

OBJECTIFS : L'objectif de ce TP est de s'initier aux langages *Html* et *Css* en créant 3 pages web qui seront mises en ligne dans votre espace personnel du site *nsibranly.fr* .

Le lien vers votre espace perso se trouve sur l'onglet *premiere / web* . En fin de tp, en cliquant sur votre *avatar*, vous serez redirigé vers votre page qui affichera quelque chose comme : →

- [Blague à 2 balles](#)

En cliquant sur le lien affiché, vous serez dirigé vers une autre page qui affichera quelque chose comme : →

En cliquant sur le bouton « *Retour sur ma page* », vous serez redirigé vers la page précédente.



En cliquant sur le bouton « *Réponse* », vous serez redirigé sur une dernière page qui affichera quelque chose comme : →

En cliquant sur le bouton « *Retour* », vous serez redirigé vers la page précédente.

On vous fait bien sûr confiance pour personnaliser le contenu de ces pages. Les objectifs du tp seront ainsi :



- de créer 3 pages html avec des liens entre elles,
- d'y insérer des images et des boutons,
- de mettre en forme ces pages avec du *Css*.

Vous apprendrez ainsi les concepts de base de l'html et du css.

1. Constitution du fichier index.html de mon espace perso sur nsibranly.fr

⇒ Créer un répertoire nommé *web* par exemple, dans votre espace perso du lycée (U:\). Ce répertoire contiendra les fichiers qui seront ensuite téléchargés sur le serveur du site *nsibranly.fr* pour être mis en lignes.

⇒ Lancer l'application *Visual Studio Code* et créer un nouveau fichier qui contiendra du *html* et que vous enregistrerez sous le nom *index.html* .

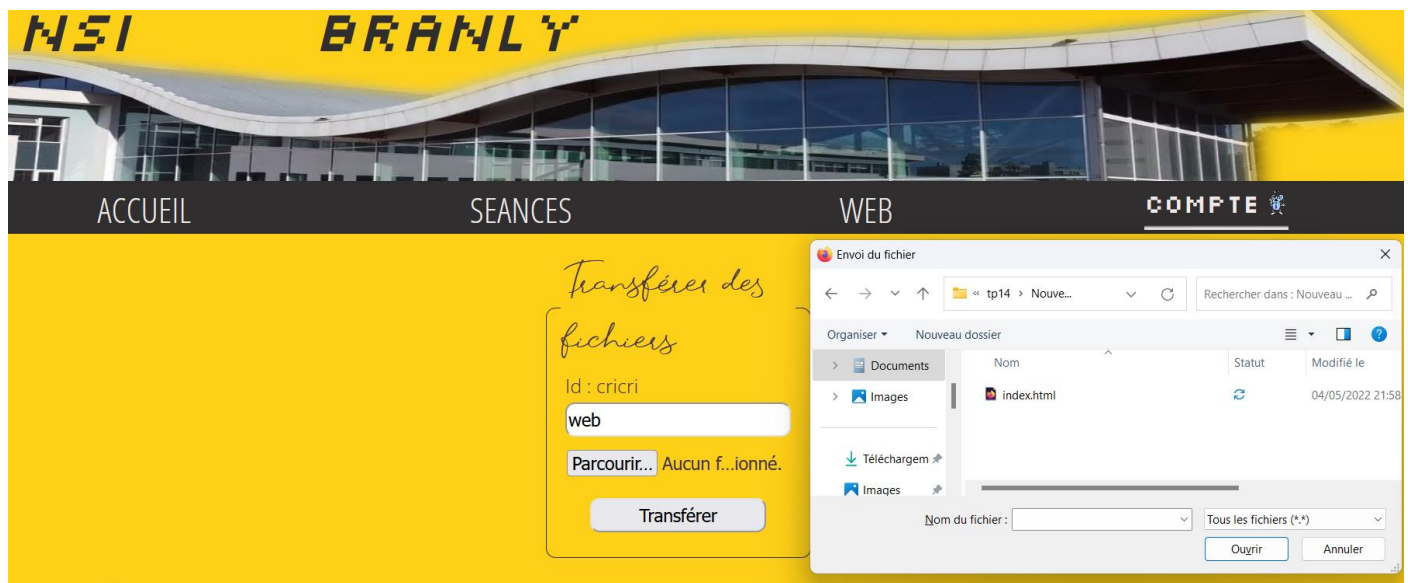
⇒ Ecrire pour l'instant, uniquement les 4 lignes suivantes et enregistrer le fichier :

```
1  <!DOCTYPE html>
2  <html lang="fr">
3  C'est bientôt l'été
4
5
6
7  </html>
```

