Jc Ghyselinck

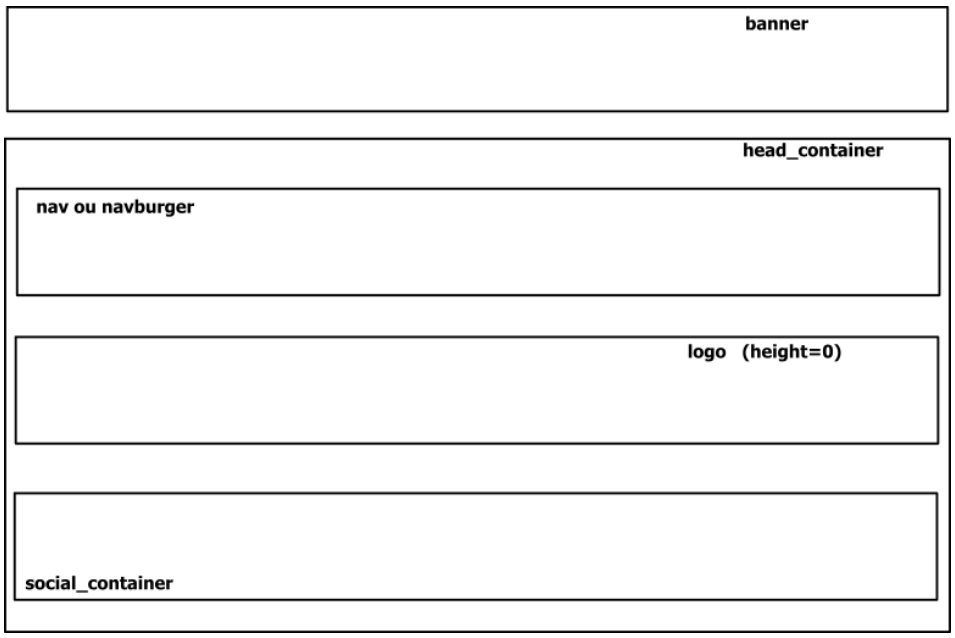
Codage du site ZEvent

Page de la boutique

_____

1\ Zoning de la page :

1a\ Le header



Le **head_contenair** est positionné en flex. Appartenant au **body**, il a une largeur à 100% ; cela permettra donc d'assurer le centrage des éléments qu'il contient, par rapport au **body**, à l'aide des directives flex placées à bon escient. Le logo ZEvent est dans une **div** qui a sa hauteur à zéro. Cela permet de garder le centrage facilement au cours du responsive.

Tout en haut, juste après la balise d'ouverture du **body**, se trouve une bannière (**banner**) d'avertissement sur la taille. Cette bannière doit disparaitre lorsque le menu de navigation est au sommet de la fenêtre. Cela est effectué par un **sticky**, et un **top** à 0px, placés sur le **header**, sachant que le **body** (le père du **header**) est positionné par défaut.

La semi-transparence de la **nav** est obtenue par les propriétés **background-color : transparent** et **backdrop-filter : blur(8px)**.

