-em">quis egestas diam
        laoreet rutrum. Donec feugiat, sapien ut consectetur pellentesque,
        magna mi dignissim ipsum,</span>id varius turpis nisi vel arcu. Proin
        malesuada lacus faucibus tellus auctor, ac commodo dolor bibendum.
    </p>
</div>
```

Fig. 24 Vert24 © Skillandyou

Et le CSS :

```css
.pixel {
  font-size:14px;
  color: #cb375a;
}
.petit-px{
  font-size:7px
}
.em {
  font-size:0.85em;
  color: #3c6bab;
}
.petit-em {
  font-size:0.85em
}
```

Fig. 25 Vert25 © Skillandyou

Donnent le résultat suivant :

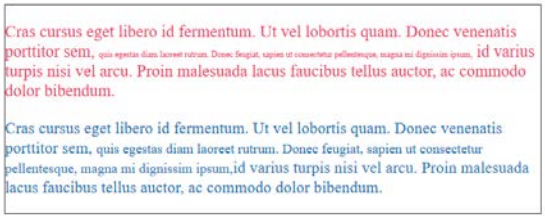

Fig. 26 Vert26 © Skillandyou

Si .petit-px (en rouge) et **.petit-em** (en bleu) ont une taille de police plus petite que celle de leur parent, **.petit-em** a la même valeur en em que son parent. Cela nous démontre que les unités em sont fonction de la taille du parent. En réalité, **.petit-em** fait 0,85 fois ou 85 % la taille de son parent.

Ajoutons une autre règle :

```css
body {
  font-size:150%;
}
```

Fig. 27 Vert27 © Skillandyou

Le rendu :

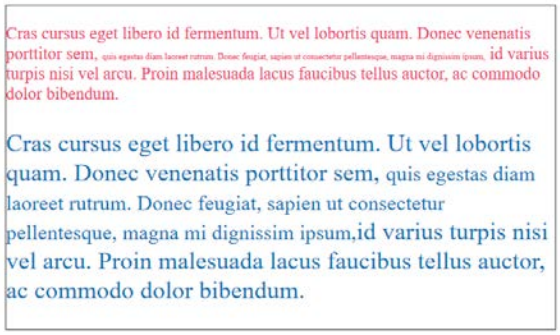

Fig. 28 Vert28 © Skillandyou

On remarque que le texte qui est écrit en **em** a grandi (en bleu), alors que celui qui est écrit en **px** n'a pas été modifié. En effet, l'unité em s'exprime en pourcentage de son parent. Il suffit donc de modifier la taille du texte sur la balise **<body>** pour que tous les éléments enfants le soient aussi.

Faisons encore une autre expérience et écrivons :

