Les pages web conçues sont destinées à être affichées sur des écrans de taille variable :

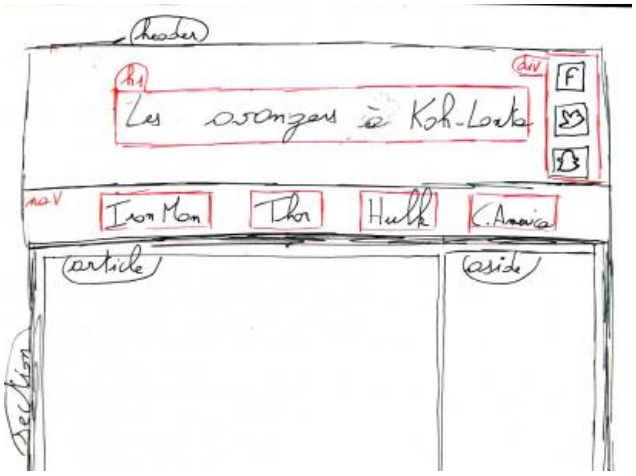
- de 480 à 720 px pour les smartphones,
- de 768 à 900 px pour les tablettes en mode portrait,
- de 900 à 1024 px pour les tablettes en mode paysage,
- de 1024 à 1200 px pour un ordinateur fixe ou portable.

Le **responsive design** fait référence à toutes les techniques de conception de site internet qui permettent d'adapter le contenu visuel à n'importe quel type et n'importe quelle taille d'écran. Grâce au responsive design, une page web et l'ensemble de son contenu (texte, image et vidéo) s'adaptent automatiquement à l'appareil que l'internaute utilise (téléphone mobile, tablette ou ordinateur). La lisibilité de la page est ainsi toujours optimisée.

On vous propose dans ce Tp de rendre un site *responsif*. On part de celui sur les *Avengers*, qui a servi de support de cours dans le chapitre précédent consacré au Css.

1. Que va-t-on faire ?

⇨ Télécharger le dossier *responsive.zip* contenant les fichiers *html* + *css* + images d'une version simplifiée du site. Décompresser et placer tous les fichiers dans un répertoire de travail.

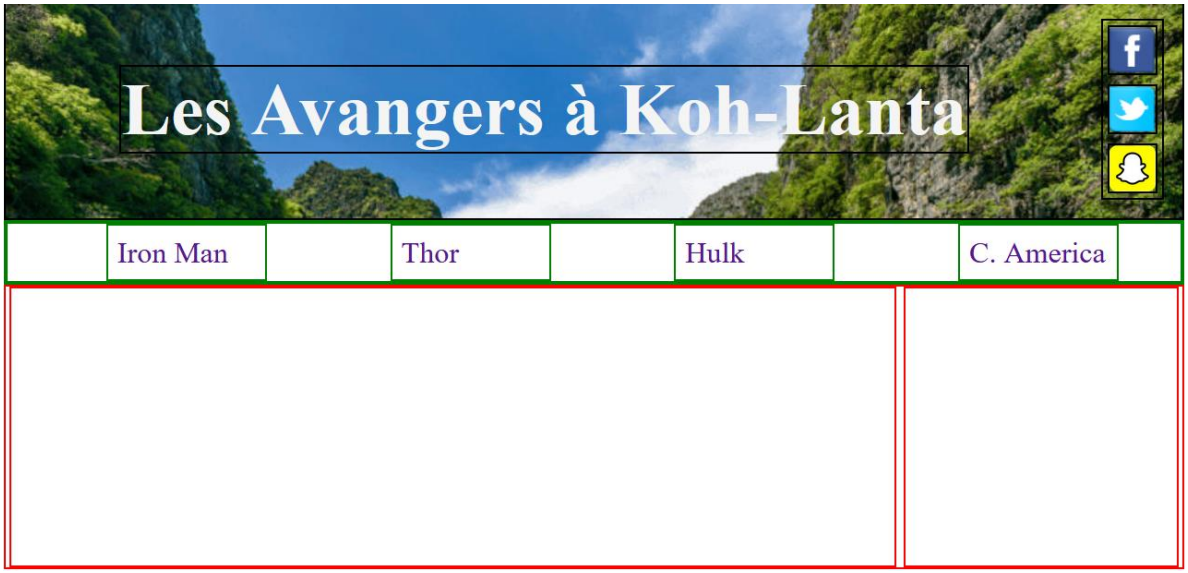


La structure de la page web de ce site est donnée sur la figure ci-contre.