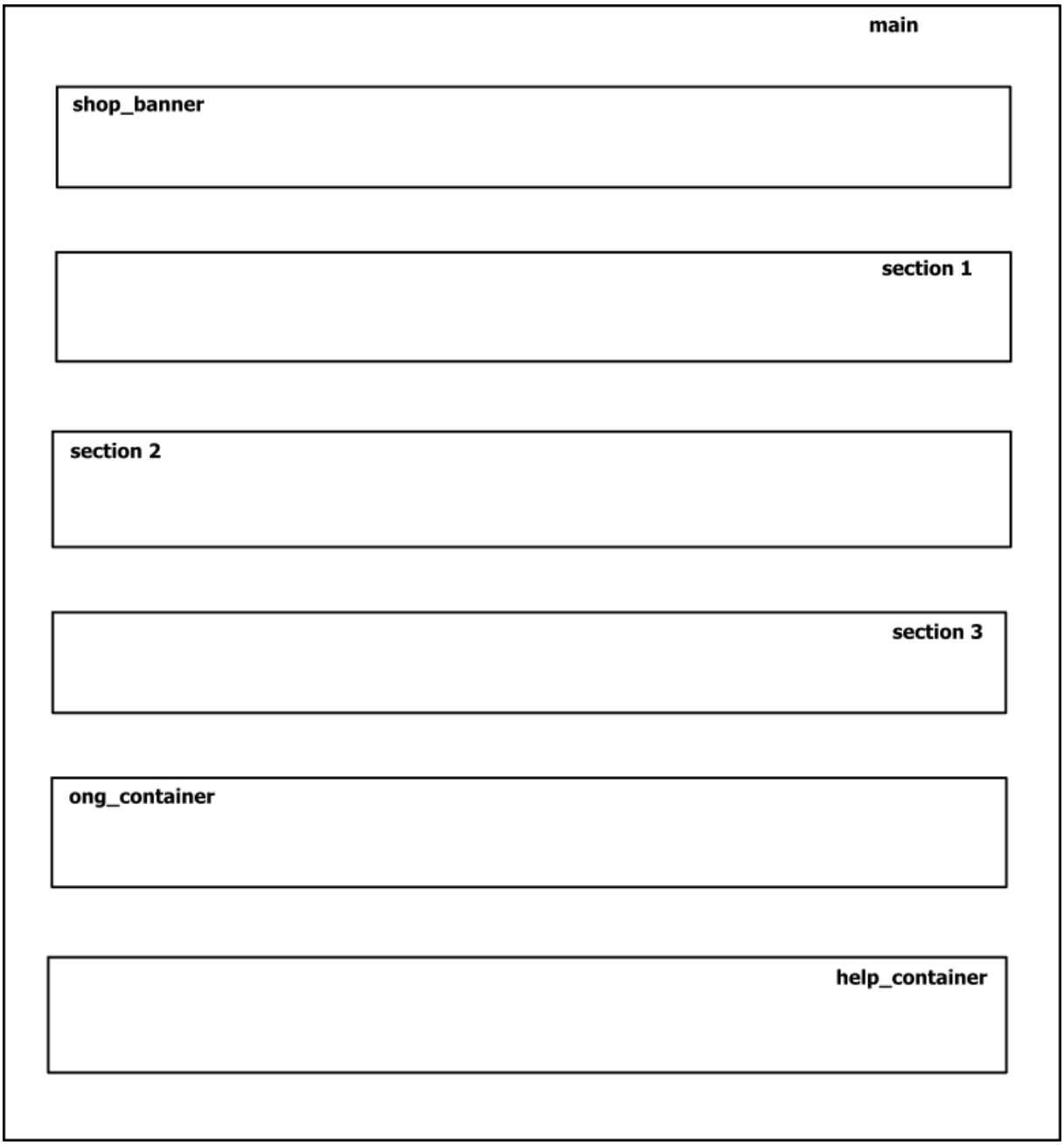
Nous verrons le menu burger dans le chapitre reponsivité.

Le reste est classique et explicite dans le CSS en lui-même.

1b/ Le corps de page

Le corps de la page est inclus dans un conteneur général appelé **main**. Ce conteneur est en **display flex**, column, center. Ce qui assure un centrage par rapport au **body** de ce qu'il contient.

Le site est full-responsive jusqu'à une largeur mini de 320px. Je n'ai pas mis de limite en largeur maxi.



Ce conteneur principal contient :

- la **shop_banner**, qui est l'image bannière de la boutique

- 3 sections qui sont différenciées (**section1**, **section2**, **section3**)

    Les Tee-shirt

    Tirage sur Dibond

    Trophée numérotés

Chacune de ces sections contient un système d'informations basé sur des onglets. Etant donné la similitude de ces onglets, ils seront codés et généré en POO JavaScript.

- une section **ong_container** contenant 4 cartes qui rappelle les ONG concernées.

