Compte-Rendu

Index.html :

```html
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Etudes | Spécialité NSI</title>
        <link rel="stylesheet" href="style.css">
        <link rel="shortcut icon" type="image/png" href="Pictures/logo_lycee.ico">
    </head>
    <body>
        <header>
            <nav>
                <a href="index.html"><img src="Pictures/logo_lycee.png" alt="logo_lycee"></a>
                <a href="programme_nsi.html" class="a_texte">Le programme de NSI</a>
                <a href="projets_nsi.html" class="a_texte">Les projets en NSI</a>
                <a href="etudes.html" class="a_texte">Les études post-bac</a>
                <a href="#contact" class="bouton">Contactez-nous !</a>
            </nav>
        </header>
```

Ceci est le début du code, il permet de créer le site et de créer un en-tête reliant aux autres pages.

Ceci est la partie du code permettant de faire les vagues présente dans le document . celles-ci étant noires.

```html
<div class="vagues_section_1">
    <svg class="waves" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 24 150 28"
        <defs>
            <path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z" />
        </defs>
        <g class="parallax">
            <use xlink:href="#gentle-wave" x="48" y="0" fill="rgba(0,0,0,0.7" />
            <use xlink:href="#gentle-wave" x="48" y="3" fill="rgba(0,0,0,0.5)" />
            <use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(0,0,0,0.3)" />
            <use xlink:href="#gentle-wave" x="48" y="7" fill="black" />
        </g>
    </svg>
</div>
```

```html
<div class="card">
    <div class="card_3">
        <div class="face face1">
            <div class="content">
                <img src="Pictures/chapeau_diplome.png">
                <h3>Etudes</h3>
            </div>
        </div>
        <div class="face face2">
            <div class="content">
                <p>Découvrez les différents types d'études que vous pouvez faire après avoir suivi la spécialité NSI</p>
                <a href="etudes.html">En savoir plus</a>
            </div>
        </div>
    </div>
</div>
```

Ceci est la partie du code permettant de faire bouger les images devant les liens en bas de la page d'accueil .

Ceci est la partie du code permettant de récupérer les informations pour recontacter les personnes allant sur le site.

```html
<section id="contact">
    <form>
        <h1>Contactez-nous !</h1>
        <label>
            <p>Votre nom : <input type="text" placeholder="Votre nom"></p>
        </label>
        <label>
            <p>Votre adresse email : <input type="email" placeholder="Votre adresse email"></p>
        </label>
        <label>
            <p>Votre message : <input type="text" placeholder="Votre message"></p>
        </label>
        <label>
            <p><input type="button" value="Envoyer" onclick="popup()"></p>
        </label>
    </form>
</section>
```

Style.css :

```css
#section_1 h1{
        font-family: Helvetica, sans-serif;
        animation: move 10s linear infinite;
        background-image: linear-gradient(to right, rgb(255, 100, 100), rgb(255, 175, 100), rgb(255, 255, 100),
        background-size: 200% auto;
        background-clip: text;
        font-size: 200px;
        padding-top: 200px;
        -webkit-text-fill-color: transparent;
        -webkit-background-clip: text;
        text-align: center;
        padding-bottom: 300px;
    }
```

Ceci est la partie du code css permettant de faire varier les couleur du titre dans la page d'accueil.

```css
.card .face.face1{
    position: relative;
    background: #333;
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 1;
    transform: translateY(100px);
}
```

Ceci est la partie du code permettant, à l'aide du troisième code HTML ci-dessus, de faire bouger les images en bas de la page d'accueil quand la souris est dessus.

```css
:root {
    --color-accent-light: #00DBDE;
    --color-accent-dark: #FC00FF;
    --color-secondary: #ccc;
    --text-color-default: #2c2c2c;
    --text-color-light: #fff;
    --font-size-default: 15px;
    --font-size-lg: 25px;
    --font-weight-regular: 400;
    --font-weight-semibold: 600;
    --font-weight-default: var(--font-weight-regular);
    --border-color-default: var(--color-secondary);
    --border-color-accent: var(--color-accent-dark);
    --bg-color-default: #fff;
    --bg-color-dark: #2c2c2c;
    --bg-color-accent--lignt: var(--color-accent-light);
    --bg-color-accent--dark: var(--color-accent-dark);
    --transition-duration: .2s;
    --transition-timing-function: linear;
    --transition-delay: 0s;
    --negative-multiplier: -1;
    --gradient-default: linear-gradient(122deg, var(--color-accent-light) 0%, var(--color-accent-dark) 100%);
}
```

