Potrebuju upravit Navbar.tsx a styly v Navbar.module.css. Ten bych chtel aby byl vic videt, mozna vetsi nebo tak neco. A ty linky by mohly mit nejaky zajimavejsi hover efekt

V Navbar.module.css mam ted tohle pro brand a linky:

.navbarBrand {

font-size: 1.5em;

font-weight: bold;

}

.brandLink {

text-decoration: none;

color: var(--color-text-primary);

}

/\* ... \*/

.navList {

list-style: none;

padding: 0;

margin: 0;

display: flex;

gap: 30px;

}

.navItem {

margin: 0;

}

.navLink {

text-decoration: none;

color: var(--color-text-primary);

font-weight: 600;

padding: 10px 15px;

border-radius: 8px;

transition: background-color 0.2s ease;

}

.navLink:hover {

background-color: var(--color-secondary-background);

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Co s tim muzu udelat?

Jasně, pojďme na to.

**Pro .navbarBrand:**

Můžeš zkusit zvětšit font-size a možná přidat nějaký letter-spacing pro modernější vzhled. Taky můžeš zvážit použití výraznější barvy, pokud máš nějakou primární barvu definovanou v globals.css.

Například:

.navbarBrand {

font-size: 1.8em; /\* Větší písmo \*/

font-weight: 700; /\* Můžeš použít číselnou hodnotu pro tloušťku \*/

letter-spacing: 1px; /\* Lehké prostrkání písmen \*/

/\* Pokud chceš použít primární barvu pro text: \*/

/\* color: var(--color-primary); \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Pro hover efekt na .navLink:**

Místo background-color můžeš zkusit změnit barvu textu nebo přidat spodní ohraničení.

Varianta 1: Změna barvy textu na primární barvu a lehká změna pozadí:

.navLink:hover {

color: var(--color-primary); /\* Použije primární modrou z globals.css \*/

background-color: rgba(0, 123, 255, 0.05); /\* Velmi jemné pozadí, přizpůsob si RGBA primární barvě \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Varianta 2: Spodní ohraničení při hoveru:

.navLink {

text-decoration: none;

color: var(--color-text-primary);

font-weight: 600;

padding: 10px 15px;

border-radius: 8px; /\* Můžeš zmenšit, pokud budeš používat border-bottom \*/

position: relative; /\* Potřebné pro absolutní pozicování pseudoelementu \*/

transition: color 0.2s ease; /\* Přechod pro barvu textu \*/

}

.navLink::after {

content: '';

position: absolute;

width: 0;

height: 2px;

bottom: 5px; /\* Umístění linky \*/

left: 50%;

transform: translateX(-50%);

background-color: var(--color-primary);

transition: width 0.3s ease-out; /\* Animace šířky \*/

}

.navLink:hover {

color: var(--color-primary); /\* Změna barvy textu při hoveru \*/

/\* background-color: transparent; /\* Odeber původní hover na pozadí, pokud chceš jen linku \*/

}

.navLink:hover::after {

width: 80%; /\* Šířka linky při hoveru (např. 80% šířky odkazu) \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Vyber si, co se ti líbí víc, nebo to zkombinuj. Nezapomeň, že hodnoty --color-primary a --color-secondary-background se berou z tvého globals.css. Pokud bys chtěl jinou barvu pro hover, můžeš si ji tam definovat nebo použít přímo hex kód.

OK, tak to menu nahore v Navbar.tsx by uz melo vypadat lip. Ted mam ale problem s tim Footer.tsx a Footer.module.css. Prijde mi ten footer takovej moc jednoduchej, hlavne ta cast .footerMain.

Chtel bych aby ty .footerColumn byly nejak lip usporadany a mozna ty .footerTitle ("Fitness Coaching", "Rychlé odkazy", "Kontakt") aby byly vyraznejsi. A ty socialni ikony v .socialLinks jsou taky takovy nemastny neslany.

Tady je kus kodu z Footer.module.css:

/\* Main Footer \*/

.footerMain {

background-color: #1a1a1a;

padding: 60px 0 40px;

border-top: 1px solid rgba(255, 255, 255, 0.05);

}

.footerContainer {

max-width: 1200px;

margin: 0 auto;

padding: 0 20px;

display: flex;

flex-wrap: wrap;

justify-content: space-between;

}

.footerColumn {

flex: 1;

min-width: 250px;

margin-bottom: 30px;

padding: 0 15px;

}

.footerTitle {

color: white;

font-size: 1.2rem;

margin-bottom: 25px;

font-weight: 600;

position: relative;

padding-bottom: 10px;

text-transform: uppercase;

letter-spacing: 1px;

}

.footerTitle::after {

content: '';

position: absolute;

bottom: 0;

left: 0;

width: 30px;

height: 2px;

background: #0088cc;

border-radius: 1px;

}

/\* ... \*/

.socialLinks {

display: flex;

gap: 12px;

margin-top: 25px;

}

.socialLink {

display: flex;

align-items: center;

justify-content: center;

width: 36px;

height: 36px;

border-radius: 4px;

background-color: rgba(255, 255, 255, 0.08);

color: white;

transition: all 0.3s ease;

}

.socialLink:hover {

background-color: #0088cc;

transform: translateY(-2px);

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Nejaky napady jak to vylepsit?

Jasně, podíváme se na ten footer.

**Pro .footerColumn a .footerTitle:**

Uspořádání sloupců pomocí flex: 1 a min-width: 250px je celkem standardní a mělo by fungovat dobře pro responzivitu. Pokud chceš, aby sloupce měly třeba přesnější šířku na větších obrazovkách, můžeš pro .footerColumn nastavit třeba flex-basis nebo max-width v kombinaci s media queries.

Co se týče .footerTitle, už tam máš podtržení a uppercase, což je dobrý základ. Můžeš zkusit:

1. **Zvětšit font:** Mírně větší písmo může pomoct.
2. **Jiná barva pro podtržení:** Použít výraznější barvu z globals.css, pokud #0088cc není tvoje hlavní.
3. **Tloušťka písma:** Zkusit font-weight: 700; pro ještě větší důraz.
4. **Odsazení:** Zvětšit margin-bottom u .footerTitle nebo padding-bottom, aby byl větší prostor mezi nadpisem a obsahem sloupce.

Příklad úpravy pro .footerTitle:

.footerTitle {

color: white;

font-size: 1.3rem; /\* Trochu větší \*/

margin-bottom: 30px; /\* Větší odsazení pod nadpisem \*/

font-weight: 700; /\* Výraznější \*/

position: relative;

padding-bottom: 12px; /\* Větší odsazení pro linku \*/

text-transform: uppercase;

letter-spacing: 1.2px; /\* Lehce větší prostrkání \*/

}

.footerTitle::after {

content: '';

position: absolute;

bottom: 0;

left: 0;

width: 40px; /\* Delší linka \*/

height: 3px; /\* Tlustší linka \*/

background: var(--color-primary); /\* Použij globální proměnnou pro konzistenci \*/

border-radius: 1.5px;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Pro .socialLinks a .socialLink:**

Tyhle ikonky můžou vypadat lépe, když budou mít jednotný styl a možná trochu jiný hover efekt.

1. **Velikost a odsazení:** Můžeš si pohrát s width, height a gap.
2. **Hover efekt:** Místo změny background-color na plnou barvu, můžeš zkusit jen zesvětlit pozadí nebo změnit barvu ikony, pokud to ikony umožňují (SVG ikony jsou na to lepší).

Příklad úpravy pro sociální ikony:

.socialLinks {

display: flex;

gap: 15px; /\* Větší mezera \*/

margin-top: 25px;

}

.socialLink {

display: flex;

align-items: center;

justify-content: center;

width: 40px; /\* O něco větší \*/

height: 40px; /\* O něco větší \*/

border-radius: 8px; /\* Více zakulacené rohy \*/

background-color: rgba(255, 255, 255, 0.1); /\* Mírně výraznější pozadí \*/

color: var(--color-footer-text); /\* Použij barvu textu z footeru z globals.css \*/

transition: background-color 0.3s ease, transform 0.3s ease, color 0.3s ease;

}

.socialLink:hover {

background-color: var(--color-primary); /\* Pozadí na primární barvu \*/

color: white; /\* Ikona bude bílá, aby byla vidět \*/

transform: translateY(-3px) scale(1.05); /\* Trochu je "nadzvedne" a zvětší \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /\* Jemný stín pro 3D efekt \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

v příloze je kód projektu, co aktuálně mám. Jde o fitness coaching web. aktuálně to vypadá jak na screenshotech. Ale hlavní stránka nevypadá dobře na telefonech. není to uplně responzivní. sám to zhodnoť na příloze č.4,5. Udělej vše pro to, aby to bylo responzivní. Budu chtít celé kódy a celé cesty, na to pamatuj

Edit

Prohlédl jsem si váš kód a screenshoty, a vidím problémy s responzivitou na mobilních zařízeních. Pojďme tyto problémy vyřešit, abychom zajistili, že váš fitness coaching web bude plně responzivní na všech zařízeních.

Hlavní problémy:

1. Obsah přetéká mimo viewport na mobilních zařízeních
2. Špatné velikosti a zarovnání prvků v mobilním zobrazení
3. Některé responzivní styly nejsou správně aplikovány

Pojďme tyto problémy opravit:

*/\* src/pages/index.module.css \*/*

.homepageContainer {

overflow-x: hidden;

width: 100%;

max-width: 100%;

}

*/\* Hero Section Styles \*/*

.heroSection {

display: flex;

align-items: center;

justify-content: space-between;

padding: 60px 20px;

background: linear-gradient(135deg, var(--color-background), var(--color-secondary-background));

position: relative;

overflow: hidden;

min-height: 500px;

width: 100%;

box-sizing: border-box;

}

.heroContent {

flex: 1;

max-width: 600px;

z-index: 2;

}

.heroTitle {

font-size: 2.8rem;

font-weight: 800;

margin-bottom: 16px;

line-height: 1.2;

color: var(--color-text-primary);

position: relative;

}

.heroTitle::after {

content: '';

position: absolute;

bottom: -10px;

left: 0;

width: 80px;

height: 4px;

background: var(--color-primary);

border-radius: 2px;

}

.heroSubtitle {

font-size: 1.5rem;

font-weight: 600;

margin-bottom: 20px;

color: var(--color-primary);

}

.heroText {

font-size: 1rem;

margin-bottom: 30px;

line-height: 1.6;

color: var(--color-text-secondary);

}

.heroButtons {

display: flex;

gap: 16px;

margin-top: 24px;

}

.primaryButton {

padding: 12px 24px;

background-color: var(--color-primary);

color: white;

border-radius: 30px;

font-weight: 600;

font-size: 1rem;

transition: all 0.3s ease;

text-align: center;

box-shadow: 0 4px 15px rgba(0, 123, 255, 0.2);

display: inline-block;

width: auto;

}

.primaryButton:hover {

background-color: var(--color-primary-hover);

transform: translateY(-3px);

box-shadow: 0 6px 20px rgba(0, 123, 255, 0.25);

}

.secondaryButton {

padding: 12px 24px;

background-color: transparent;

color: var(--color-text-primary);

border: 2px solid var(--color-primary);

border-radius: 30px;

font-weight: 600;

font-size: 1rem;

transition: all 0.3s ease;

text-align: center;

display: inline-block;

width: auto;

}

.secondaryButton:hover {

background-color: rgba(0, 123, 255, 0.05);

transform: translateY(-3px);

}

.heroImageContainer {

flex: 1;

display: flex;

justify-content: center;

align-items: center;

position: relative;

z-index: 1;

max-width: 45%;

}

.heroImage {

border-radius: 20px;

box-shadow: 0 20px 40px rgba(0, 0, 0, 0.15);

object-fit: cover;

max-width: 100%;

height: auto;

}

*/\* Marquee Section Styles \*/*

.marqueeSection {

overflow: hidden;

background-color: var(--color-primary);

margin: 0;

padding: 12px 0;

width: 100%;

box-sizing: border-box;

}

.marqueeWrapper {

display: flex;

width: 100%;

overflow: hidden;

}

@keyframes marqueeAnimation {

0% {

transform: translateX(0);

}

100% {

transform: translateX(-100%);

}

}

.marqueeContent {

display: flex;

align-items: center;

white-space: nowrap;

animation: marqueeAnimation 30s linear infinite;

padding-left: 100%;

}

.marqueeContent span {

color: white;

font-size: 1.1rem;

font-weight: 700;

text-transform: uppercase;

padding: 0 15px;

letter-spacing: 1px;

}

.marqueeSeparator {

color: rgba(255, 255, 255, 0.6);

font-size: 1.3rem;

margin: 0 8px;

}

*/\* Services Section Styles \*/*

.servicesSection {

padding: 50px 15px;

text-align: center;

width: 100%;

box-sizing: border-box;

}

.servicesSection h2 {

font-size: 2rem;

margin-bottom: 25px;

}

.servicesGrid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 25px;

max-width: 1200px;

margin: 0 auto;

padding: 0 10px;

box-sizing: border-box;

}

.serviceCard {

background-color: var(--color-background);

border-radius: 12px;

padding: 25px;

display: flex;

flex-direction: column;

align-items: flex-start;

box-shadow: 0 4px 6px var(--color-box-shadow);

transition: transform 0.3s ease, box-shadow 0.3s ease;

box-sizing: border-box;

width: 100%;

}

.serviceCard:hover {

transform: translateY(-8px);

box-shadow: 0 12px 24px rgba(0, 0, 0, 0.15);

}

.serviceTitle {

font-size: 1.4rem;

font-weight: 900;

margin: 0 0 20px 0;

letter-spacing: 0.5px;

color: var(--color-text-primary);

}

.serviceCard:first-child .serviceTitle {

background: linear-gradient(45deg, #3B82F6, #60A5FA);

-webkit-background-clip: text;

background-clip: text;

-webkit-text-fill-color: transparent;

}

.serviceFeatures {

list-style: none;

padding: 0;

margin: 0 0 auto 0;

width: 100%;

}

.serviceFeatures li {

color: var(--color-text-secondary);

margin-bottom: 12px;

font-size: 0.95rem;

display: flex;

align-items: center;

text-align: left;

}

.priceContainer {

margin: 20px 0;

display: flex;

align-items: baseline;

gap: 5px;

}

.price {

font-size: 1.8rem;

font-weight: 900;

color: var(--color-primary);

}

.period {

color: var(--color-text-secondary);

font-size: 0.9rem;

}

.serviceButton {

background: none;

border: 1px solid var(--color-text-primary);

padding: 8px 18px;

font-size: 0.9rem;

cursor: pointer;

display: flex;

align-items: center;

transition: all 0.3s ease;

color: var(--color-text-primary);

}

.serviceButton:hover {

background-color: rgba(0, 0, 0, 0.05);

}

*/\* CTA Section styles \*/*

.ctaSection {

background-color: var(--color-secondary-background);

padding: 50px 20px;

text-align: center;

border-radius: 16px;

margin: 40px auto;

max-width: 900px;

box-sizing: border-box;

}

.ctaSection h2 {

font-size: 1.8rem;

margin-bottom: 20px;

color: var(--color-text-primary);

}

.ctaSection p {

margin-bottom: 25px;

font-size: 1rem;

color: var(--color-text-secondary);

}

.ctaButton {

display: inline-block;

padding: 12px 28px;

background-color: var(--color-button-primary);

color: white;

border-radius: 30px;

font-weight: 600;

font-size: 1rem;

transition: all 0.3s ease;

text-align: center;

box-shadow: 0 4px 15px rgba(40, 167, 69, 0.2);

}

.ctaButton:hover {

background-color: var(--color-button-primary-hover);

transform: translateY(-3px);

box-shadow: 0 6px 20px rgba(40, 167, 69, 0.25);

}

*/\* Media query pro střední obrazovky \*/*

@media (max-width: 1024px) {

.servicesGrid {

grid-template-columns: repeat(2, 1fr);

padding: 0 15px;

}

.heroTitle {

font-size: 2.5rem;

}

.heroSubtitle {

font-size: 1.4rem;

}

.heroImageContainer {

max-width: 45%;

}

}

*/\* Media query pro tablety \*/*

@media (max-width: 900px) {

.heroSection {

flex-direction: column;

padding: 40px 20px;

text-align: center;

}

.heroContent {

max-width: 100%;

margin-bottom: 30px;

}

.heroImageContainer {

max-width: 80%;

}

.heroTitle::after {

left: 50%;

transform: translateX(-50%);

}

.heroButtons {

justify-content: center;

}

}

*/\* Media query pro menší obrazovky (mobilní telefony) \*/*

@media (max-width: 768px) {

.servicesGrid {

grid-template-columns: 1fr;

padding: 0 10px;

}

.serviceCard {

padding: 20px;

}

.serviceTitle {

font-size: 1.3rem;

}

.serviceFeatures li {

font-size: 0.9rem;

}

.price {

font-size: 1.7rem;

}

.heroSection {

padding: 30px 15px;

min-height: auto;

}

.heroTitle {

font-size: 2rem;

}

.heroSubtitle {

font-size: 1.3rem;

}

.heroText {

font-size: 0.95rem;

}

.heroButtons {

flex-direction: column;

gap: 12px;

width: 100%;

}

.primaryButton, .secondaryButton {

width: 100%;

padding: 12px 20px;

box-sizing: border-box;

}

.heroImageContainer {

width: 100%;

max-width: 100%;

}

.heroImage {

width: 100%;

max-height: 350px;

}

.marqueeContent span {

font-size: 0.9rem;

padding: 0 10px;

}

.marqueeSeparator {

margin: 0 5px;

}

.ctaSection {

margin: 30px 15px;

padding: 40px 15px;

}

.ctaSection h2 {

font-size: 1.5rem;

}

.ctaButton {

width: 100%;

padding: 12px 20px;

}

}

*/\* Media query pro velmi malé obrazovky \*/*

@media (max-width: 480px) {

.heroSection {

padding: 25px 12px;

}

.heroTitle {

font-size: 1.8rem;

}

.heroSubtitle {

font-size: 1.2rem;

}

.heroImage {

max-height: 300px;

}

.marqueeContent span {

font-size: 0.8rem;

padding: 0 8px;

}

.serviceCard {

padding: 18px;

}

.ctaSection {

padding: 30px 15px;

margin: 20px 10px;

}

.ctaSection h2 {

font-size: 1.4rem;

}

}

*// src/pages/index.tsx*

import React, { useEffect, useRef } from 'react';

import styles from './index.module.css';

import Link from 'next/link';

import Image from 'next/image';

import Head from 'next/head';

const HomePage: React.FC = () => {

const marqueeRef = useRef<HTMLDivElement>(null);

*// Efekt pro duplikaci obsahu v marquee pro plynulý scrolling*

useEffect(() => {

if (marqueeRef.current) {

*// Duplikaní obsahu je již řešeno v HTML pomocí aria-hidden="true"*

*// Tento efekt lze případně rozšířit pro další funkce*

}

}, []);

return (

<>

<Head>

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0" />

</Head>

<div className={styles.homepageContainer}>

<section className={styles.heroSection}>

<div className={styles.heroContent}>

<h1 className={styles.heroTitle}>Dosáhněte svých fitness cílů</h1>

<h2 className={styles.heroSubtitle}>Profesionální coaching</h2>

<p className={styles.heroText}>

Pomůžu vám transformovat vaše tělo a životní styl pomocí personalizovaných tréninkových plánů

a odborného vedení.

</p>

<div className={styles.heroButtons}>

<Link href="/kontakt" className={styles.primaryButton}>

Začít hned teď

</Link>

<Link href="/aboutme" className={styles.secondaryButton}>

Více o mně

</Link>

</div>

</div>

<div className={styles.heroImageContainer}>

<Image

src="/images/trener.jpg"

alt="Fitness trenér"

width={500}

height={600}

className={styles.heroImage}

priority

/>

</div>

</section>

{*/\* Marquee sekce \*/*}

<section className={styles.marqueeSection}>

<div className={styles.marqueeWrapper} ref={marqueeRef}>

<div className={styles.marqueeContent}>

<span>PŘEKONEJ SVÉ LIMITY</span>

<span className={styles.marqueeSeparator}>•</span>

<span>STAŇ SE NEJLEPŠÍ VERZÍ SEBE SAMA</span>

<span className={styles.marqueeSeparator}>•</span>

<span>TRANSFORMACE ZAČÍNÁ DNES</span>

<span className={styles.marqueeSeparator}>•</span>

<span>SILNĚJŠÍ KAŽDÝM DNEM</span>

<span className={styles.marqueeSeparator}>•</span>

</div>

<div className={styles.marqueeContent} aria-hidden="true">

<span>PŘEKONEJ SVÉ LIMITY</span>

<span className={styles.marqueeSeparator}>•</span>

<span>STAŇ SE NEJLEPŠÍ VERZÍ SEBE SAMA</span>

<span className={styles.marqueeSeparator}>•</span>

<span>TRANSFORMACE ZAČÍNÁ DNES</span>

<span className={styles.marqueeSeparator}>•</span>

<span>SILNĚJŠÍ KAŽDÝM DNEM</span>

<span className={styles.marqueeSeparator}>•</span>

</div>

</div>

</section>

<section className={styles.servicesSection}>

<h2>Moje Služby</h2>

<div className={styles.servicesGrid}>

{*/\* COACHING+ \*/*}

<div className={styles.serviceCard}>

<h3 className={styles.serviceTitle}>COACHING+</h3>

<ul className={styles.serviceFeatures}>

<li>✓ 1 společný trénink měsíčně</li>

<li>✓ Jídelníček</li>

<li>✓ Tréninkový plán na míru</li>

<li>✓ Video vysvětlivky cviků</li>

<li>✓ 2x týdně kontroly</li>

<li>✓ WhatsApp chat</li>

</ul>

<div className={styles.priceContainer}>

<span className={styles.price}>2500,-</span>

<span className={styles.period}>/ 1 měsíc</span>

</div>

<button className={styles.serviceButton}>

MÁM ZÁJEM →

</button>

</div>

{*/\* COACHING CLASSIC \*/*}

<div className={styles.serviceCard}>

<h3 className={styles.serviceTitle}>COACHING CLASSIC</h3>

<ul className={styles.serviceFeatures}>

<li>✓ Jídelníček</li>

<li>✓ Tréninkový plán na míru</li>

<li>✓ Video vysvětlivky cviků</li>

<li>✓ 2x týdně kontroly</li>

<li>✓ WhatsApp chat</li>

</ul>

<div className={styles.priceContainer}>

<span className={styles.price}>2000,-</span>

<span className={styles.period}>/ 1 měsíc</span>

</div>

<button className={styles.serviceButton}>

MÁM ZÁJEM →

</button>

</div>

{*/\* COACHING STUDENT \*/*}

<div className={styles.serviceCard}>

<h3 className={styles.serviceTitle}>COACHING STUDENT</h3>

<ul className={styles.serviceFeatures}>

<li>✓ Jídelníček</li>

<li>✓ Tréninkový plán na míru</li>

<li>✓ Video vysvětlivky cviků</li>

<li>✓ 2x týdně kontroly</li>

<li>✓ WhatsApp chat</li>

</ul>

<div className={styles.priceContainer}>

<span className={styles.price}>2000,-</span>

<span className={styles.period}>/ 1. měsíc, dále 1500,-</span>

</div>

<button className={styles.serviceButton}>

MÁM ZÁJEM →

</button>

</div>

{*/\* PORADENSTVÍ \*/*}

<div className={styles.serviceCard}>

<h3 className={styles.serviceTitle}>PORADENSTVÍ</h3>

<ul className={styles.serviceFeatures}>

<li>✓ Suplementace</li>

<li>✓ Optimalizace zdraví</li>

<li>✓ Odběry</li>

</ul>

<div className={styles.priceContainer}>

<span className={styles.price}>2500,-</span>

<span className={styles.period}>/ 1 měsíc</span>

</div>

<button className={styles.serviceButton}>

MÁM ZÁJEM →

</button>

</div>

{*/\* JÍDELNÍČEK \*/*}

<div className={styles.serviceCard}>

<h3 className={styles.serviceTitle}>JÍDELNÍČEK</h3>

<ul className={styles.serviceFeatures}>

<li>✓ 12 variant pro každé jídlo</li>

<li>✓ Náročnost přípravy dle chuti</li>

<li>✓ Jídla na míru chutím</li>

<li>✓ 2x týdně kontroly</li>

<li>✓ WhatsApp chat</li>

</ul>

<div className={styles.priceContainer}>

<span className={styles.price}>1500,-</span>

<span className={styles.period}>/ 1 měsíc</span>

</div>

<button className={styles.serviceButton}>

MÁM ZÁJEM →

</button>

</div>

{*/\* TRÉNINKOVÝ PLÁN \*/*}

<div className={styles.serviceCard}>

<h3 className={styles.serviceTitle}>TRÉNINKOVÝ PLÁN</h3>

<ul className={styles.serviceFeatures}>

<li>✓ Jednorázový</li>

<li>✓ Na míru dle stanovených cílů</li>

<li>✓ Na míru dle vybavení</li>

<li>✓ Na míru dle časových možností</li>

</ul>

<div className={styles.priceContainer}>

<span className={styles.price}>2000,-</span>

</div>

<button className={styles.serviceButton}>

MÁM ZÁJEM →

</button>

</div>

</div>

</section>

<section className={styles.ctaSection}>

<h2>Začněte svou cestu k lepšímu zdraví ještě dnes!</h2>

<p>Kontaktujte mě pro úvodní konzultaci zdarma.</p>

<Link href="/kontakt" className={styles.ctaButton}>

Kontaktujte mě

</Link>

</section>

</div>

</>

);

};

export default HomePage;

*/\* src/styles/globals.css \*/*

*/\* Definice barev pro Light mode (výchozí) \*/*

:root {

--color-background: #fff; */\* Bílé pozadí \*/*

--color-text-primary: #333; */\* Tmavě šedý hlavní text \*/*

--color-text-secondary: #555; */\* Šedý sekundární text \*/*

--color-primary: #007bff; */\* Modrá primární barva (tlačítka, odkazy) \*/*

--color-primary-hover: #0056b3; */\* Tmavší modrá pro hover \*/*

--color-secondary-background: #f8f8f8; */\* Světle šedé pozadí sekcí \*/*

--color-box-shadow: rgba(0, 0, 0, 0.1); */\* Stín pro boxy \*/*

--color-input-border: #ccc; */\* Border inputů \*/*

--color-input-focus-border: #007bff; */\* Border inputů při focusu \*/*

--color-input-focus-shadow: rgba(0, 123, 255, 0.2); */\* Stín inputů při focusu \*/*

--color-button-primary: #28a745; */\* Zelená barva primárního tlačítka \*/*

--color-button-primary-hover: #218838; */\* Tmavší zelená pro hover tlačítka \*/*

--color-navbar-background: white; */\* Bílé pozadí navbaru \*/*

--color-navbar-shadow: rgba(0, 0, 0, 0.1); */\* Stín navbaru \*/*

--color-footer-background: #333; */\* Tmavě šedé pozadí footeru \*/*

--color-footer-text: white; */\* Bílý text ve footeru \*/*

}

*/\* Definice barev pro Dark mode \*/*

body.dark-mode {

--color-background: #121212; */\* Tmavé pozadí \*/*

--color-text-primary: #eee; */\* Světle šedý hlavní text \*/*

--color-text-secondary: #ccc; */\* Světle šedý sekundární text \*/*

--color-primary: #64b5f6; */\* Světlejší modrá primární barva \*/*

--color-primary-hover: #42a5f5; */\* Ještě světlejší modrá pro hover \*/*

--color-secondary-background: #212121; */\* Tmavší šedé pozadí sekcí \*/*

--color-box-shadow: rgba(255, 255, 255, 0.08); */\* Světlejší stín pro boxy \*/*

--color-input-border: #555; */\* Border inputů v dark mode \*/*

--color-input-focus-border: #64b5f6; */\* Border inputů při focusu v dark mode \*/*

--color-input-focus-shadow: rgba(100, 181, 246, 0.4); */\* Stín inputů při focusu v dark mode \*/*

--color-button-primary: #4caf50; */\* Světlejší zelená barva primárního tlačítka v dark mode \*/*

--color-button-primary-hover: #43a047; */\* Ještě světlejší zelená pro hover tlačítka v dark mode \*/*

--color-navbar-background: #1e1e1e; */\* Tmavší pozadí navbaru v dark mode \*/*

--color-navbar-shadow: rgba(255, 255, 255, 0.08); */\* Světlejší stín navbaru v dark mode \*/*

--color-footer-background: #000; */\* Černé pozadí footeru v dark mode \*/*

--color-footer-text: #ddd; */\* Světle šedý text ve footeru v dark mode \*/*

}

*/\* RESET - KRITICKÉ PRO MOBILNÍ ZOBRAZENÍ \*/*

\*, \*::before, \*::after {

box-sizing: border-box;

margin: 0;

padding: 0;

}

html {

scroll-behavior: smooth;

-webkit-text-size-adjust: 100%;

}

html, body {

overflow-x: hidden;

width: 100%;

position: relative;

margin: 0;

padding: 0;

font-family: 'Arial', sans-serif;

color: var(--color-text-primary);

background-color: var(--color-background);

line-height: 1.6;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

transition: background-color 0.3s ease, color 0.3s ease;

}

body {

display: flex;

flex-direction: column;

min-height: 100vh;

overflow-x: hidden;

}

#\_\_next {

width: 100%;

display: flex;

flex-direction: column;

min-height: 100vh;

overflow-x: hidden;

}

*/\* Další základní styly \*/*

a {

color: inherit;

text-decoration: none;

}

h1, h2, h3, h4, h5, h6 {

font-weight: bold;

margin-bottom: 1rem;

}

p {

margin-bottom: 1rem;

}

img {

max-width: 100%;

height: auto;

display: block;

}

button {

cursor: pointer;

font-family: inherit;

}

*/\* Opravy přetečení na mobilních zařízeních \*/*

.layoutContainer {

display: flex;

flex-direction: column;

min-height: 100vh;

width: 100%;

overflow-x: hidden;

}

.mainContent {

flex: 1;

width: 100%;

box-sizing: border-box;

overflow-x: hidden;

}

*/\* Specifická pravidla pro mobilní zařízení \*/*

@media screen and (max-width: 768px) {

body {

-webkit-overflow-scrolling: touch;

touch-action: manipulation;

}

h1 {

font-size: 1.8rem;

}

h2 {

font-size: 1.5rem;

}

h3 {

font-size: 1.3rem;

}

}

/\*

Taky se ujisti, že v Footer.tsx máš správně nastavené ikony (např. size prop u react-icons), aby se přizpůsobily velikosti .socialLink.