On a 3 blocs qui occupent toute la largeur de la page.

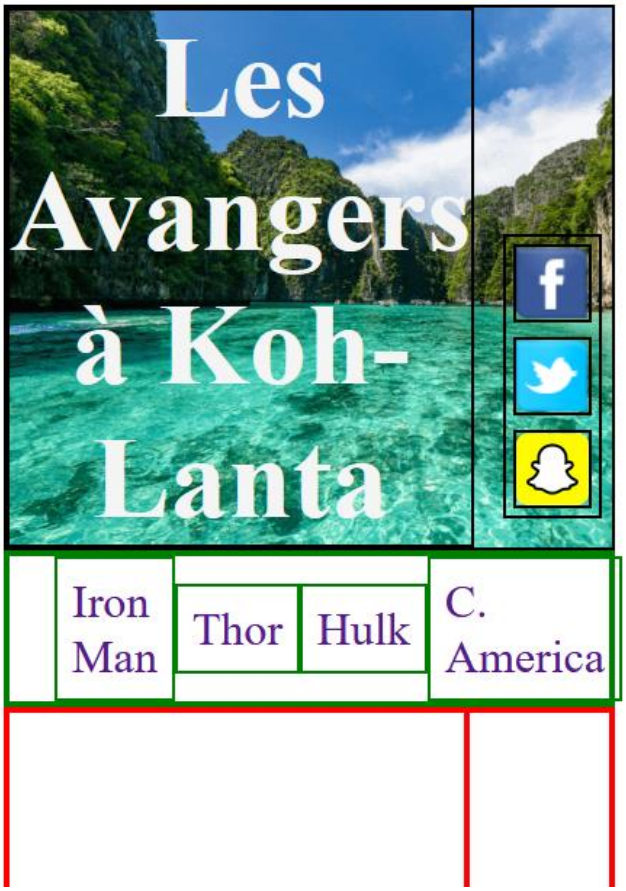
- <header> qui contient :
  - <h1>
  - <div class=''reseaux''>
- <nav> qui contient une liste <ul> contenant 4 lignes
- <section> qui contient <article> et <aside>

⇨ Afficher la page *avengers.html* dans le navigateur. ….. On obtient :



⇨ Réduire la largeur de la fenêtre. ……. On obtient l'affichage ci-contre. On y repère plusieurs défauts évidents :

- le titre du site prend beaucoup trop de place,
- Les différents items du bandeau de navigation se retrouvent collés et comprennent désormais un saut de ligne,
- la largeur du bloc <*aside*> définie dans le *Css* avec un *width:23%* , devient trop étroite pour y afficher correctement du contenu.



On propose, dans la suite, des solutions qui permettent de corriger ces défauts.

2. Peut-on avoir un Css qui s'adapte à la taille de l'écran ?

Oui, heureusement, c'est l'outil principal qui permettra de rendre le site *responsif*.

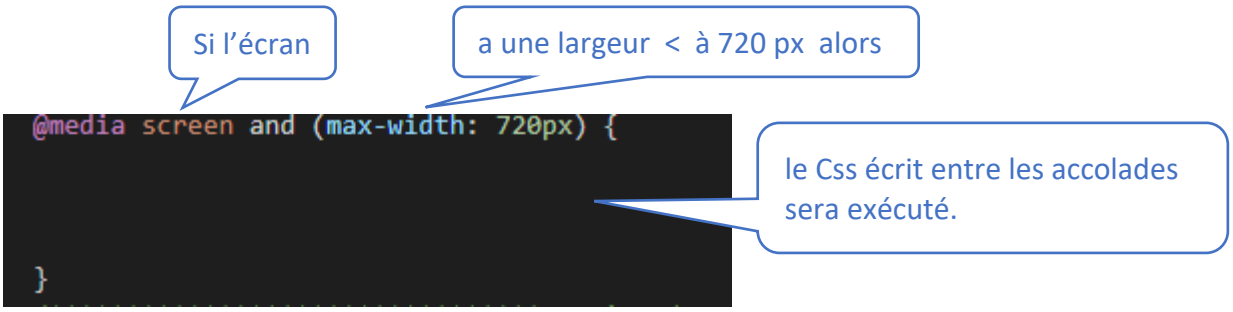
2.1. On rajoute des **Media Queries** dans le fichier Css