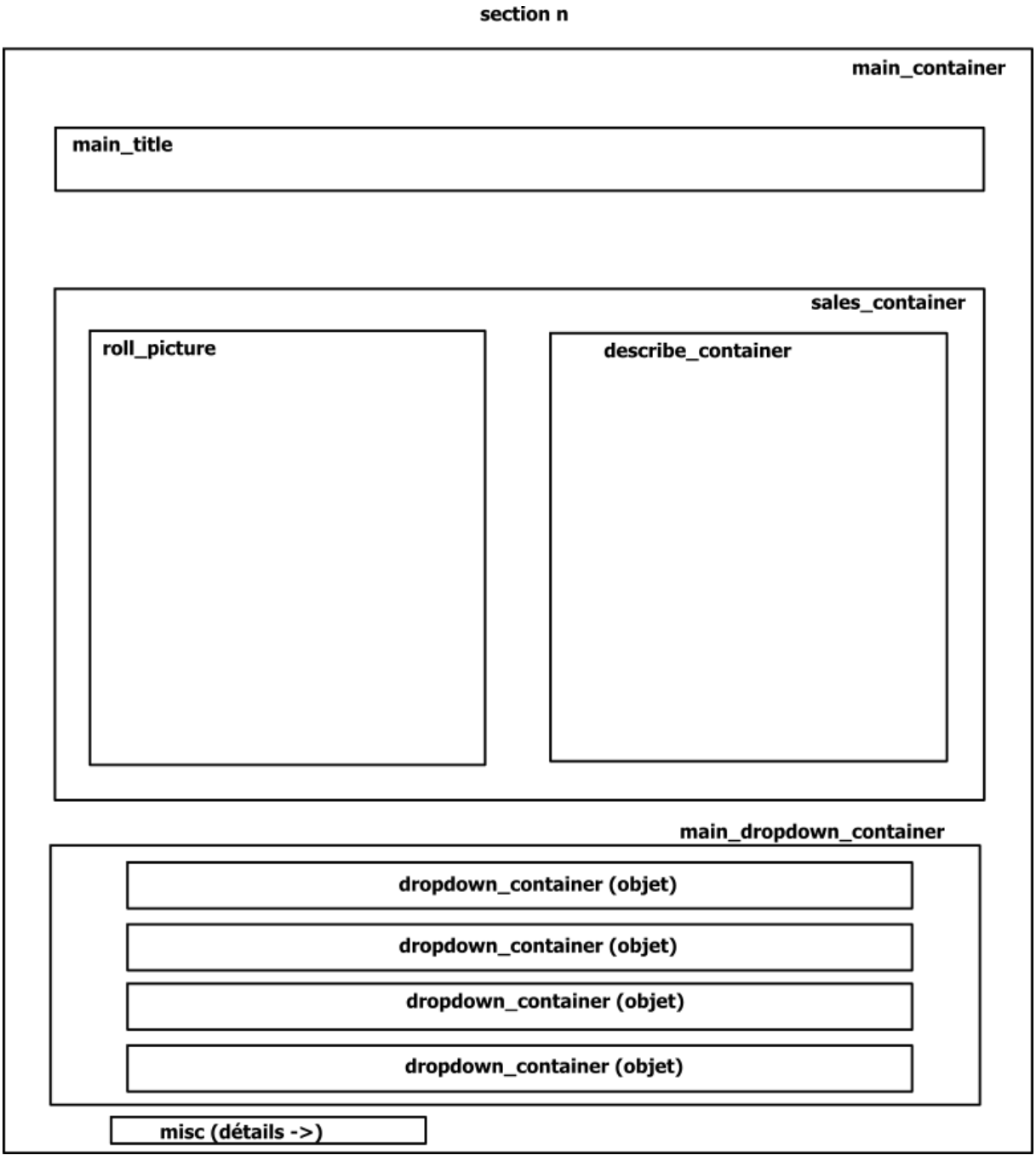
- Une section Aide et FAQ (**help_container**)

Soit 6 sous-ensembles en tout.

Reprenons en détails chacun de ces ensembles :

*La photo de **shop_banner*** étant placée dans une **div**, celle-ci s'adapte automatiquement à la largeur de la fenêtre.

Les trois sections bien que différenciées ont une structure commune que voici :



Le conteneur ***roll_picture*** contiendra un carousel de démonstration qui défilera de façon infinie simulant les produits en stock ou non.

Le carousel est codé en pur CSS à l'aide des **@keyframes** et de la directive **animation**.

2 images A et B sont placées en ligne de cette façon : A B A. Cette « ligne » est placée dans une div (**roll_picture**) à la largeur d'une image, placée en **overflow : hidden** de sorte qu'une seule image n'apparaisse dans la **div**. Ensuite une translation de la largeur de chaque image, et ce, en boucle assure l'effet. Il va sans dire que les images, différentes pour chaque section, ont dû être retaillées en largeur.

Dans la ***section 1***, le **describe_container** contient un ensemble de 7 boutons. Ces boutons ont été codés à l'aide d'un **form** de type **radio**. Les coches ont été surchargées avec des **div** qui contiennent les labels des boutons.

La gestion de ces boutons et leurs CSS sont confiée au JavaScript.

Ces boutons sont positionnés avec la propriété **grid**, le formulaire étant placé dans un sous-conteneur **aux_describe_container**, enfant du conteneur **describe_container**.