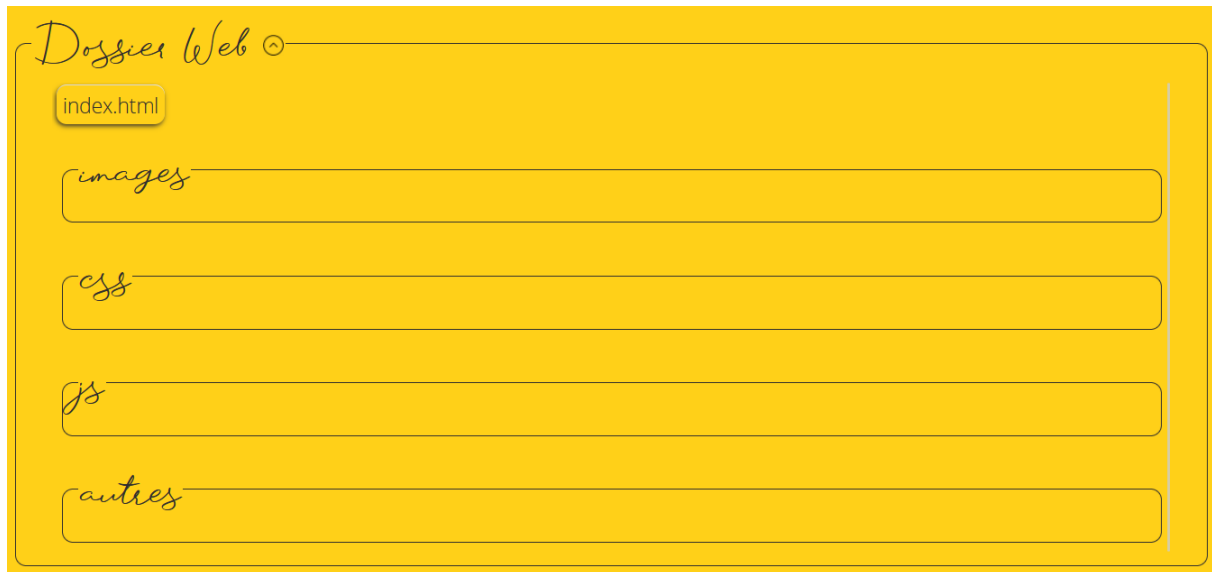
Balise qui indique que ce fichier contient de l'html

Bloc qui contient de l'html

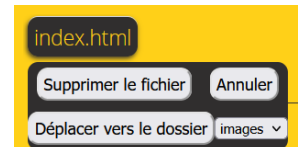
⇒ Télécharger ce fichier *index.html* dans votre zone perso sur le site *nsibranly.fr* . Pour cela, se loguer et transférer ensuite ce fichier *index.html* en utilisant le code **web**. Avec ce code, les fichiers sont directement transférés dans votre espace perso, sur le serveur qui héberge le site.



⇒ Vous pouvez gérer les fichiers déposés sur le serveur en ouvrant le volet *Dossier Web* disponible sur l'onglet *Compte* :



Le fichier *index.html* déposé se trouve dans la racine de votre espace personnel sur le serveur de *nsibranly.fr*. En cliquant sur le nom du fichier vous pouvez le supprimer ou le déplacer vers 4 sous-répertoires. Mais ici, ce fichier *index.html* doit rester sur la racine.



⇒ Visualiser votre page à partir du site *nsibranly.fr* dans un autre onglet de préférence (1 onglet pour la page *Mon compte* et un autre pour votre page perso), en cliquant sur votre avatar de la page WEB :

C'est bientôt l'été

Vous pouvez accéder directement à votre page en utilisant l'adresse URL :

nsibranly.fr/membres/guillaumeD par exemple pour la page perso de **guillaume** Dupin (prénom en minuscule + 1^{ère} lettre nom de famille en majuscule)

⇒ Visualiser également cette page sur votre smartphone.

On constate que les caractères qui ne sont pas ASCII ne sont pas affichés correctement. Autre problème, sur smartphone, les caractères sont affichés bien trop petits. On remédie à cela en ajoutant un bloc d'en-tête :

```

1  <!DOCTYPE html>
2  <html lang="fr">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Chez Nils</title>
7          <link rel="icon" type="img/png" href="ico.png"/>
8      </head>
9      <body>
10         C'est bientôt l'été
11     </body>
12 </html>

```

Ce bloc est délimité par les balises `<head> ... </head>`. Les éléments qu'il contient ne sont pas affichés et sont dans l'ordre :

- une balise orpheline `<meta>` avec l'attribut *charset* qui indique l'encodage du texte de la page. En UTF-8 les caractères sont définis avec un nombre d'octets compris entre 1 et 4. Si on n'indique pas la valeur de cet attribut, le navigateur n'accepte que des caractères ASCII qui sont eux encodés sur 1 octet.
- une balise orpheline `<meta>` avec l'attribut *viewport* qui permet d'adapter l'affichage aux écrans étroits de smartphone.
- une balise `<title>` qui permet d'afficher un texte court **dans l'onglet du navigateur**
- une balise orpheline `<link>` avec l'attribut *icon* qui permet d'afficher **dans l'onglet du navigateur** une image *.png* de petite taille (30 px environ). Vous pourrez télécharger l'image ci-contre dans votre répertoire (lien en cliquant sur l'image ci-contre pour l'ouvrir dans le navigateur, puis « *click droit et enregistrer sous* »)



⇒ Transférez les fichiers *index.html* et *ico.png* dans votre espace perso de *nsibranly.fr* (vous pouvez sélectionner les 2 fichiers pour les transférer en même temps) et observer le résultat sur écran d'ordinateur et sur votre smartphone.