Zkus tyhle úpravy a dej vědět, jak to vypadá. Můžeme to dál ladit.

Super, navbar a footer mame trochu poladeny. Ted bych chtel pridat to prepinani mezi svetlym a tmavym rezimem. Uz mam soubor ThemeToggle.tsx a ThemeToggle.module.css.

V ThemeToggle.tsx to mam takhle:

import React, { useState, useEffect } from 'react';

import styles from './ThemeToggle.module.css';

import { BsSun, BsMoonStarsFill } from 'react-icons/bs';

const ThemeToggle: React.FC = () => {

const [darkMode, setDarkMode] = useState(false);

useEffect(() => {

const storedTheme = localStorage.getItem('theme');

if (storedTheme === 'dark') {

setDarkMode(true);

} else if (storedTheme === 'light') {

setDarkMode(false);

} else {

const prefersDarkMode = window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches;

setDarkMode(prefersDarkMode);

}

}, []);

useEffect(() => {

localStorage.setItem('theme', darkMode ? 'dark' : 'light');

if (darkMode) {

document.body.classList.add('dark-mode');

} else {

document.body.classList.remove('dark-mode');

}

}, [darkMode]);

const toggleDarkMode = () => {

setDarkMode(!darkMode);

};

return (

<button className={styles.themeToggleButton} onClick={toggleDarkMode}>

{darkMode ? <BsSun size="20" color="white"/> : <BsMoonStarsFill size="20" />}

</button>

);

};

export default ThemeToggle;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

A v ThemeToggle.module.css je jen:

.themeToggleButton {

background: none;

border: none;

cursor: pointer;

padding: 5px;

color: var(--color-text-primary);

transition: color 0.3s ease;

}

.themeToggleButton:hover {

opacity: 0.8;

}

.themeToggleButton:focus {

outline: none;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Tohle tlacitko chci dat do Navbar.tsx na desktopu i v mobilnim menu. Ale nevim presne jak to napojit na ty barvy v globals.css, kde mam definovany --color-background, --color-text-primary atd. pro light mode a pak v body.dark-mode ty samy promenny pro dark mode.

Jak docilim toho, aby se po kliknuti na tlacitko opravdu zmenily barvy cele stranky podle tech promennych v globals.css? Ta logika s document.body.classList.add('dark-mode') by mela fungovat, ne? Ale prijde mi, ze se to neprojevuje na vsech prvcich, hlavne na tech co maji barvy nastavene primo.

Máš to skoro hotové a logika s document.body.classList.add('dark-mode') a document.body.classList.remove('dark-mode') je přesně to, co potřebuješ, aby se aplikovaly CSS proměnné definované v globals.css pod selektorem body.dark-mode.

**Proč se to nemusí projevovat všude:**

1. **Přímé nastavení barev:** Pokud máš někde v CSS modulech nastavenou barvu přímo (např. background-color: #fff; nebo color: #333;) místo použití CSS proměnné (např. background-color: var(--color-background); nebo color: var(--color-text-primary);), tak se tato přímá hodnota **nepřepíše** změnou třídy na body. CSS proměnné fungují tak, že změníš jejich hodnotu centrálně (což děláš přidáním/odebráním třídy dark-mode na body), a všechny prvky, které tuto proměnnou používají, se automaticky aktualizují.
   * **Řešení:** Projdi si všechny své CSS soubory (hlavně Navbar.module.css, Footer.module.css a další, které budeš vytvářet) a ujisti se, že pro barvy pozadí, textu, ohraničení atd. používáš CSS proměnné z globals.css (např. var(--color-background), var(--color-text-primary), var(--color-primary) atd.).
2. **Specifičnost CSS selektorů:** Ve vzácných případech může být problém se specifičností. Pokud máš nějaký velmi specifický selektor, který nastavuje barvu, mohl by přebít obecnější pravidla z body.dark-mode. To ale u tvé současné struktury nepředpokládám jako hlavní problém.

**Kontrola globals.css:**

Ujisti se, že tvůj globals.css má správně definované proměnné pro oba módy. Struktura by měla vypadat nějak takto:

/\* src/styles/globals.css \*/

/\* Definice barev pro Light mode\*/

:root {

--color-background: #fff;

--color-text-primary: #333;

--color-text-secondary: #555;

--color-primary: #007bff;

/\* ... a další proměnné pro light mode ... \*/

--color-navbar-background: white;

--color-footer-background: #333;

--color-footer-text: white;

}

/\* Definice barev pro Dark mode \*/

body.dark-mode {

--color-background: #121212;

--color-text-primary: #eee;

--color-text-secondary: #ccc;

--color-primary: #64b5f6;

/\* ... a další proměnné pro dark mode ... \*/

--color-navbar-background: #1e1e1e;

--color-footer-background: #000;

--color-footer-text: #ddd;

}

/\* Základní styly používající proměnné \*/

html, body {

/\* ... \*/

color: var(--color-text-primary);

background-color: var(--color-background);

transition: background-color 0.3s ease, color 0.3s ease; /\* Přidá plynulý přechod \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Důležitý je ten transition na body, aby změna barev byla plynulá.

**Integrace ThemeToggle do Navbar.tsx:**

To máš v plánu správně.  
V Navbar.tsx:

1. Importuj ThemeToggle: import ThemeToggle from './ThemeToggle';
2. Umísti komponentu <ThemeToggle /> tam, kde ji chceš mít:
   * V .desktopNav (třeba na konec ul nebo vedle něj).
   * V .mobileMenu jako jednu z položek li.

Příklad pro desktop:

// Navbar.tsx

// ... imports ...

import ThemeToggle from './ThemeToggle';

// ... component ...

<div className={styles.desktopNav}>

<ul className={styles.navList}>

{/\* ... nav items ... \*/}

</ul>

<ThemeToggle /> {/\* Přidáno zde \*/}

</div>

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

Příklad pro mobilní menu:

// Navbar.tsx

// ... imports ...

import ThemeToggle from './ThemeToggle'; // Pokud už není importováno

// ... component ...

<ul className={styles.mobileMenuList}>

{/\* ... other mobile menu items ... \*/}

<li className={styles.mobileMenuItem}>

<ThemeToggle /> {/\* Přidáno zde \*/}

</li>

</ul>

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

**Testování:**  
Po těchto úpravách a kontrole CSS souborů by mělo přepínání fungovat globálně. Otevři si vývojářské nástroje v prohlížeči a sleduj, zda se třída dark-mode správně přidává/odebírá z elementu <body> a zda se v záložce "Computed" (Vypočtené styly) u prvků mění hodnoty CSS proměnných.

Pokud by to stále někde nefungovalo, podívej se přesně na ten prvek ve vývojářských nástrojích, jaké styly se na něj aplikují a zda tam není nějaká "tvrdě" nakódovaná barva.

Chtel bych, aby barva tech ikonek v ThemeToggle.tsx taky reagovala na zmenu motivu a pouzivala --color-text-primary z globals.css, aby byla vzdycky kontrastni k pozadi tlacitka nebo navbaru.

Tady je ten return v ThemeToggle.tsx:

return (

<button className={styles.themeToggleButton} onClick={toggleDarkMode}>

{darkMode ? <BsSun size="20" color="white"/> : <BsMoonStarsFill size="20" />}

</button>

);

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

A CSS pro tlacitko v ThemeToggle.module.css:

.themeToggleButton {

background: none;

border: none;

cursor: pointer;

padding: 5px;

color: var(--color-text-primary); /\* Tohle by melo barvit ikonu, ne? \*/

transition: color 0.3s ease;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Puvodne jsem si myslel, ze color: var(--color-text-primary); v .themeToggleButton obarvi i tu ikonu, ale zda se, ze color="white" v komponente <BsSun /> to prebiji. Jak to udelat spravne?

Máš pravdu, color="white" přímo v JSX komponentě <BsSun /> má vyšší prioritu než CSS vlastnost color na rodičovském <button>. Ikony z react-icons často přijímají color prop, který inline styluje SVG element.

Máš několik způsobů, jak to vyřešit:

**1. Odebrat color prop z ikon a spolehnout se na CSS:**

Nejčistší řešení je odebrat color prop přímo z JSX a nechat CSS, aby se postaralo o barvu. Vlastnost color na tlačítku by se pak měla dědit na SVG ikonu uvnitř, pokud ikona sama o sobě nemá nastavenou pevnou barvu (což ikony z react-icons většinou nemají, pokud jim explicitně nenastavíš fill nebo stroke atributy).

Uprav ThemeToggle.tsx takto:

return (

<button className={styles.themeToggleButton} onClick={toggleDarkMode}>

{darkMode ? <BsSun size="20" /> : <BsMoonStarsFill size="20" />}

</button>

);

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

A tvůj ThemeToggle.module.css by měl pak fungovat správně:

.themeToggleButton {

background: none;

border: none;

cursor: pointer;

padding: 5px;

color: var(--color-text-primary); /\* Toto by nyní mělo obarvit ikonu \*/

transition: color 0.3s ease;

display: flex; /\* Přidáno pro lepší zarovnání ikony, pokud by byl problém \*/

align-items: center; /\* Přidáno pro lepší zarovnání ikony \*/

justify-content: center; /\* Přidáno pro lepší zarovnání ikony \*/

}

.themeToggleButton:hover {

opacity: 0.8;

}

.themeToggleButton:focus {

outline: none;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Přidal jsem display: flex a align-items/justify-content pro případ, že by ikona nebyla v tlačítku dokonale vycentrovaná.

**2. Dynamicky nastavovat color prop v JSX pomocí CSS proměnné (méně doporučené pro tento případ):**

Teoreticky bys mohl zkusit nastavit barvu ikon dynamicky pomocí JavaScriptu, který by četl CSS proměnnou, ale to je zbytečně komplikované pro tento případ. Spoléhání na dědičnost CSS je zde jednodušší a správnější.

**3. Specifické stylování SVG uvnitř tlačítka (pokud by první možnost nefungovala):**

Pokud by z nějakého důvodu ikona stále nedědila barvu (což by bylo neobvyklé pro react-icons), mohl bys explicitně cílit na SVG elementy uvnitř tlačítka v ThemeToggle.module.css:

.themeToggleButton svg {

fill: currentColor; /\* Nebo fill: var(--color-text-primary); \*/

/\* Nebo pokud ikona používá stroke: \*/

/\* stroke: currentColor; \*/

/\* stroke: var(--color-text-primary); \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

currentColor je speciální CSS hodnota, která říká elementu, aby použil hodnotu vlastnosti color svého rodiče.

**Doporučení:**  
Začni s **možností 1**. Je to nejčistší a nejvíce odpovídá tomu, jak by CSS dědičnost měla fungovat. Odeber color="white" z <BsSun /> a nech color: var(--color-text-primary); v .themeToggleButton dělat svou práci.

Po této úpravě by se ikony měly správně přebarvovat podle aktuálně nastavené CSS proměnné --color-text-primary definované v :root nebo body.dark-mode v globals.css.

Zatím nemám žádný HTML/JSX kód pro ty kartičky, ani CSS, takže začínám od nuly. Můžeš mi navrhnout strukturu JSX pro jednu kartičku a k tomu základní CSS v index.module.css (nebo page.module.css)?

Rozumím, pustíme se do návrhu kartiček pro služby.

**Návrh JSX struktury pro jednu kartičku služby (v index.tsx nebo page.tsx):**

Mohlo by to vypadat nějak takto pro jednu kartu. Toto bys pak opakoval pro každou službu, ideálně procházením pole dat.

// Předpokládejme, že máš nějaký kontejner pro všechny karty, např. <div className={styles.servicesGrid}>

<div className={styles.serviceCard}>

<div className={styles.serviceIconContainer}>

{/\* Zde může být ikona, např. z react-icons nebo vlastní SVG \*/}

{/\* <FaDumbbell className={styles.serviceIcon} /> \*/}

<span className={styles.serviceIconPlaceholder}>💪</span> {/\* Prozatímní placeholder \*/}

</div>

<h3 className={styles.serviceTitle}>Název Služby (např. Osobní Trénink)</h3>

<p className={styles.serviceDescription}>

Krátký a výstižný popis služby, co klient získá, pro koho je vhodná atd.

</p>

<div className={styles.servicePriceContainer}>

<span className={styles.servicePrice}>1500 Kč</span>

<span className={styles.servicePeriod}>/ měsíc</span>

</div>

<a href="/kontakt" className={styles.serviceButton}>

Zjistit více

</a>

</div>

// Tento blok bys opakoval nebo generoval v cyklu

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Návrh základního CSS pro kartičky (v index.module.css nebo page.module.css):**

Začneme s kontejnerem pro mřížku a stylem pro samotnou kartičku.

/\* Kontejner pro všechny kartičky služeb \*/

.servicesGrid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); /\* Responzivní mřížka \*/

gap: 30px; /\* Mezery mezi kartičkami \*/

padding: 40px 0; /\* Odsazení sekce \*/

max-width: 1200px; /\* Maximální šířka obsahu \*/

margin: 0 auto; /\* Vycentrování \*/

}

/\* Styly pro jednotlivou kartičku služby \*/

.serviceCard {

background-color: var(--color-secondary-background);

border-radius: 12px;

padding: 25px;

box-shadow: 0 4px 12px var(--color-box-shadow);

display: flex;

flex-direction: column; /\* Uspořádá obsah pod sebe \*/

transition: transform 0.3s ease, box-shadow 0.3s ease;

border: 1px solid transparent; /\* Pro případný border při hoveru \*/

}

.serviceCard:hover {

transform: translateY(-5px);

box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15); /\* Výraznější stín při hoveru \*/

/\* border-color: var(--color-primary); \*/ /\* Možnost přidat ohraničení při hoveru \*/

}

.serviceIconContainer {

width: 60px;

height: 60px;

border-radius: 50%;

background-color: var(--color-primary);

color: white;

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 20px;

font-size: 1.8rem; /\* Pro placeholder ikonu \*/

/\* Pokud používáš react-icons, nastav velikost ikony přímo v JSX \*/

}

.serviceIconPlaceholder { /\* Pokud používáš span jako placeholder \*/

line-height: 1;

}

/\* .serviceIcon { \*/ /\* Pokud používáš react-icons \*/

/\* font-size: 2rem; \*/

/\* } \*/

.serviceTitle {

font-size: 1.5rem;

font-weight: 700;

color: var(--color-text-primary);

margin-bottom: 15px;

}

.serviceDescription {

font-size: 0.95rem;

color: var(--color-text-secondary);

line-height: 1.6;

margin-bottom: 20px;

flex-grow: 1; /\* Zajistí, že popis zabere dostupný prostor a tlačítko bude dole \*/

}

.servicePriceContainer {

margin-bottom: 20px;

text-align: left; /\* Nebo center, podle preferencí \*/

}

.servicePrice {

font-size: 1.8rem;

font-weight: 700;

color: var(--color-primary);

margin-right: 8px;

}

.servicePeriod {

font-size: 0.9rem;

color: var(--color-text-secondary);

}

.serviceButton {

display: inline-block;

background-color: var(--color-primary);

color: white;

padding: 12px 25px;

border-radius: 8px;

text-decoration: none;

font-weight: 600;

text-align: center;

transition: background-color 0.3s ease, transform 0.2s ease;

margin-top: auto; /\* Posune tlačítko na dno karty, pokud popis není dost dlouhý \*/

}

.serviceButton:hover {

background-color: var(--color-primary-hover);

transform: translateY(-2px);

}

/\* Media query pro menší obrazovky, pokud by bylo potřeba upravit počet sloupců jinak \*/

@media (max-width: 768px) {

.servicesGrid {

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 20px;

}

.serviceCard {

padding: 20px;

}

.serviceTitle {

font-size: 1.3rem;

}

.servicePrice {

font-size: 1.6rem;

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Vysvětlení CSS:**

* **.servicesGrid**: Používá CSS Grid k vytvoření responzivní mřížky. repeat(auto-fit, minmax(300px, 1fr)) znamená, že se pokusí vejít co nejvíce sloupců o minimální šířce 300px a maximální šířce 1fr (flexibilní část dostupného prostoru).
* **.serviceCard**: Základní styly pro kartu, včetně pozadí, zaoblení, stínu a přechodů pro hover efekt. display: flex a flex-direction: column pomáhá s uspořádáním obsahu uvnitř karty.
* **.serviceIconContainer**: Styly pro kontejner ikony.
* **.serviceDescription**: flex-grow: 1; je důležité, aby popis vyplnil dostupný vertikální prostor, což způsobí, že cena a tlačítko budou zarovnány na spodní části karty, i když popisy budou mít různou délku.
* **.serviceButton**: margin-top: auto; také pomáhá tlačítko "přilepit" dolů.

Tohle je dobrý výchozí bod. Můžeš si samozřejmě pohrát s barvami, velikostmi písma, odsazením a dalšími detaily, aby to co nejlépe odpovídalo tvé představě. Až budeš mít karty vytvořené v JSX, uvidíš, jak to vypadá, a můžeme to dál ladit.

.heroSection {

position: relative;

min-height: 90vh;

/\* ... další styly pro full width ... \*/

color: white;

display: flex;

align-items: center;

overflow: hidden;

background: linear-gradient(120deg, #000 0%, #071a36 70%, #0c2d61 100%);

/\* ... \*/

}

.heroContent {

display: flex;

align-items: center;

justify-content: space-between;

padding: 50px 40px;

width: 100%;

max-width: 1440px; /\* Omezení pro obsah \*/

margin: 0 auto; /\* Centrování obsahu \*/

}

.heroTextContent {

flex: 1;

max-width: 600px;

padding-right: 40px;

/\* ... \*/

}

.heroImageContainer {

flex: 0 0 40%;

display: flex;

justify-content: flex-end;

position: relative;

}

.heroImage {

max-width: 100%;

height: auto;

border-radius: 5px;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.4);

/\* ... \*/

}

/\* A pak mam media queries: \*/

@media (max-width: 992px) {

.heroSection {

padding: 40px 0; /\* Změněno z min-height na padding \*/

}

.heroContent {

flex-direction: column-reverse; /\* Tady to obracím \*/

text-align: center;

padding: 30px 40px;

}

.heroTextContent {

max-width: 100%;

padding-right: 0;

padding-top: 40px;

}

.heroImageContainer {

max-width: 400px;

justify-content: center;

margin-bottom: 30px; /\* Přidána mezera pod obrázkem \*/

flex: 0 0 auto; /\* Změna flex vlastnosti \*/

}

/\* ... další menší úpravy ... \*/

}

/\* A pro ještě menší obrazovky: \*/

@media (max-width: 576px) {

.heroTitle {

font-size: 2.8rem;

}

.heroImageContainer {

max-width: 280px; /\* Zmenšení obrázku \*/

}

.heroContent {

padding: 30px 20px;

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Ta media query pro @media (max-width: 992px) s flex-direction: column-reverse; sice da text pod obrazek, ale porad to neni ono. Jak to lip vyladit pro mobily, aby byl text citelny a obrazek neprekazel nebo nebyl zdeformovany? Mozna by ten obrazek mohl byt mensi nebo uplne schovany na nejmensich mobilech?

Máš pravdu, responzivita hero sekce může být ošemetná, zvlášť když kombinuješ text a obrázek vedle sebe a pak je skládáš pod sebe.

Tady je několik nápadů a úprav, jak to vylepšit pro mobilní zařízení:

**1. Úprava min-height a padding v Hero sekci pro mobily:**

Na menších obrazovkách často nepotřebuješ, aby hero sekce zabírala 90% výšky viewportu (min-height: 90vh). Místo toho je lepší použít padding a nechat výšku přizpůsobit obsahu. To už částečně děláš v @media (max-width: 992px) s padding: 40px 0;. Můžeš zvážit úplné odstranění min-height pro mobilní verze, pokud padding stačí.

**2. Velikost obrázku a jeho kontejneru:**

* **.heroImageContainer**:
  + V @media (max-width: 992px) máš flex: 0 0 auto; a max-width: 400px;. To je dobrý začátek.
  + Můžeš zkusit pro menší breakpointy (např. @media (max-width: 768px) nebo @media (max-width: 576px)) ještě více zmenšit max-width pro .heroImageContainer, nebo dokonce nastavit width: 80%; (nebo podobnou procentuální hodnotu), aby se obrázek lépe přizpůsobil šířce obrazovky a nebyl příliš dominantní.
* **.heroImage**:
  + Ujisti se, že max-width: 100%; a height: auto; zůstávají, aby si obrázek zachoval poměr stran a nepřetékal ze svého kontejneru.

**3. Úprava textového obsahu:**

* **.heroTextContent**:
  + Na mobilu (@media (max-width: 992px)) už máš max-width: 100%; a padding-right: 0;.
  + Odsazení padding-top: 40px; dává smysl, když je obrázek nahoře.
* **Velikosti písma:** Zmenšení font-size pro .heroTitle a .heroText na menších obrazovkách je klíčové. To už děláš.

**4. Možnost skrýt obrázek na velmi malých obrazovkách:**

Pokud je na nejmenších obrazovkách (např. pod 480px) stále málo místa, můžeš obrázek úplně skrýt.

@media (max-width: 480px) { /\* Nebo jiný breakpoint podle potřeby \*/

.heroImageContainer {

display: none; /\* Skryje celý kontejner obrázku \*/

}

.heroTextContent {

padding-top: 0; /\* Pokud obrázek není, není potřeba odsazení shora \*/

}

.heroContent {

/\* Můžeš upravit padding, pokud se odstraní obrázek \*/

padding: 40px 20px;

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Konkrétní návrhy úprav pro index.module.css:**

/\* ... (tvoje stávající styly) ... \*/

@media (max-width: 992px) {

.heroSection {

min-height: auto; /\* Nech výšku určit obsahem a paddingem \*/

padding: 60px 0; /\* Můžeš padding i zvětšit, pokud je potřeba více místa \*/

}

.heroContent {

flex-direction: column-reverse;

text-align: center;

padding: 30px 20px; /\* Zmenšení paddingu pro konzistenci s menšími obrazovkami \*/

}

.heroTextContent {

max-width: 100%;

padding-right: 0;

padding-top: 40px; /\* Odsazení od obrázku, který je teď nad textem \*/

}

.heroImageContainer {

flex-basis: auto; /\* Místo flex: 0 0 auto; \*/

width: 70%; /\* Například, aby nebyl přes celou šířku, ale přizpůsobil se \*/

max-width: 380px; /\* Omez maximální velikost \*/

margin: 0 auto 30px auto; /\* Vycentrování a mezera pod \*/

justify-content: center;

}

.heroTitle {

font-size: 3.5rem; /\* Už máš, jen pro kontext \*/

}

.heroText {

font-size: 1.1rem; /\* Možná bude potřeba také zmenšit \*/

max-width: 500px; /\* Omezit šířku textu i na mobilu, když je na střed \*/

margin-left: auto;

margin-right: auto;

}

}

@media (max-width: 768px) {

.heroSection {

padding: 50px 0;

}

.heroImageContainer {

width: 80%;

max-width: 320px; /\* Další zmenšení \*/

margin-bottom: 25px;

}

.heroTitle {

font-size: 3rem; /\* Další zmenšení \*/

}

.heroTagline { /\* Pokud máš tagline, uprav i ten \*/

font-size: 1rem;

}

.heroText {

font-size: 1rem;

}

.heroButtons { /\* Pokud máš tlačítka \*/

flex-direction: column; /\* Tlačítka pod sebou \*/

gap: 15px;

align-items: center; /\* Centrování tlačítek \*/

}

.primaryButton, .secondaryButton { /\* Styly pro tlačítka, pokud je máš \*/

width: 80%;

max-width: 300px;

}

}

@media (max-width: 480px) {

.heroSection {

padding: 40px 0;

}

.heroImageContainer {

/\* Pokud se rozhodneš skrýt: \*/

display: none;

}

.heroTextContent {

padding-top: 0; /\* Není potřeba, když je obrázek skrytý \*/

}

.heroTitle {

font-size: 2.5rem;

}

.heroButtons {

margin-bottom: 20px; /\* Menší odsazení tlačítek \*/

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Co zkontrolovat a vyzkoušet:**

1. **Testuj na různých šířkách:** Použij vývojářské nástroje v prohlížeči (režim responzivního designu) k testování na různých šířkách.
2. **Výška sekce:** Sleduj, jak se chová výška .heroSection po odstranění min-height a použití padding.
3. **Obrázek:** Pozoruj, zda se obrázek správně zmenšuje a zda si zachovává poměr stran. Případně uprav width a max-width pro .heroImageContainer.
4. **Čitelnost textu:** Ujisti se, že text je stále dobře čitelný a má dostatečné odsazení.
5. **Tlačítka a další prvky:** Nezapomeň na responzivitu tlačítek nebo jiných prvků v hero sekci.

Tímto bys měl dosáhnout mnohem lepšího zobrazení hero sekce na mobilních zařízeních.

Je to jenom zlomek vteřiny, ale je to vidět a nevypadá to dobře. Jako by se nejdřív načetla defaultní hodnota darkMode (což je false v useState(false)) a až pak useEffect zjistil, jaký motiv má být skutečně nastaven.