Tout ce système de conteneurs est piloté en **flex** pour assurer à la fois le centrage par rapport au **body** et le responsive. Celui-ci est assuré par une directive **flex-wrap**, placée sur **sales_container**. Chacun des contenus de ce conteneur n'excède pas 320px, ce qui permet d'atteindre le full-responsive sans aléa.

Quant aux éléments block contenant du texte, le responsive est automatique. De plus leurs tailles s'adaptent automatiquement à la fenêtre.

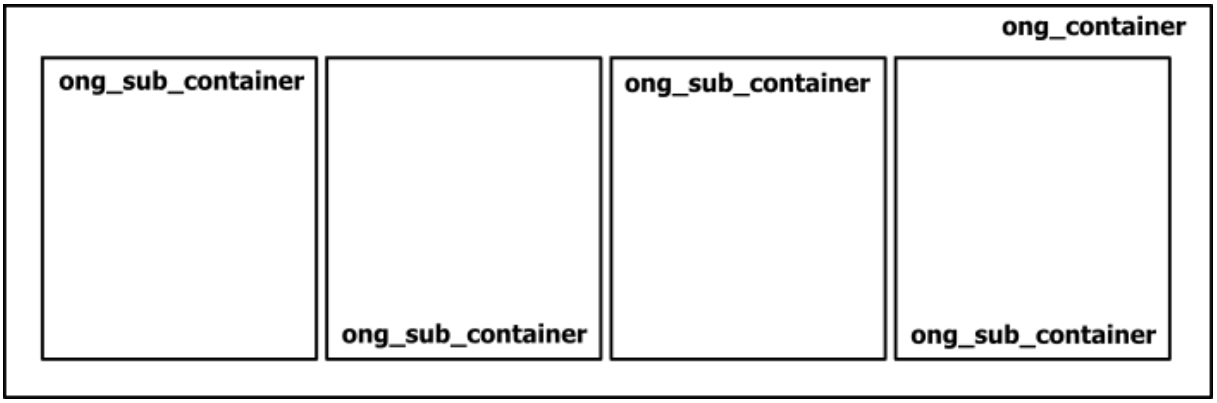
Ce conteneur contient également un bouton acheter maintenant qui devient vert au clic et reprend sa couleur initiale (pour la démo) lorsque l'on arrête de le survoler. J'ai confié ces effets au JavaScript.

Venons-en maintenant aux onglets de descriptions contenus dans **drop_down_container**. Ces onglets sont générés en JavaScript, une classe **Element** a dans son constructeur trois arguments : l'url de la petite icône du déclencheur de la dropbox, l'intitulé du déclencheur, et enfin le contenu de la dropbox.

3 x 4 = 12 objets ont ainsi été créés et placés au bon endroit de chaque section. Comme les onglets et leurs contenus ne contiennent que du texte et de petites icones, ils s'adaptent automatiquement à la largeur de la page.

Le reste n'appelle aucun commentaire particulier et est implicite dans le CSS très commenté.

Suivent les cartes contenant les ONG. La structure est la suivante :

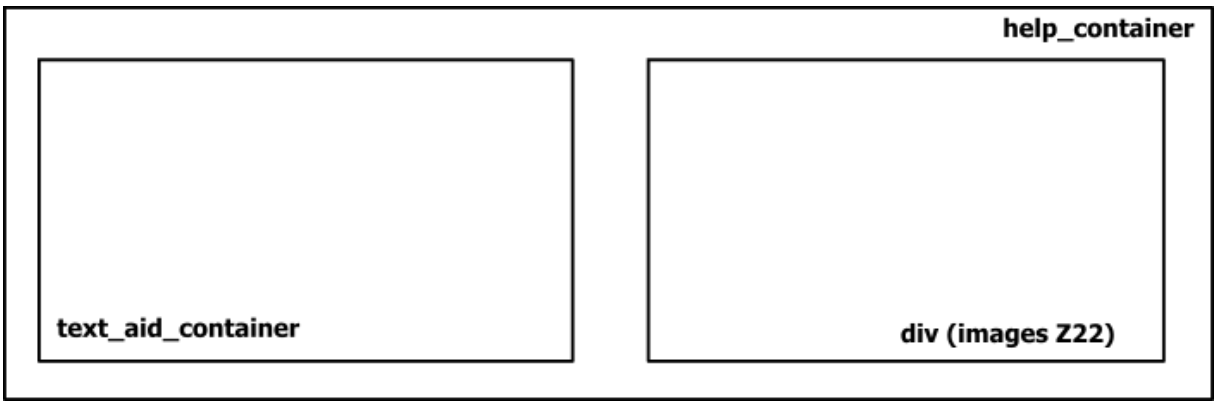


Cette section n'appelle aucun commentaire particulier. Le conteneur **ong_container** est placé en flex. Le responsive est assuré par la clause **flex-wrap : wrap** sur **ong_container** lui-même. La dimension des **ong_sub_container** est commune, juste en dessous de 320px, pour aller taper sans aléa le 320px mini du cahier des charge.