⇒ Modifier la partie `<body> .... </body>` afin d'y insérer une liste qui ne contiendra pour l'instant qu'une seule ligne :

```
1  <!DOCTYPE html>
2  <html lang="fr">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Chez Nils</title>
7          <link rel="icon" type="img/png" href="ico.png"/>
8      </head>
9      <body>
10         <ul>
11             <li>Ma blague à 2 balles</li>
12         </ul>
13     </body>
14 </html>
```

⇒ Transférez le fichier *index.html* modifié (cela écrase le précédent) et vérifiez le rendu

⇒ Créer un nouveau fichier *html* que vous sauvegarderez sous le nom *blague.html* et que vous laisserez vide pour l'instant.

⇒ Insérer dans le fichier *index.html* un lien qui permettra une redirection vers la page *blague.html* en cliquant sur la ligne « *Ma blague à 2 balles* » :

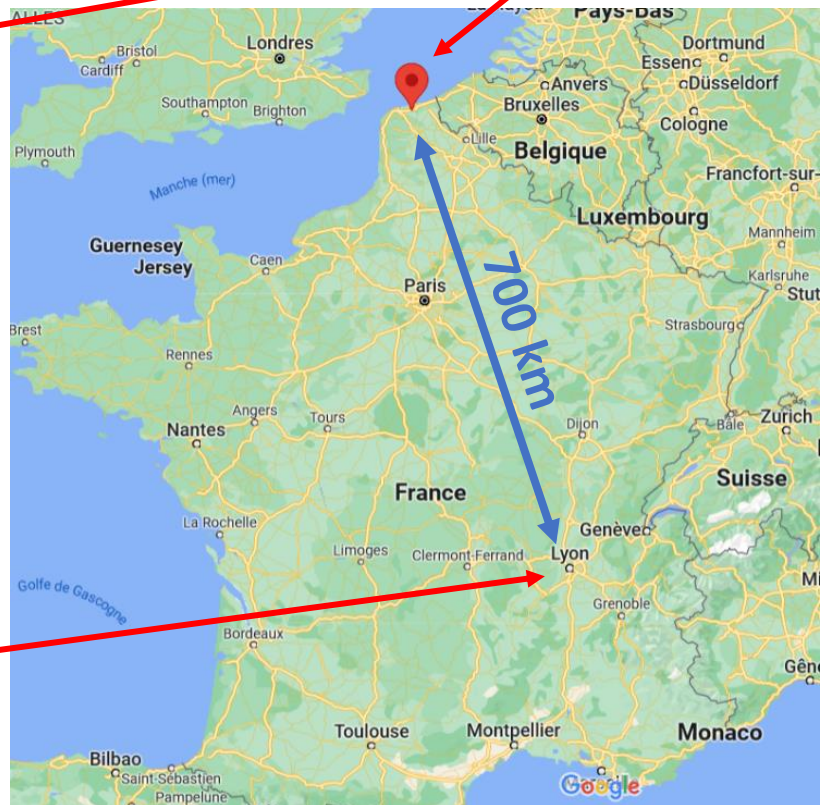
```
<body>
  <ul>
    <a href="blague.html">  <li>Ma blague à 2 balles</li>  </a>
  </ul>
</body>
```

Pour créer ce lien, le bloc `<li> ... </li>` a été placé à l'intérieur d'un bloc `<a> ... </a>`. L'attribut *href* de la balise ouvrante `<a>` a comme valeur le nom du fichier vers lequel la redirection est faite. Ce fichier *blague.html* se trouve ici dans le même répertoire que le fichier *index.html*.

⇒ Télécharger les fichiers *index.html* et *blague.html* et vérifier le bon fonctionnement du lien.

2. Comment la liaison serveur / client s'effectue-t-elle ?

Le site nsibranly.fr est hébergé par la société OVH sur un serveur qui se trouve à **Gravelines** dans le département du Nord.