Ceci est un code copié sur un site (dans nos sources) mettant les bases à la création de la timeline présent dans la page du programme de NSI.

```css
@media(min-width: 700px) { /*Mettre la barre en couleur*/
    .timeline__step-active-marker {
        display: block;
    }
}

.timeline__step.is-active .timeline__step-title:before { /*Mettre les ronds en couleur*/
    transition: background-color var(--transition-duration) var(--transition-timing-function) var(--transition-delay);
    background-color: var(--color-accent-dark);
    border-color: var(--border-color-accent);
}
```

```css
.timeline__stepper { /*Barre gradient*/
    --step-border-width: 3px;
    --offset-to-slider-content: 30px;
    position: relative;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 30px;
    margin-bottom: var(--offset-to-slider-content);
}
```

Ces codes, comme celui précédent, ont été copiés avec lui sur l'une de nos sources. Ces codes permettent de créer la barre de la timeline, les points, de mettre la ligne et les points en couleur et afficher les textes lorsque les points sont sélectionnés.

```css
.timeline__step-title:before, /*Rond sur la barre*/
.timeline__step-title:after {
    position: absolute;
    top: var(--dot-top-position);
    left: 50%;
    display: block;
    width: var(--timeline-dot-dimensions);
    height: var(--timeline-dot-dimensions);
    content: '';
    border-radius: 50%;
}
```

```css
.timeline__slide.is-active { /* Afficher le texte quand on le sélectionne*/
    transition: opacity var(--transition-duration) var(--transition-timing-function) var(--transition-delay);
    opacity: 1;
    visibility: visible;
}
```

Script.js :

```
const STEP_ACTIVE_MARKEP_CUSTOM_PROPS = {
    width: "--slide-width",
    posX: "--slide-pos-x",
    posY: "--slide-pos-y",
};
```

Cette partie du code permet de définir des valeurs dynamiques pour la largeur et la position. Cela permet de créer des animation fluides.

Cette fonction permet de faire effectuer une action lorsque la timeline est cliquée (dans ce code il envoie au code ci-dessous).

```
timeline === null || timeline === void 0 ? void 0 : timeline.addEventListener("click", (event) => {
    const { target } = event;
    if (!target ||
        !(target instanceof Element) ||
        !target.closest(`.${DOM.timelineStep}`)) {
        return;
    }
    const currentStep = target.closest(`.${DOM.timelineStep}`);
    if (!currentStep) {
        return;
    }
```

```
function deactivateSteps() {
    const steps = document.querySelectorAll(`.${DOM.timelineStep}`);
    steps === null || steps === void 0 ? void 0 : steps.forEach((elem) => elem.classList.remove(`${DOM.timelineStepActive}`));
}
function activateCurrentStep(currentStep) {
    currentStep === null || currentStep === void 0 ? void 0 : currentStep.classList.add(`${DOM.timelineStepActive}`);
}
function deactivateSlides() {
    timelineSlides === null || timelineSlides === void 0 ? void 0 : timelineSlides.forEach((elem) => elem.classList.remove(`${DOM.timelineSlideActive}`));
}
function activateCurrentSlide() {
    const currentSlide = getCurrentSlide();
    if (!currentSlide) {
        return;
    }
```

Ces fonctions permettent de faire passer la timeline active sur la seul timeline sélectionner grâce à la fonction ci-dessus.

Cette fonction détermine quelle est la timeline active et lui met un marqueur actif pour le mettre en valeur.

```
function createStepActiveMarker() {
    const stepActiveMarker = document.createElement("div");
    stepActiveMarker.classList.add(`${DOM.timelineStepActiveMarker}`);
    timelineStepper === null || timelineStepper === void 0 ? void 0 : timelineStepper.appendChild(stepActive
    const positionProps = getStepActiveMarkerProps();
    if (!positionProps) {
        return;
    }
    setStepActiveMarkerProps(Object.assign({ stepActiveMarker }, positionProps));
}
```

```
function setStepActiveMarkerProps({ stepActiveMarker, posX, posY, width, }) {
    stepActiveMarker.style.setProperty(`${STEP_ACTIVE_MARKEP_CUSTOM_PROPS.width}`, `${width}px`);
    stepActiveMarker.style.setProperty(`${STEP_ACTIVE_MARKEP_CUSTOM_PROPS.posX}`, `${posX}px`);
    if (typeof posY === "number") {
        stepActiveMarker.style.setProperty(`${STEP_ACTIVE_MARKEP_CUSTOM_PROPS.posY}`, `${posY}px`);
    }
}
```

Cette fonction applique des propriétés de style (telles que la largeur et la position) au marqueur actif

Cette fonction récupère les propriétés nécessaires pour positionner correctement le marqueur actif dans l'interface utilisateur.

```
function getStepActiveMarkerProps() {
    const { currentStep } = getCurrentStep();
    if (!currentStep) {
        return null;
    }
    const width = getElementWidth(currentStep);
    const posX = getStepActiveMarkerPosX(currentStep);
    const posY = getStepActiveMarkerPosY();
    if (typeof posX !== "number" || typeof posY !== "number") {
        return null;
    }
    return { posX, posY, width };
}
```

Maxime : HTML / CSS

Maël : HTML / CSS

Antoine : JavaScript / CSS

Sacha : Compte Rendu / CSS