```
.em {
  font-size:1.5em;
  color: #3c6bab;
}
.petit-em {
  font-size:1em
}
```

Fig. 29 Vert29 © Skillandyou

Le rendu :

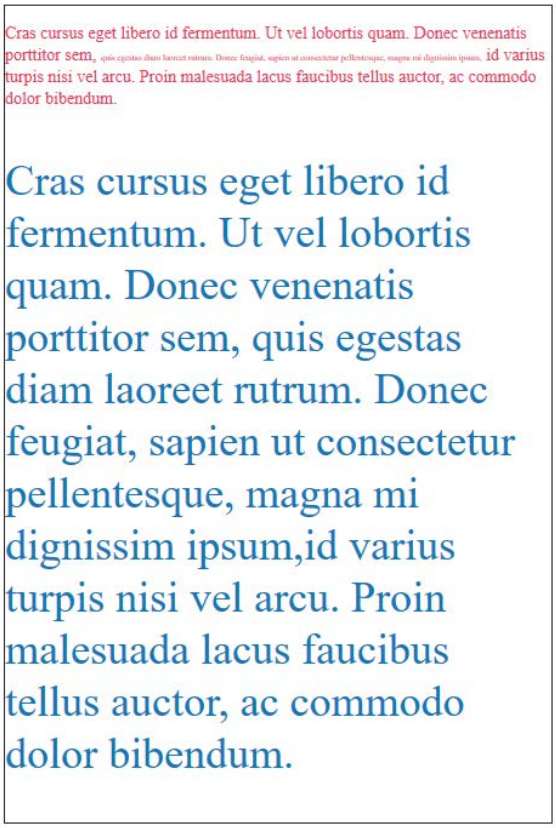

Fig. 30 Vert30 © Skillandyou

Cette fois-ci, **.em** et **.petit-em** font la même taille (on ne distingue plus de différence de caractère dans le paragraphe en bleu). En effet, **.em** demande à être 1,5 fois plus gros que la police définie en body. Mais **.petit-em**, qui est l'enfant de **.em** demande à être 1 fois plus grand que **.em**, c'est-à-dire à faire la même taille.

B. Equivalence em et pixel

On peut tenter de réaliser une équivalence entre les em et les pixels, ces derniers sont en effet plus simples à manipuler. Pour cela, on joue sur le font-size de body, afin qu'à 1em corresponde 10px. Il suffit en effet d'indiquer une valeur de 62,5 % au font-size de la balise body. Pour comprendre cette correspondance de valeur il faut se souvenir qu'une police à 100 % de sa taille fait 16px.

Pour connaître la valeur en em de 10px, on divise 10 par 16. En faisant ce calcul on tombe sur 10 / 16 = 0,625. On retombe donc sur nos pieds.

Ainsi, il suffit d'écrire :

```css
body {
  font-size:62.5%;
}
```

Fig.31 Vert31 © Skillandyou

Désormais, si on écrit **font-size:1.2em**, on aura l'équivalent de 12px, à 0.8em on aura 8px. Il faudra cependant faire attention à la cascade CSS, c'est-à-dire l'héritage. Pour conserver la correspondance px/em quelle que soit l'imbrication, il faudrait être certain de se référer à la taille de police définie à la racine, qui peut être différente de celle du parent direct. Notre premier exemple nous le montre : 0.85em de la classe **petit-em** se réfère à la définition de police du parent **em** et non de **body**.

C. Les unités rem

L'idéal serait d'avoir une taille définie de référence qui n'apparaîtrait qu'une fois pour toutes dans la feuille de style CSS. Or, il existe une unité de mesure qui permet de le faire : il s'agit du rem. Avec une déclaration en rem, on revient à la racine de la valeur qui définit la taille de police. Et, contrairement à ce que nous avons vu plus haut, cette première déclaration doit obligatoirement se faire sur la balise **HTML**.

Observons alors les différences entre les unités **px, em et rem**. On ajoute un paragraphe en **rem** entre les deux premiers, soit :

```html
<div class="container">
    <p class="pixel">
        Cras cursus eget libero id fermentum. Ut vel lobortis quam.
        Donec venenatis porttitor sem, <span class="petit-px"> quis
        egestas diam laoreet rutrum. Donec feugiat, sapien ut
        consectetur pellentesque, magna mi dignissim ipsum,</span>
        id varius turpis nisi vel arcu. Proin malesuada lacus
        faucibus tellus auctor, ac commodo dolor bibendum.
    </p>
    <p class="rem">
        Cras cursus eget libero id fermentum. Ut vel lobortis quam.
        Donec venenatis porttitor sem, <span class="petit-rem">quis
        egestas diam laoreet rutrum. Donec feugiat, sapien ut
        consectetur pellentesque, magna mi dignissim ipsum,</span>
        id varius turpis nisi vel arcu. Proin malesuada lacus
        faucibus tellus auctor, ac commodo dolor bibendum.
    </p>
    <p class="em">
        Cras cursus eget libero id fermentum. Ut vel lobortis quam.
        Donec venenatis porttitor sem, <span class="petit-em">quis
        egestas diam laoreet rutrum. Donec feugiat, sapien ut
        consectetur pellentesque, magna mi dignissim ipsum,</span>
        id varius turpis nisi vel arcu. Proin malesuada lacus
        faucibus tellus auctor, ac commodo dolor bibendum.
    </p>
</div>
```

Fig.32 Vert32 © Skillandyou

Le CSS :