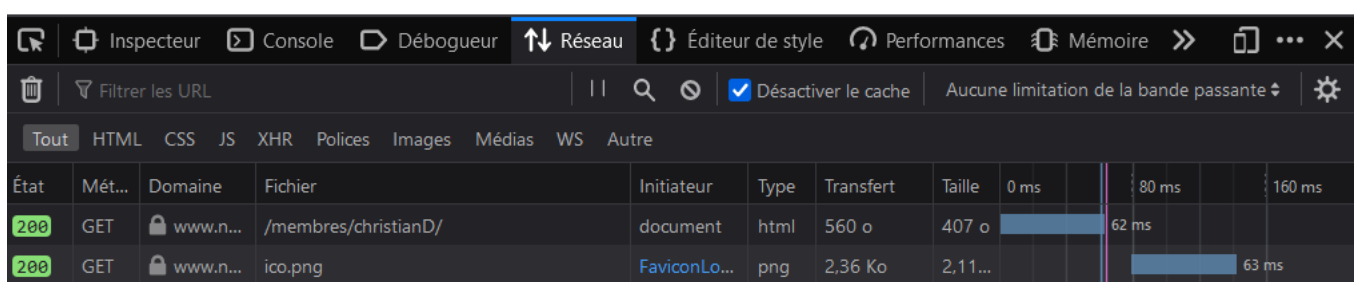


Le fichier *index.html* que vous avez créé se trouve actuellement dans un espace dédié, sur le serveur OVH de Gravelines. Lorsque vous voulez accéder à cette page depuis votre smartphone ou depuis l'ordinateur qui se trouve à Lyon, à 700 km du serveur, que se passe-t-il ?

Pour apporter une première réponse à cette question, on utilise les outils de développement mis à disposition par votre navigateur Firefox ou Chrome :

⇒ Ouvrir la fenêtre *Outils de développement* sur votre navigateur de l'ordinateur, en tapant sur la touche F12 du clavier ou avec un clic droit sur la fenêtre du navigateur et en choisissant « inspecter ».

⇒ Sur la fenêtre qui apparaît, choisir l'onglet *Réseau* (ou Network) et recharger votre page :



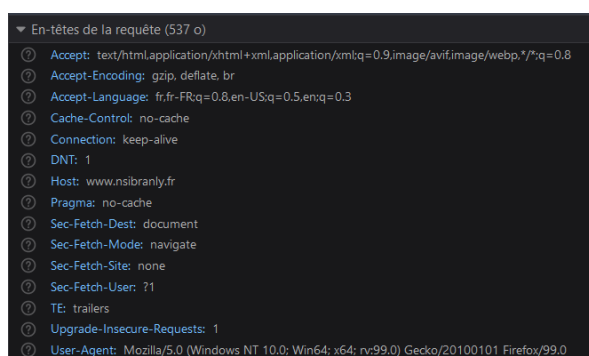
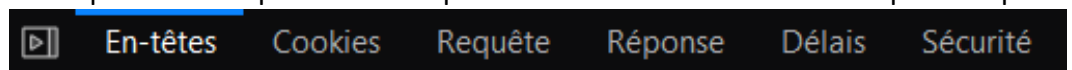
On voit sur la capture d'écran précédente, que pour charger la page, le navigateur a effectué 2 requêtes. La première pour obtenir le fichier *index.html* de taille 407 octets a nécessité 62 ms, la seconde pour obtenir le fichier image *ico.png* de taille 2110 octets a nécessité 63 ms.

Chronologiquement, les choses se déroulent ainsi :

- Vous êtes ce qu'on appelle un **Client**. Sur votre ordinateur à Lyon, vous saisissez dans la barre d'URL, l'adresse *nsibranly.fr/membres/guillaumeD* par exemple.
- Le navigateur de votre ordinateur crée une requête appelée **GET**. Elle est codée en binaire et est composée d'une demande du fichier *index.html* qui se trouve dans le répertoire *membres/nilsM* du site *nsibranly.fr*. L'adresse IP de votre ordinateur est bien sûr insérée dans cette requête pour pouvoir réceptionner la réponse ensuite.
- Le navigateur envoie cette requête par l'intermédiaire de la *carte réseau* de l'ordinateur, sur le serveur du lycée Branly, situé au 1^{er} étage du bâtiment A.
- Ce serveur la redirige vers l'adresse passerelle qui raccorde le lycée Branly au réseau Internet (adresse gateway). Cette requête est alors transmise de routeurs en routeurs et arrive à Gravelines, situé à 700 km de Lyon, sur le serveur OVH qui héberge le site *nsibranly.fr*.
- Ce serveur répond aussitôt, en renvoyant en retour le fichier *index.html* demandé.
- La réponse de cette première requête **GET** arrive sur votre ordinateur en parcourant à nouveau 700 km dans l'autre sens. Elle est décodée par votre navigateur. Dans la fenêtre « *Outils de développement* », l'**état 200** indique que cette requête a été réalisée avec succès.
- Le fichier *index.html* est alors interprété par le navigateur. Celui-ci constate que pour afficher la page correctement, il est nécessaire de réaliser une autre requête pour obtenir le fichier image *ico.png*. Cette seconde requête est réalisée de la même façon que la première et se termine quelques millisecondes plus tard avec succès (état 200 affiché).
- Le navigateur peut alors afficher à présent la page demandée.