⇨ Ajouter le *media queries* ci-dessous dans le fichier *avengers.css,* afin de réduire la taille de la police du titre <*h1*> , les marges dans le bloc <*header*> et la taille des images dans <*header*>, lorsque la taille de l'écran est inférieure à 720 px :

```
/*********************************Gestion du header ********************************/
> header  {···
  }
> .reseaux{···
  }
> #icone {···
  }
header h1{ border:2px solid black;margin:auto;font-size: 5em;text-align: center;color:whitesmoke;}
.reseaux img{ border:2px solid black;width:50px;margin:5px;}
header img:hover{ border:4px transparent black;width:55px;margin:5px;border-radius: 10px;;}
@media screen and (max-width: 720px) {
    header h1{font-size: 3em;}
    .reseaux{margin:0px;}
    .reseaux img{ width:30px;}
}
```

Bloc média queries

La syntaxe de ce bloc média querie est la suivante :



⇨ Tester le bon fonctionnement de ce script équivalent à une sorte de structure ***if***, en réduisant la largeur de la fenêtre de votre navigateur.

⇨ Ajouter un autre *media querie* dans le fichier *avengers.css* afin de ne plus positionner les blocs <*article*> et <*aside*> l'un à côté de l'autre, lorsque la taille de l'écran est inférieure à 720px :

⇨ Tester le bon fonctionnement de ce script en réduisant la largeur de la fenêtre de votre navigateur.

```
/*********************************Gestion de section ***
section , article , aside  {···
}
section {···
}
article {
    width:75%;
    height:300px;
}

aside{
    width:23%;
    height:300px;
}
@media screen and (max-width: 720px) {
    article {
        width:100%;
    }
    aside{
        width:100%;
    }
}
```

On constate que parmi les défauts évoqués précédemment, il subsiste ceux liés au bloc <nav>. Là, on préfèrera résoudre le problème en disposant le bloc <nav> sous forme d'une liste verticale. Elle sera affichée uniquement après click sur une icône *menu*. C'est l'objectif du paragraphe 3 …. mais avant …

2.2. On rajoute une balise ***meta viewport*** dans l'*html*

⇨ On rajoute uniquement une petite ligne dans la partie des métas du fichier *avengers.html*, entre les balises <*head*> et </*head*> :

```
<head>
    <title>Avengers</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width= device-width, initial-scale=1.0">
    <link rel="stylesheet" href="avengers.css">
</head>
```

Pourquoi rajouter cette balise méta ? …. En voici l'explication :

Le *viewport* correspond, sur un smartphone, à la fenêtre dans laquelle s'affiche la page web. Les écrans de smartphones ont une largeur comprise entre 480px et 720px. Par contre la résolution des écrans est importante ce qui leur permet d'afficher tout de même correctement des largeurs de page de jusqu'à 2000px. Sur un site *non responsif*, le navigateur va afficher la page en dézoomant automatiquement afin qu'elle occupe toute la largeur de l'écran. Celle-ci apparait alors souvent trop petite pour être lue. Une taille de pixels fictive est créée. La balise *viewport* indique au navigateur que les largeurs en px définies dans les *media queries* doivent être interprétées à l'échelle 1 :1 . Sans cette balise, la prise en compte de ces tailles est souvent mauvaise.

3. On s'occupe de bandeau de navigation ?

3.1. Média queries pour modifier le Css du bloc <*nav*> afin de l'afficher en liste verticale pour les smartphones

⇨ on rajoute dans un premier temps un bloc *media querie* dans le fichier *avengers.css*, pour la partie qui gère le <nav>.