Tady je ten ThemeToggle.tsx ještě jednou:

import React, { useState, useEffect } from 'react';

import styles from './ThemeToggle.module.css';

import { BsSun, BsMoonStarsFill } from 'react-icons/bs';

const ThemeToggle: React.FC = () => {

const [darkMode, setDarkMode] = useState(false); // Defaultně je light mode

useEffect(() => {

// Tento useEffect se spustí až po prvním renderu na klientovi

const storedTheme = localStorage.getItem('theme');

if (storedTheme === 'dark') {

setDarkMode(true);

} else if (storedTheme === 'light') {

setDarkMode(false);

} else {

// Pokud nic není v localStorage, zkusí systémové nastavení

const prefersDarkMode = window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches;

setDarkMode(prefersDarkMode);

}

}, []); // Prázdné pole závislostí, spustí se jen jednou po mountnutí

useEffect(() => {

// Tento useEffect aplikuje třídu na body a ukládá do localStorage

localStorage.setItem('theme', darkMode ? 'dark' : 'light');

if (darkMode) {

document.body.classList.add('dark-mode');

} else {

document.body.classList.remove('dark-mode');

}

}, [darkMode]); // Spustí se vždy, když se změní stav darkMode

const toggleDarkMode = () => {

setDarkMode(!darkMode);

};

return (

<button className={styles.themeToggleButton} onClick={toggleDarkMode}>

{/\* Ikona se renderuje na základě počátečního stavu 'darkMode', než se useEffect stihne spustit \*/}

{darkMode ? <BsSun size="20" /> : <BsMoonStarsFill size="20" />}

</button>

);

};

export default ThemeToggle;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

Jak můžu tenhle "flash" špatné ikony (a potenciálně i motivu, i když to je asi rychlejší) odstranit? Chci, aby se hned při prvním načtení stránky zobrazila správná ikona a motiv bez toho probliknutí.

Máš naprostou pravdu, tohle je klasický problém známý jako FOUC (Flash Of Unstyled Content) nebo v tomto případě spíše "Flash of incorrect theme". Je to způsobeno tím, že:

1. React komponenta se nejprve renderuje s počátečním stavem (darkMode je false).
2. Až **poté**, co se komponenta "připojí" (mountne) do DOMu na straně klienta, se spustí první useEffect, který zkontroluje localStorage a systémové preference a případně aktualizuje stav darkMode.
3. Tato aktualizace stavu způsobí nový render, tentokrát se správnou ikonou.

Mezi krokem 1 a 3 je ten krátký okamžik, kdy vidíš nesprávný stav.

**Řešení pro Next.js (a podobné SSR/SSG frameworky):**

Protože Next.js může stránky renderovat na serveru (SSR) nebo generovat staticky (SSG), localStorage a window.matchMedia nejsou na serveru dostupné. Musíme tedy počáteční motiv nastavit co nejdříve na klientovi, ideálně ještě před prvním plnohodnotným renderem Reactu, nebo alespoň minimalizovat dopad.

**Možnost 1: Skript v <Head> pro nastavení třídy před hydratací (doporučeno pro zamezení FOUC u stylů)**

Toto je nejrobustnější způsob, jak zabránit probliknutí celého motivu (nejen ikony). Vložíš malý skript přímo do <Head> tvého \_document.tsx (nebo layout.tsx pro App Router, pokud používáš <html> a <body> tagy tam). Tento skript se vykoná ještě předtím, než React vůbec začne "hydratovat" stránku.

Uprav \_document.tsx (pokud používáš Pages Router):

// pages/\_document.tsx

import { Html, Head, Main, NextScript } from 'next/document'

export default function Document() {

const setInitialTheme = `

function getUserPreference() {

if(window.localStorage.getItem('theme')) {

return window.localStorage.getItem('theme')

}

return window.matchMedia('(prefers-color-scheme: dark)').matches

? 'dark'

: 'light'

}

document.body.classList.add(getUserPreference());

`;

return (

<Html>

<Head>

{/\* Jiné meta tagy a linky \*/}

</Head>

<body>

<script dangerouslySetInnerHTML={{ \_\_html: setInitialTheme }} />

<Main />

<NextScript />

</body>

</Html>

)

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

*Nebo pro App Router v app/layout.tsx:*

// app/layout.tsx

export default function RootLayout({ children }: { children: React.ReactNode }) {

const setInitialTheme = `

// ... (stejný skript jako výše) ...

`;

return (

<html lang="en">

<head />

<body>

<script dangerouslySetInnerHTML={{ \_\_html: setInitialTheme }} />

{children}

</body>

</html>

);

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Co dělá tento skript:**

* Definuje funkci getUserPreference, která zkontroluje localStorage a pak systémové preference.
* Okamžitě přidá třídu (dark nebo light) na <body>. To zajistí, že CSS proměnné z globals.css (pokud je máš správně nastavené s body.dark a body.light nebo jen body.dark a default pro light) se aplikují ještě před Reactem.

**Úprava ThemeToggle.tsx pro synchronizaci s tímto skriptem:**

Teď tvůj ThemeToggle.tsx nemusí nastavovat třídu dark-mode sám, ale musí správně číst počáteční stav a také inicializovat darkMode stav tak, aby odpovídal třídě na body, kterou nastavil skript.

import React, { useState, useEffect, useCallback } from 'react';

import styles from './ThemeToggle.module.css';

import { BsSun, BsMoonStarsFill } from 'react-icons/bs';

const ThemeToggle: React.FC = () => {

// Funkce pro získání počátečního stavu, která může běžet na klientovi

const getInitialMode = () => {

if (typeof window !== 'undefined') { // Ujistíme se, že jsme na klientovi

const storedTheme = localStorage.getItem('theme');

if (storedTheme) {

return storedTheme === 'dark';

}

// Kontrolujeme třídu na body, kterou mohl nastavit inline skript

if (document.body.classList.contains('dark')) { // Předpokládáme, že inline skript přidává 'dark' nebo 'light'

return true;

}

if (document.body.classList.contains('light')) {

return false;

}

// Fallback na systémové preference, pokud třída není (měla by být)

return window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches;

}

return false; // Výchozí pro SSR/build time

};

const [darkMode, setDarkMode] = useState(getInitialMode());

// Tento useEffect už nebude potřeba pro nastavení třídy na body, pokud používáš inline skript

// useEffect(() => {

// if (darkMode) {

// document.body.classList.add('dark-mode'); // nebo jen 'dark'

// document.body.classList.remove('light-mode'); // nebo jen 'light'

// } else {

// document.body.classList.remove('dark-mode');

// document.body.classList.add('light-mode');

// }

// localStorage.setItem('theme', darkMode ? 'dark' : 'light');

// }, [darkMode]);

// Místo toho jen useEffect pro localStorage a případnou změnu třídy, pokud inline skript nepokrývá vše

useEffect(() => {

const newTheme = darkMode ? 'dark' : 'light';

localStorage.setItem('theme', newTheme);

// Ujisti se, že třída na body je konzistentní s tím, co nastavuje inline skript

document.body.classList.remove(darkMode ? 'light' : 'dark');

document.body.classList.add(newTheme);

}, [darkMode]);

// Tento useEffect pro čtení z localStorage při mountu už není tak kritický,

// pokud getInitialMode() funguje správně, ale může sloužit jako pojistka

// nebo pro aktualizaci, pokud by se localStorage změnilo v jiném tabu.

// Můžeme ho zjednodušit nebo zvážit jeho potřebu.

useEffect(() => {

const handleStorageChange = () => {

const storedTheme = localStorage.getItem('theme');

if (storedTheme) {

setDarkMode(storedTheme === 'dark');

}

};

// Aktualizace, pokud se motiv změnil v localStorage (např. v jiném tabu)

window.addEventListener('storage', handleStorageChange);

// Také zkontroluj při mountu, zda se nezměnilo od doby, kdy `getInitialMode` běželo

handleStorageChange();

return () => {

window.removeEventListener('storage', handleStorageChange);

};

}, []);

const toggleDarkMode = () => {

setDarkMode(prevMode => !prevMode);

};

// Stav pro "mounted", aby se ikona zobrazila až po hydrataci a správném nastavení darkMode

const [mounted, setMounted] = useState(false);

useEffect(() => {

setMounted(true);

}, []);

if (!mounted) {

// Můžeš vrátit null nebo placeholder, dokud není komponenta plně mountnutá a stav darkMode není jistý

// To pomůže zabránit probliknutí nesprávné ikony

return <button className={styles.themeToggleButton} style={{ opacity: 0 }} disabled aria-label="Načítání motivu"></button>;

}

return (

<button className={styles.themeToggleButton} onClick={toggleDarkMode} aria-label={darkMode ? "Přepnout na světlý režim" : "Přepnout na tmavý režim"}>

{darkMode ? <BsSun size="20" /> : <BsMoonStarsFill size="20" />}

</button>

);

};

export default ThemeToggle;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

A v globals.css bys měl mít:

/\* globals.css \*/

:root { /\* Toto je default (light) \*/

--color-background: #fff;

--color-text-primary: #333;

/\* ... další light proměnné ... \*/

}

body.dark { /\* Použij .dark místo .dark-mode, aby to odpovídalo skriptu \*/

--color-background: #121212;

--color-text-primary: #eee;

/\* ... další dark proměnné ... \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Všimni si body.dark (místo body.dark-mode), aby to odpovídalo tomu, co nastavuje inline skript. Můžeš si to samozřejmě pojmenovat, jak chceš, ale musí to být konzistentní.

**Vysvětlení úprav v ThemeToggle.tsx:**

* getInitialMode(): Funkce, která se snaží co nejlépe určit počáteční stav na klientovi.
* useState(getInitialMode()): Inicializuje stav pomocí této funkce.
* useEffect pro aplikaci třídy na body je upraven, aby byl konzistentní.
* mounted stav: Zobrazí ikonu až po prvním useEffect na klientovi, což zajistí, že darkMode stav je již synchronizován. Během serverového renderu a těsně po něm (než se mounted nastaví na true) se tlačítko buď nerenderuje, nebo je neviditelné/neaktivní.

**Možnost 2: Použití knihovny jako next-themes**

Knihovna next-themes je navržena přesně pro řešení těchto problémů v Next.js. Stará se o synchronizaci s localStorage, systémovými preferencemi a správné nastavení třídy na <html> element (což je často lepší než na <body>), a také řeší FOUC.

Pokud bys chtěl jednodušší implementaci bez psaní vlastního skriptu a logiky, next-themes je skvělá volba.

**Klíčové:**  
Nejdůležitější je, aby třída (dark nebo light) byla na <body> (nebo <html>) co nejdříve. Inline skript v \_document.tsx (nebo layout.tsx) je pro to nejlepší. Komponenta ThemeToggle pak musí tento počáteční stav respektovat. Trik s mounted stavem pomáhá skrýt ikonu, dokud

není stav plně synchronizován.

tak jsem upravoval ten Footer.tsx a chtěl jsem přidat takovou malou oddělovací čáru nad copyright textem, aby to bylo vizuálně hezčí. Přidal jsem tam obyčejný <hr /> tag, ale teď mi Visual Studio Code řve chybu přímo v tom souboru a stránka se ani nezkompiluje.

Chyba vypadá takhle, když najedu na ten podtržený <hr />:

Property 'stlye' does not exist on type 'DetailedHTMLProps<HTMLHRElement, HTMLHRElement>'. Did you mean 'style'?ts(2551)

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).

A tady je ten upravený kus kódu v Footer.tsx v části .footerCopyright, kde jsem to přidal:

// ... ostatní části Footer.tsx ...

<div className={styles.footerColumn}>

<h3 className={styles.footerTitle}>Kontakt</h3>

<address className={styles.footerAddress}>

{/\* ... adresa ... \*/}

</address>

{/\* ZDE JSEM PŘIDAL HR A UPRAVIL DIV \*/}

<div className={styles.footerBottomSection}> {/\* Přidal jsem nový div pro lepší strukturu \*/}

<hr stlye={{ borderTop: '1px solid rgba(255, 255, 255, 0.1)', margin: '20px 0' }} /> {/\* TADY JE TA CHYBA \*/}

<div className={styles.footerCopyright}>

© {currentYear} Fitness Coaching - Všechna práva vyhrazena

</div>

</div>

</div>

</div>

</div>

</footer>

);

};

export default Footer;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

V CSS (Footer.module.css) jsem pro .footerCopyright měl:

.footerCopyright {

color: rgba(255, 255, 255, 0.5);

margin-top: 30px;

font-size: 0.9rem;

border-top: 1px solid rgba(255, 255, 255, 0.05); /\* TOTO jsem chtěl nahradit <hr> \*/

padding-top: 20px;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Ten border-top z .footerCopyright jsem chtěl odstranit a nahradit ho tím <hr /> pro větší flexibilitu se stylováním. Co je špatně s tím stlye atributem u <hr />?

Ta chyba Property 'stlye' does not exist on type 'DetailedHTMLProps<HTMLHRElement, HTMLHRElement>'. Did you mean 'style'?ts(2551) ti přesně říká, co je špatně.

Máš tam překlep v názvu atributu pro inline styly. Místo stlye má být správně style.

**Oprava v Footer.tsx:**

Stačí opravit ten překlep v atributu u <hr /> elementu:

// ... ostatní části Footer.tsx ...

<div className={styles.footerColumn}>

<h3 className={styles.footerTitle}>Kontakt</h3>

<address className={styles.footerAddress}>

{/\* ... adresa ... \*/}

</address>

<div className={styles.footerBottomSection}>

{/\* OPRAVA ZDE: stlye -> style \*/}

<hr style={{ border: 'none', borderTop: '1px solid rgba(255, 255, 255, 0.1)', marginTop: '30px', marginBottom: '20px' }} />

<div className={styles.footerCopyright}>

© {currentYear} Fitness Coaching - Všechna práva vyhrazena

</div>

</div>

</div>

</div>

</div>

</footer>

);

};

export default Footer;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Vysvětlení a další doporučení pro <hr />:**

1. **Překlep stlye vs style**: Jak chyba napovídá, správný název atributu pro inline styly v JSX (a HTML) je style.
2. **Stylování <hr />**:
   * Element <hr /> má často výchozí styly od prohlížeče (např. 3D efekt, vlastní border). Pokud chceš mít plnou kontrolu nad jeho vzhledem, je dobré tyto výchozí styly resetovat.
   * V inline stylu jsem přidal border: 'none' pro resetování výchozího ohraničení a pak explicitně nastavil borderTop.
   * margin: '20px 0' jsem rozdělil na marginTop a marginBottom pro větší explicitnost v JSX stylu. V JSX se CSS vlastnosti píšou camelCase (např. borderTop, marginTop).
3. **CSS třída vs. inline styl**: Pro složitější stylování nebo pro znovupoužitelnost je obecně lepší vytvořit si CSS třídu v Footer.module.css pro <hr /> než používat dlouhé inline styly.

Například v Footer.module.css:

.footerSeparator {

border: none; /\* Reset výchozího borderu \*/

border-top: 1px solid rgba(255, 255, 255, 0.1); /\* Tvoje linka \*/

margin-top: 30px;

margin-bottom: 20px;

width: 100%; /\* Aby byla přes celou šířku sloupce \*/

}

.footerCopyright {

color: rgba(255, 255, 255, 0.5);

/\* margin-top: 30px; -- Odsazení už řeší hr \*/

font-size: 0.9rem;

/\* border-top: 1px solid rgba(255, 255, 255, 0.05); -- Odebráno \*/

/\* padding-top: 20px; -- Odsazení už řeší hr \*/

text-align: center; /\* Často chceme copyright na střed \*/

}

.footerBottomSection {

/\* Tato třída je v pořádku, pokud ji potřebuješ pro další účely \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

A v Footer.tsx:

<div className={styles.footerBottomSection}>

<hr className={styles.footerSeparator} />

<div className={styles.footerCopyright}>

© {currentYear} Fitness Coaching - Všechna práva vyhrazena

</div>

</div>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Tímto způsobem udržuješ styly oddělené v CSS souboru.

Po opravě překlepu na style by chyba měla zmizet a tvůj <hr /> by se měl zobrazit s tvými inline styly. Doporučuji zvážit přesun stylů do CSS modulu pro lepší organizaci.

Taky to tlačítko .serviceButton bych chtěl mít vždycky na spodku karty, i když je popis krátký. Teď když je popis krátký, tak je tlačítko hned pod ním a pod tlačítkem je prázdné místo (pokud ostatní karty v řádku jsou vyšší).

Tady je struktura jedné karty v index.tsx:

<div className={styles.serviceCard}>

<div className={styles.serviceIconContainer}>

<span className={styles.serviceIconPlaceholder}>💪</span>

</div>

<h3 className={styles.serviceTitle}>COACHING+</h3>

<p className={styles.serviceDescription}>

Obsahuje 1 společný trénink měsíčně, jídelníček na míru, tréninkový plán na míru, video vysvětlivky cviků, 2x týdně kontroly a WhatsApp chat. Tohle je delší popis pro testování.

</p>

<div className={styles.servicePriceContainer}>

<span className={styles.servicePrice}>2500 Kč</span>

<span className={styles.servicePeriod}>/ měsíc</span>

</div>

<a href="/kontakt" className={styles.serviceButton}>

MÁM ZÁJEM

</a>

</div>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A relevantní CSS z index.module.css:

.servicesGrid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 30px;

/\* ... \*/

}

.serviceCard {

background-color: var(--color-secondary-background);

border-radius: 12px;

padding: 25px;

box-shadow: 0 4px 12px var(--color-box-shadow);

display: flex;

flex-direction: column;

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.serviceDescription {

font-size: 0.95rem;

color: var(--color-text-secondary);

line-height: 1.6;

margin-bottom: 20px;

flex-grow: 1; /\* Tohle už tam mám, ale asi to nestačí pro grid? \*/

}

.serviceButton {

display: inline-block;

/\* ... styly tlačítka ... \*/

margin-top: auto; /\* Tohle taky, ale asi nefunguje jak má v gridu \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Ten flex-grow: 1 na popisu a margin-top: auto na tlačítku mi fungovaly, když jsem měl karty vedle sebe pomocí flexboxu, ale teď když používám display: grid na .servicesGrid, tak to se stejnou výškou karet nefunguje. Jak to opravit, aby karty v řádku měly stejnou výšku a tlačítko bylo vždy dole?

Máš pravdu, flex-grow: 1 na potomkovi (.serviceDescription) a margin-top: auto na jiném potomkovi (.serviceButton) fungují skvěle pro zarovnání uvnitř *jednoho flex kontejneru* (.serviceCard, který má display: flex; flex-direction: column;). To ti zajistí, že tlačítko je na spodku *své vlastní karty*.

Problém, který popisuješ – že karty v rámci jednoho řádku v CSS Gridu nemají stejnou výšku – je způsoben tím, jak CSS Grid standardně zachází s výškou svých položek. Každá položka gridu (v tvém případě .serviceCard) si standardně vezme jen tolik výšky, kolik potřebuje její obsah.

**Řešení pro stejnou výšku karet v CSS Gridu:**

Pokud chceš, aby všechny karty v jednom řádku gridu měly stejnou výšku (výšku nejvyšší karty v daném řádku), CSS Grid sám o sobě toto přímo neřeší tak jednoduše jako Flexbox s align-items: stretch. Máš ale několik možností:

**1. Nechat .serviceCard být flex kontejnerem (což už je) a zajistit, že se roztáhne:**  
Když je .servicesGrid definován jako display: grid, jeho přímí potomci (.serviceCard) se standardně výškově nepřizpůsobují ostatním položkám v řádku.  
Ale protože .serviceCard je už display: flex; flex-direction: column;, vnitřní logika s flex-grow: 1 na .serviceDescription a margin-top: auto na .serviceButton bude fungovat správně pro *každou jednotlivou kartu*. Potřebujeme jen zajistit, aby se všechny karty v řádku natáhly na stejnou výšku.

CSS Grid má vlastnost grid-auto-rows s hodnotou 1fr, která by mohla pomoci, ale ta by udělala všechny řádky stejně vysoké, což nemusí být žádoucí, pokud máš v různých řádcích různě vysoké karty.

Častým řešením je **nastavit explicitní výšku pro řádky gridu nebo nechat výchozí chování a spolehnout se na vnitřní strukturu karet.**

Ve tvém případě už máš .serviceCard jako flex kontejner. Problém tedy není v tom, že by se tlačítko neposouvalo na spodek *své* karty, ale v tom, že karty samotné nejsou stejně vysoké.

**Nejjednodušší přístup pro moderní prohlížeče (pokud to podporují všechny tvé cílové):**  
CSS Grid sám o sobě nezaručí stejnou výšku položek v řádku, pokud mají různý obsah. Ale protože tvůj .serviceCard je už flex kontejner, vnitřní zarovnání by mělo fungovat, jakmile karta dosáhne své maximální výšky v daném řádku.

Můžeš zkusit zajistit, aby se .serviceCard sám o sobě roztáhl na výšku buňky gridu, pokud by to pomohlo. Standardně by to měl dělat.

**Co je pravděpodobně problém a jak to řešit:**

flex-grow: 1 na .serviceDescription a margin-top: auto na .serviceButton **JSOU SPRÁVNĚ** pro to, aby se tlačítko zarovnalo na spodek *každé jednotlivé karty*.

Problém "karty v řádku nemají stejnou výšku" je oddělený.

* **Pokud chceš, aby se všechny karty v gridu natáhly na výšku nejvyšší karty v celém gridu:** To je složitější a často vyžaduje JavaScript nebo velmi specifické CSS (např. grid-template-rows: masonry; což je experimentální).
* **Pokud chceš, aby se karty v *každém jednotlivém řádku* natáhly na výšku nejvyšší karty v *tomto konkrétním řádku*:**  
  CSS Grid ve výchozím nastavení toto sám od sebe nedělá. Položky gridu se roztahují na výšku svého obsahu.

**Nejčastější a nejrobustnější řešení, když používáš Grid pro layout a Flexbox uvnitř položek:**  
Nechat Grid, aby si určil rozložení karet. Každá karta (.serviceCard) je display: flex; flex-direction: column;. Díky tomu:

* 1. h3 (.serviceTitle) zabere svou výšku.
  2. p (.serviceDescription) s flex-grow: 1; se pokusí zabrat všechen zbývající volný vertikální prostor uvnitř karty.
  3. div (.servicePriceContainer) zabere svou výšku.
  4. a (.serviceButton) s margin-top: auto; se "odtlačí" na úplný spodek karty.

**Zkontroluj následující:**

* 1. Ujisti se, že žádný prvek uvnitř .serviceCard nemá nastavenou pevnou výšku, která by bránila .serviceDescription v růstu.
  2. CSS Grid (.servicesGrid) by měl automaticky vytvářet řádky. Pokud máš hodně karet, vytvoří se více řádků. Výška karet v *různých* řádcích může být různá.

**Problém může být, že flex-grow: 1 na .serviceDescription a margin-top: auto na .serviceButton spolupracují na tom, aby vyplnily výšku samotné karty, ale pokud karty samotné nejsou stejně vysoké kvůli Gridu, tak to stále bude vypadat nestejně.**

**Možné řešení pro stejnou výšku karet v řádku Gridu (pokud výše uvedené nestačí):**

Pokud chceš vynutit stejnou výšku pro všechny karty v rámci jednoho implicitního řádku gridu, můžeš zkusit explicitně nastavit, aby se položky gridu vertikálně roztahovaly. Standardně je align-items: stretch (pro blokovou osu, což je zde vertikální směr implicitních řádků), což by mělo fungovat.

**Upřímně, tvůj současný CSS pro .serviceCard, .serviceDescription a .serviceButton by MĚL fungovat tak, jak chceš (tlačítko na spodku každé karty).** Problém nestejné výšky karet v řádku je vlastnost Gridu.

**Zkusme se zaměřit na to, aby KAŽDÁ KARTA měla tlačítko dole, bez ohledu na výšku ostatních karet:**  
Tohle už máš vyřešené pomocí:

.serviceCard {

display: flex;

flex-direction: column;

/\* ... \*/

}

.serviceDescription {

/\* ... \*/

flex-grow: 1;

}

.serviceButton {

/\* ... \*/

margin-top: auto;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Toto je správně. Pokud to takto nefunguje, problém bude jinde.

**Pokud trváš na tom, aby všechny karty v JEDNOM ŘÁDKU měly stejnou výšku:**  
Toto je trik. CSS Grid sám o sobě ti výšku položek v řádku nesjednotí, pokud je jejich obsah různě vysoký.  
Mohl bys zkusit:

1. **Nastavit align-items: stretch; na .servicesGrid (což je výchozí, ale pro jistotu).** To by mělo roztáhnout položky gridu (karty) na výšku nejvyšší položky v jejich oblasti.
2. Ujistit se, že .serviceCard (jako položka gridu) se může vertikálně roztáhnout.

**Reálnější scénář a ověření:**  
Vytvoř si několik karet s různě dlouhými popisy.

1. Ověř, zda tlačítko v KAŽDÉ kartě je na jejím spodku. Pokud ano, tvůj flexbox uvnitř karty funguje.
2. Pokud karty v jednom řádku nemají stejnou výšku, je to vlastnost gridu. Pro dosažení stejné výšky bys musel buď:
   * Nastavit pevnou minimální výšku pro .serviceCard (což není ideální pro responzivitu).
   * Použít JavaScript k sjednocení výšek po renderu (což je overkill).
   * Nebo akceptovat, že v CSS Gridu mohou mít položky s různým obsahem různou výšku, pokud to explicitně neřešíš (např. pomocí grid-auto-rows: 1fr; by všechny řádky měly stejnou výšku, a tedy i karty v nich, ale to roztáhne i ty řádky s kratšími kartami).

**Doporučení:**  
Soustřeď se na to, aby **každá karta individuálně** měla tlačítko dole pomocí tvé stávající flexbox logiky. To je nejdůležitější.

Pokud se ti nelíbí, že karty v řádku Gridu nejsou stejně vysoké:  
V index.module.css přidej k .servicesGrid:

.servicesGrid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 30px;

align-items: stretch; /\* Toto je výchozí, ale pro jistotu \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

A ujisti se, že .serviceCard nemá nic, co by mu bránilo se vertikálně roztáhnout (např. pevná výška).  
Tvoje stávající nastavení display: flex; flex-direction: column; pro .serviceCard a flex-grow:1 pro .serviceDescription by mělo zajistit, že když se karta natáhne, popis vyplní prostor a tlačítko zůstane dole.

Problém může být v tom, že flex-grow se uplatní až *poté*, co se určí výška samotné karty gridem.

**Možná je problém v tom, že .serviceCard jako položka gridu se nenatahuje.**  
Zkus explicitně nastavit, aby se položky gridu chovaly jako flex kontejnery, které se natáhnou:

.serviceCard {

background-color: var(--color-secondary-background);

border-radius: 12px;

padding: 25px;

box-shadow: 0 4px 12px var(--color-box-shadow);

display: flex; /\* Důležité: Karta je flex kontejner \*/

flex-direction: column; /\* Obsah karty jde pod sebe \*/

/\* align-self: stretch; Toto by mohlo pomoci, pokud by to align-items na gridu nestačilo \*/

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Klíčem je, že .serviceCard (položka gridu) se musí vertikálně natáhnout, aby vyplnila buňku gridu. A zároveň musí být flex kontejnerem, aby jeho vnitřní prvky (.serviceDescription a .serviceButton) mohly využít flex-grow a margin-top: auto.

Pokud by i toto selhalo, znamená to, že implicitní řádky gridu nemají jednotnou výšku, a pak je to složitější CSS problém.

Tady je aktuální kód formuláře v kontakt.tsx:

// ... (importy a začátek komponenty) ...