Les « outils de développement » du navigateur nous permettent de voir plus en détail ce qu'il y a dans ces requêtes :

⇒ Cliquer sur l'état 200 de la première requête afin de pouvoir visualiser les différentes parties qui la composent :



Informations envoyées par le navigateur du Client au Serveur

```

▼ En-têtes de la réponse (301 o)
① accept-ranges: none
② cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
③ content-encoding: gzip
④ content-length: 258
⑤ content-type: text/html
⑥ date: Fri, 29 Apr 2022 13:34:47 GMT
⑦ pragma: no-cache
⑧ server: Apache
⑨ vary: Accept-Encoding
X-Firefox-Spdy: h2

```

Informations envoyées par le
Serveur au navigateur du Client

```

HTML Brut

```

• [Ma blague à 2 balles](#)

```

HTML Brut
1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Chez Nils</title>
7     <link rel="icon" type="img/png" href="ico.png"/>
8   </head>
9   <body>
10    <ul>
11      <a href="blague.html"> <li>Ma blague à 2 balles</li> </a>
12    </ul>
13  </body>
14 </html>
15
16

```

Réponse renvoyée interprétée

Réponse renvoyée brute (code
html)

⇒ Cliquer sur l'état 200 de la seconde requête et observer les 2 onglets En-têtes et l'onglet Réponse .

3. Constitution du fichier `blague.html` : insertion du titre et du paragraphe

On continue notre TP en complétant à présent le fichier `blague.html` qui devra à la fin de ce paragraphe, afficher un contenu qui ressemble à :



La mise au point des fichiers nécessite de les tester régulièrement sur votre navigateur.

Il est plus facile de les tester **localement** en demandant au navigateur de lire directement les fichiers qui sont sur votre ordinateur. La mise en ligne se fera en fin de Tp, lorsque tous les fichiers seront au point.

⇒ Avec un copié-collé, écrire dans le fichier `blague.html` la structure de base suivante :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Chez Nils</title>
7     <link rel="icon" type="img/png" href="ico.png"/>
8   </head>
9   <body>
10
11
12   </body>
13 </html>

```

⇒ Insérer dans la partie <head> </head> la balise orpheline <link> qui permet de lier le fichier *blague.html* à un fichier que l'on nommera *blague.css* et qui contiendra le code Css.

Ce code permettra de mettre en forme le contenu de la page.

```
1  <!DOCTYPE html>
2  <html lang="fr">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Chez Nils</title>
7          <link rel="icon" type="img/png" href="ico.png"/>
8          <link rel="stylesheet" href="blague.css">
9      </head>
10     <body>
11
12
13     </body>
14 </html>
```

⇒ Créer ce nouveau fichier qui contiendra du Css et qui sera nommé *blague.css*

⇒ Dans ce fichier, écrire pour l'instant :

```
1  * {
2      border: 1px solid black;
3  }
4
5  body {
6      background-color: #415fc8;
7  }
```

Veut dire que l'on insère une bordure noire de 1px de large sur chacun des éléments de la page. Cela permet de visualiser les contours de chaque élément durant la phase de mise au point

Veut dire que la couleur de fond de tout le bloc <body> doit être bleue (ici le code couleur rgb est en hexadécimal : 41 pour rouge, 5f pour green et c8 pour blue.

⇒ Faire un click droit sur l'onglet du fichier *blague.html* dans **Visual Studio Code** et exécuter : « Copier le chemin ». Coller ce chemin dans la barre URL du navigateur afin de lui demander de lire ce fichier. Ne plus afficher ici, les outils du développeur.

La fenêtre affichée est bleue et n'affiche pour l'instant aucun contenu.

⇒ Compléter le fichier *blague.html* en y insérant les blocs <h1></h1> et <p></p> suivants :

```
<body>
    <h1>Ma blague</h1>
    <p>Que fait un crocodile lorsqu'il rencontre une superbe femelle ?</p>
</body>
```

Voir sur le navigateur le rendu de ces 2 lignes. Les balises <h1> et <p> permettent de créer des blocs qui occupent la largeur de la page. Ce sont des éléments de type **blocs**.

⇒ Appuyer sur la touche F12 du clavier pour remettre les "outils du développeur ».

⇒ Cliquer sur l'icône  et modifier la largeur de l'écran pour avoir un rendu en version smartphone.

⇒ Afin d'obtenir le rendu attendu pour le titre « *Ma blague* », on complète par étapes le fichier *blague.css* , en testant à chaque fois le rendu dans le navigateur :

- On demande un fond blanc dans le bloc titre <h1> :

⇒ tester le rendu

```
h1 {
  background-color: white;
}
```

- On demande une autre couleur du texte inséré dans l'élément <h1> :

⇒ tester le rendu