```css
html {
  font-size:62.5%;
}
.pixel {
  font-size:18px;
  color: #cb375a;
}
.petit-px{
  font-size:12px;
}
.em {
  font-size:1.8em;
  color: #3c6bab;
}
.petit-em {
  font-size:1.2em;
}
.rem {
  font-size:1.8rem;
  color: #428027;
}
.petit-rem {
  font-size:1.2rem;
}
```

Fig.33 Vert33 © Skillandyou

Le rendu :

Fig.34 Vert34 © Skillandyou

On observe que les blocs en pixels (rouge) et en unités **rem** (vert) ont exactement la même taille, par contre le paragraphe en **em** (bleu) n'est pas identique. Si on observe attentivement, on comprend que le texte de **p.em** possède la même taille (on a défini un font-size dans la balise HTML pour qu'il en soit ainsi pour les enfants directs) mais que **span.petit-em** est paradoxalement plus grand que les autres polices.

D. Les unités vw

Il existe une dernière valeur proportionnelle que l'on peut utiliser pour les polices : il s'agit des **vw**. Il s'agit de calculer une taille en rapport avec la largeur du viewport (de la fenêtre). En effet, **vw** veut dire **viewport-width**, à mettre en rapport avec l'équivalent pour les hauteurs **vh** (**viewport-height**).

Soit l'exemple suivant :

On a deux paragraphes, le premier avec des valeurs de **h1**, **h2** et **p** par défaut, sachant que l'on n'a rien indiqué sur **body** ni **HTML** et qu'il s'agit dans les deux cas d'une police identique.

Le CSS :

```css
.vm h1 {
  font-size: 4vw;
}
.vm h2 {
  font-size: 3vw;
}
.vm p {
  font-size: 1.4vw;
}
```

Fig. 35 Vert35 © Skillandyou

Le résultat pour un écran de près de 1400px (le premier paragraphe est en pixels, le second en **vw**) :

Fig. 36 Vert36 © Skillandyou

Dès qu'on rétrécit l'écran (ici 650px):

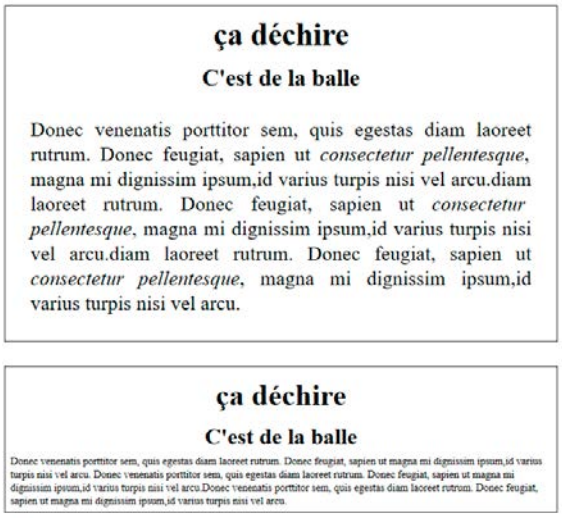

Fig. 37 Vert37 © Skillandyou

Le texte devient très petit, ce qui sera compliqué si l'on travaille des pages pour qu'elles soient *responsive*. À utiliser avec prudence.

On se tournera donc de préférence vers les unités rem ou em si l'on désire travailler avec des unités relatives.

III. Grille verticale

A. Grille verticale

Qu'est-ce qu'une grille verticale ?

La grille verticale est la manière d'agencer le texte afin que les lignes de texte tombent toujours sur le même niveau. Un exemple visuel étant plus facile à comprendre qu'un long discours, on va comparer la différence entre le texte d'une page sans grille verticale et celui d'une page réalisée à l'aide de celle-ci.

Sans grille :

Fig. 38 Sans grille © Skill and You

Avec une grille verticale :

Fig. 39 Avec une grille verticale © Skill and You

On remarque que sur la deuxième image, le texte (balise **p**) de la colonne de gauche est aligné sur celle de droite. Ce qui n'est pas le cas dans notre première image.

La grille verticale est donc une série de règles CSS concernant les éléments textuels et permettant de répartir les éléments sur cette grille.

La grille verticale d'une page web se définit à l'aide de trois valeurs : la taille de la police (**font-size**), la hauteur de ligne ou hauteur d'x basée sur la taille que prend la lettre x dans un jeu de police (**line-height**), et les valeurs des marges internes et ou externes. L'unité de base va donc être la hauteur de ligne (line-height).

Les traits que l'on aperçoit sur notre copie d'écran plus haut sont ceux de la hauteur de ligne du texte – c'est-à-dire en fait de p, l'élément qui reçoit le plus de texte. La hauteur de ligne va donc être notre unité de référence, de même que lorsque l'on a étudié les grilles de mise en page on avait une unité de référence (base-6, base-12, etc.).

Une grille verticale typographique peut s'avérer compliquée à mettre en œuvre. Il est intéressant d'utiliser un petit trait en arrière-plan afin de nous repérer dans la page.

On peut le réaliser avec un simple dégradé sur body :