J'ai laissé en centrage les cartes, même quand elles « wrappent » pour des raisons d'esthétique par rapport aux autres éléments.

Enfin vient la section Aide et FAQ

Sa structure est la suivante :



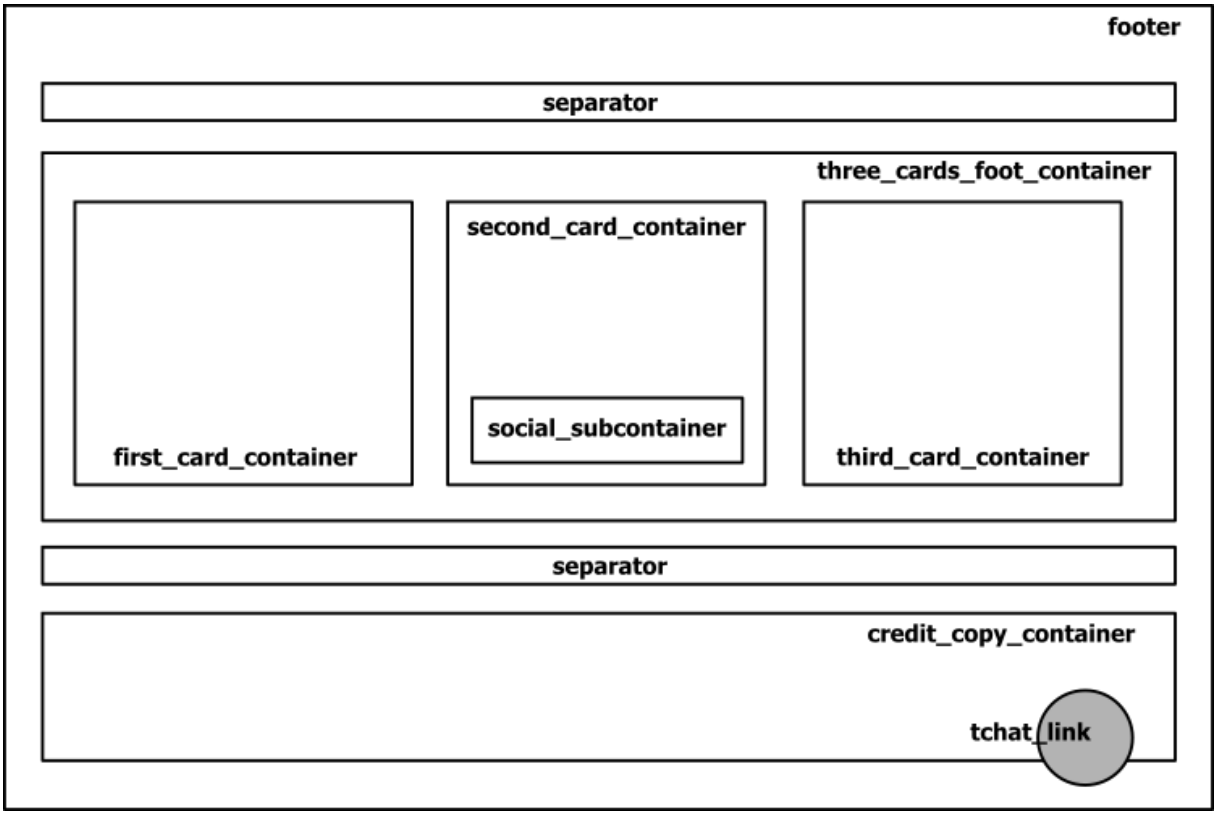
Comme d'habitude le **help_container** est placé en **flex**. Cependant lors du responsive, on veut que l'image Z22 se place au-dessus du texte de l'aide. Cela sera assuré par une directive **flex-wrap: wrap-reverse** placée sur ce même container.

Comme le texte de l'aide est assez long, lors de la configuration mobile, il faudra prévoir un redimensionnement en hauteur des éléments. Comme l'image a ses propres proportions, il faudra prévoir un « retaillage » de celle-ci. C'est pour cela qu'il existe deux images **calibre** et **calibre2**. Tout cela sera précisé dans la rubrique des média-queries. La largeur de l'image finira à 310px pour aller chercher la taille mini imposée par les mobiles.