```
h1 {
  background-color: white;
  color: #415fc8;
}
```

- On réduit la largeur de ce bloc à 50 % de la largeur de l'écran (tester le rendu en modifiant la largeur de l'écran)

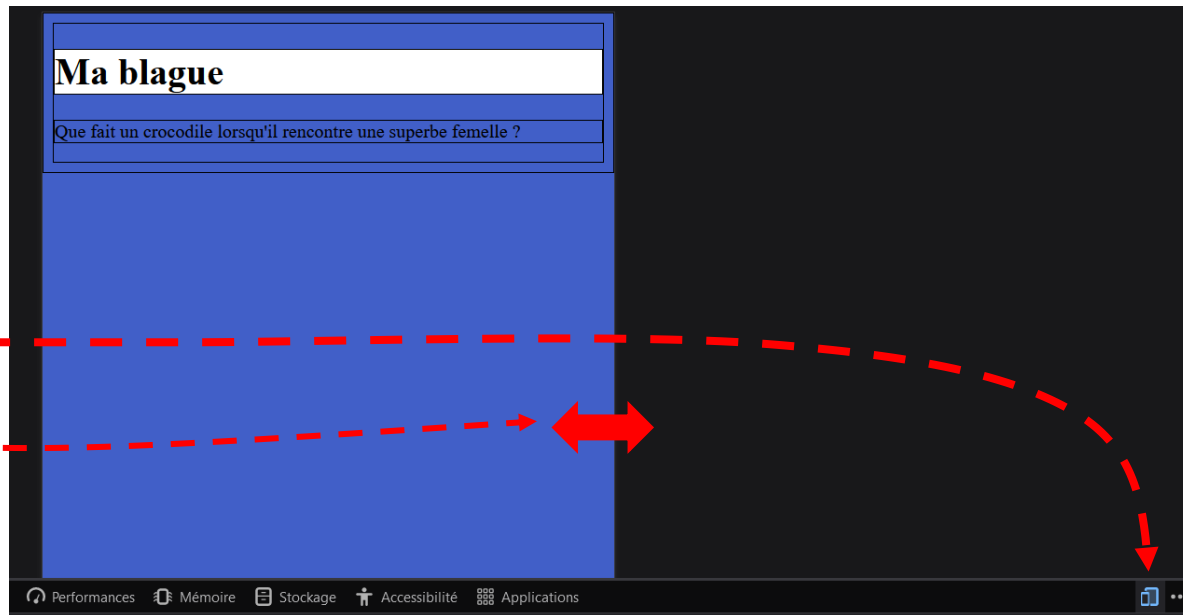
⇒ tester le rendu

```
h1 {
  background-color: white;
  color: #415fc8;
  width: 50%;
}
```

- On centre le texte contenu dans le bloc <h1> :

⇒ tester le rendu

```
h1 {
  background-color: white;
  color: #415fc8;
  width: 50%;
  text-align: center;
}
```



- On centre le bloc <h1> lui-même dans son bloc parent qui est le bloc <body> :

⇒ tester le rendu

```
h1 {  
    background-color: white;  
    color: #415fc8;  
    width: 50%;  
    text-align: center;  
    margin: auto;  
}
```

```
h1 {  
    background-color: white;  
    color: #415fc8;  
    width: 50%;  
    text-align: center;  
    margin: 30px auto;  
}
```

- On rajoute une marge extérieure de 30px en haut et en bas :

⇒ tester le rendu

- On rajoute une marge intérieure de 10 px, en haut, sur le côté gauche, sur le côté droit et en bas. Cette marge intérieure est appelée **padding** :

⇒ tester le rendu

```
h1 {  
    background-color: white;  
    color: #415fc8;  
    width: 50%;  
    text-align: center;  
    margin: 30px auto;  
    padding: 10px;  
}
```

```
h1 {  
    background-color: white;  
    color: #415fc8;  
    width: 50%;  
    text-align: center;  
    margin: 30px auto;  
    padding: 10px;  
    border-radius: 15px;  
}
```

- On crée des coins arrondis sur les contours du bloc <h1> :

⇒ tester le rendu

Vérifier que cela fonctionne pour des largeurs d'écrans différentes.

⇒ Appliquer la même mise en forme sur le bloc <p> en rajoutant simplement

⇒ tester le rendu

```
h1 , p {
    background-color: white;
    color: #415fc8;
    width: 50%;
    text-align: center;
    margin: 30px auto;
    padding: 10px;
    border-radius: 15px;
}
```

⇒ On continue la mise en forme de ces blocs <h1> et <p> en demandant au navigateur d'afficher le mot crocodile en vert et en gras. Pour cela :

- Modifier le fichier blague.html en insérant ce mot entre 2 balises afin de l'identifier :

```
<body>
  <h1>Ma blague</h1>
  <p>Que fait un <span>crocodile</span> lorsqu'il rencontre une superbe femelle ?</p>
</body>
```

- On rajoute dans le css

Une bordure s'affiche autour de ce bloc, c'est normal car demandé sur la première instruction

```
p span {
    color: green;
    font-weight: bold;
}
```

⇒ On termine cette mise en forme de ces 2 blocs, en rajoutant dans le fichier Css une propriété qui ne s'appliquera qu'en cas de survol avec la souris de ces éléments (méthode :hover) :

- En survolant le bloc <h1>, la bordure devra changer légèrement :

```
h1:hover{
    border: 3px solid transparent;
    box-shadow: -1px 2px 5px #752116,
               0 -1px 0 white;
}
```

- En survolant le mot crocodile la couleur du mot deviendra rouge et sa taille plus importante (font-size : larger) : à faire vous-même.