<form className={styles.contactForm} onSubmit={handleSubmit}>

{/\* Chyby a úspěch se zobrazují zde \*/}

{submitStatus?.success === false && (

<div className={styles.errorMessage}><p>{submitStatus.message}</p></div>

)}

<div className={styles.formColumns}>

<div className={styles.formColumn}>

<div className={styles.formGroup}>

<label htmlFor="name">Jméno a Příjmení\*</label>

<input type="text" id="name" name="name" required className={styles.formInput} value={formData.name} onChange={handleChange} placeholder="Vaše jméno" />

</div>

{/\* ... další inputy v prvním sloupci (email, phone, occupation) ... \*/}

<div className={styles.formGroup}>

<label htmlFor="email">E-mail\*</label>

<input type="email" id="email" name="email" required className={styles.formInput} value={formData.email} onChange={handleChange} placeholder="vas@email.cz"/>

</div>

<div className={styles.formGroup}>

<label htmlFor="phone">Telefonní číslo</label>

<input type="tel" id="phone" name="phone" className={styles.formInput} value={formData.phone} onChange={handleChange} placeholder="+420 XXX XXX XXX"/>

</div>

<div className={styles.formGroup}>

<label htmlFor="occupation">Povolání / Studium</label>

<select id="occupation" name="occupation" className={styles.formSelect} value={formData.occupation} onChange={handleChange}>

<option value="">-- Vyberte možnost --</option>

{/\* ... options ... \*/}

</select>

</div>

</div>

<div className={styles.formColumn}>

<div className={styles.formRow}>

<div className={styles.formGroup}>

<label htmlFor="age">Věk</label>

<input type="number" id="age" name="age" className={styles.formInput} value={formData.age} onChange={handleChange} placeholder="25"/>

</div>

{/\* ... další inputy v řádku (height, weight) ... \*/}

<div className={styles.formGroup}>

<label htmlFor="height">Výška (cm)</label>

<input type="number" id="height" name="height" className={styles.formInput} value={formData.height} onChange={handleChange} placeholder="175"/>

</div>

<div className={styles.formGroup}>

<label htmlFor="weight">Hmotnost (kg)</label>

<input type="number" id="weight" name="weight" step="0.1" className={styles.formInput} value={formData.weight} onChange={handleChange} placeholder="70"/>

</div>

</div>

<div className={styles.formGroup}>

<label htmlFor="sportFrequency">Fyzická aktivita / Jak často, jak dlouho? Sporty?</label>

<textarea id="sportFrequency" name="sportFrequency" rows={2} className={styles.formTextarea} value={formData.sportFrequency} onChange={handleChange} placeholder="Popište vaše současné fyzické aktivity..."></textarea>

</div>

<div className={styles.formGroup}>

<label htmlFor="service">Vyberte službu, která Vás zajímá\*</label>

<select id="service" name="service" required className={styles.formSelect} value={formData.service} onChange={handleChange}>

<option value="">-- Vyberte službu --</option>

{/\* ... options ... \*/}

</select>

</div>

</div>

</div>

<div className={styles.formFullWidth}>

{/\* ... textarea pro cíl, preference jídla, zprávu ... \*/}

<div className={styles.formGroup}>

<label htmlFor="goal">Jaké máš se svojí postavou ambice, co bys s ní chtěl dělat?\*</label>

<textarea id="goal" name="goal" rows={3} required className={styles.formTextarea} value={formData.goal} onChange={handleChange} placeholder="Popište vaše cíle a očekávání..."></textarea>

</div>

<div className={styles.formGroup}>

<label htmlFor="foodPreferences">Jaké máš stravovací návyky? Alergie? Co ti chutná/nechutná?</label>

<textarea id="foodPreferences" name="foodPreferences" rows={3} className={styles.formTextarea} value={formData.foodPreferences} onChange={handleChange} placeholder="Popište vaše stravovací návyky, preference a případné alergie..."></textarea>

</div>

<div className={styles.formGroup}>

<label htmlFor="message">Zpráva / Upřesnění</label>

<textarea id="message" name="message" rows={3} className={styles.formTextarea} value={formData.message} onChange={handleChange} placeholder="Doplňující informace nebo dotazy..."></textarea>

</div>

<div className={styles.formGroup}>

<div className={styles.checkboxGroup}>

<input type="checkbox" id="privacyConsent" name="privacyConsent" required className={styles.formCheckbox} checked={formData.privacyConsent} onChange={handleChange}/>

<label htmlFor="privacyConsent" className={styles.privacyLabel}>

Souhlasím s{' '}

<Link href="/obchodni-podminky" className={styles.privacyLink}>

obchodními podmínkami a zásadami ochrany osobních údajů

</Link>.\*

</label>

</div>

</div>

<button type="submit" className={styles.submitButton} disabled={isSubmitting}>

{isSubmitting ? 'Odesílání...' : 'Odeslat formulář'}

</button>

</div>

</form>

// ... (konec komponenty) ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A relevantní CSS v kontakt.module.css:

.contactForm {

width: 100%;

background-color: var(--color-secondary-background);

border-radius: 12px;

padding: 40px;

box-shadow: 0 8px 25px rgba(0, 0, 0, 0.08);

}

.formColumns {

display: flex;

gap: 30px;

margin-bottom: 30px; /\* Původně bylo 0, přidal jsem mezeru \*/

}

.formColumn {

flex: 1;

display: flex;

flex-direction: column;

gap: 20px; /\* Mezera mezi formGroup uvnitř sloupce \*/

}

.formRow { /\* Použito pro věk, výšku, váhu \*/

display: flex;

flex-wrap: wrap; /\* wrap pro případ, že by se nevešly vedle sebe \*/

gap: 15px; /\* Mezera mezi inputy v řádku \*/

width: 100%;

}

.formRow .formGroup { /\* .formGroup uvnitř .formRow \*/

flex: 1; /\* Aby se rovnoměrně rozdělily \*/

min-width: calc(33.333% - 10px); /\* Snaha o 3 sloupce, pokud je místo \*/

}

.formFullWidth { /\* Pro textareu a checkbox \*/

display: flex;

flex-direction: column;

gap: 20px; /\* Mezera mezi formGroup \*/

}

.formGroup {

/\* margin-bottom: 0; Odebráno, mezery řeší gap na .formColumn, .formRow, .formFullWidth \*/

}

.formInput, .formSelect, .formTextarea {

/\* ... stávající styly ... \*/

background-color: var(--color-background); /\* Důležité pro kontrast v dark mode \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Co bych chtěl vylepšit:

1. **Placeholdery a popisky:** Možná nějaký efekt, když uživatel klikne do pole? Třeba aby popisek (label) nezmizel, ale zmenšil se a posunul nahoru?
2. **Validace polí:** Zatím mám jen required. Bylo by fajn mít nějakou vizuální indikaci, když pole není validní (např. červený okraj), ale až po tom, co se uživatel pokusí formulář odeslat nebo opustí pole (onBlur).
3. **Vzhled select elementu:** Ten výchozí vzhled selectu je nic moc. Dá se nějak lépe nastylovat, aby ladil se zbytkem formuláře?
4. **Odesílací tlačítko:** Možná nějakou ikonku k textu "Odeslat formulář"?

Můžeš mi k těmto bodům dát nějaké tipy a ukázky kódu (hlavně CSS, případně drobné úpravy JSX)?

Výborně, pojďme ten formulář posunout na další úroveň.

**1. Vylepšené popisky (Floating Labels):**

Efekt, kdy se popisek při focusu nebo vyplnění inputu zmenší a posune nahoru, je populární. Vyžaduje trochu HTML struktury a CSS.

* **Úprava JSX v kontakt.tsx (pro jeden formGroup jako příklad):**  
  Obal input a label do dalšího divu, kterému dáme position: relative. Label bude absolutně pozicován.
* <div className={styles.formGroup}> {/\* Tento div už máš \*/}
* <div className={styles.inputWrapper}> {/\* Nový obalovač \*/}
* <input
* type="text"
* id="name"
* name="name"
* required
* className={styles.formInput}
* value={formData.name}
* onChange={handleChange}
* placeholder=" " /\* Důležité: placeholder musí být (i mezera), aby fungoval :placeholder-shown selektor \*/
* />
* <label htmlFor="name" className={styles.formLabel}>Jméno a Příjmení\*</label>
* </div>

</div>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Tuto strukturu (.inputWrapper s input a label uvnitř) aplikuj na všechny formGroup, kde chceš tento efekt.

* **CSS v kontakt.module.css:**
* .inputWrapper {
* position: relative;
* }
* .formInput { /\* Tvoje stávající styly pro .formInput \*/
* /\* ... \*/
* padding: 18px 15px 6px 15px; /\* Uprav padding, aby bylo místo pro label nahoře \*/
* }
* .formLabel {
* position: absolute;
* left: 15px;
* top: 50%; /\* Původně na středu \*/
* transform: translateY(-50%);
* color: var(--color-text-secondary);
* background-color: var(--color-background); /\* Aby překrylo border inputu při posunu \*/
* padding: 0 5px;
* margin: 0; /\* Resetuj defaultní margin labelu \*/
* transition: all 0.2s ease-out;
* pointer-events: none; /\* Aby kliknutí prošlo na input \*/
* font-size: 1rem; /\* Původní velikost labelu \*/
* }
* /\* Když je input focusnutý NEBO když má hodnotu (není prázdný placeholder) \*/
* .formInput:focus + .formLabel,
* .formInput:not(:placeholder-shown) + .formLabel {
* top: 0px; /\* Posun nahoru nad border \*/
* transform: translateY(-50%) scale(0.85); /\* Zmenšení a posun \*/
* font-size: 0.80rem; /\* Zmenšená velikost písma \*/
* color: var(--color-primary); /\* Můžeš změnit barvu aktivního labelu \*/
* }
* /\* Pro select to bude trochu jiné, protože select nemá :placeholder-shown tak spolehlivě \*/
* /\* Pro select bych zvážil jednodušší přístup nebo JavaScript pro detekci hodnoty \*/
* .formSelect {
* padding: 12px 15px; /\* Select nepotřebuje tolik místa nahoře, pokud label zůstane nad ním \*/
* }
* /\* Pokud chceš floating label i pro select, budeš muset label nechat nad ním klasicky,

nebo použít JS pro přidání třídy, když je vybrána hodnota. \*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Poznámka k selectu:** Floating label pro <select> je komplikovanější, protože :placeholder-shown se na něj nechová stejně. Často se label nechává staticky nad ním, nebo se používá JavaScript. Pro jednoduchost bych u selectu nechal klasický label nad ním.

**2. Vizuální indikace validace:**

Můžeš použít CSS pseudotřídy :invalid a :valid po interakci. Pro lepší kontrolu (např. zobrazení chyb až po "touched" stavu) bys potřeboval JavaScript a stavy pro každé pole. Začneme s CSS.

* **CSS v kontakt.module.css:**
* .formInput, .formSelect, .formTextarea {
* /\* ... stávající styly ... \*/
* border: 1px solid var(--color-input-border); /\* Výchozí border \*/
* }
* /\* Styl pro input, který byl "dotknut" a je nevalidní \*/
* .formInput.touched:invalid,
* .formTextarea.touched:invalid {
* border-color: #e53e3e; /\* Červená pro chybu \*/
* box-shadow: 0 0 0 3px rgba(229, 62, 62, 0.2);
* }
* /\* Styl pro input, který je validní (volitelné) \*/
* /\*
* .formInput.touched:valid {
* border-color: #38a169; // Zelená pro úspěch
* }

\*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

* **V kontakt.tsx bys musel přidat logiku pro "touched" stav:**
  + Přidat do formData (nebo samostatného stavu) pro každé pole boolean touched (např. nameTouched: false).
  + Nastavit touched na true v onBlur handleru pro každý input.
  + Podmíněně přidávat třídu .touched do className inputu:  
    className={${styles.formInput} ${formData.nameTouched && !formData.name ? styles.touchedInvalid : ''}` // Zjednodušený příklad

**Jednodušší CSS-only přístup (méně UX přívětivý, protože chyby ukazuje hned):**  
Pokud nechceš zatím řešit "touched" stavy v JS:

.formInput:user-invalid:not(:focus), /\* Zobrazí chybu, když pole ztratí focus a je nevalidní \*/

.formTextarea:user-invalid:not(:focus) {

border-color: #e53e3e; /\* Červená pro chybu \*/

}

/\* :user-invalid je novější pseudotřída, podpora se může lišit.

Alternativně jen :invalid, ale to může být agresivní. \*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Doporučuji spíše implementovat "touched" logiku pro lepší UX.

**3. Vzhled <select> elementu:**

Styling <select> je notoricky známý problém. Nejlepší je ho "resetovat" a pak nastylovat, jak nejvíc to jde, případně přidat vlastní šipku.

* **CSS v kontakt.module.css:**
* .formSelect {
* appearance: none; /\* Odstraní výchozí vzhled prohlížeče \*/
* -webkit-appearance: none;
* -moz-appearance: none;
* background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='16' height='16' fill='%23333' viewBox='0 0 16 16'%3E%3Cpath fill-rule='evenodd' d='M1.646 4.646a.5.5 0 0 1 .708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z'/%3E%3C/svg%3E");
* background-repeat: no-repeat;
* background-position: right 15px center;
* background-size: 16px 12px; /\* Velikost šipky \*/
* padding-right: 40px; /\* Místo pro vlastní šipku \*/
* /\* ... tvoje ostatní styly pro .formSelect ... \*/
* }
* /\* Pro dark mode můžeš chtít jinou barvu šipky \*/
* :global(.dark-mode) .formSelect {
* background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='16' height='16' fill='%23eee' viewBox='0 0 16 16'%3E%3Cpath fill-rule='evenodd' d='M1.646 4.646a.5.5 0 0 1 .708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z'/%3E%3C/svg%3E");

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

SVG pro šipku je inline Data URL. Můžeš si vytvořit vlastní SVG nebo použít ikonu. fill='%23333' je barva šipky pro light mode, fill='%23eee' pro dark mode.

**4. Ikonka k odesílacímu tlačítku:**

Použij react-icons nebo jiné SVG ikony.

* **Úprava JSX v kontakt.tsx:**  
  Importuj ikonu, např. FaPaperPlane z react-icons/fa.
* import { FaPaperPlane } from 'react-icons/fa'; // Nebo jiná ikona
* // ...
* <button type="submit" className={styles.submitButton} disabled={isSubmitting}>
* {isSubmitting ? 'Odesílání...' : (
* <>
* Odeslat formulář <FaPaperPlane style={{ marginLeft: '8px' }} />
* </>
* )}

</button>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

* **CSS v kontakt.module.css (pro tlačítko):**  
  Můžeš přidat display: inline-flex a align-items: center pro lepší zarovnání textu a ikony.
* .submitButton {
* /\* ... tvoje stávající styly ... \*/
* display: inline-flex;
* align-items: center;
* justify-content: center; /\* Pokud chceš obsah vycentrovat \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Těmito úpravami by měl tvůj formulář vypadat podstatně lépe a být přívětivější. Implementace "touched" stavů pro validaci je trochu víc práce v JavaScriptu, ale výrazně zlepší UX.

Můžeš mi navrhnout základní strukturu JSX pro tyto sekce v aboutme.tsx a k tomu odpovídající základní CSS třídy a styly v aboutme.module.css? Chci, aby to bylo přehledné, moderní a samozřejmě responzivní. Barvy a písma by měly co nejvíc vycházet z globals.css.

Jasně, stránka "O mně" je důležitá pro budování důvěry. Pojďme navrhnout strukturu a základní styly.

**Návrh JSX struktury (v aboutme.tsx):**

// pages/aboutme.tsx nebo app/aboutme/page.tsx

import React from 'react';

import Head from 'next/head';

import Image from 'next/image'; // Pro optimalizované obrázky

import Link from 'next/link'; // Pro CTA tlačítko

import styles from './aboutme.module.css';

// Ikonky si můžeš naimportovat z react-icons, např.

import { FaAward, FaDumbbell, FaStar, FaUserGraduate, FaRunning, FaHeartbeat } from 'react-icons/fa';

const AboutMePage: React.FC = () => {

return (

<>

<Head>

<title>O mně | Fitness Coaching</title>

<meta name="description" content="Poznejte svého osobního trenéra..." />

</Head>

<div className={styles.aboutContainer}> {/\* Hlavní kontejner stránky \*/}

{/\* 1. Hero Sekce \*/}

<section className={styles.heroSection}>

<span className={styles.subtitle}>POZNEJTE SVÉHO TRENÉRA</span>

<h1 className={styles.title}>Kdo jsem?</h1>

<div className={styles.separator}></div>

</section>

{/\* 2. Intro Sekce \*/}

<section className={styles.introSection}>

<div className={styles.introWrapper}> {/\* Obalovač pro layout \*/}

<div className={styles.imageContainer}>

<div className={styles.imageFrame}>

<Image

src="/images/trener.jpg" // Cesta k obrázku trenéra

alt="Osobní trenér"

width={500} // Nastav skutečnou šířku obrázku

height={600} // Nastav skutečnou výšku obrázku

className={styles.trainerImage}

priority // Pokud je obrázek důležitý pro LCP

/>

<div className={styles.experienceBadge}>

<span className={styles.experienceYears}>10+</span>

<span className={styles.experienceText}>Let zkušeností</span>

</div>

</div>

</div>

<div className={styles.introContent}>

<h2 className={styles.introTitle}>Vítejte ve světě fitness se mnou</h2>

<p className={styles.introParagraph}>

Ahoj, jmenuji se [Jméno Trenéra] a fitness je moje vášeň i profese. Pomohl jsem již stovkám klientů dosáhnout jejich cílů, ať už šlo o redukci váhy, nabrání svalové hmoty nebo zlepšení celkové kondice.

</p>

<p className={styles.introParagraph}>

Mým cílem je nejen sestavit vám efektivní tréninkový plán a jídelníček, ale také vás naučit principy zdravého životního stylu, které si udržíte dlouhodobě.

</p>

<div className={styles.signatureContainer}>

{/\* Můžeš sem dát obrázek podpisu, pokud máš \*/}

{/\* <Image src="/images/podpis.png" alt="Podpis" width={150} height={80} className={styles.signatureImage} /> \*/}

<div className={styles.signatureInfo}>

<p className={styles.signatureName}>[Jméno Příjmení Trenéra]</p>

<p className={styles.signatureTitle}>Certifikovaný Osobní Trenér</p>

</div>

</div>

</div>

</div>

</section>

{/\* 3. Sekce "Můj přístup" \*/}

<section className={styles.approachSection}>

<div className={styles.approachWrapper}>

<h2 className={styles.sectionTitle}>Můj Přístup k Tréninku</h2>

<p className={styles.approachText}>

Věřím v individuální přístup ke každému klientovi. Každý jsme jiný a co funguje pro jednoho, nemusí fungovat pro druhého. Proto kladu důraz na...

</p>

<div className={styles.featuresGrid}>

<div className={styles.featureCard}>

<div className={styles.featureIconContainer}><FaDumbbell /></div>

<h3 className={styles.featureTitle}>Personalizované Plány</h3>

<p className={styles.featureDescription}>Tréninky a jídelníčky sestavené přesně na míru vašim potřebám a cílům.</p>

</div>

<div className={styles.featureCard}>

<div className={styles.featureIconContainer}><FaHeartbeat /></div>

<h3 className={styles.featureTitle}>Zdraví na Prvním Místě</h3>

<p className={styles.featureDescription}>Udržitelný přístup s důrazem na vaše dlouhodobé zdraví a pohodu.</p>

</div>

<div className={styles.featureCard}>

<div className={styles.featureIconContainer}><FaUserGraduate /></div>

<h3 className={styles.featureTitle}>Vzdělávání a Motivace</h3>

<p className={styles.featureDescription}>Nejen cvičíme, ale také se učíme, jak tělo funguje a jak si udržet motivaci.</p>

</div>

<div className={styles.featureCard}>

<div className={styles.featureIconContainer}><FaRunning /></div>

<h3 className={styles.featureTitle}>Flexibilita</h3>

<p className={styles.featureDescription}>Plány, které respektují váš životní styl a časové možnosti.</p>

</div>

</div>

</div>

</section>

{/\* 4. Statistiky \*/}

<section className={styles.statsSection}>

<div className={styles.statsContainer}>

<div className={styles.statItem}>

<span className={styles.statNumber}>500+</span>

<span className={styles.statText}>Spokojených klientů</span>

</div>

<div className={styles.statItem}>

<span className={styles.statNumber}>10+</span>

<span className={styles.statText}>Let zkušeností</span>

</div>

<div className={styles.statItem}>

<span className={styles.statNumber}>99%</span>

<span className={styles.statText}>Úspěšnost klientů</span>

</div>

<div className={styles.statItem}>

<span className={styles.statNumber}>24/7</span>

<span className={styles.statText}>Online podpora</span>

</div>

</div>

</section>

{/\* 5. Reference (Testimonials) \*/}

<section className={styles.testimonialsSection}>

<div className={styles.testimonialsWrapper}>

<h2 className={styles.sectionTitle}>Co Říkají Moji Klienti</h2>

<div className={styles.testimonialsContainer}>

{/\* Příklad jedné reference - opakuj pro další \*/}

<div className={styles.testimonialCard}>

<div className={styles.testimonialStars}>

<FaStar /><FaStar /><FaStar /><FaStar /><FaStar />

</div>

<p className={styles.testimonialText}>

"Díky [Jméno Trenéra] jsem konečně dosáhl/a svých cílů! Skvělý přístup a motivace."

</p>

<div className={styles.testimonialAuthor}>

<span className={styles.authorName}>Jméno Klienta</span>

<span className={styles.authorInfo}>Cíl: Redukce váhy</span>

</div>

</div>

{/\* Další reference ... \*/}

</div>

</div>

</section>

{/\* 6. Certifikáty/Kvalifikace \*/}

<section className={styles.certificatesSection}>

<div className={styles.certificatesWrapper}>

<h2 className={styles.sectionTitle}>Kvalifikace a Certifikace</h2>

<div className={styles.certificatesGrid}>

<div className={styles.certificateCard}>

<div className={styles.certificateIcon}><FaAward /></div>

<h3 className={styles.certificateTitle}>Certifikovaný Osobní Trenér</h3>

<p className={styles.certificateDescription}>Mezinárodně uznávaná certifikace...</p>

</div>

{/\* Další certifikáty ... \*/}

</div>

</div>

</section>

{/\* 7. CTA Sekce \*/}

<section className={styles.ctaSection}>

<div className={styles.ctaContainer}>

<h2 className={styles.ctaTitle}>Připraveni Začít Svou Proměnu?</h2>

<p className={styles.ctaText}>

Neváhejte mě kontaktovat pro nezávaznou konzultaci a společně najdeme tu nejlepší cestu k vašim cílům.

</p>

<Link href="/kontakt" className={styles.ctaButton}>

Kontaktujte Mě

</Link>

</div>

</section>

</div>

</>

);

};

export default AboutMePage;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Návrh základního CSS (v aboutme.module.css):**

/\* aboutme.module.css \*/

.aboutContainer {

max-width: 100%;

overflow-x: hidden;

}

/\* --- Hero Sekce --- \*/

.heroSection {

padding: 80px 20px;

text-align: center;

background-color: var(--color-secondary-background); /\* Lehké pozadí \*/

}

.subtitle {

font-size: 1.1rem;

font-weight: 600;

color: var(--color-primary);

margin-bottom: 10px;

letter-spacing: 2px;

text-transform: uppercase;

}

.title {

font-size: 3rem;

font-weight: 800;

color: var(--color-text-primary);

margin-bottom: 20px;

}

.separator {

width: 80px;

height: 4px;

background-color: var(--color-primary);

margin: 0 auto;

border-radius: 2px;

}

/\* --- Intro Sekce --- \*/

.introSection {

padding: 60px 20px;

background-color: var(--color-background);

}

.introWrapper {

display: flex;

gap: 40px;

align-items: center;

max-width: 1100px;

margin: 0 auto;

}

.imageContainer {

flex: 1;

max-width: 400px; /\* Omez šířku obrázku \*/

position: relative; /\* Pro badge \*/

}

.imageFrame {

border-radius: 10px;

overflow: hidden;

box-shadow: 0 10px 30px var(--color-box-shadow);

}

.trainerImage { /\* Next/Image si nastaví width a height sám \*/

display: block;

width: 100%;

height: auto; /\* Zachová poměr stran \*/

}

.experienceBadge {

position: absolute;

bottom: 20px;

right: -25px; /\* Částečně přesahuje obrázek \*/

background-color: var(--color-primary);

color: white;

padding: 10px 15px;

border-radius: 5px;

box-shadow: 0 5px 15px rgba(0,0,0,0.2);

text-align: center;

}

.experienceYears {

display: block;

font-size: 1.8rem;

font-weight: 700;

}

.experienceText {

font-size: 0.8rem;

text-transform: uppercase;

}

.introContent {

flex: 1.5; /\* Více místa pro text \*/

}

.introTitle {

font-size: 2.2rem;

font-weight: 700;

color: var(--color-text-primary);

margin-bottom: 20px;

position: relative;

padding-bottom: 10px;

}

.introTitle::after { /\* Podtržení nadpisu \*/

content: '';

position: absolute;

left: 0;

bottom: 0;

width: 50px;

height: 3px;

background-color: var(--color-primary);

}

.introParagraph {

font-size: 1rem;

line-height: 1.7;

color: var(--color-text-secondary);

margin-bottom: 15px;

}

.signatureContainer { margin-top: 20px; }

.signatureName { font-weight: bold; color: var(--color-text-primary); }

.signatureTitle { font-size: 0.9rem; color: var(--color-text-secondary); }

/\* --- Sekce "Můj přístup" --- \*/

.approachSection {

padding: 60px 20px;

background-color: var(--color-secondary-background);

}

.approachWrapper {

max-width: 1100px;

margin: 0 auto;

text-align: center; /\* Pro nadpis a úvodní text \*/

}

.sectionTitle { /\* Společný styl pro nadpisy sekcí \*/

font-size: 2.5rem;

font-weight: 700;

color: var(--color-text-primary);

margin-bottom: 20px;

position: relative;

display: inline-block; /\* Aby se ::after správně pozicovalo \*/

padding-bottom: 15px;

}

.sectionTitle::after { /\* Podtržení \*/

content: '';

position: absolute;

left: 50%;

bottom: 0;

transform: translateX(-50%);

width: 70px;

height: 3px;

background-color: var(--color-primary);

}

.approachText {

font-size: 1.1rem;

color: var(--color-text-secondary);

max-width: 700px;

margin: 0 auto 40px auto; /\* Odsazení pod textem \*/

line-height: 1.7;

}

.featuresGrid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 30px;

text-align: left; /\* Karty samotné budou mít text vlevo \*/

}

.featureCard {

background-color: var(--color-background);

padding: 25px;

border-radius: 8px;

box-shadow: var(--color-box-shadow);

transition: transform 0.3s ease;

}

.featureCard:hover {

transform: translateY(-5px);

}

.featureIconContainer {

font-size: 2rem; /\* Velikost ikonky \*/

color: var(--color-primary);

margin-bottom: 15px;

display: inline-block; /\* Aby se ikona chovala jako blok \*/

/\* Nebo pokud chceš kruhové pozadí:

width: 60px; height: 60px; border-radius: 50%;

background-color: var(--color-primary); color: white;

display: flex; align-items: center; justify-content: center; \*/

}

.featureTitle {

font-size: 1.3rem;

font-weight: 600;

color: var(--color-text-primary);

margin-bottom: 10px;

}

.featureDescription {

font-size: 0.9rem;

color: var(--color-text-secondary);

line-height: 1.6;

}

/\* --- Statistiky --- \*/

.statsSection {

padding: 50px 20px;

background-color: var(--color-primary); /\* Výrazné pozadí \*/

color: white;

}

.statsContainer {

display: flex;

justify-content: space-around;

align-items: center;

flex-wrap: wrap; /\* Pro responzivitu \*/

gap: 20px;

max-width: 900px;

margin: 0 auto;