Le reste est implicite dans le CSS.

1c/ Le footer

La structure est à la page suivante :



Rien de bien particulier, les séparateurs sont effectués avec une **div** de 1px de haut peinte en gris. Tout le système de conteneur est piloté en **flex**.

Le centrage des cartes est assuré par un **justify-content: space-evenly** placé sur **three_cards_foot_container**. Le responsive par un **wrap** sur ce même conteneur.

Le bouton du chat est placé en **position : absolute** par rapport à son père **credit_copy_container**, lui-même positionné en **relatif**. Afin de centrer l'icône de ce bouton, il est lui-même placé en **flex** avec les directives de centrage qui vont bien.

Pour qu'elle soit en blanc, une directive **filter: invert(100%)** a été placée sur l'icône du chat fournie en gris.

2\ Le responsive, les média-queries :

Le responsive est confié aux media-queries. Il y a 4 ruptures.

2a\ 1ere rupture à 1286px

Il s'agit de la gestion de l'inversion texte (**text_aid_container**) et de l'image Z22 (**help22_calibre.webp**) de la section de l'aide (**help_container**).

On a vu précédemment que le placement de l'image au-dessus du texte est fait avec un wrap inverse. Il faut cependant aussi inverser les border-radius. C'est tout simplement ce que fait cette **@media**.

Il est à noter que pour des raisons d'esthétique, de réglage et de responsivité, j'ai taillé l'image fournie à une hauteur de 310px, sachant que sa largeur deviendra sa hauteur lors d'une prochaine rupture. Sa largeur (635px), identique au conteneur du texte (en vert) est calculée pour que la largeur totale (conteneur texte + image) en vue desktop, soit approximativement la même que celle du conteneur des cartes des o.n.g. Leurs dimensions respectives sont : 635px x 310px et 314px x 227px.

La première image (format paysage) a été renommée **help22_calibre.webp**, la seconde (format portrait) **help22_calibre2.webp**. Ces deux images sont placées dans une même **div**, la deuxième est donc masquée par un **display : none** dans le **CSS** hors **@media**.

2b\ 2eme rupture à 960px

Cette rupture sert à augmenter la hauteur de la bannière d'avertissement pour la taille, à faire disparaitre le menu **nav** « traditionnel » et faire apparaitre le menu à icônes. Quelques icônes en noir sur blanc ont été inversées en blanc sur noir.

3c\ 3eme rupture à 640px

Cette rupture sert à la fois de mettre la section aide en configuration mobile et de placer le logo ZEvent à gauche du **header**.

On redimensionne le conteneur du texte en 314px, on fait disparaitre l'image Z22 en format paysage et on fait apparaitre l'image en format portrait. Ce bloc obtenu à ainsi 314px de large, prêt au format mobile.

Le réalignement du menu lors du passage de **ZEvent** à gauche est implicite dans le **CSS**. On en profite pour réajuster les titres des sections.

4c\ 4eme rupture à 440px

Cette rupture atteint la mise en format mobile et pourra se réduire jusque 320px.

On fait donc disparaitre le menu icônes, et apparaitre le symbole du menu burger qui va commander le sous menu de la **nav (navburger)**. La commande de ce menu sera confiée au JavaScript. Un ré-ajustage des titres de sections est également fait. Le bouton chat replacé.

3\ Le menu burger

La fenêtre du menu burger est pré-fabriquée dans la classe **navburger** et bien sûr placée en **display : none** hors **@media**.

Il est à noter que j'ai placé le bouton de fermeture (la croix) dans la même classe que le menuburger, constituant ainsi une mini fenêtre modale. Cela évite une réapparition, du plus mauvais effet, de l'icône menu burger si l'on agrandit la fenêtre lorsque le menuburger est ouvert. Ceci dit, cela n'a pas d'importance car avec un mobile ce ré-agrandissement n'est pas possible.

La croix est donc positionnée en **absolu** par rapport à son père le **header**, et est réglée de manière à masquer l'icône menu burger d'ouverture, lorsque le menu est activé.

J'ai également décidé de faire disparaitre la bannière de mise en garde des tailles lors de l'ouverture du menuburger. Cela évite un décalage, si le menu est ouvert pendant que cette bannière est visible.

L'ouverture et la fermeture du menu, la disparition et réapparition de la bannière sont confiées à quelques lignes de JavaScript.

Fin du document