```
p span:hover{
}
}
```

⇒ Vérifier que cela fonctionne pour des largeurs d'écrans différentes.

4. Constitution du fichier *blague.html* : insertion des 2 images

On continue notre histoire en complétant le fichier *blague.html* qui devra à la fin de ce paragraphe, afficher un contenu qui ressemble à :



⇒ Pour ne pas perdre de temps, vous pourrez télécharger les images *croco.png* et *croco2.png* ci-contre (lien en cliquant sur chacune des images ci-contre) et les placer dans votre dossier.



⇒ Insérer les images dans votre fichier *blague.html* :

Les balises `<img>` sont des balises **orphelines**. L'attribut *alt* permet de donner un nom à l'image (pour les robots), si le

fichier image n'est pas trouvé, c'est ce nom qui s'affichera. L'attribut *title* permet d'afficher un message lors du survol de l'image par la souris. Tester le rendu dans le navigateur.

```
<body>
```

```
<h1>Ma blague</h1>
```

```
<p>Que fait un <span>crocodile</span> lorsqu'il rencontre une superbe femelle ?</p>
```

```

```

```

```

```
</body>
```

⇒ Les images sont un peu grandes. On peut avec le *Css* réduire la taille de leur affichage, en imposant par exemple 250 px de hauteur :

```
img {  
    height:250px;  
}
```

Les images sont des entités de type **inline**. Contrairement aux entités `<h1>` et `<p>` qui sont de type **bloc**, un élément de type **inline** n'occupe pas toute la largeur de la page. Il n'y a pas de retour à la ligne automatique, les éléments **inline** se placent les uns après les autres sur la même ligne.

Pour centrer une entité de type **bloc**, on utilise la propriété *margin:auto*. Pour le centrage d'une entité de type **inline**, une solution consiste à placer cet élément **inline** dans un élément de type **bloc** `<div>`. Ce bloc `<div>` occupera toute la largeur de la page et l'élément **inline** qu'il contient sera centré en utilisant la propriété *text-align:center*. Cela donne dans les fichiers *blague.html* et *blague.css* :

⇒ Pour le fichier *html* :

```
<body>
  <h1>Ma blague</h1>
  <p>Que fait un <span>crocodile</span> lorsqu'il rencontre une superbe femelle ?</p>
  <div>
    
    
  </div>
</body>
```

⇒ Pour le fichier *css* :

```
div {
  text-align: center;
}
```

⇒ Réduire la taille de l'écran et vérifier le rendu pour des écrans de smartphones.....On peut alors constater que l'image du crocodile femelle est rognée pour une largeur d'écran trop petite.

Pour éviter ceci, on spécifie dans le *Css*, que la largeur maximale de chacune des images est égale à 100% de la largeur de l'élément parent qui est ici le *body*, c'est-à-dire la largeur de l'écran.

Vérifier le rendu sur le navigateur en modifiant la largeur de l'écran.

```
img {
  height: 250px;
  max-width: 100%;
}
```

⇒ On améliore ce rendu, en rajoutant plus d'espace entre les 2 images. Pour cela, une solution consiste à rajouter une marge droite sur le crocodile male. Pour discerner l'image du crocodile male parmi les 2 images, on ajoute un attribut *class* à celui-ci dans sa balise **

```
<div>
  
  
</div>
```

Cet attribut *class* permet de repérer cette balise dans le *Css*.

Pour repérer une *class*, on place un point devant le nom de la *class*. Cela donne : *.male* {}

Vérifier le rendu sur le navigateur, avec des largeurs d'écran différentes.

```
.male{
  margin-right: 50px;
}
```

⇒ Pour améliorer encore ce rendu, on vous demande de rajouter dans le *Css*, une propriété *:hover* qui permette de mettre une couleur d'arrière-plan rouge (*background-color : red ;*) et des coins arrondis (*border-radius : 50px ;*) lorsqu'une image est survolée.

5. Constitution du fichier `blague.html` : insertion des 2 boutons

On termine à présent le codage du fichier `blague.html` qui devra à la fin de ce paragraphe, afficher un contenu qui ressemble à :



Pour insérer un bouton, il existe 2 façons de faire. On utilise la première méthode pour le bouton *Réponse* et la seconde pour le bouton *Retour sur ma page*.

⇒ Insérer le premier bouton en utilisant une balise orpheline `<input type= 'button' value='Réponse'>` . Pour le second, on utilise la balise `<button></button>`. Cela donne :

```
<body>
  <h1>Ma blague</h1>
  <p>Que fait un <span>crocodile</span> lorsqu'il rencontre une superbe femelle ?</p>
  <div>
    
    
  </div>
  <input type="button" value="Réponse">
  <button>Retour sur ma page</button>
</body>
```

On constate que les boutons sont des entités de type *inline*, comme les images. Ils se placent les uns après les autres sur la même ligne, sans retour à la ligne. Pour les centrer, on utilise la même technique que pour les boutons. On les place dans un élément de type *block* `<div>` . Ce bloc `<div>` occupera toute la largeur de la page. Les boutons *inline* qu'il contient seront centrés en utilisant la propriété `text-align:center`. Cela donne dans les fichiers `blague.html` et `blague.css` :

⇒ Pour le fichier *html* :