}

.statItem {

text-align: center;

min-width: 150px; /\* Minimální šířka pro každou statistiku \*/

}

.statNumber {

font-size: 2.8rem;

font-weight: 800;

display: block;

}

.statText {

font-size: 1rem;

text-transform: uppercase;

opacity: 0.9;

}

/\* --- Reference (Testimonials) --- \*/

.testimonialsSection {

padding: 60px 20px;

background-color: var(--color-background);

}

.testimonialsWrapper { max-width: 1100px; margin: 0 auto; }

.testimonialsContainer {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 30px;

}

.testimonialCard {

background-color: var(--color-secondary-background);

padding: 25px;

border-radius: 8px;

box-shadow: var(--color-box-shadow);

border-left: 4px solid var(--color-primary); /\* Akcent na levé straně \*/

}

.testimonialStars {

color: #FFD700; /\* Zlatá barva hvězd \*/

margin-bottom: 15px;

font-size: 1.2rem; /\* Velikost hvězd \*/

}

.testimonialText {

font-style: italic;

color: var(--color-text-secondary);

margin-bottom: 15px;

line-height: 1.6;

}

.authorName {

font-weight: 600;

color: var(--color-text-primary);

}

.authorInfo {

font-size: 0.85rem;

color: var(--color-text-secondary);

}

/\* --- Certifikáty/Kvalifikace --- \*/

.certificatesSection {

padding: 60px 20px;

background-color: var(--color-secondary-background);

}

.certificatesWrapper { max-width: 1100px; margin: 0 auto; }

.certificatesGrid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 30px;

}

.certificateCard {

background-color: var(--color-background);

padding: 20px;

border-radius: 8px;

text-align: center;

box-shadow: var(--color-box-shadow);

}

.certificateIcon {

font-size: 2.5rem;

color: var(--color-primary);

margin-bottom: 15px;

}

.certificateTitle {

font-size: 1.2rem;

font-weight: 600;

color: var(--color-text-primary);

margin-bottom: 10px;

}

.certificateDescription {

font-size: 0.9rem;

color: var(--color-text-secondary);

line-height: 1.5;

}

/\* --- CTA Sekce --- \*/

.ctaSection {

padding: 70px 20px;

background-color: var(--color-primary-hover); /\* Tmavší odstín primární barvy nebo jiná akcentní barva \*/

color: white;

text-align: center;

}

:global(.dark-mode) .ctaSection {

background-color: var(--color-primary); /\* V dark modu může být světlejší z primárních barev \*/

}

.ctaContainer { max-width: 700px; margin: 0 auto; }

.ctaTitle {

font-size: 2.2rem;

font-weight: 700;

margin-bottom: 15px;

color: white; /\* Explicitně bílá, protože pozadí je tmavé \*/

}

/\* Odstranění ::after pro .ctaTitle, pokud má .sectionTitle ::after a .ctaTitle sdílí .sectionTitle třídu \*/

.ctaTitle.sectionTitle::after {

display: none;

}

.ctaText {

font-size: 1.1rem;

margin-bottom: 30px;

line-height: 1.7;

opacity: 0.9;

}

.ctaButton {

display: inline-block;

padding: 14px 35px;

background-color: white; /\* Kontrastní tlačítko \*/

color: var(--color-primary); /\* Text v primární barvě \*/

border-radius: 8px;

text-decoration: none;

font-weight: 700;

font-size: 1rem;

transition: all 0.3s ease;

box-shadow: 0 4px 10px rgba(0,0,0,0.1);

}

.ctaButton:hover {

background-color: #f0f0f0; /\* Lehce tmavší bílá \*/

color: var(--color-primary-hover);

transform: translateY(-2px);

box-shadow: 0 6px 15px rgba(0,0,0,0.15);

}

/\* --- Responzivita --- \*/

@media (max-width: 992px) {

.introWrapper {

flex-direction: column;

text-align: center;

}

.imageContainer {

margin-bottom: 30px;

max-width: 350px; /\* Menší obrázek na tabletech \*/

}

.introContent {

flex: 1; /\* Plná šířka \*/

}

.introTitle::after { left: 50%; transform: translateX(-50%); } /\* Centrování podtržení \*/

.signatureContainer { justify-content: center; }

}

@media (max-width: 768px) {

.title { font-size: 2.5rem; }

.sectionTitle { font-size: 2rem; }

.introTitle { font-size: 1.8rem; }

.ctaTitle { font-size: 1.8rem; }

.featuresGrid, .testimonialsContainer, .certificatesGrid {

grid-template-columns: 1fr; /\* Vše pod sebou \*/

}

.statsContainer {

flex-direction: column;

gap: 30px;

}

.statItem { min-width: auto; }

.experienceBadge { right: 10px; bottom: 10px; padding: 8px 12px; }

.experienceYears { font-size: 1.5rem; }

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Klíčové body a doporučení:**

* **Responzivita:** Používám flex a grid pro layouty, které by se měly dobře přizpůsobovat. Media queries na konci upravují vzhled pro menší obrazovky.
* **Obrázky:** Použij komponentu <Image> z next/image pro optimalizaci. Nezapomeň nastavit width a height nebo layout="fill" a objectFit. Cesty k obrázkům (/images/trener.jpg) si uprav podle své struktury.
* **Ikonky:** react-icons je skvělá volba. Můžeš si vybrat jakékoliv ikony, které se ti hodí.
* **Konzistence:** Snaž se používat CSS proměnné z globals.css pro barvy, aby byl vzhled konzistentní a snadno upravitelný.
* **Obsah:** Tento návrh je kostra. Doplň reálné texty, obrázky a reference.
* **CTA:** Ujisti se, že CTA tlačítko na konci stránky jasně směřuje uživatele tam, kam chceš (např. na kontaktní stránku).

Tohle je poměrně obsáhlý základ. Začni implementací jednotlivých sekcí a postupně je vylaďuj.

// ...

<section className={styles.introSection}>

<div className={styles.introWrapper}>

<div className={styles.imageContainer}> {/\* Tento kontejner má position: relative \*/}

<Image

src="/images/trener.jpg"

alt="Osobní trenér"

width={500}

height={600}

className={styles.trainerImage}

priority

/>

{/\* TENTO ODZNAK ZLOBÍ \*/}

<div className={styles.experienceBadge}>

<span className={styles.experienceYears}>10+</span>

<span className={styles.experienceText}>Let zkušeností</span>

</div>

</div>

</div>

{/\* ... introContent ... \*/}

</div>

</section>

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A CSS z aboutme.module.css:

/\* ... \*/

.imageContainer {

flex: 1;

max-width: 400px;

position: relative; /\* Toto je důležité pro absolutní pozicování badge \*/

}

.imageFrame { /\* Nově přidaný, pokud ho používáš pro rámeček nebo stín obrázku \*/

border-radius: 10px;

overflow: hidden; /\* Pokud obrázek přesahuje \*/

box-shadow: 0 10px 30px var(--color-box-shadow);

position: relative; /\* Badge by měl být relativní k tomuto, pokud je uvnitř \*/

/\* Nebo nechat badge relativní k .imageContainer \*/

}

.experienceBadge {

position: absolute;

bottom: 20px; /\* Pozice od spodního okraje .imageContainer (nebo .imageFrame) \*/

right: -25px; /\* Pozice od pravého okraje - tady to může zlobit \*/

background-color: var(--color-primary);

color: white;

padding: 10px 15px;

border-radius: 5px;

box-shadow: 0 5px 15px rgba(0,0,0,0.2);

text-align: center;

z-index: 10; /\* Aby byl nad obrázkem, pokud by se překrývaly \*/

}

/\* ... \*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Jak mám ten .experienceBadge správně napozicovat, aby byl vždycky hezky umístěný třeba vpravo dole u toho obrázku, ale aby se dobře choval i při změně velikosti okna a aby třeba moc nezasahoval do textu vedle, pokud je layout responzivní a obrázek se zmenší? Možná by měl být spíš uvnitř obrázku, ne takhle napůl venku?

Máš pravdu, absolutní pozicování může být někdy zrádné, zvláště když se kombinuje s responzivitou a procentuálními hodnotami nebo flex layoutem.

Problém s right: -25px; je ten, že odznak je pozicován *mimo* svého rodičovského kontejneru (.imageContainer nebo .imageFrame), což může způsobovat problémy při zmenšování, kdy se vedlejší obsah může začít s odznakem překrývat nebo odznak "ulítne".

**Možnosti řešení pro .experienceBadge:**

**Řešení 1: Umístění odznaku dovnitř .imageFrame a pozicování vůči němu.**

Toto je často stabilnější, protože odznak zůstává vizuálně spojen s obrázkem.

1. **HTML struktura:** Ujisti se, že .experienceBadge je uvnitř .imageFrame, pokud chceš, aby byl pozicován relativně k rámečku (což máš).
2. **CSS úpravy:**
   * **.imageFrame**: Musí mít position: relative; (to už máš naznačeno, tak se ujisti, že to tak je).
   * **.experienceBadge**: Změň hodnoty bottom a right tak, aby byl odznak uvnitř rámečku.
3. /\* aboutme.module.css \*/
4. .imageContainer {
5. /\* ... beze změny ... \*/
6. /\* position: relative; už není tak kritické, pokud je badge v .imageFrame \*/
7. }
8. .imageFrame {
9. border-radius: 10px;
10. overflow: hidden; /\* Důležité, pokud by badge přesahoval a chtěl jsi ho oříznout k rámečku \*/
11. box-shadow: 0 10px 30px var(--color-box-shadow);
12. position: relative; /\* Klíčové pro pozicování badge uvnitř \*/
13. /\* display: inline-block; /\* Může pomoci, pokud by se imageFrame choval divně \*/
14. }
15. .experienceBadge {
16. position: absolute;
17. bottom: 15px; /\* Např. 15px od spodního okraje imageFrame \*/
18. right: 15px; /\* Např. 15px od pravého okraje imageFrame \*/
19. background-color: var(--color-primary);
20. color: white;
21. padding: 8px 12px; /\* Mírně menší padding, pokud je uvnitř \*/
22. border-radius: 5px;
23. box-shadow: 0 3px 10px rgba(0,0,0,0.2);
24. text-align: center;
25. z-index: 10;
26. /\* Pro lepší čitelnost na různých obrázcích můžeš přidat lehké poloprůhledné pozadí,
27. pokud by barva odznaku splývala s obrázkem: \*/
28. /\* background-color: rgba(0, 123, 255, 0.9); /\* Primární barva s trochou průhlednosti \*/
29. }
30. .experienceYears {
31. display: block;
32. font-size: 1.6rem; /\* Mírně menší, pokud je uvnitř \*/
33. font-weight: 700;
34. }
35. .experienceText {
36. font-size: 0.75rem; /\* Mírně menší \*/
37. text-transform: uppercase;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Řešení 2: Použití transformace pro přesnější centrování/pozicování (pokud chceš odznak částečně překrývat roh).**

Pokud chceš, aby odznak elegantně seděl na rohu, můžeš použít transform.

.imageFrame {

/\* ... jako výše ... \*/

position: relative;

}

.experienceBadge {

position: absolute;

/\* Chceme ho do pravého dolního rohu, ale pak ho posuneme \*/

bottom: 0;

right: 0;

/\* Posuneme ho o polovinu jeho výšky nahoru a o polovinu jeho šířky doleva,

aby "seděl" na rohu. Velikosti paddingu a písma ovlivní jeho celkovou velikost. \*/

transform: translate(25%, 25%); /\* Experimentuj s procenty; např. 25% ho posune ven z rohu \*/

/\* Nebo pro umístění přesně na roh, ale částečně ven: transform: translate(50%, 50%); \*/

/\* Nebo pokud chceš, aby byl v rohu, ale celý viditelný: transform: translate(-10px, -10px); (záleží na paddingu) \*/

background-color: var(--color-primary);

color: white;

padding: 10px 15px;

border-radius: 5px;

/\* ... zbytek stylů ... \*/

z-index: 10;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Tento přístup s transform může být trochu složitější na vyladění, aby vypadal dobře na všech velikostech, protože procentuální transformace se vztahuje k velikosti samotného elementu .experienceBadge.

**Řešení 3: Responzivní úpravy pro odznak.**

Můžeš také přidat media queries specificky pro .experienceBadge, aby se jeho pozice nebo velikost měnila na menších obrazovkách.

/\* ... stávající styly pro .experienceBadge ... \*/

@media (max-width: 768px) {

.experienceBadge {

/\* Pokud je badge uvnitř (Řešení 1) \*/

bottom: 10px;

right: 10px;

padding: 6px 10px;

}

.experienceYears {

font-size: 1.3rem;

}

.experienceText {

font-size: 0.7rem;

}

}

@media (max-width: 480px) {

.experienceBadge {

/\* Možná ještě menší nebo jiná pozice \*/

bottom: 5px;

right: 5px;

font-size: 0.9em; /\* Zmenší text uvnitř proporcionálně \*/

padding: 5px 8px;

}

.experienceYears {

font-size: 1.1rem;

}

.experienceText {

font-size: 0.6rem;

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Doporučení:**

1. **Začni s Řešením 1 (odznak uvnitř .imageFrame).** Je to obvykle nejstabilnější a nejjednodušší na údržbu.
2. Ujisti se, že .imageFrame (nebo .imageContainer, pokud badge není vnořen do dalšího divu) má position: relative.
3. Hodnoty bottom a right pro .experienceBadge nastav tak, aby byl plně nebo částečně uvnitř obrázku/rámečku podle tvé preference. Kladné hodnoty ho posouvají dovnitř od okrajů.
4. Testuj na různých šířkách obrazovky a případně dolaď pozici/velikost pomocí media queries (Řešení 3).

Pokud by odznak i přesto "utíkal", zkontroluj, zda nějaký nadřazený prvek nemá overflow: hidden, které by ho mohlo ořezávat, nebo zda není problém se z-index vůči jiným prvkům. Ale s position: absolute by měl být vyjmut z normálního toku a pozicovat se vůči nejbližšímu position: relative (nebo absolute/fixed/sticky) předkovi.

Mam v .env.local nastaveny EMAIL\_USER a EMAIL\_APP\_PASSWORD (pro Gmail a heslo pro aplikace).

Tady je ten pages/api/send-email.ts:

// pages/api/send-email.ts

import type { NextApiRequest, NextApiResponse } from 'next';

import nodemailer from 'nodemailer';

type ResponseData = {

success: boolean;

message?: string;

error?: string;

};

export default async function handler(

req: NextApiRequest,

res: NextApiResponse<ResponseData>

) {

if (req.method !== 'POST') {

res.setHeader('Allow', ['POST']);

return res.status(405).json({ success: false, error: `Method ${req.method} Not Allowed` });

}

try {

const transporter = nodemailer.createTransport({

service: 'gmail', // Pouzivam Gmail

auth: {

user: process.env.EMAIL\_USER,

pass: process.env.EMAIL\_APP\_PASSWORD, // Heslo pro aplikace

},

});

const { name, email, message, service /\* ... a další pole ... \*/ } = req.body;

const mailOptions = {

from: `"Fitness web" <${process.env.EMAIL\_USER}>`,

to: 'mojedavidjmeno@gmail.com', // Můj email kam to má chodit

subject: `Nová poptávka: ${service} od ${name}`,

html: `

<h2>Nová poptávka</h2>

<p><strong>Služba:</strong> ${service || 'Neuvedeno'}</p>

<p><strong>Jméno:</strong> ${name}</p>

<p><strong>Email:</strong> ${email}</p>

<p><strong>Zpráva:</strong> ${message || 'Žádná zpráva'}</p>

{/\* Tady bych mohl přidat i další pole z formData \*/}

`,

};

await transporter.sendMail(mailOptions);

console.log('Email odeslán na:', mailOptions.to); // Log pro mě

res.status(200).json({ success: true, message: 'E-mail úspěšně odeslán' });

} catch (error: unknown) {

console.error('CHYBA V API:', error); // Loguju chybu na serveru

// Chci aby se ta chyba poslala i na klienta, aby uzivatel vedel, ze je neco spatne

const errorMessage = error instanceof Error ? error.message : 'Neznámá chyba serveru.';

res.status(500).json({

success: false,

error: `Chyba při odesílání e-mailu: ${errorMessage}. Zkuste to prosím později nebo mě kontaktujte jinak.`

});

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

Co s tím může být? Ty process.env.EMAIL\_USER a process.env.EMAIL\_APP\_PASSWORD by se měly načítat správně, ne? A ten transporter pro Nodemailer vypadá taky podle dokumentace. Může být problém s Gmailem nebo tak něco? Nebo dělám něco blbě v tom try...catch bloku?

Chápu, problémy s odesíláním emailů přes API můžou být frustrující. Chyba 500 a errory od Nodemaileru často souvisí s konfigurací transporteru, přístupem k emailové službě, nebo s proměnnými prostředí.

Pojďme se podívat na možné příčiny a co zkontrolovat:

1. **Proměnné prostředí (.env.local):**
   * **Správné načtení:** Ujisti se, že Next.js správně načítá tvé proměnné. Po každé změně v .env.local **musíš restartovat vývojový server** (npm run dev nebo yarn dev), aby se změny projevily.
   * **Názvy proměnných:** Jsou názvy EMAIL\_USER a EMAIL\_APP\_PASSWORD v .env.local přesně takto? Bez překlepů?
   * **Hodnoty:** Je EMAIL\_USER tvůj celý Gmail email (např. tvojemail@gmail.com)? Je EMAIL\_APP\_PASSWORD opravdu 16místné heslo pro aplikace vygenerované v nastavení Google účtu, a ne tvé běžné heslo k Gmailu?
   * **Prefix NEXT\_PUBLIC\_:** Pokud bys tyto proměnné potřeboval i na straně klienta (což pro heslo určitě ne!), musely by mít prefix NEXT\_PUBLIC\_. Ale pro serverovou API route to není potřeba a je to takhle bezpečnější.
2. **Nastavení Gmail účtu:**
   * **Heslo pro aplikace:** Ještě jednou ověř, že máš pro svůj Google účet vygenerované "Heslo pro aplikace" a že jsi ho správně zkopíroval. Běžné heslo zde fungovat nebude a může vést k chybám autentizace.
   * **Méně bezpečné aplikace (Less Secure App Access):** Pro použití Nodemaileru s Gmailem (i s heslem pro aplikace) už bys neměl potřebovat povolovat "Přístup méně bezpečných aplikací", protože hesla pro aplikace jsou právě tím bezpečnějším způsobem. Ale pokud by problémy přetrvávaly, můžeš dočasně (pro testování) zkusit toto nastavení v Google účtu zkontrolovat, i když by to nemělo být nutné. **Pozor: Povolení méně bezpečných aplikací je bezpečnostní riziko.**
   * **Dvoufázové ověření:** Pokud máš na Google účtu zapnuté dvoufázové ověření (což je doporučené), musíš používat právě heslo pro aplikace.
3. **Konfigurace Nodemailer Transporteru:**
   * Tvá konfigurace service: 'gmail' a auth vypadá správně.
   * Někdy může být problém s tím, že Gmail detekuje pokus o přihlášení z nového "zařízení" (tvůj server/localhost) a může ho dočasně blokovat nebo vyžadovat dodatečné ověření. Zkontroluj si emailovou schránku (tu, kterou používáš pro EMAIL\_USER), jestli tam nepřišlo nějaké bezpečnostní upozornění od Googlu.
4. **Chybové hlášky z Nodemaileru (v terminálu):**
   * **Podívej se na detailní chybovou hlášku, kterou ti Nodemailer vypisuje v terminálu, když dojde k chybě 500.** Ta hláška často napoví víc:
     + Invalid login: Problém s EMAIL\_USER nebo EMAIL\_APP\_PASSWORD.
     + Connection timeout nebo ETIMEDOUT: Problém se síťovým připojením tvého serveru k SMTP serverům Googlu. Může to být firewallem, dočasným výpadkem atd.
     + Missing credentials: Nenalezl user nebo pass.
     + Něco ohledně SSL/TLS: Většinou by to service: 'gmail' mělo řešit automaticky.
5. **Logování v API route:**
   * Před await transporter.sendMail(mailOptions); si můžeš přidat console.log('Pokus o odeslání s user:', process.env.EMAIL\_USER); abys viděl, zda se proměnná vůbec načetla. **Nikdy neloguj heslo!**
6. **Zpracování chyby v catch bloku:**
   * Tvoje zpracování chyby vypadá dobře. error instanceof Error ? error.message : 'Neznámá chyba serveru.' je dobrý způsob, jak získat text chyby.
   * Posílání této zprávy na klienta je také v pořádku, aby uživatel věděl, že se něco stalo.

**Co zkusit teď:**

1. **Restartuj vývojový server** (Ctrl+C a pak npm run dev).
2. **Důkladně zkontroluj hodnoty EMAIL\_USER a EMAIL\_APP\_PASSWORD v .env.local** (překlepy, správné heslo pro aplikace).
3. **Podívej se na přesnou chybovou hlášku v terminálu (na serveru), když dojde k chybě.** Zkopíruj ji sem, pokud si s ní nebudeš vědět rady.
4. Dočasně zkus v mailOptions nastavit to: na stejný email jako je EMAIL\_USER, abys vyloučil problémy s cílovou adresou.

le kdyz jsem zkusil zakomentovat ten radek await transporter.sendMail(mailOptions); a dat tam jen res.status(200).json({ success: true, message: 'Testovaci uspech' });, tak to NAJEDNOU proslo a formular na klientovi dostal odpoved! Takze problem bude asi fakt v tom samotnym transporter.sendMail.

Co by tam mohlo byt spatne? Konfigurace transporter vypada takhle:

const transporter = nodemailer.createTransport({

service: 'gmail',

auth: {

user: process.env.EMAIL\_USER, // Tohle se loguje jako NASTAVENO

pass: process.env.EMAIL\_APP\_PASSWORD, // Tohle taky

},

});

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

A mailOptions mam taky vyplneny. Nenapada te, co by mohlo ten sendMail takhle zaseknout, kdyz to predtim jakoby "nekdy" slo? Nemuze Gmail neco blokovat, i kdyz pouzivam heslo pro aplikace? Nebo je nejaky timeout, co bych mel nastavit? V terminalu zadna specificka chyba od Nodemaileru neni, proste to jen "stoji".

Aha, tak jestli to projde, když zakomentuješ transporter.sendMail(), tak je to jasné – problém je v komunikaci s SMTP serverem Googlu nebo v samotném procesu odesílání přes Nodemailer. To, že to "někdy" šlo, může naznačovat problémy se sítí, dočasné blokace ze strany Googlu, nebo limity.

Když sendMail "visí" a nevrací ani chybu hned, často to znamená, že se snaží připojit, ale nemůže, nebo čeká na odpověď, která nepřichází.

Zde jsou běžné příčiny a co zkusit:

1. **Dočasné síťové problémy nebo problémy na straně Googlu:**
   * Někdy může mít samotný Gmail nebo tvé připojení k němu dočasné potíže. To by vysvětlovalo, proč to "někdy" šlo. Zkus to o pár minut/hodin později.
2. **Blokace ze strany Googlu kvůli "podezřelé aktivitě":**
   * I s heslem pro aplikace, pokud se připojuješ z nové IP adresy (např. tvůj lokální vývojový server má dynamickou IP, nebo pokud jsi projekt deploynul a server má novou IP), Google může dočasně omezit přihlášení nebo odesílání, dokud si neověří, že jsi to ty.
   * **Co dělat:** Zkontroluj emailovou schránku process.env.EMAIL\_USER. Hledej jakékoliv emaily od Googlu ohledně bezpečnostních upozornění, zablokovaných pokusů o přihlášení nebo žádostí o ověření aktivity. Někdy tam může být odkaz typu "Byl jste to vy?" a po potvrzení se to může odblokovat.
3. **Limity pro odesílání Gmailu:**
   * Gmail má limity na počet emailů, které můžeš odeslat za den přes SMTP (obvykle kolem 500 pro běžné účty, ale může se lišit). Pokud jsi hodně testoval, mohl jsi narazit na limit. Tento limit se obvykle resetuje po 24 hodinách.
   * Také existují limity na frekvenci odesílání.
4. **Firewall nebo antivirový program:**
   * Pokud testuješ lokálně, tvůj firewall nebo antivirový program by teoreticky mohl blokovat odchozí spojení na port, který Nodemailer používá pro SMTP (obvykle 465 s SSL nebo 587 s TLS). Je to méně časté, ale možné.
5. **Nastavení secure a port v nodemailer.createTransport (pro pokročilejší ladění):**
   * Ačkoliv service: 'gmail' by si to měl nastavit automaticky, pro explicitní kontrolu můžeš zkusit:
   * const transporter = nodemailer.createTransport({
   * host: 'smtp.gmail.com',
   * port: 465, // Nebo 587, pokud by 465 nefungovalo
   * secure: true, // true pro port 465, false pro port 587 (pak vyžaduje STARTTLS)
   * auth: {
   * user: process.env.EMAIL\_USER,
   * pass: process.env.EMAIL\_APP\_PASSWORD,
   * },
   * // Můžeš zkusit přidat debug a logger pro více informací
   * // debug: true,
   * // logger: true

});

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

Pokud použiješ port 587, secure by mělo být false a Gmail obvykle vyžaduje, aby klient inicioval STARTTLS. Nodemailer by to měl zvládnout. S port: 465 a secure: true by to mělo být přímo SSL spojení.  
Zapnutí debug: true a logger: true ti vypíše do konzole serveru mnohem více informací o tom, co Nodemailer dělá v pozadí během pokusu o připojení a odeslání. To může odhalit, ve které fázi se to zasekává. **Nezapomeň to pak vypnout, hlavně debug, protože loguje i citlivé informace.**

1. **Timeouty v Nodemaileru:**
   * Nodemailer má výchozí timeouty. Pokud je tvé připojení velmi pomalé nebo server Googlu odpovídá pomalu, teoreticky by to mohlo vypršet. Můžeš zkusit nastavit delší connectionTimeout:
   * const transporter = nodemailer.createTransport({
   * service: 'gmail',
   * auth: { /\* ... \*/ },
   * connectionTimeout: 10000, // 10 sekund (výchozí je myslím 30s nebo více) - spíš bych zkusil VĚTŠÍ
   * // např. 60000 (60 sekund) pro test

});

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).TypeScript

Ale pokud to visí "donekonečna", tak to spíš timeoutem nebude, protože by to po čase mělo spadnout s chybou timeoutu.

**Co bych udělal teď:**

1. **Zkontroluj emailovou schránku (EMAIL\_USER) na jakákoliv bezpečnostní upozornění od Googlu.** To je velmi častý problém.
2. Pokud nic nenajdeš, zkus v nodemailer.createTransport přidat debug: true a logger: true (a pak restartuj server). Sleduj logy v terminálu serveru, když se pokusíš odeslat email. Měly by ti ukázat kroky SMTP komunikace.
3. Pokud logy z Nodemaileru ukazují, že se snaží připojit, ale nedostane odpověď, nebo se zasekne na nějakém kroku (např. EHLO, AUTH), může to být síťový problém nebo blokace.
4. Zkus explicitní konfiguraci s host, port, secure, jak jsem ukázal výše (nejprve s portem 465).

Dej vědět, co zjistíš z logů Nodemaileru (pokud zapneš debug) nebo jestli najdeš nějaké upozornění od Googlu. Ty debug logy bývají dost upovídané, ale často tam je vidět, kde přesně se to zastaví.

Jak to udelat, aby se ten text v .marqueeTrack fakt hejbal zprava doleva a porad dokola? A aby to bylo plynuly.