```
body {
  background:  linear-gradient(to bottom, #E7E3CF .1em, transparent .1em );
  background-size: 100% 1.5em;
}
```

Fig. 40 Ajout d'un dégradé sur body © Skill and You

La première ligne nous permet de réaliser un trait horizontal de .1em, puis grâce à **background-size** on indique l'unité de base choisie pour notre grille verticale, c'est-à-dire la hauteur de ligne (ici une hauteur de ligne de 1.5).

En théorie, c'est simple. Chaque fois que l'on ajoute un espace vertical, c'est-à-dire un bloc ou un élément textuel (titre, paragraphe) qui possède une marge (**margin** et/ou **padding**), il faut faire en sorte que le bloc ajouté soit un multiple de notre unité de référence.

Par exemple, si mon unité de grille est de 30 pixels, je vais ajouter des espaces internes ou externes correspondant à 30 ou un multiple de 30px.

Ce qui ne veut pas dire que la marge doit toujours être de 30px, on peut avoir **padding-top** 10px et **margin-bottom** de 20px – attention à la fusion des marges.

B. Le calcul du rythme vertical en pixels

Pour avoir un rythme vertical cohérent, il nous faut donc deux ingrédients : des hauteurs de ligne (line-height) et des marges en cohérence avec la hauteur de ligne.

Mais comment trouver ou calculer la hauteur de ligne qui deviendra l'unité de référence de notre grille verticale ?

Elle va être obtenue à partir de la hauteur de ligne de la page qui sera un ratio généralement compris entre 1.2 et 1.8, en tout cas autour de 1.5.

Dans le cas d'une taille de texte de 16px, si notre hauteur de ligne possède un ratio de 1.5, on va multiplier l'un par l'autre, 16 x 1.5 = **24** . Toutes les hauteurs de ligne et les marges inférieures doivent obligatoirement être des multiples de 24px.

Ainsi le CSS suivant :

```
html {
  font-size: 16px;
}
body {
  font-size: 100%; /*soit 16px*/
  line-height: 1.5
  /*soit 24px (= 1.5 * 16)
  => unité de référence */
}
h1{
  font-size: 32px;
  line-height: 48px; /*2 unités*/
  margin-bottom:48px; /*2 unités*/
}
h2  {
  font-size: 25px;
  line-height: 48px; /*2 unités*/
  padding-top: 12px; /*1/2 unités*/
  margin-bottom:12px; /*1/2 unités*/
}
h3 {
  font-size: 18px;
  line-height: 24px; /*1 unité*/
  padding-top: 10px;
  margin-bottom:14px;
  /*10 + 14 = 24px
  => 1 unité */
}
p {
  margin-top: 0px;
  margin-bottom:24px; /*1 unité*/
}
```

Fig. 41 Fichier CSS © Skill and You

On obtient le résultat suivant :

Fig.42 Affichage sur un navigateur © Skill and You

La grille fonctionne parfaitement.

C. Grille verticale en em

Maintenant si l'on veut utiliser des valeurs **em** , cela va être un peu plus complexe, en effet chaque fois que l'on aura :