```
<body>
  <h1>Ma blague</h1>
  <p>Que fait un <span>crocodile</span> lorsqu'il rencontre une superbe femelle ?</p>
  <div>
    
    
  </div>
  <div>
    <input type="button" value="Réponse">
    <button>Retour sur ma page</button>
  </div>
</body>
```



⇒ Pour le fichier *css* : AUCUN CHANGEMENT, car la propriété *text-align : center* **est déjà** défini sur tous les blocs `<div>` de cette page.

⇒ Vérifier le rendu pour des largeurs d'écran différentes.

⇒ On améliore le rendu des boutons en augmentant leur taille. Pour cela on peut jouer sur les propriétés *width* et *height* :

```
input , button {
  width:200px;
  height:50px;
}
```

Une autre solution est **de remplacer** les 2 propriétés précédentes par une propriété de marge intérieure de 10px sur le haut, la droite, le bas et la gauche de chacun des boutons :

```
input , button {
  padding:10px;
}
```

Si on veut des marges intérieures de 10px en haut, 11px à droite, 12px en bas et 13px à gauche, on écrit :

```
input , button {
  padding:10px 11px 12px 13px;
}
```

⇒ On améliore encore le rendu en créant une marge extérieure de 20px au-dessus des boutons et de 30px à gauche et à droite. On en profite pour demander des coins arrondis :

```
input , button {
  padding:10px 11px 12px 13px;
  margin:20px 30px 0px 30px;
  border-radius: 10px;
}
```

⇒ On termine ce traitement des boutons, en rajoutant un lien sur le bouton *Réponse* qui permettra de renvoyer l'utilisateur vers une nouvelle page définie dans un nouveau fichier que l'on nommera *reponse.html* . Pour cela, on insère le bouton entre 2 balises `<a>` et `</a>` :

```
<div>
  <a href="reponse.html"><input type="button" value="Réponse"></a>
  <button>Retour sur ma page</button>
</div>
```

Tester le fonctionnement du lien. N'oubliez pas de créer dans votre répertoire, ce nouveau fichier *reponse.html* qui contiendra ultérieurement du code html.

⇒ Modifier encore une dernière fois ce fichier *blague.html* , en ajoutant un lien sur le bouton *Retour sur ma page*, qui permet de renvoyer l'utilisateur sur votre page *index.html* . Tester le bon fonctionnement.

6. Première mise en ligne de votre travail

⇒ Sur le fichier *blague.css* , mettre en commentaire la propriété qui permet d'afficher une bordure autour de tous les éléments html de la page : */* ceci est en commentaire */*

```
/* ←
* {
  border:1px solid black;
}
*/ ←
```

⇒ Transférer en une étape, tous les fichiers de votre répertoire de travail (*blague.html* , *blague.css* , *reponse.html* , *croco.png* , *croco2.png*) sur le serveur de *nsibranly.fr* .

⇒ Tester le bon fonctionnement de l'ensemble sur ordinateur et sur votre smartphone.

⇒ Sur ordinateur, taper sur la touche F12 afin d'afficher les *outils du développeur*. Dans l'onglet *Réseau*, faire afficher les requêtes GET initiées par votre navigateur en direction du serveur de *nsibranly.fr* . Combien de temps prennent ces requêtes ? Pour chacune d'elles, qu'elles sont les en-têtes échangés, les données transférées ?

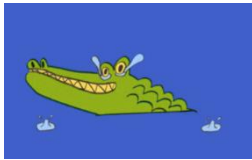
État	Méthode	Domaine	Fichier	Initiateur	Type	Transfert	Taille	0 ms	319 ms	639 ms
200	GET	www.nsibranly.fr	blague.html	document	html	728 o	862 o	85 ms		
200	GET	www.nsibranly.fr	blague.css	stylesheet	css	679 o	812 o	172 ms		
200	GET	www.nsibranly.fr	croco.png	img	png	101.89 Ko	101.64 Ko	218 ms		
200	GET	www.nsibranly.fr	croco2.png	img	png	141.31 Ko	141.06 Ko	360 ms		
200	GET	www.nsibranly.fr	ico.png	FaviconLoader.jsm:191 (img)	png	2.36 Ko	2.11 Ko	47 ms		

7. Constitution du fichier reponse.html : à vous de jouer ...

⇒ On vous demande de créer le code du fichier *reponse.html* .

En cliquant sur le bouton « *Retour* », vous devrez être redirigé vers la page précédente.

⇒ Vous pourrez télécharger l'image *rep.gif* ci-contre (lien en cliquant sur l'image ci-dessous) et la placer dans votre dossier.



⇒ Transférer en une étape, tous les fichiers de votre répertoire de travail sur le serveur de *nsibranly.fr* .

⇒ Tester le bon fonctionnement de l'ensemble sur ordinateur et sur votre smartphone.