Tady je ten kus HTML co bych tam tak mel:

<section class="marqueeSection">

<div class="marqueeTrack">

<div class="marqueeContent">

<span>SLOGAN 1</span>

<span class="marqueeSeparator">•</span>

<span>SLOGAN 2</span>

<span class="marqueeSeparator">•</span>

<span>SLOGAN 3</span>

{/\* Tady bych to zduplikoval, aby to bylo nekonecny \*/}

<span>SLOGAN 1</span>

<span class="marqueeSeparator">•</span>

<span>SLOGAN 2</span>

<span class="marqueeSeparator">•</span>

<span>SLOGAN 3</span>

</div>

</div>

</section>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Html

A CSS zatim skoro zadny nemam, jen:

.marqueeSection {

padding: 20px 0;

background-color: var(--color-primary); /\* Aby to bylo videt \*/

overflow: hidden; /\* Tohle je asi dulezity \*/

color: white;

}

.marqueeTrack {

white-space: nowrap; /\* Aby se to nezalamovalo \*/

/\* Tady asi prijde ta animace? \*/

}

.marqueeContent {

display: flex; /\* Aby byly spany vedle sebe \*/

/\* align-items: center; \*/

}

.marqueeContent span {

padding: 0 20px;

font-weight: bold;

}

.marqueeSeparator {

padding: 0 10px;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Staci tam jenom nejaka CSS animace nebo je to slozitejsi?

Jo, ten běžící text (marquee) se dá udělat celkem jednoduše pomocí CSS animace. overflow: hidden; na .marqueeSection a white-space: nowrap; na .marqueeTrack jsou správné základy.

**CSS Animace:**

Potřebuješ @keyframes animaci, která bude posouvat .marqueeTrack doleva. Aby to bylo nekonečné a plynulé, text v .marqueeContent musí být dostatečně dlouhý (ideálně alespoň dvakrát tak dlouhý jako šířka viditelné oblasti, proto jsi správně uvažoval o zduplikování).

1. **Definuj @keyframes animaci:**
2. @keyframes marqueeAnimation {
3. 0% {
4. transform: translateX(0%); /\* Začátek - žádný posun \*/
5. }
6. 100% {
7. transform: translateX(-50%); /\* Posun o polovinu šířky tracku.
8. Pokud je tvůj obsah v marqueeContent přesně 2x tak dlouhý
9. jako viditelná část, posun o -50% způsobí, že když
10. první sada dojede na konec, druhá sada je přesně
11. na začátku, čímž vznikne iluze nekonečna. \*/
12. }

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

1. **Aplikuj animaci na .marqueeTrack:**
2. .marqueeTrack {
3. white-space: nowrap;
4. display: inline-block; /\* Nebo display: flex; pokud .marqueeContent je jeho jediný potomek \*/
5. animation: marqueeAnimation 30s linear infinite; /\* Název animace, doba trvání, plynulost, nekonečné opakování \*/
6. /\* width: max-content; - Tohle je v tvém finálním kódu, to je dobré, aby se šířka přizpůsobila obsahu \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Úpravy v JSX/HTML a CSS:**

* **Duplikace obsahu:** Jak jsi správně poznamenal, pro plynulý nekonečný efekt je potřeba mít obsah v .marqueeContent (nebo přímo v .marqueeTrack, pokud .marqueeContent je jediný prvek) alespoň dvakrát. Tvoje struktura s .marqueeContent uvnitř .marqueeTrack je v pořádku. Ujisti se, že ten <span>SLOGAN 1</span> ... <span>SLOGAN 3</span> blok je tam opravdu dvakrát (nebo i vícekrát, pokud chceš delší pauzu, než se začne opakovat).
* **Šířka .marqueeTrack a .marqueeContent:**  
  V tvém finálním index.module.css máš pro .marqueeTrack vlastnost width: max-content;. To je dobré, protože to zajistí, že .marqueeTrack bude tak široký, jak je jeho obsah.  
  .marqueeContent má display: flex;, což je také v pořádku.

**Upravené CSS (kombinace tvého a doplnění animace):**

/\* index.module.css \*/

.marqueeSection {

padding: 20px 0;

background-color: var(--color-primary);

overflow: hidden;

color: white;

/\* Tvé další styly pro full-width, pokud je máš \*/

width: 100vw !important;

max-width: 100vw !important;

margin-left: calc(-50vw + 50%) !important;

margin-right: calc(-50vw + 50%) !important;

position: relative; /\* Pokud bys chtěl něco absolutně pozicovat \*/

box-sizing: border-box;

left: 0;

right: 0;

}

@keyframes marqueeAnimation {

0% { transform: translateX(0%); }

100% { transform: translateX(-50%); } /\* Posun o polovinu obsahu marqueeContent \*/

}

.marqueeTrack {

white-space: nowrap;

/\* display: inline-block; // Není nutné, pokud používáš width: max-content \*/

animation: marqueeAnimation 30s linear infinite; /\* Uprav rychlost (30s) podle potřeby \*/

width: max-content; /\* Důležité, aby se track přizpůsobil šířce obsahu \*/

/\* will-change: transform; // Může pomoci s výkonem animace \*/

}

.marqueeContent {

display: flex; /\* Slogany vedle sebe \*/

/\* align-items: center; // Pokud je potřeba vertikální zarovnání \*/

}

.marqueeContent span { /\* Obecný styl pro spany v marquee \*/

display: inline-flex; /\* Aby se align-items uplatnilo, pokud bys ho přidal do .marqueeContent \*/

align-items: center; /\* Pro případ, že by slogany měly různou výšku \*/

padding: 0 20px;

font-weight: 700; /\* Změněno z bold na číselnou hodnotu \*/

font-size: 1.3rem; /\* Podle tvého finálního kódu \*/

letter-spacing: 1px;

text-transform: uppercase;

}

.marqueeSeparator {

/\* padding: 0 10px; // Tvé původní \*/

color: rgba(255, 255, 255, 0.6); /\* Podle tvého finálního kódu \*/

font-size: 1.2rem; /\* Podle tvého finálního kódu \*/

padding: 0 10px; /\* Podle tvého finálního kódu \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Důležité:**

* **Doba trvání animace (30s):** Uprav si ji tak, aby rychlost textu byla příjemná. Delší čas = pomalejší pohyb.
* **Obsah .marqueeContent:** Musí být dostatečně dlouhý. Pokud je tvůj translateX(-50%), tak div s .marqueeContent musí obsahovat alespoň dvojnásobek textu, který chceš, aby se plynule opakoval. V tvém finálním kódu máš slogany zduplikované, což je správně.

Takhle by to mělo fungovat. Ta CSS animace je na to ideální.

To je blby, chtel bych, aby se to menu samo zavrelo, kdyz kliknu na jakykoliv odkaz v nem.

Tady je ten kus kodu z Navbar.tsx, kde mam to mobilni menu:

// Navbar.tsx

// ... (useState pro isMobileMenuOpen a toggleMobileMenu) ...

return (

<nav className={styles.navbar}>

{/\* ... navbarBrand ... \*/}

{/\* ... desktopNav ... \*/}

<div className={styles.mobileNav}>

<button className={styles.menuToggleButton} onClick={toggleMobileMenu}>

{isMobileMenuOpen ? <BsX size="25" /> : <BsList size="25" />}

</button>

{/\* TADY JE TO MENU, KTERÉ SE NEZAVÍRÁ \*/}

{isMobileMenuOpen && (

<div className={`${styles.mobileMenu} ${isMobileMenuOpen ? styles.open : ''}`}>

<ul className={styles.mobileMenuList}>

<li className={styles.mobileMenuItem}>

<Link href="/" className={styles.mobileMenuLink}> {/\* CHCI ABY PO KLIKU ZAVŘELO MENU \*/}

Domů

</Link>

</li>

<li className={styles.mobileMenuItem}>

<Link href="/aboutme" className={styles.mobileMenuLink}> {/\* TADY TAKY \*/}

O mně

</Link>

</li>

<li className={styles.mobileMenuItem}>

<Link href="/kontakt" className={styles.mobileMenuLink}> {/\* A TADY \*/}

Kontakt

</Link>

</li>

<li className={styles.mobileMenuItem}>

<ThemeToggle /> {/\* ThemeToggle je taky v menu, ale to asi zavírat nemusí \*/}

</li>

</ul>

</div>

)}

</div>

</nav>

);

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Jak tam jednoduse pridam to, aby se po kliknuti na <Link> zavolala ta funkce toggleMobileMenu() (nebo spis setIsMobileMenuOpen(false))?

Jasně, to je běžný požadavek na mobilní menu. Chceš, aby se po kliknutí na odkaz menu zavřelo.

Máš dvě hlavní možnosti, jak toho dosáhnout:

**1. Přidat onClick handler přímo na každý <Link> v mobilním menu:**

Toto je nejjednodušší a přímočaré řešení.

Uprav Navbar.tsx takto:

// Navbar.tsx

// ... (useState pro isMobileMenuOpen a toggleMobileMenu) ...

const closeMobileMenu = () => {

setIsMobileMenuOpen(false);

};

return (

<nav className={styles.navbar}>

{/\* ... navbarBrand ... \*/}

{/\* ... desktopNav ... \*/}

<div className={styles.mobileNav}>

<button className={styles.menuToggleButton} onClick={toggleMobileMenu}>

{isMobileMenuOpen ? <BsX size="25" color="var(--color-text-primary)"/> : <BsList size="25" color="var(--color-text-primary)"/>}

</button>

{/\* Podmíněné renderování menu je lepší dělat přes CSS třídu pro animace,

ale pokud používáš {isMobileMenuOpen && (...)}, tak onClick na Linku bude fungovat \*/}

<div className={`${styles.mobileMenu} ${isMobileMenuOpen ? styles.open : ''}`}>

{isMobileMenuOpen && ( // Pokud stále používáš toto pro renderování obsahu

<ul className={styles.mobileMenuList}>

<li className={styles.mobileMenuItem}>

<Link href="/" className={styles.mobileMenuLink} onClick={closeMobileMenu}> {/\* PŘIDÁN onClick \*/}

Domů

</Link>

</li>

<li className={styles.mobileMenuItem}>

<Link href="/aboutme" className={styles.mobileMenuLink} onClick={closeMobileMenu}> {/\* PŘIDÁN onClick \*/}

O mně

</Link>

</li>

<li className={styles.mobileMenuItem}>

<Link href="/kontakt" className={styles.mobileMenuLink} onClick={closeMobileMenu}> {/\* PŘIDÁN onClick \*/}

Kontakt

</Link>

</li>

<li className={styles.mobileMenuItem}>

{/\* ThemeToggle nemá přímý vliv na navigaci, takže zde onClick není nutný

pro zavření menu, pokud nechceš, aby i přepnutí tématu zavřelo menu.

Pokud ano, obal ThemeToggle do divu s onClick={closeMobileMenu}

nebo předej closeMobileMenu jako prop do ThemeToggle, aby ho mohl zavolat.

V tvém případě je ThemeToggle v kódu jako poslední, takže to nevadí.

\*/}

<ThemeToggle />

</li>

</ul>

)}

</div>

</div>

</nav>

);

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Vytvořil jsem pomocnou funkci closeMobileMenu, která jednoduše nastaví isMobileMenuOpen na false. Tuto funkci pak voláš v onClick atributu každého odkazu.  
V tvém kódu máš BsX size="25" color="var(--color-text-primary)" atd. Toto je správně pro konzistenci barev.

**Vylepšení s CSS animací (pokud .open třída řídí viditelnost):**

Pokud viditelnost a animaci otevírání/zavírání menu řídíš hlavně pomocí CSS třídy .open (jak máš v transform: scaleY(0); a transform: scaleY(1);), tak stačí, když isMobileMenuOpen nastavíš na false.

**Jak to funguje:**  
Když uživatel klikne na odkaz, nejprve se spustí closeMobileMenu(), což změní stav isMobileMenuOpen na false. To způsobí, že:

* Pokud menu renderuješ podmíněně ({isMobileMenuOpen && ...}), tak se celé menu odrenderuje (zmizí z DOMu).
* Pokud viditelnost řídíš třídou .open, tak se tato třída odebere a CSS animace zavření se spustí.

Obojí povede k zavření menu. Navigace na novou stránku pomocí <Link> proběhne následně.

**Důležitá poznámka k ThemeToggle:**  
Pokud máš ThemeToggle uvnitř li jako v tvém kódu a nechceš, aby kliknutí na něj zavíralo menu, tak je to v pořádku. Pokud bys chtěl, aby i přepnutí tématu zavřelo menu, musel bys onClick={closeMobileMenu} přidat i na obalující prvek ThemeToggle nebo upravit ThemeToggle tak, aby mohl volat funkci closeMobileMenu (např. předáním jako prop). V tvém případě je to poslední položka, takže to není kritické.

Tohle je nejčastější a nejjednodušší způsob, jak toho docílit.

Chtel bych to udelat jako takovej ten posuvnik (slider/carousel), aby se tam ty transformace daly prochazet, hlavne na mobilu, kde se mi to pod sebe nelibi.

V index.tsx mam neco takovyho (zjednodusene pro jednu transformaci, normalne jich tam mam vic):

// index.tsx

// ... (useState pro isMobile, currentSlide atd. tam mam z tveho kodu) ...

<section className={styles.transformationsSection} id="transformations">

<div className={styles.sectionHeader}>

<h2>ÚSPĚŠNÉ PROMĚNY</h2>

<p className={styles.sectionSubtitle}>Inspirujte se skutečnými výsledky mých klientů</p>

</div>

<div className={styles.transformationsContainer}> {/\* Tento kontejner je asi dulezity \*/}

<div

className={`${styles.transformationsGrid} ${isMobile ? styles.transformationsSlider : ''}`}

ref={transformationsRef}

onTouchStart={handleTouchStart}

onTouchMove={handleTouchMove}

onTouchEnd={handleTouchEnd}

>

{/\* JEDNA KARTA TRANSFORMACE - TOTO SE OPAKUJE \*/}

<div className={`${styles.transformationCard} ${isMobile ? styles.transformationSlide : ''}`}>

<div className={styles.transformationImages}>

<div className={styles.transformationImageBefore}>

<img src="/images/1.jpg" alt="Před transformací" />

<span className={styles.transformationLabel}>PŘED</span>

</div>

<div className={styles.transformationImageAfter}>

<img src="/images/2.jpg" alt="Po transformaci" />

<span className={styles.transformationLabel}>PO</span>

</div>

</div>

<div className={styles.transformationContent}>

<h3 className={styles.transformationName}>David, 18 let</h3>

<p className={styles.transformationText}>

"Otočil jsem svůj život..."

</p>

<div className={styles.transformationStats}>

{/\* ... statistiky ... \*/}

</div>

</div>

</div>

{/\* DALŠÍ KARTY ... \*/}

</div>

{/\* Tady jsou ty tečky a šipky, co mám v plném kódu \*/}

{isMobile && (

<div className={styles.sliderDots}>

{/\* ... mapování teček ... \*/}

</div>

)}

{/\* Na desktopu bych taky chtěl šipky, pokud to není slider jen pro mobil \*/}

{/\* <button className={`${styles.sliderArrow} ${styles.sliderArrowLeft}`}><</button> \*/}

{/\* <button className={`${styles.sliderArrow} ${styles.sliderArrowRight}`}>></button> \*/}

</div>

</section>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A v index.module.css mam pro to tyhle styly:

.transformationsSection { /\* ... \*/ }

.sectionHeader { /\* ... \*/ }

.transformationsContainer { /\* Toto je nový obalovač, který jsem viděl v tvém kódu \*/

position: relative; /\* Pro absolutní pozicování šipek \*/

width: 100%;

max-width: 1400px; /\* Nebo jiná šířka \*/

margin: 0 auto;

}

.transformationsGrid {

display: grid; /\* Na desktopu je to grid \*/

grid-template-columns: repeat(auto-fill, minmax(350px, 1fr));

gap: 40px;

margin-top: 60px;

/\* max-width: 1400px; \*/ /\* Toto je teď na .transformationsContainer \*/

/\* margin-left: auto; \*/

/\* margin-right: auto; \*/

}

/\* Styly pro mobilní slider verzi \*/

.transformationsSlider { /\* Tato třída se přidává na mobilu \*/

display: flex; /\* Místo gridu použijeme flex pro horizontální scrollování \*/

grid-template-columns: none; /\* Vypneme grid sloupce \*/

gap: 0; /\* Vypneme gap mezi slidy, pokud chceme plynulý přechod \*/

overflow-x: scroll; /\* Umožní horizontální scrollování \*/

scroll-snap-type: x mandatory; /\* Přichytávání slidů \*/

scrollbar-width: none; /\* Skryje scrollbar (Firefox) \*/

-ms-overflow-style: none; /\* Skryje scrollbar (IE/Edge) \*/

touch-action: pan-x; /\* Lepší dotykové ovládání \*/

/\* width: 100%; // Už je na .transformationsContainer \*/

padding: 20px 0 30px; /\* Odsazení pro slider \*/

}

.transformationsSlider::-webkit-scrollbar { display: none; } /\* Skryje scrollbar (Chrome/Safari) \*/

.transformationSlide { /\* Tato třída se přidává na mobilu pro každou kartu \*/

flex: 0 0 100%; /\* Každý slide zabere celou šířku kontejneru slideru \*/

scroll-snap-align: center; /\* Přichytí slide na střed \*/

/\* margin-right: 20px; // Mezera mezi slidy, pokud nechceš plynulý přechod \*/

width: calc(100% - 40px); /\* Šířka slidu s odsazením, aby bylo vidět kousek dalšího \*/

/\* Toto možná upravit podle toho, jak chceš aby to vypadalo \*/

/\* transition: all 0.3s ease; // Může způsobovat problémy se snapováním \*/

}

/\* .transformationSlide:last-child { margin-right: 0; } \*/

.transformationCard { /\* Styly pro samotnou kartu \*/

background-color: var(--color-secondary-background);

border-radius: 16px;

overflow: hidden;

/\* ... zbytek stylů karty ... \*/

}

/\* Styly pro tečky a šipky (z tvého plného kódu) \*/

.sliderDots { /\* ... \*/ }

.sliderDot { /\* ... \*/ }

.sliderDotActive { /\* ... \*/ }

.sliderArrow {

position: absolute;

top: 50%;

transform: translateY(-50%); /\* Vertikální centrování \*/

width: 40px;

height: 40px;

/\* ... další styly šipek ... \*/

z-index: 10; /\* Aby byly nad obsahem \*/

/\* display: none; // Na mobilu jsou asi schované? Nebo je chceš? \*/

}

.sliderArrowLeft { left: 5px; }

.sliderArrowRight { right: 5px; }

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Ten onTouchStart, onTouchMove, onTouchEnd a currentSlide stav uz tam nejak mam pro mobilni swipe, ale chtel bych, aby to bylo robustnejsi. A hlavne, jak udelat, aby ty sipky .sliderArrowLeft a .sliderArrowRight fungovaly i na desktopu (a mozna i na mobilu, pokud tam nebudu chtit jen swipe)? A jak zajistit, aby ten posuvnik (.transformationsSlider) ukazoval vzdycky jeden .transformationSlide hezky uprostred? To scroll-snap-type by melo pomoct, ne?

Jak mam upravit ten JavaScript v index.tsx aby ty sipky posouvaly slidy a ten useEffect co tam mam pro currentSlide a isMobile to spravne scrolloval?

Jasně, udělat z toho robustní slider chce trochu víc práce s JavaScriptem pro ovládání šipkami a synchronizaci se stavem. scroll-snap-type je skvělý základ pro plynulé přichytávání, ale pro ovládání šipkami a tečkami potřebujeme i programové scrollování.

Máš tam stavy isMobile, currentSlide, totalSlides a ref transformationsRef, což je super základ.

**1. Funkce pro ovládání šipkami:**

Do index.tsx přidej funkce, které budou měnit currentSlide:

// index.tsx

// ... (stávající importy, stavy: isMobile, currentSlide, totalSlides, transformationsRef) ...

// totalSlides by měl být nastaven na počet tvých transformačních karet

const nextSlide = () => {

setCurrentSlide(prev => (prev === totalSlides - 1 ? 0 : prev + 1)); // Cyklické procházení

// Alternativně, pokud nechceš cyklické:

// setCurrentSlide(prev => Math.min(prev + 1, totalSlides - 1));

};

const prevSlide = () => {

setCurrentSlide(prev => (prev === 0 ? totalSlides - 1 : prev - 1)); // Cyklické procházení

// Alternativně, pokud nechceš cyklické:

// setCurrentSlide(prev => Math.max(prev - 1, 0));

};

// Funkce pro přechod na specifický slide (pro tečky)

const goToSlide = (index: number) => {

setCurrentSlide(index);

};

// ... (useEffect pro detekci isMobile, handleTouchStart, Move, End) ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**2. useEffect pro scrollování na aktivní slide:**

Tvůj stávající useEffect pro scrollování je dobrý, jen ho trochu upravíme, aby fungoval i pro desktop (pokud bys chtěl i na desktopu posuv ovládaný JS, i když tam máš grid) a hlavně pro mobilní slider.

// index.tsx

// ...

useEffect(() => {

if (transformationsRef.current) {

// Výpočet pozice pro scroll

// Šířka jednoho slidu. Pokud máš .transformationSlide padding/margin, musíš to zohlednit.

// Pro tvůj CSS s width: calc(100% - 40px) to bude složitější, pokud bys chtěl přesné zarovnání

// na střed, když má slide menší šířku než kontejner.

// Pro flex: 0 0 100% by to bylo jednodušší:

// const slideWidth = transformationsRef.current.offsetWidth / totalSlides; // Toto by platilo, pokud by všechny slidy byly vidět najednou

// Pro tvůj případ, kdy je vidět jeden slide:

const sliderElement = transformationsRef.current;

if (sliderElement.children[currentSlide]) { // Zkontroluj, zda slide existuje

const slideToScrollTo = sliderElement.children[currentSlide] as HTMLElement;

// Výpočet pro scrollování, aby byl slide na začátku kontejneru (nebo na středu)

// Pro scroll-snap-type: x mandatory; a scroll-snap-align: center; by měl prohlížeč

// sám zajistit zarovnání po programovém scrollu blízko cíle.

let scrollToPosition;

if (isMobile) { // Pouze pro mobilní slider s flex a overflow-x: scroll

// transformationsRef.current.children[currentSlide] je samotný slide

// offsetLeft dá pozici slidu relativně k transformationsRef.current

scrollToPosition = slideToScrollTo.offsetLeft;

// Pokud má .transformationSlide nějaký margin, např. pro "peek" efekt,

// musíš to zohlednit, nebo nastavit kontejneru padding.

// Pro tvůj CSS .transformationSlide: width: calc(100% - 40px);

// a pokud máš padding: 20px 0 30px; na .transformationsSlider,

// pak offsetLeft by měl být relativní k vnitřku .transformationsSlider.

// Pokud máš gap: 0; a slidy jsou hned vedle sebe, offsetLeft je OK.

} else {

// Pro desktop grid toto scrollování nepotřebuješ,

// ale pokud bys chtěl i na desktopu slider, logika by byla podobná.

// Prozatím na desktopu nic neděláme.

return;

}

sliderElement.scrollTo({

left: scrollToPosition,

behavior: 'smooth' // Plynulý scroll

});

}

}

}, [currentSlide, isMobile, totalSlides]); // Přidáno totalSlides pro případ, že by se měnil dynamicky

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Vysvětlení useEffectu pro scrollování:**

* Spustí se vždy, když se změní currentSlide nebo isMobile.
* Získá referenci na kontejner slideru (transformationsRef.current).
* Získá referenci na aktuální slide (sliderElement.children[currentSlide]).
* Vypočítá scrollToPosition na základě offsetLeft aktuálního slidu. Toto je pozice levého okraje slidu. Pokud používáš scroll-snap-align: center;, prohlížeč by se měl postarat o finální vycentrování.
* Použije sliderElement.scrollTo() pro programové posunutí. behavior: 'smooth' zajistí plynulou animaci.

**3. Připojení funkcí k šipkám a tečkám v JSX:**

// index.tsx

// ...

<div className={styles.transformationsContainer}>

{/\* Šipky - zobrazíme je, pokud je více než 1 slide \*/}

{totalSlides > 1 && (

<>

<button

className={`${styles.sliderArrow} ${styles.sliderArrowLeft}`}

onClick={prevSlide}

aria-label="Předchozí proměna"

// Můžeš skrýt šipky na mobilu, pokud chceš jen swipe,

// nebo je nechat a skrýt/zobrazit pomocí CSS @media queries

// style={{ display: isMobile ? 'none' : 'flex' }} // Příklad skrývání na mobilu

>

< {/\* Nebo ikona \*/}

</button>

<button

className={`${styles.sliderArrow} ${styles.sliderArrowRight}`}

onClick={nextSlide}

aria-label="Následující proměna"

// style={{ display: isMobile ? 'none' : 'flex' }} // Příklad

>

> {/\* Nebo ikona \*/}

</button>

</>

)}

<div

className={`${styles.transformationsGrid} ${isMobile ? styles.transformationsSlider : ''}`}

ref={transformationsRef}

onTouchStart={isMobile ? handleTouchStart : undefined} // Zapnout dotyky jen na mobilu

onTouchMove={isMobile ? handleTouchMove : undefined}

onTouchEnd={isMobile ? handleTouchEnd : undefined}

>

{/\* ... mapování tvých .transformationCard ... \*/}

{/\* Příklad mapování, pokud máš data v poli 'transformationsData' \*/}

{transformationsData.map((transformace, index) => (

<div key={index} className={`${styles.transformationCard} ${isMobile ? styles.transformationSlide : ''}`}>

{/\* ... obsah karty ... \*/}

</div>

))}

</div>

{/\* Tečky - zobrazíme je, pokud je více než 1 slide \*/}

{totalSlides > 1 && ( // Zobraz tečky jen pokud je to smysluplné

<div className={styles.sliderDots}>

{Array.from({ length: totalSlides }).map((\_, index) => (

<span

key={index}

className={`${styles.sliderDot} ${currentSlide === index ? styles.sliderDotActive : ''}`}

onClick={() => goToSlide(index)}

aria-label={`Přejít na proměnu ${index + 1}`}

></span>

))}

</div>

)}

</div>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Nezapomeň si transformationsData nahradit skutečným polem dat pro tvé transformace a totalSlides nastavit na délku tohoto pole.

**4. CSS úpravy pro .transformationsSlider a .transformationSlide (pro lepší kontrolu scrollování):**

Tvůj CSS pro slider je dobrý základ. scroll-snap-type: x mandatory; a scroll-snap-align: center; jsou klíčové.

* **Šířka slidu .transformationSlide:**  
  Pokud máš width: calc(100% - 40px); a chceš, aby se slide zarovnal na střed kontejneru, tak samotný offsetLeft nemusí být dostatečný pro scrollTo, protože levý okraj slidu nebude na 0, pokud má slider padding. scroll-snap-align: center; by si s tím měl poradit, ale někdy je potřeba jemné doladění.  
  Jednodušší je často mít .transformationSlide s flex: 0 0 100%; (nebo width: 100%;) a padding řešit uvnitř slidu nebo na kontejneru slideru.

Pokud chceš efekt "nahlížení" dalšího slidu (peek effect), pak tvůj přístup s width: calc(100% - 40px); je správný, ale offsetLeft bude stále odkazovat na levý okraj tohoto už zúženého slidu. scroll-snap-align: center by se měl postarat o vycentrování tohoto zúženého slidu.

* **Problém s transition: all 0.3s ease; na .transformationSlide:**  
  Tato CSS transition může interferovat s behavior: 'smooth' u scrollTo nebo s scroll-snap efektem. Obvykle je lepší ji pro slidy, které jsou programově scrollovány a snapovány, odebrat nebo použít transition jen na specifické vlastnosti (např. opacity), pokud je to potřeba.