On modifie la propriété *display* du bloc <*ul*> et on remplace donc *display :flex* ; par *display :block*.
Ainsi l'affichage du bloc <*ul*> devient standard et les blocs <*li*> enfants de <*ul*> se mettent les uns sous les autres, comme dans une liste standard.
On rajoute un *padding :0* ; car la balise <*li*> crée par défaut un *padding* pour décaler les items de la liste.

```
/*******************************Gestion du nav **
nav{
    border:2px solid green;
}
nav ul {
    border:2px solid green;
    display:flex;
    flex-direction: row;
    justify-content: space-around;
    align-items:center;
    margin:0;
}
nav li{
    border:2px solid green;
    list-style: none;
    padding:10px;
    width:150px;
}
nav li a{…
}
@media screen and (max-width: 720px) {
    nav{

    }
    nav ul {
        display:block;
        padding:0;
    }
}
```

⇨ Tester le bon fonctionnement de ce script en réduisant la largeur de la fenêtre de votre navigateur.

3.2. On insère une icône dans l'html qui nous permettra de gérer l'affichage du menu <nav> en position « smartphone »

Les fichiers *ouverture.png* et *fermeture.png* contenus dans le dossier uploadé en début de Tp permettront d'afficher une icone que l'on va placer dans le coin bas-gauche du bloc <*header*>. La taille de ces images est de 40x40 px.

⇨ Insérer l'image *ouverture.png* en début du block <*nav*> :

```
<header>
    <img src="ouverture.png" alt="icone d'ouverture menu" title="Menu">
    <h1>Les Avangers à Koh-Lanta</h1>
    <div class="reseaux"><img src='facebook.png'><img src='twitter.png'><img src='snap.png'></div>
</header>
```

L'affichage de la page est alors le suivant :

⇨ Modifier quelques propriétés du Css initial du bloc <*header*> afin d'afficher l'icône en partie basse, avec une petite marge sur la gauche et en enlevant l'effet *:hover* qui est gênant ici :

```
/*********************************Gestion du header *********************************/
header  {
    border:2px solid black;
    display:flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: flex-end;
    background-image: url('fond.jpg');
    background-size: cover;
}
.reseaux{···
}
#icone {···
}
header h1{ border:2px solid black;margin:auto;font-size: 5em;text-align: center;color:whitesmoke;}
header img{margin:0 0 2px 2px;}
.reseaux img{ border:2px solid black;width:50px;margin:5px;}
.reseaux img:hover{ border:4px transparent black;width:55px;margin:5px;border-radius: 10px;;}
@media screen and (max-width: 720px) {
    header h1{font-size: 3em;}
    .reseaux{margin:0px;}
    .reseaux img{ width:30px;}
}
```

On remplace *center* par *flex-end<*

On rajoute une marge de 2px

On remplace *header par .reseaux* pour cibler les images des réseaux sociaux uniquement

⇨ Tester le bon fonctionnement de ce script en réduisant la largeur de la fenêtre de votre navigateur.

3.3. On commence par gérer l'affichage de l'icône et du block <*nav*> dans le Css

On veut que :

- En écran ordinateur :
  - le bloc <nav> s'affiche comme initialement, c'est-à-dire horizontalement,
  - l'icône ne s'affiche pas.
- En écran smartphone :
  - le bloc <nav> se dispose en liste verticale (déjà fait avant), **sans s'afficher**,
  - l'icône s'affiche.

Pour obtenir ce comportement, on utilise la propriété *display* toujours à laquelle on donnera soit la valeur *none*, soit la valeur *block* (*display: none* ou *display: block*).

⇨ On rajoute tout d'abord un *id* à l'icône :

```
<header>
    <img src="ouverture.png" alt="icone d'ouverture menu" title="Menu" id="icone">
    <h1>Les Avangers à Koh-Lanta</h1>
    <div class="reseaux"><img src='facebook.png'><img src='twitter.png'><img src='snap.png'></div>
</header>
```

⇨ On rajoute des *display: none* ou *display: block* dans le Css :

```
#icone {
    display:none;
}
@media screen and (max-width: 720px) {
    nav{
        display:none;
    }
    nav ul {
        display:block;
        padding:0;
    }
    #icone {
        display:block;
    }
}
```

⇨ Tester le bon fonctionnement de ce script en réduisant la largeur de la fenêtre de votre navigateur.