```
h1{
  font-size: 32px;
  line-height: 48px;
  margin-bottom: 48px;
}
```

Fig.43 Valeurs px © Skill and You

Si l'on écrit en em :

```
h1{
  font-size: 2em;
  line-height: 1.5;
  margin-bottom: 2em;
}
```

Fig.44 Valeurs em © Skill and You

On obtient :

Fig.45 Affichage sur un navigateur © Skill and You

Après le h1 (« ça déchire ») sur la colonne de gauche, notre grille ne fonctionne plus.

On est donc obligé de réaliser une conversion des pixels en em (il est plus simple de réaliser la grille verticale en pixels qu'en unités relatives). On va simplement utiliser une règle de trois. Mais attention les em étant une unité relative au parent, il faut distinguer la taille de la police du texte que l'on veut exprimer en rapport avec la taille de la police dans le body. Puis, l'interlignage et les marges qui devront être exprimés en rapport avec la taille de la police du texte de notre élément.

On retiendra deux formules :

1. **Taille de texte en em** : taille du texte désiré en pixels divisé par notre unité de référence en pixels (sur body). Soit

Tem (font-size) = Tpx / Tpxbody

2. **Interlignage et marges** : interlignage initial en pixels divisé par la taille du texte désiré en pixels. Soit INTem = INTpx / Tpx

Dans notre exemple, la taille du texte en body est de 16px.

La taille du texte désirée est donc : 32 / 16 = 1.3333, soit 1.3333em

La taille de notre interlignage se calcule à partir de la taille du texte désiré en px, soit 32.

Taille de l'interlignage : 48 / 32 = 2em

On peut utiliser la fonction **calc** pour simplifier les choses, et en regardant l'ensemble des règles, si cela n'est pas encore clair, vous allez comprendre :

```
body {
    font-size: 16px;
    line-height: 1.5;
}
h1{
  font-size: calc(32em / 16);
  line-height: calc(48em / 32);
  margin-bottom: calc(48em / 32);
}

h2  {
  font-size: calc(25em / 16);
  line-height: calc(48em / 25);
  padding-top: calc(12em / 25);
  margin-bottom:calc(12em / 25);
}
h3 {
  font-size: calc(18em / 16);
  line-height: calc(24em / 18);
  padding-top: calc(10em / 18);
  margin-bottom: calc(14em / 18);
}
p {
  margin-top: 0px;
  margin-bottom: calc(24em / 16);
}
```

Fig. 46 Fichier CSS © Skill and You

Vous remarquez que la taille de la police est toujours calculée par rapport à la taille de la police de référence (16), mais que les autres valeurs sont calculées par rapport à font-size. La marge de p étant particulière puisque 16 est la valeur de police de référence.

Résultat :

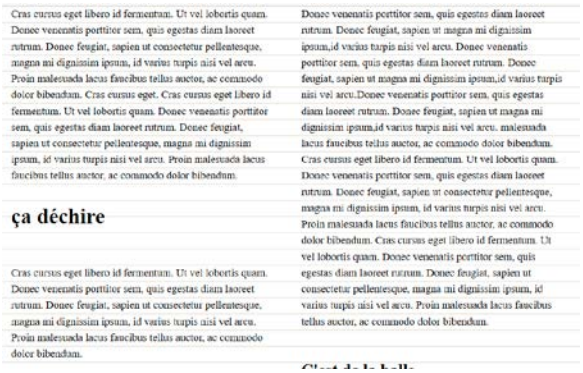

Fig. 47 Affichage sur un navigateur © Skill and You

D. Grille verticale en rem

Pour les rem, c'est encore plus simple, tous les éléments se trouvent en rapport avec la définition en px de la taille de la police en **body**.

On retiendra deux formules :

Taille de texte, de l'interlignage et des marges en em : taille du texte désirée en pixels divisée par notre unité de référence en pixels (sur body).

Soit

Tem = Tpx / Tpxbody

INTem = Tpx / Tpxbody

Notre CSS devient :