**Doporučení pro CSS:**

/\* index.module.css \*/

.transformationsSlider {

display: flex;

overflow-x: auto; /\* 'auto' je lepší než 'scroll', scrollbar se ukáže jen když je potřeba (i když ho skrýváš) \*/

scroll-snap-type: x mandatory;

-webkit-overflow-scrolling: touch; /\* Pro plynulejší scrollování na iOS \*/

/\* padding: 0 20px; /\* Přidat padding na kontejner slideru \*/

/\* Místo nastavování šířky slidu na calc(100% - 40px) \*/

/\* scrollbar-width: none; ... (tvé skrývací styly) ... \*/

}

.transformationSlide {

flex: 0 0 100%; /\* Každý slide zabere 100% šířky .transformationsSlider (pokud má padding) \*/

/\* Nebo pokud .transformationsSlider nemá padding: \*/

/\* flex: 0 0 calc(100% - 40px); /\* Pokud chceš peek effect a máš 20px mezeru na každé straně \*/

/\* margin: 0 20px; /\* Pak bys potřeboval margin na slidech \*/

scroll-snap-align: center;

/\* Odeber obecnou transition, pokud způsobuje problémy \*/

/\* transition: opacity 0.3s ease; /\* Jen příklad, pokud bys něco animoval \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Pokud dáš padding na .transformationsSlider (např. padding: 0 20px;), pak .transformationSlide s flex: 0 0 100% vyplní dostupnou šířku *uvnitř* tohoto paddingu, což může vytvořit ten "peek" efekt, pokud slider samotný je oříznutý (což je).

**Testování:**  
Po těchto úpravách bys měl mít:

* Funkční šipky pro posun na desktopu i mobilu.
* Funkční tečky pro přímý skok na slide.
* Dotykové ovládání (swipe) na mobilu.
* Plynulé přichytávání slidů díky scroll-snap.

Důkladně otestuj interakce na různých zařízeních. Může být potřeba doladit hodnoty offsetLeft nebo způsob výpočtu scrollToPosition, pokud by zarovnání nebylo perfektní, ale scroll-snap by měl většinu práce odvést.

Chtěl bych, aby byly na desktopu třeba trochu větší a na mobilu menší, nebo aby na mobilu nebyly vůbec, když tam mám swipe. A co když nemám dostatek transformací, aby se vůbec dalo scrollovat? Třeba jen jednu nebo dvě. Pak by ty šipky a tečky neměly být vidět.

Tady je kus toho useEffectu a JSX pro šipky z index.tsx:

// index.tsx

// ... (stavy currentSlide, isMobile, totalSlides, transformationsRef) ...

useEffect(() => {

if (transformationsRef.current && isMobile) { // Teď je to jen pro mobil

const sliderElement = transformationsRef.current;

if (sliderElement.children[currentSlide]) {

const slideToScrollTo = sliderElement.children[currentSlide] as HTMLElement;

// TOTO POČÍTÁNÍ MI PŘIJDE NEPŘESNÉ S PADDINGEM SLIDERU

const scrollToPosition = slideToScrollTo.offsetLeft;

sliderElement.scrollTo({

left: scrollToPosition,

behavior: 'smooth'

});

}

}

// Chtěl bych aby to fungovalo i pro desktop šipky, nejen pro isMobile swipe

}, [currentSlide, isMobile, totalSlides]);

// ... (JSX pro transformationsContainer) ...

{/\* ŠIPKY \*/}

{totalSlides > 1 && ( // Toto je dobrá podmínka

<>

<button

className={`${styles.sliderArrow} ${styles.sliderArrowLeft}`}

onClick={prevSlide}

aria-label="Předchozí proměna"

// Tady bych chtěl řešit zobrazení/velikost pro mobil/desktop

> < </button>

<button

className={`${styles.sliderArrow} ${styles.sliderArrowRight}`}

onClick={nextSlide}

aria-label="Následující proměna"

// Tady taky

> > </button>

</>

)}

{/\* ... zbytek slideru ... \*/}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A CSS pro šipky v index.module.css:

.sliderArrow {

position: absolute;

top: 50%;

transform: translateY(-50%);

width: 40px; /\* Velikost na desktopu? \*/

height: 40px; /\* Velikost na desktopu? \*/

background: rgba(0, 0, 0, 0.6);

color: white;

/\* ... další styly ... \*/

z-index: 10;

/\* Jak tohle nastylovat jinak pro mobil? \*/

}

.sliderArrowLeft { left: 5px; }

.sliderArrowRight { right: 5px; }

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Můžeš mi poradit, jak spolehlivěji vycentrovat ten aktivní slide pomocí scrollTo, i když má kontejner slideru padding? A jak nejlépe řešit tu responzivitu a podmíněné zobrazení šipek a teček?

Rozumím, vyladění scrollování a responzivity ovládacích prvků slideru může být detailní práce. Pojďme se na to podívat.

**1. Přesnější scrollování na střed slidu (i s paddingem kontejneru):**

Pokud má tvůj kontejner .transformationsSlider horizontální padding (např. padding: 0 20px; pro "peek" efekt) a tvé slidy .transformationSlide mají flex: 0 0 100%; (takže zabírají 100% šířky *uvnitř* toho paddingu), pak offsetLeft slidu je jeho pozice od levého okraje *vnitřku* kontejneru (tedy už za levým paddingem).

Aby se slide vycentroval, potřebuješ, aby střed slidu byl ve středu viditelné části kontejneru.

* sliderElement.clientWidth je šířka viditelné části kontejneru (bez scrollbarů, včetně paddingu).
* slideToScrollTo.offsetWidth je plná šířka samotného slidu.
* slideToScrollTo.offsetLeft je pozice levého okraje slidu od levého okraje *obsahové oblasti* (content-box) sliderElement.

Nový výpočet pro scrollToPosition v useEffect:

// index.tsx

useEffect(() => {

if (transformationsRef.current) {

const sliderElement = transformationsRef.current;

if (sliderElement.children[currentSlide]) {

const slideToScrollTo = sliderElement.children[currentSlide] as HTMLElement;

// Výpočet pro scrollování na střed slidu

// 1. Pozice levého okraje slidu

const slideLeft = slideToScrollTo.offsetLeft;

// 2. Šířka slidu

const slideWidth = slideToScrollTo.offsetWidth;

// 3. Šířka viditelné části kontejneru slideru

const sliderVisibleWidth = sliderElement.clientWidth;

// Chceme, aby střed slidu byl ve středu kontejneru.

// Střed slidu je na: slideLeft + slideWidth / 2

// Střed kontejneru je na: sliderVisibleWidth / 2

// Hledáme takové `scrollLeft`, aby:

// scrollLeft + sliderVisibleWidth / 2 = slideLeft + slideWidth / 2

// scrollLeft = slideLeft + slideWidth / 2 - sliderVisibleWidth / 2

// scrollLeft = slideLeft - (sliderVisibleWidth / 2) + (slideWidth / 2)

const scrollToPosition = slideLeft - (sliderVisibleWidth / 2) + (slideWidth / 2);

// Ujisti se, že scrollToPosition není menší než 0 nebo větší než maximální scroll

const maxScrollLeft = sliderElement.scrollWidth - sliderVisibleWidth;

const finalScrollToPosition = Math.max(0, Math.min(scrollToPosition, maxScrollLeft));

// Tento useEffect by se měl spouštět pro jakoukoliv interakci měnící currentSlide,

// nejen pro isMobile, pokud chceš, aby fungoval i s desktopovými šipkami.

// Podmínku isMobile můžeš odstranit, pokud chceš jednotné chování.

// if (isMobile) { // Tuto podmínku můžeš odstranit nebo upravit

sliderElement.scrollTo({

left: finalScrollToPosition,

behavior: 'smooth'

});

// }

}

}

}, [currentSlide, totalSlides]); // isMobile už tu není potřeba, pokud to má být obecné

// totalSlides je tu pro případ, že by se dynamicky měnil počet

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Vysvětlení výpočtu:**  
Cílem je dostat střed slidu na střed viditelné části slideru. scrollLeft hodnota, kterou hledáme, je taková, která posune viewport slideru tak, aby se to stalo.

* slideLeft - (sliderVisibleWidth / 2): Posune levý okraj slidu na pozici, kde by byl, kdyby byl jeho střed na levém okraji viewportu.
* + (slideWidth / 2): Přidá polovinu šířky slidu, čímž se efektivně posune viewport tak, aby střed slidu byl ve středu viewportu.
* finalScrollToPosition: Omezuje scroll, aby nepřesáhl hranice obsahu.

**Důležité:** Tento výpočet předpokládá, že scroll-snap-align: center; na slidech a scroll-snap-type: x mandatory; na slideru stále fungují a pomáhají "dokončit" snapování po programovém scrollu. Někdy může behavior: 'smooth' a scroll-snap trochu soupeřit. Pokud by to dělalo problémy, můžeš zkusit behavior: 'auto' a nechat snapování dělat svou práci, nebo dočasně odstranit scroll-snap-type během programového scrollování (což je složitější). Obvykle ale smooth a snap spolupracují dobře.

**2. Responzivita a podmíněné zobrazení šipek a teček:**

* **Podmíněné zobrazení na základě totalSlides:**  
  To už máš správně: {totalSlides > 1 && ( ... )}. Tím zajistíš, že se ovládací prvky zobrazí jen tehdy, když je alespoň jeden další slide, na který se dá přejít.
* **Responzivní šipky (CSS):**  
  Pro různé velikosti šipek na mobilu a desktopu použij media queries v index.module.css.
* /\* index.module.css \*/
* .sliderArrow {
* position: absolute;
* top: 50%;
* transform: translateY(-50%);
* background: rgba(0, 0, 0, 0.5); /\* Mírně světlejší pro lepší viditelnost \*/
* color: white;
* border: none;
* border-radius: 50%; /\* Kulaté šipky vypadají často lépe \*/
* cursor: pointer;
* z-index: 10;
* display: flex; /\* Pro centrování ikony/textu uvnitř \*/
* align-items: center;
* justify-content: center;
* transition: background-color 0.3s ease, opacity 0.3s ease;
* opacity: 0.7; /\* Lehce průhledné, dokud není hover \*/
* /\* Desktop velikost \*/
* width: 50px;
* height: 50px;
* font-size: 1.5rem; /\* Velikost textu/ikony šipky \*/
* }
* .sliderArrow:hover {
* background-color: rgba(0, 0, 0, 0.8);
* opacity: 1;
* }
* .sliderArrowLeft {
* left: 15px; /\* Dále od kraje na desktopu \*/
* }
* .sliderArrowRight {
* right: 15px; /\* Dále od kraje na desktopu \*/
* }
* /\* Šipky pro mobil - menší a blíže ke kraji, nebo skryté \*/
* @media (max-width: 768px) {
* .sliderArrow {
* /\* Pokud je chceš menší: \*/
* width: 35px;
* height: 35px;
* font-size: 1rem;
* /\* Pokud je chceš blíže ke kraji: \*/
* /\* left: 5px; (pro levou) \*/
* /\* right: 5px; (pro pravou) \*/
* }
* .sliderArrowLeft { left: 10px; } /\* Uprav podle potřeby \*/
* .sliderArrowRight { right: 10px; } /\* Uprav podle potřeby \*/
* /\* Pokud je chceš na mobilu úplně skrýt, když je swipe: \*/
* /\* .sliderArrow { display: none; } \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Můžeš se rozhodnout, zda šipky na mobilu zmenšíš, nebo je úplně skryješ (display: none;), pokud spoléháš primárně na swipe.

* **Podmíněné zobrazení šipek v JSX na základě isMobile (pokud chceš v JS):**  
  Pokud bys chtěl skrývání řídit čistě z JavaScriptu (např. zobrazit jen když !isMobile):
* {totalSlides > 1 && !isMobile && ( // Přidána podmínka !isMobile
* <>
* <button className={`${styles.sliderArrow} ${styles.sliderArrowLeft}`} onClick={prevSlide}><</button>
* <button className={`${styles.sliderArrow} ${styles.sliderArrowRight}`} onClick={nextSlide}>></button>
* </>

)}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Obecně je pro responzivní zobrazení/skrývání prvků lepší použít CSS media queries, protože se to děje bez nutnosti re-renderu React komponenty jen kvůli změně velikosti okna. Ale pokud je logika složitější, JS může být nutný.

**3. Logika pro totalSlides <= 1:**  
Podmínka totalSlides > 1 pro zobrazení šipek a teček je správná. Pokud je jen jeden slide (nebo žádný), ovládací prvky se nezobrazí.

**Další vylepšení:**

* **Deaktivace šipek na koncích:** Pokud nechceš cyklické procházení, můžeš šipkám přidat disabled atribut, když jsi na prvním/posledním slidu, a nastylovat .sliderArrow:disabled.
* // V JSX pro levou šipku:
* // disabled={currentSlide === 0 && !cyklickeProchazeni}
* // V JSX pro pravou šipku:

// disabled={currentSlide === totalSlides - 1 && !cyklickeProchazeni}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

* **aria-hidden a tabindex pro šipky:** Pokud jsou šipky skryty (např. na mobilu), zvaž přidání aria-hidden="true" a tabindex="-1", aby nebyly dostupné pro screen readery a klávesnicovou navigaci.

Testuj důkladně na různých zařízeních a šířkách. Centrování pomocí scrollTo může někdy vyžadovat drobné korekce v závislosti na přesném box modelu tvých slidů a kontejneru (padding, border, margin). Debugování ve vývojářských nástrojích a logování hodnot slideLeft, slideWidth, sliderVisibleWidth a finalScrollToPosition ti pomůže pochopit, co se děje.

Chtel bych aby ta aktivni tecka byla nejak vyraznejsi, treba jina barva nebo vetsi. A aby ty neaktivni byly taky trochu hezci.

Tady mam to CSS pro ty tecky z index.module.css:

.sliderDots {

display: flex;

justify-content: center;

margin-top: 30px; /\* Mozna moc velka mezera? \*/

gap: 10px;

}

.sliderDot {

width: 12px;

height: 12px;

border-radius: 50%;

background-color: rgba(0, 0, 0, 0.15); /\* Tohle je ta hnusna seda \*/

cursor: pointer;

transition: all 0.3s ease;

border: 2px solid transparent; /\* Proc tu mam transparentni border? \*/

}

/\* Tady je ten aktivni, ale moc to nedela rozdil \*/

.sliderDotActive {

background-color: var(--color-primary); /\* Tohle je sice primarni barva, ale porad to neni ono \*/

transform: scale(1.2); /\* Jenom se trochu zvetsi \*/

box-shadow: 0 0 10px rgba(0, 123, 255, 0.4); /\* Ten stin je skoro neviditelny \*/

}

/\* A jeste mam v globals.css tohle pro dark mode, to by melo fungovat i tady ne? \*/

/\* :global(.dark) .sliderDot { background-color: rgba(255, 255, 255, 0.15); } \*/

/\* :global(.dark) .sliderDotActive { box-shadow: 0 0 10px rgba(0, 123, 255, 0.6); } \*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

A v index.tsx se ty tecky generujou takhle:

// ...

{totalSlides > 1 && (

<div className={styles.sliderDots}>

{Array.from({ length: totalSlides }).map((\_, index) => (

<span

key={index}

className={`${styles.sliderDot} ${currentSlide === index ? styles.sliderDotActive : ''}`}

onClick={() => goToSlide(index)}

aria-label={`Přejít na proměnu ${index + 1}`}

></span>

))}

</div>

)}

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Co s tema teckama muzu udelat aby vypadaly lip? A ten border: 2px solid transparent; na .sliderDot je tam k necemu? Mozna by se dal vyuzit?

Jasně, ty tečky (dots/pagination) u slideru jsou důležitý vizuální prvek. Pojďme je vylepšit.

**1. Vylepšení neaktivních teček (.sliderDot):**

* **Barva:** Místo rgba(0, 0, 0, 0.15) můžeš použít světlejší šedou nebo barvu, která lépe ladí s tvým designem. Můžeš taky použít CSS proměnnou, pokud máš nějakou vhodnou.
* **Transparentní border:** Ten border: 2px solid transparent; je tam pravděpodobně proto, aby se při přidání borderu aktivní tečce nerozhodil layout (aby tečky neposkakovaly kvůli změně velikosti). To je dobrá technika.
* **Velikost:** 12px je celkem standardní, ale můžeš zkusit menší (10px) nebo větší (14px) podle toho, co se ti líbí.

Upravený .sliderDot:

.sliderDot {

width: 10px; /\* Zkusíme menší \*/

height: 10px; /\* Zkusíme menší \*/

border-radius: 50%;

background-color: var(--color-input-border); /\* Použijeme barvu borderu inputů pro konzistenci, nebo jinou světle šedou \*/

/\* V dark modu by se měla tato proměnná taky změnit, pokud máš správně globals.css \*/

cursor: pointer;

transition: background-color 0.3s ease, transform 0.3s ease; /\* Přidána transformace pro hover \*/

border: 2px solid transparent; /\* Necháme pro stabilitu layoutu \*/

margin: 0 4px; /\* Menší gap přes margin, pokud by gap property dělala neplechu s bordeerm \*/

}

.sliderDot:hover { /\* Přidáme hover efekt pro neaktivní tečky \*/

background-color: var(--color-text-secondary); /\* Ztmavne při hoveru \*/

transform: scale(1.1); /\* Lehce se zvětší \*/

}

/\* Přizpůsobení pro dark mode pro .sliderDot, pokud by var(--color-input-border) nebyla dostatečně kontrastní \*/

:global(.dark-mode) .sliderDot {

background-color: #555; /\* Tmavší šedá pro dark mode \*/

}

:global(.dark-mode) .sliderDot:hover {

background-color: #777;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Místo .sliderDots { gap: 10px; } můžeš použít margin na .sliderDot, jak je naznačeno výše (margin: 0 4px;), abys měl jemnější kontrolu nad mezerami, zvláště pokud by border nějak ovlivňoval gap.

**2. Vylepšení aktivní tečky (.sliderDotActive):**

* **Výraznější odlišení:** Místo pouhé změny barvy a lehkého zvětšení můžeme udělat víc.
  + **Plný border:** Místo background-color můžeme použít border a vnitřek nechat třeba transparentní nebo jen lehce vyplněný.
  + **Větší zvětšení:** transform: scale(1.4);
  + **Výraznější stín:** Stín může být barevný (v odstínu primární barvy).

Upravený .sliderDotActive:

.sliderDotActive {

background-color: var(--color-primary); /\* Stále primární barva jako základ \*/

/\* Alternativa: Použít border a transparentní vnitřek \*/

/\* background-color: transparent; \*/

/\* border: 2px solid var(--color-primary); \*/

transform: scale(1.3); /\* Výraznější zvětšení \*/

box-shadow: 0 0 8px 2px var(--color-input-focus-shadow); /\* Použijeme stín podobný focusu inputů,

ale můžeš si vytvořit vlastní výraznější,

např. 0 0 12px var(--color-primary); \*/

}

/\* Pokud bys šel cestou s borderem pro aktivní tečku: \*/

/\*

.sliderDot {

width: 10px;

height: 10px;

border-radius: 50%;

background-color: var(--color-input-border);

cursor: pointer;

border: 2px solid var(--color-input-border); // Border má stejnou barvu jako pozadí, takže není vidět

transition: all 0.3s ease;

}

.sliderDotActive {

background-color: transparent; // Nebo var(--color-primary-light) pro jemnou výplň

border-color: var(--color-primary); // Výrazný border v primární barvě

transform: scale(1.1); // Jen lehké zvětšení, protože border už ji opticky zvětší

}

\*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**3. Úprava .sliderDots kontejneru:**  
Můžeš zmenšit margin-top, pokud je moc velký, a zajistit, aby byly tečky vždy hezky vycentrované.

.sliderDots {

display: flex;

justify-content: center;

align-items: center; /\* Pro případ, že by tečky měly různou výšku po transformaci \*/

margin-top: 25px; /\* Trochu menší odsazení \*/

/\* gap: 8px; // Pokud nepoužíváš margin na .sliderDot \*/

padding-bottom: 10px; /\* Malý padding dole, aby nebyly nalepené na další sekci \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Příklad celkového vzhledu teček (kombinace):**

/\* index.module.css \*/

.sliderDots {

display: flex;

justify-content: center;

align-items: center;

margin-top: 25px;

padding-bottom: 10px;

}

.sliderDot {

width: 10px;

height: 10px;

border-radius: 50%;

background-color: var(--color-input-border); /\* Světle šedá \*/

margin: 0 5px; /\* Mezery mezi tečkami \*/

cursor: pointer;

transition: all 0.3s ease;

border: 2px solid transparent; /\* Aby se neměnil rozměr při přidání borderu aktivní \*/

}

:global(.dark-mode) .sliderDot {

background-color: #4a4a4a; /\* Tmavší šedá pro dark mode \*/

}

.sliderDot:not(.sliderDotActive):hover { /\* Hover pro neaktivní \*/

background-color: var(--color-text-secondary);

transform: scale(1.15);

}

:global(.dark-mode) .sliderDot:not(.sliderDotActive):hover {

background-color: #6e6e6e;

}

.sliderDotActive {

background-color: var(--color-primary);

transform: scale(1.35);

box-shadow: 0 0 10px 0px var(--color-primary-hover); /\* Stín v barvě primary-hover \*/

/\* Pokud bys chtěl "dutou" aktivní tečku: \*/

/\* background-color: transparent; \*/

/\* border: 2px solid var(--color-primary); \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Ten border: 2px solid transparent; na .sliderDot je tam přesně k tomu, aby když aktivní tečce přidáš viditelný border (např. pro "dutý" vzhled), tak se neaktivní tečky nerozjedou, protože už tu "neviditelnou" rezervu na border mají. V tomto případě, kdy měníš hlavně background-color a transform: scale, ten transparentní border není tak kritický, ale nevadí tam.

Zkus si pohrát s těmito hodnotami, hlavně s barvami a transform: scale pro aktivní tečku, abys dosáhl vzhledu, který se ti bude líbit. Ten stín u aktivní tečky může taky hodně pomoct.

A taky, ta barva pozadi .marqueeSection je ted var(--color-primary). Co kdyby to byl nejakej gradient nebo tak neco?

Tady je ten relevantni kus CSS z index.module.css:

.marqueeSection {

padding: 20px 0;

background-color: var(--color-primary); /\* Tohle bych chtel zmenit \*/

overflow: hidden;

color: white;

/\* ... styly pro full-width ... \*/

width: 100vw !important; /\* Force full viewport width \*/

max-width: 100vw !important;

margin-left: calc(-50vw + 50%) !important;

margin-right: calc(-50vw + 50%) !important;

/\* ... \*/

}

.marqueeTrack {

/\* ... animace ... \*/

width: max-content;

}

.marqueeContent {

display: flex;

}

.marqueeContent span { /\* Obecny styl pro spany, tedy i slogany \*/

display: inline-flex;

align-items: center;

padding: 0 20px;

color: white; /\* Tohle je OK \*/

font-weight: 700;

font-size: 1.3rem; /\* Mozna vetsi? Jiny font? \*/

letter-spacing: 1px;

text-transform: uppercase;

}

.marqueeSeparator { /\* Tady je ta tecka \*/

color: rgba(255, 255, 255, 0.6);

font-size: 1.2rem;

padding: 0 10px;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

A v index.tsx ten .marqueeContent vypada nejak takhle:

<div className={styles.marqueeContent}>

<span>PŘEKONEJ SVÉ LIMITY</span>

<span className={styles.marqueeSeparator}>•</span>

<span>STAŇ SE NEJLEPŠÍ VERZÍ SEBE SAMA</span>

<span className={styles.marqueeSeparator}>•</span>

{/\* ... dalsi slogany a separatory ... \*/}

</div>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Nemas nejaky napady, jak to ozivit? Hlavne ten font, separator a pozadi.

Jasně, pojďme tu marquee sekci trochu "vytunit", aby byla poutavější!

**1. Vylepšení textu sloganů (.marqueeContent span):**

* **Font:** Zkus použít nějaký výraznější nebo modernější font. Pokud máš v projektu definovaný nějaký speciální font pro nadpisy (např. přes @font-face v globals.css nebo importovaný Google Font), můžeš ho zde použít. Pokud ne, můžeš zkusit nějaký systémový, který vypadá dobře ve velkém a uppercase.
* **Velikost:** 1.3rem je slušná velikost, ale můžeš experimentovat s větší, např. 1.5rem nebo 1.8rem, podle toho, kolik textu tam máš a jak rychle se pohybuje.
* **Styl písma:** Můžeš zvážit font-style: italic; pro dynamičtější vzhled, nebo font-weight: 800; či 900; pro ještě větší tučnost.
* **Textový stín:** Jemný textový stín může textu dodat hloubku.

Příklad úpravy pro .marqueeContent span:

.marqueeContent span { /\* Cílíme na spany, které obsahují slogany, ne na separátory \*/

display: inline-flex;

align-items: center;

padding: 0 25px; /\* Trochu větší padding \*/

color: white;

font-family: 'Impact', 'Arial Black', sans-serif; /\* Příklad výraznějšího fontu, nahraď svým \*/

font-weight: 700; /\* Nebo 800 \*/

font-size: 1.6rem; /\* Větší písmo \*/

letter-spacing: 1.5px; /\* Větší prostrkání \*/

text-transform: uppercase;

/\* text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.3); \*/ /\* Volitelný jemný stín \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Pokud chceš odlišit styl samotných sloganů od stylu separátorů (pokud bys separátory nechal jako span), musel bys sloganům dát specifickou třídu nebo použít selektor typu .marqueeContent > span:not(.marqueeSeparator). Ale jelikož separátor má vlastní třídu, tak se ti styly nebudou plést.

**2. Vylepšení oddělovače (.marqueeSeparator):**

Místo textové tečky • můžeš použít:

* **SVG Ikonu:** Ideální pro ostré a škálovatelné ikonky. Můžeš použít ikonu z react-icons nebo vlastní SVG.
* **CSS Tvary:** Pomocí CSS ::before nebo ::after pseudoelementů můžeš vytvořit různé tvary (kosočtverec, hvězdička atd.).
* **Obrázek:** Malý PNG nebo SVG obrázek.

**Varianta A: Použití SVG ikony z react-icons (např. šipka nebo hvězda):**  
V index.tsx bys musel importovat ikonu a vložit ji místo tečky:

// index.tsx

import { FaChevronRight, FaStar } from 'react-icons/fa'; // Příklad ikon

// ...

<div className={styles.marqueeContent}>

<span>PŘEKONEJ SVÉ LIMITY</span>

<span className={styles.marqueeSeparator}>

<FaChevronRight /> {/\* Nebo <FaStar /> \*/}

</span>

<span>STAŇ SE NEJLEPŠÍ VERZÍ SEBE SAMA</span>

<span className={styles.marqueeSeparator}>

<FaChevronRight />

</span>

{/\* ... \*/}

</div>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A CSS pro .marqueeSeparator by se upravil, aby správně formátoval ikonu:

.marqueeSeparator {

display: inline-flex; /\* Aby se ikona dobře zarovnala \*/

align-items: center;

justify-content: center;

color: rgba(255, 255, 255, 0.7); /\* Barva ikony \*/

font-size: 1rem; /\* Velikost ikony (react-icons obvykle reagují na font-size) \*/

padding: 0 15px; /\* Odsazení kolem ikony \*/

/\* Můžeš přidat transformaci, pokud chceš ikonu otočit nebo upravit \*/

/\* transform: scale(0.8); \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Varianta B: CSS Tvar (např. kosočtverec):**  
HTML zůstane stejné (s tečkou nebo bez ní, text uvnitř spanu nebude vidět).

.marqueeSeparator {

display: inline-block; /\* Nebo inline-flex \*/

align-items: center;

width: 10px; /\* Šířka kosočtverce \*/

height: 10px; /\* Výška kosočtverce \*/

background-color: rgba(255, 255, 255, 0.6);

margin: 0 20px; /\* Odsazení \*/

transform: rotate(45deg); /\* Otočení čtverce na kosočtverec \*/

position: relative; /\* Pokud bys chtěl přidat další detaily \*/

top: -2px; /\* Malá korekce vertikálního zarovnání, pokud je potřeba \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**3. Pozadí sekce (.marqueeSection):**

Místo jedné barvy můžeš použít gradient.