3.4. JavaScript pour afficher ou pas, le menu en cliquant sur l'icône, pour un écran smartphone :

On écrira le code *JavaScript* dans un fichier séparé. On appellera ce fichier *avengers.js* .

⇨ Créer un nouveau fichier et l'enregistrer sous le nom *avengers.js* ,

⇨ Ajouter dans le <*head*> du fichier *.html* la ligne qui indique, lors du chargement de la page, qu'il y a du script JS à lire.

```
<head>
    <title>Avengers</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width= device-width, initial-scale=1.0">
    <link rel="stylesheet" href="avengers.css">
    <script src="avengers.js" async></script>
</head>
```

⇨ On commence par écrire ces quelques lignes dans le fichier *.js* , pour :

- repérer les éléments html avec la méthode *querySelector(),*
- créer une fonction qui sera appelé au click sur l'icone,
- créer l'évènement lié au click sur l'icone

```
/* Gestion du bloc <nav> pour les écrans smartphone */
// Repèrage des éléments html
let icone = document.querySelector("#icone");
let nav = document.querySelector("nav");
// Fonction qui gère l'affichage du <nav>
function menu(){
    console.log("menu a été appelé")
}
// Evènements lié au click sur l'icone
icone.addEventListener("click",menu);

/*------------------- FIN gestion bloc <nav> -------------------*/
```

⇨ Afficher la console de votre navigateur et tester le bon fonctionnement de ce script.

La structure du script étant écrite et commentée, on se lance dans l'écriture du code de la fonction *menu*. Il devra permettre :

- Si le menu <nav> n'est pas déjà affiché :
  - de l'afficher, en modifiant sa propriété Css *display :none;* pour lui donner la valeur *display :block;*
  - de modifier le fichier image lié à l'icône afin de lui donner l'apparence 
- Si le menu <nav> est déjà affiché :
  - de ne plus l'afficher, en modifiant sa propriété Css *display :block;* pour lui donner la valeur *display :none;*
  - de modifier le fichier image lié à l'icône afin de lui redonner l'apparence 

⇨ Afin de vous aider à écrire ce script sans aide, tester les méthodes javascript qui seront utilisés, en les exécutant les unes après les autres, dans la console du navigateur. Observer les effets produits :

```
» nav.style.display="block"
» nav.style.display="none"
» icone.getAttribute("src")
» icone.setAttribute("src","fermeture.png")
» icone.setAttribute("src","ouverture.png")
```

⇨ Compléter à présent le script de la fonction menu :

```
// Fonction qui gère l'affichage du <nav>
function menu(){
    let fichier = icone.getAttribute("src");
    if (fichier == "ouverture.png") {
        [redacted].style.display="block";
        icone.setAttribute("src",[redacted]
    }else{
        [redacted]
    }
}
```

⇨ Tester le bon fonctionnement de ce script.

3.5. Encore une petite modification du Css pour terminer

Tout se passe correctement ….. sauf que, en affichant le bloc <*nav*>, celui-ci décale les blocs qui suivent vers le bas, <*nav*> occupant toute la largeur de la page.

Pour remédier à ce défaut tout de même important, on rajoute dans le fichier *avengers.css*, dans le media queries de <*nav*>, une ligne qui indique que lors du chargement de la page *html*, l'affichage de <*nav*> sera réalisé au bon endroit, mais les blocs qui suivront, seront affichés comme si <*nav*> n'était pas là, en se superposant à lui.

```
@media screen and (max-width: 720px) {
    nav{
        display:none;
        position:absolute;
        background-color: green;
    }
```

On dit que <*nav*> a été sorti du flux d'affichage de la page web.

⇨ Tester le bon fonctionnement du script complet en réduisant la largeur de la fenêtre de votre navigateur et en cliquant sur l'icône.

4. Et à présent, que fait-on ?

⇨ Utiliser ce qui a été vu dans ce travail pour l'appliquer à **votre site**, déjà en ligne sur *nsibranly.fr* .

⇨ Tester le bon fonctionnement en réduisant la largeur de l'écran et aussi, en visionnant votre site sur votre smartphone.