```css
body {
    font-size: 16px;
    line-height: 1.5;
}
h1{
  font-size: calc(32rem / 16);
  line-height: calc(48rem / 16);
  margin-bottom: calc(48rem / 16);
}

h2  {
  font-size: calc(25rem / 16);
  line-height: calc(48rem / 16);
  padding-top: calc(12rem / 16);
  margin-bottom:calc(12rem / 16);
}
h3 {
  font-size: calc(18rem / 16);
  line-height: calc(24rem / 16);
  padding-top: calc(10rem / 16);
  margin-bottom: calc(14rem / 16);
}
p {
  margin-top: 0px;
  margin-bottom: calc(24rem / 16);
}
```

Fig. 48 Fichier CSS © Skill and You

Le résultat est le même.

E. Le cas des images

Il faut dès lors faire attention aux images. Comme tous les autres éléments de la page, les images vont avoir un effet sur l'espace vertical.

Pour conserver le rythme vertical on pourra par exemple donner une taille dans la valeur relative de notre grille verticale.

Soit le HTML suivant :

```html
<h1>ça déchire  </h1>
<figure>
    <img src="img/shoes.jpg">
    <figcaption>explanatory caption</figcaption>
</figure>
<p>
    Cras cursus eget libero id fermentum. Ut vel lobortis quam. Donec venenatis porttitor
    sem,  quis egestas diam laoreet rutrum. Donec feugiat, sapien ut consectetur pellentesque,
    magna mi dignissim ipsum, id varius turpis nisi vel arcu. Proin malesuada lacus faucibus
    tellus auctor, ac commodo dolor bibendum.
</p>
```

Fig. 49 Fichier HTML © Skill and You

On a une balise **figcaption** dont il faudra s'occuper. Ici, on va simplement lui demander une couleur différente et de l'italique.

Ensuite il suffit de donner une hauteur minimale à **figure** qui reprenne notre unité de base et d'indiquer une largeur maximale à **img**. Soit :

```css
p, figcaption {
  margin-top: 0px;
  margin-bottom: calc(24em / 16);
}
figcaption {
  font-style: italic;
  color: #888;
}
img {
  max-width: 100%
}
figure {
  min-height: 22.5em;
}
```

Fig. 50 Fichier CSS © Skill and You

Soit :

Fig. 51 Affichage sur un navigateur © Skill and You

C'est donc exactement le même principe que pour les autres éléments. L'inconvénient vient du fait qu'au redimensionnement de la page, on pourra avoir des décalages.

Si on utilise une unité en rem, en revanche :

Écran large :

Fig. 52 Affichage sur un navigateur avec un écran large © Skill and You

Écran rétréci :

Fig. 53 Affichage sur un navigateur avec un écran rétréci © Skill and You

Cela peut s'avérer très complexe. Aussi est-il plus intéressant de n'avoir qu'une seule colonne lorsqu'on a des images qui ne perturberont pas le rythme vertical mais le repousseront, ou de les faire systématiquement flotter autour du texte.

IV. Conclusion

La typographie est un élément trop souvent négligé par les webdesigners et/ou intégrateurs. En particulier, le rythme vertical est compliqué à maîtriser lorsque l'on a beaucoup d'images et d'autres éléments visuels sur nos pages. Car s'il faut impérativement savoir manipuler les objets avec des unités relatives, le choix de typographies exotiques pour lesquelles il n'existe pas de valeur standard universelle ne simplifie pas le travail du typographe pour le Web.

On consultera avec profit les liens suivants.

Générateur de grille typographique :
https://drewish.com/tools/vertical-rhythm/
https://soqr.fr/vertical-rhythm/
https://www.gridlover.net/try
https://topfunky.com/baseline-rhythm-calculator/

Des tailles avec un ratio d'augmentation :
https://type-scale.com/