.marqueeSection {

padding: 20px 0;

/\* background-color: var(--color-primary); // Původní \*/

background: linear-gradient(90deg, var(--color-primary-hover) 0%, var(--color-primary) 50%, var(--color-primary-hover) 100%);

/\* Příklad gradientu: začíná tmavší modrou, přechází do světléjší primární a zase do tmavší.

Uprav si barvy a úhel (90deg) podle sebe. \*/

/\* Nebo diagonální:

background: linear-gradient(135deg, #0056b3 0%, #007bff 100%);

\*/

overflow: hidden;

color: white;

/\* ... zbytek tvých stylů pro full-width atd. ... \*/

/\* Volitelný border pro oddělení od ostatních sekcí \*/

border-top: 2px solid rgba(255, 255, 255, 0.1);

border-bottom: 2px solid rgba(255, 255, 255, 0.1);

}

/\* V tvém finálním kódu máš i toto pro light mode, což je dobré pro konzistenci,

pokud bys chtěl jiný gradient/barvu v light modu: \*/

:global(.light) .marqueeSection {

/\* Například použít tmavší variantu primární barvy pro light mode,

aby byl stále kontrast s bílým textem, nebo úplně jiný styl \*/

background: linear-gradient(90deg, #0c47a1 0%, #1976d2 100%); /\* Příklad pro light mode \*/

/\* Nebo použít --color-primary-dark z globals.css, pokud tam máš takovou proměnnou \*/

/\* background-color: var(--color-primary-dark); \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Celkový dojem:**  
Kombinací výraznějšího fontu pro slogany, zajímavějšího oddělovače (např. SVG ikony) a gradientu na pozadí můžeš tuto sekci výrazně oživit. Neboj se experimentovat s různými fonty a ikonami, dokud nenajdeš kombinaci, která se ti líbí a ladí se zbytkem webu.  
Například, pokud je tvůj web zaměřený na energii a dynamiku, šipka jako oddělovač může být dobrá volba. Pokud je to víc o úspěchu a kvalitě, malá hvězdička by mohla fungovat.

Napadlo me, ze bych mohl mit treba 5 hvezdicek a podle hodnoceni klienta (rekneme ze bych mel nekde ulozene hodnoceni 1-5) by se jich prislusny pocet vybarvilo (treba tou zlatou) a zbytek by zustal sedy nebo jenom ohraniceny. A kdyz na ne najedu mysi, tak by se taky mohly nejak zvyraznit.

Tady je ten JSX pro jednu referenci v aboutme.tsx:

// aboutme.tsx

// ...

<div className={styles.testimonialCard}>

<div className={styles.testimonialStars}>

{/\* TADY JSOU TY HVĚZDIČKY - TEĎ JE JICH TAM NAKÓDOVÁNO 5 \*/}

<FaStar /> <FaStar /> <FaStar /> <FaStar /> <FaStar />

</div>

<p className={styles.testimonialText}>

"Super trenink, diky moc!"

</p>

<div className={styles.testimonialAuthor}>

<span className={styles.authorName}>Petr Novak</span>

{/\* <span className={styles.authorInfo}>Info o klientovi</span> \*/}

</div>

</div>

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A CSS z aboutme.module.css:

.testimonialCard {

/\* ... styly karty ... \*/

border-left: 4px solid var(--color-primary);

}

.testimonialStars {

color: #FFD700; /\* Zlata barva pro vsechny hvezdy \*/

margin-bottom: 15px;

font-size: 1.2rem; /\* Velikost hvezd \*/

display: flex; /\* Aby byly vedle sebe \*/

gap: 5px; /\* Mezera mezi hvezdami \*/

}

.testimonialText { /\* ... \*/ }

.authorName { /\* ... \*/ }

.authorInfo { /\* ... \*/ }

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Jak bych mel upravit to JSX a CSS, abych tam mohl dynamicky zobrazovat pocet vybarvenych hvezd? A jak udelat ten hover efekt na jednotlivy hvezdicky? Musel bych asi kazdou hvezdu obalit zvlast do nejakyho spanu nebo tak neco?

Jasně, interaktivní hvězdičkové hodnocení vypadá mnohem lépe. Pojďme na to.

**1. Dynamické generování hvězdiček v JSX:**

Nejprve potřebuješ způsob, jak v datech pro každou referenci uchovávat hodnocení (číslo od 1 do 5). Pak v JSX vygeneruješ hvězdičky v cyklu.

* **Předpokládejme, že máš data pro reference nějak takto:**
* const testimonialsData = [
* {
* id: 1,
* text: "Super trenink, diky moc!",
* author: "Petr Novak",
* rating: 5, // Hodnocení klienta
* },
* {
* id: 2,
* text: "Pomohl mi zhubnout a citim se skvele.",
* author: "Jana Mala",
* rating: 4,
* },
* // ... další reference

];

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).JavaScript

* **Úprava JSX v aboutme.tsx pro generování hvězdiček:**
* // aboutme.tsx
* // ...
* {testimonialsData.map(testimonial => ( // Procházíš data referencí
* <div key={testimonial.id} className={styles.testimonialCard}>
* <div className={styles.testimonialStars}>
* {/\* Dynamické generování hvězdiček \*/}
* {[...Array(5)].map((\_, index) => {
* const starValue = index + 1;
* return (
* <span
* key={index}
* className={`${styles.starIcon} ${starValue <= testimonial.rating ? styles.starActive : styles.starInactive}`}
* // Pro případný hover efekt na jednotlivé hvězdy, pokud bys chtěl měnit barvu při hoveru celé skupiny
* // onMouseEnter={() => handleStarHover(starValue)}
* // onMouseLeave={() => handleStarHover(0)} // Reset hoveru
* // onClick={() => handleStarClick(starValue)} // Pokud bys chtěl i klikací hodnocení
* >
* {/\* Můžeš použít plnou hvězdu a měnit její barvu přes CSS,
* nebo dvě různé ikony (plnou a prázdnou) \*/}
* <FaStar />
* </span>
* );
* })}
* </div>
* <p className={styles.testimonialText}>
* "{testimonial.text}"
* </p>
* <div className={styles.testimonialAuthor}>
* <span className={styles.authorName}>{testimonial.author}</span>
* {/\* <span className={styles.authorInfo}>Info o klientovi</span> \*/}
* </div>
* </div>
* ))}

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Vysvětlení JSX:**

* + [...Array(5)] vytvoří pole s 5 prvky (pro 5 hvězdiček).
  + Mapujeme přes toto pole, index jde od 0 do 4.
  + starValue = index + 1 je hodnota hvězdy (1 až 5).
  + Každá <FaStar /> je obalena do <span> s třídou .starIcon.
  + Podmíněně přidáváme třídu .starActive (pokud je starValue menší nebo rovno hodnocení klienta) nebo .starInactive.

**2. CSS styly pro aktivní a neaktivní hvězdičky (aboutme.module.css):**

/\* aboutme.module.css \*/

.testimonialStars {

/\* display: flex; \*/ /\* Už máš \*/

/\* gap: 5px; \*/ /\* Už máš \*/

margin-bottom: 15px;

font-size: 1.3rem; /\* Mírně větší hvězdy \*/

line-height: 1; /\* Pro lepší zarovnání, pokud by byly problémy \*/

}

.starIcon { /\* Obalovač pro každou hvězdu \*/

display: inline-block; /\* Nebo inline-flex \*/

cursor: default; /\* Výchozí kurzor, pokud nejsou klikací \*/

transition: color 0.2s ease, transform 0.2s ease;

color: #e0e0e0; /\* Výchozí barva pro všechny (neaktivní/placeholder) \*/

}

.starIcon svg { /\* Cílení na samotnou SVG ikonu uvnitř spanu \*/

display: block; /\* Odstraní případný malý prostor pod ikonou \*/

}

.starActive { /\* Aktivní (vybarvená) hvězda \*/

color: #FFD700; /\* Zlatá barva \*/

/\* transform: scale(1.05); // Jemné zvětšení pro aktivní, pokud chceš \*/

}

.starInactive { /\* Neaktivní (šedá) hvězda \*/

color: var(--color-input-border); /\* Světle šedá, stejná jako border inputů \*/

/\* Nebo jiná decentní šedá: #ccc; \*/

/\* opacity: 0.7; // Můžeš je udělat i lehce průhledné \*/

}

:global(.dark-mode) .starInactive {

color: #555; /\* Tmavší šedá pro neaktivní hvězdy v dark modu \*/

}

/\* Hover efekt na jednotlivé hvězdy (pokud je chceš interaktivní při najetí) \*/

.testimonialStars:hover .starIcon { /\* Když najedu na kontejner hvězd \*/

/\* color: #FFD700; \*/ /\* Všechny se obarví zlatě - jednoduchý hover \*/

/\* opacity: 0.7; \*/

}

.testimonialStars .starIcon:hover { /\* Když najedu na konkrétní hvězdu \*/

transform: scale(1.25); /\* Zvětší se hvězda pod kurzorem \*/

/\* color: #ffc107; // Můžeš mírně změnit odstín zlaté \*/

}

/\* Pokud bys chtěl efekt, že se při hoveru na hvězdu obarví ona a všechny před ní:

To by vyžadovalo trochu JavaScriptu pro přidávání/odebírání tříd při hoveru,

nebo velmi složité CSS s obecnými sourozeneckými selektory (~), což nedoporučuji

pro tento případ kvůli složitosti a potenciálním problémům s výkonem.

Pro zobrazení hodnocení stačí třídy .starActive a .starInactive.

\*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Jak to funguje:**

1. V JSX se pro každou referenci vygeneruje 5 <span> elementů, každý obsahující ikonu <FaStar />.
2. Každý span.starIcon dostane třídu .starActive nebo .starInactive podle toho, zda jeho "hodnota" spadá do hodnocení klienta.
3. CSS pak definuje, jak vypadá .starActive (zlatá) a .starInactive (šedá).
4. Přidali jsme jednoduchý hover efekt na .starIcon:hover, který zvětší hvězdu pod kurzorem.

**Interaktivní hover (obarvení hvězd až po tu, na které je myš):**  
Pokud bys chtěl ten pokročilejší hover efekt (že když najedeš na třetí hvězdu, tak se první tři obarví), to už by bylo složitější udělat čistě v CSS tak, aby to bylo robustní a dobře fungovalo s Reactem. Vyžadovalo by to JavaScript:

* Stav v komponentě pro "hoverované hodnocení".
* onMouseEnter a onMouseLeave eventy na každé hvězdě, které by aktualizovaly tento stav.
* Podmíněné třídy nebo styly na hvězdách založené jak na reálném testimonial.rating, tak na tomto dočasném "hoverovaném hodnocení".

Pro pouhé zobrazení uloženého hodnocení je ale výše uvedené řešení s třídami .starActive a .starInactive plně dostačující a jednoduché.

Zkus implementovat to dynamické generování a základní CSS styly. Mělo by to vypadat mnohem lépe!

Chtel bych, aby se ty cisla v .statNumber (jako "500+", "10+") nejak animovane "napocitaly" od nuly do te cilove hodnoty, kdyz uzivatel na tu sekci doscrolluje. Aby to nebylo jen tak plácnutý tam.

Tady je ten JSX pro statistiky z aboutme.tsx:

// aboutme.tsx

// ...

<section className={styles.statsSection}> {/\* Tato sekce má barevné pozadí \*/}

<div className={styles.statsContainer}>

<div className={styles.statItem}>

<span className={styles.statNumber}>500+</span> {/\* TOTO ČÍSLO CHCI ANIMOVAT \*/}

<span className={styles.statText}>Spokojených klientů</span>

</div>

<div className={styles.statItem}>

<span className={styles.statNumber}>10+</span> {/\* TOTO TAKY \*/}

<span className={styles.statText}>Let zkušeností</span>

</div>

<div className={styles.statItem}>

<span className={styles.statNumber}>99%</span> {/\* A TOTO \*/}

<span className={styles.statText}>Úspěšnost klientů</span>

</div>

<div className={styles.statItem}>

<span className={styles.statNumber}>24/7</span> {/\* TOTO MOŽNÁ NE, JE TO TEXT \*/}

<span className={styles.statText}>Podpora klientů</span>

</div>

</div>

</section>

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A CSS z aboutme.module.css:

.statsSection {

padding: 50px 20px;

background-color: var(--color-primary); /\* Výrazné pozadí \*/

color: white;

}

.statsContainer {

display: flex;

justify-content: space-around;

align-items: center;

flex-wrap: wrap;

gap: 20px;

max-width: 900px;

margin: 0 auto;

}

.statItem {

text-align: center;

min-width: 150px;

}

.statNumber {

font-size: 2.8rem;

font-weight: 800;

display: block; /\* Aby byl text pod ním \*/

}

.statText {

font-size: 1rem;

text-transform: uppercase;

opacity: 0.9;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Jak bych mohl udelat tu animaci napocitavani cisel? Asi to bude chtit nejaky JavaScript, co? A jak zaridit, aby se to spustilo, az kdyz je ta sekce viditelna na obrazovce? To "24/7" asi animovat nechci, to neni uplne cislo.

Máš pravdu, animované načítání čísel (často nazývané "counter-up" efekt) je populární způsob, jak oživit statistiky. A ano, bude to vyžadovat JavaScript.

Zde je postup, jak toho dosáhnout:

**1. Vytvoření vlastní React Hook pro animaci čísla a detekci viditelnosti:**

Je dobré si logiku pro animaci a detekci viditelnosti oddělit do znovupoužitelného hooku.

* **Hook pro detekci viditelnosti (Intersection Observer API):**  
  Vytvoříme hook useIntersectionObserver, který nám řekne, zda je element viditelný.
* // hooks/useIntersectionObserver.ts (vytvoř nový soubor)
* import { useState, useEffect, RefObject } from 'react';
* interface IntersectionObserverOptions {
* root?: Element | null;
* rootMargin?: string;
* threshold?: number | number[];
* }
* function useIntersectionObserver(
* elementRef: RefObject<Element>,
* {
* threshold = 0.1, // Jaké procento elementu musí být viditelné
* root = null,
* rootMargin = '0%',
* }: IntersectionObserverOptions
* ): boolean {
* const [isVisible, setIsVisible] = useState(false);
* useEffect(() => {
* if (!elementRef.current) {
* return;
* }
* const observer = new IntersectionObserver(
* ([entry]) => {
* // Spustí se, když se změní viditelnost
* if (entry.isIntersecting) {
* setIsVisible(true);
* observer.unobserve(entry.target); // Přestane sledovat, jakmile je jednou viditelný (pro animaci jednou)
* }
* },
* { threshold, root, rootMargin }
* );
* observer.observe(elementRef.current);
* return () => {
* if (elementRef.current) {
* // eslint-disable-next-line react-hooks/exhaustive-deps
* observer.unobserve(elementRef.current);
* }
* };
* // eslint-disable-next-line react-hooks/exhaustive-deps
* }, [elementRef, threshold, root, rootMargin]); // Závislosti hooku
* return isVisible;
* }

export default useIntersectionObserver;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

* **Komponenta pro animované číslo:**  
  Vytvoříme komponentu AnimatedNumber, která použije useIntersectionObserver.
* // components/AnimatedNumber.tsx (vytvoř nový soubor)
* import React, { useState, useEffect, useRef } from 'react';
* import useIntersectionObserver from '../hooks/useIntersectionObserver'; // Uprav cestu podle potřeby
* interface AnimatedNumberProps {
* targetNumber: number;
* duration?: number; // Doba trvání animace v ms
* className?: string; // Pro předání CSS třídy
* suffix?: string; // Např. "+" nebo "%"
* }
* const AnimatedNumber: React.FC<AnimatedNumberProps> = ({
* targetNumber,
* duration = 2000, // Výchozí doba 2 sekundy
* className,
* suffix = '',
* }) => {
* const [currentNumber, setCurrentNumber] = useState(0);
* const numberRef = useRef<HTMLSpanElement>(null); // Ref pro span, který budeme sledovat
* const isVisible = useIntersectionObserver(numberRef, { threshold: 0.5 }); // Spustí se, když je 50% viditelné
* useEffect(() => {
* if (isVisible && currentNumber < targetNumber) {
* let startTimestamp: number | null = null;
* const step = (timestamp: number) => {
* if (!startTimestamp) startTimestamp = timestamp;
* const progress = Math.min((timestamp - startTimestamp) / duration, 1);
* const nextNumber = Math.floor(progress \* targetNumber);
* setCurrentNumber(nextNumber);
* if (progress < 1) {
* requestAnimationFrame(step);
* } else {
* setCurrentNumber(targetNumber); // Zajistí přesné dosažení cílového čísla
* }
* };
* requestAnimationFrame(step);
* }
* }, [isVisible, targetNumber, duration, currentNumber]); // Přidáno currentNumber, aby se useEffect nespouštěl zbytečně, pokud už bylo dosaženo
* // Pokud targetNumber není skutečné číslo (např. "24/7"), zobrazíme ho rovnou
* if (isNaN(targetNumber)) {
* return <span ref={numberRef} className={className}>{targetNumber}{suffix}</span>;
* }
* return (
* <span ref={numberRef} className={className}>
* {currentNumber}{suffix}
* </span>
* );
* };

export default AnimatedNumber;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**2. Použití AnimatedNumber v aboutme.tsx:**

// aboutme.tsx

import React, { useRef } from 'react'; // Přidán useRef

// ... další importy ...

import AnimatedNumber from '../components/AnimatedNumber'; // Uprav cestu

const AboutMePage: React.FC = () => {

const statsSectionRef = useRef<HTMLElement>(null); // Ref pro celou sekci statistik

// Tento ref použijeme pro useIntersectionObserver, pokud bys chtěl spouštět všechny animace najednou

// Ale v komponentě AnimatedNumber už máme vlastní ref a observer pro každé číslo zvlášť.

return (

<>

{/\* ... Head ... \*/}

<div className={styles.aboutContainer}>

{/\* ... Hero, Intro, Approach sections ... \*/}

<section className={styles.statsSection} ref={statsSectionRef}>

<div className={styles.statsContainer}>

<div className={styles.statItem}>

{/\* Použití AnimatedNumber komponenty \*/}

<AnimatedNumber

targetNumber={500}

className={styles.statNumber}

suffix="+"

/>

<span className={styles.statText}>Spokojených klientů</span>

</div>

<div className={styles.statItem}>

<AnimatedNumber

targetNumber={10}

className={styles.statNumber}

suffix="+"

duration={1500} // Můžeš nastavit různou dobu

/>

<span className={styles.statText}>Let zkušeností</span>

</div>

<div className={styles.statItem}>

<AnimatedNumber

targetNumber={99}

className={styles.statNumber}

suffix="%"

/>

<span className={styles.statText}>Úspěšnost klientů</span>

</div>

<div className={styles.statItem}>

{/\* Pro "24/7" nepoužijeme AnimatedNumber, protože to není číslo k animaci \*/}

<span className={styles.statNumber}>24/7</span>

<span className={styles.statText}>Podpora klientů</span>

</div>

</div>

</section>

{/\* ... Testimonials, Certificates, CTA sections ... \*/}

</div>

</>

);

};

export default AboutMePage;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Vysvětlení:**

* **useIntersectionObserver Hook:**
  + Používá IntersectionObserver API prohlížeče.
  + Sleduje daný element (elementRef).
  + Jakmile je element viditelný (na základě threshold), nastaví isVisible na true a přestane sledovat (aby se animace nespouštěla opakovaně při každém scrollu tam a zpět).
* **AnimatedNumber Komponenta:**
  + Vezme targetNumber (cílové číslo) a duration (dobu animace).
  + Má vlastní useRef (numberRef) pro span, ve kterém se číslo zobrazuje. Tento ref se předá do useIntersectionObserver.
  + Když isVisible (z useIntersectionObserver) je true a currentNumber je menší než targetNumber:
    - Spustí se animace pomocí requestAnimationFrame. Tato funkce plynule zvyšuje currentNumber od 0 (nebo od aktuální hodnoty, pokud by se animace přerušila a znovu spustila) až po targetNumber během dané duration.
  + Pokud targetNumber není číslo (např. pro tvoje "24/7"), jednoduše zobrazí text.
  + Přidává volitelný suffix (jako "+" nebo "%") za číslo.
* **Použití v aboutme.tsx:**
  + Místo přímého <span className={styles.statNumber}>500+</span> použiješ <AnimatedNumber targetNumber={500} className={styles.statNumber} suffix="+" />.
  + Pro "24/7", které se nemá animovat jako číslo, ponecháš původní span. Komponenta AnimatedNumber by to sice zvládla zobrazit, ale je to sémanticky čistší.

**CSS (aboutme.module.css):**  
Pro .statNumber a ostatní styly statistik není potřeba žádná změna, protože AnimatedNumber komponenta používá className={styles.statNumber}, takže se na ni aplikují tvé stávající styly.

**Možná vylepšení:**

* **Easing funkce:** Pro hladší animaci můžeš implementovat easing funkci místo lineárního nárůstu.
* **Formátování čísel:** Pro velká čísla můžeš přidat formátování (např. oddělovače tisíců).
* **Spouštění všech animací najednou:** Pokud bys chtěl, aby se všechna čísla ve statsSection začala animovat najednou, až když je celá sekce viditelná, pak bys useIntersectionObserver použil na statsSectionRef v AboutMePage a stav isVisible bys předal jako prop do každé AnimatedNumber komponenty. Současné řešení animuje každé číslo individuálně, jakmile se stane viditelným.

Toto řešení by mělo fungovat dobře a dodat tvé statistické sekci dynamiku. Nezapomeň vytvořit složku hooks a components (nebo jakou máš strukturu) a umístit do ní nové soubory.

**ešení přímo v aboutme.tsx:**

Budeme potřebovat:

1. useRef pro každý .statNumber, který chceme animovat, nebo pro celou .statsSection.
2. useState pro uchování aktuálně zobrazeného čísla pro každý animovaný prvek.
3. useState pro sledování, zda už byla sekce (nebo konkrétní číslo) viditelná a animace proběhla.
4. useEffect pro nastavení IntersectionObserver API, který bude sledovat viditelnost.

**Příklad implementace přímo v aboutme.tsx:**

// aboutme.tsx

import React, { useState, useEffect, useRef } from 'react';

import Head from 'next/head';

import Image from 'next/image';

import Link from 'next/link';

import styles from './aboutme.module.css';

import { FaAward, FaDumbbell, FaStar, FaUserGraduate, FaRunning, FaHeartbeat } from 'react-icons/fa';

// Pomocná komponenta pro animované číslo (může být i přímo v AboutMePage)

interface CountUpNumberProps {

target: number;

duration?: number;

className?: string;

suffix?: string;

startCondition: boolean; // Podmínka, kdy se má animace spustit

}

const CountUpNumber: React.FC<CountUpNumberProps> = ({

target,

duration = 2000,

className,

suffix = '',

startCondition,

}) => {

const [count, setCount] = useState(0);

const [hasAnimated, setHasAnimated] = useState(false); // Sleduje, zda už animace proběhla

useEffect(() => {

// Spustí se jen pokud je startCondition true A animace ještě neproběhla

if (startCondition && !hasAnimated) {

setHasAnimated(true); // Označíme, že animace začíná

let start = 0;

const end = target;

if (start === end) return;

let startTime: number | null = null;

const animate = (timestamp: number) => {

if (!startTime) startTime = timestamp;

const progress = Math.min((timestamp - startTime!) / duration, 1);

const currentNum = Math.floor(progress \* (end - start) + start);

setCount(currentNum);

if (progress < 1) {

requestAnimationFrame(animate);

} else {

setCount(end); // Zajistí přesné dosažení cíle

}

};

requestAnimationFrame(animate);

}

}, [startCondition, target, duration, hasAnimated]); // Přidáno hasAnimated do závislostí

// Pokud už animace proběhla, zobraz rovnou cílové číslo

// Jinak zobrazuj aktuální stav počítadla (nebo 0, pokud se ještě nezačalo)

return (

<span className={className}>

{hasAnimated ? target : count}{suffix}

</span>

);

};

const AboutMePage: NextPage = () => { // Pokud používáš NextPage typ

const statsSectionRef = useRef<HTMLElement>(null); // Ref pro celou sekci

const [isStatsSectionVisible, setIsStatsSectionVisible] = useState(false);

// Stavy pro jednotlivá čísla (pokud bys nechtěl komponentu CountUpNumber)

// const [clientsCount, setClientsCount] = useState(0);

// const [experienceYears, setExperienceYears] = useState(0);

// const [successRate, setSuccessRate] = useState(0);

// const [animationTriggered, setAnimationTriggered] = useState(false);

useEffect(() => {

const sectionNode = statsSectionRef.current; // Uložíme si ref do proměnné

if (!sectionNode) return;

const observer = new IntersectionObserver(

([entry]) => {

if (entry.isIntersecting) {

setIsStatsSectionVisible(true);

observer.unobserve(sectionNode); // Přestane sledovat, jakmile je jednou viditelný

}

},

{

threshold: 0.3, // Spustí se, když je 30% sekce viditelné

}

);

observer.observe(sectionNode);

return () => {

if (sectionNode) {

observer.unobserve(sectionNode);

}

};

}, []); // Prázdné pole závislostí, aby se observer nastavil jen jednou

// Data pro statistiky (pro čistší kód)

const statsData = [

{ id: 'clients', target: 500, suffix: '+', text: 'Spokojených klientů', duration: 2000 },

{ id: 'experience', target: 10, suffix: '+', text: 'Let zkušeností', duration: 1500 },

{ id: 'success', target: 99, suffix: '%', text: 'Úspěšnost klientů', duration: 1800 },

];

return (

<>

<Head>

<title>O mně | Fitness Coaching</title>

<meta name="description" content="Poznejte svého osobního trenéra. Profesionální fitness coaching založený na letech zkušeností a stovkách spokojených klientů." />

</Head>

<div className={styles.aboutContainer}>

{/\* ... Hero, Intro, Approach sections ... \*/}

{/\* Zde použijeme ref={statsSectionRef} na celou sekci \*/}

<section className={styles.statsSection} ref={statsSectionRef}>

<div className={styles.statsContainer}>

{statsData.map(stat => (

<div key={stat.id} className={styles.statItem}>

<CountUpNumber

target={stat.target}

duration={stat.duration}

className={styles.statNumber}

suffix={stat.suffix}

startCondition={isStatsSectionVisible} // Předáváme podmínku pro start

/>

<span className={styles.statText}>{stat.text}</span>

</div>

))}

{/\* Statická položka, která se nebude animovat \*/}

<div className={styles.statItem}>

<span className={styles.statNumber}>24/7</span>

<span className={styles.statText}>Podpora klientů</span>

</div>

</div>

</section>

{/\* ... Testimonials, Certificates, CTA sections ... \*/}

</div>

</>

);

};

export default AboutMePage;

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

**Vysvětlení změn v aboutme.tsx:**

1. **Pomocná komponenta CountUpNumber:**
   * Vytvořil jsem malou komponentu CountUpNumber přímo v souboru aboutme.tsx (můžeš ji později vyčlenit, pokud budeš chtít).
   * Tato komponenta přijímá target (cílové číslo), duration, className, suffix a nově startCondition.
   * Má vlastní interní stav count (aktuální číslo) a hasAnimated (zda už animace proběhla).
   * useEffect uvnitř CountUpNumber se spustí, když startCondition je true **A** hasAnimated je false.
   * Jakmile animace začne, setHasAnimated(true) zabrání jejímu opakovanému spuštění pro tuto instanci komponenty.
   * V JSX zobrazuje buď target (pokud už animace proběhla) nebo aktuální count.
2. **AboutMePage komponenta:**
   * statsSectionRef = useRef<HTMLElement>(null);: Ref je nyní na celé .statsSection.
   * isStatsSectionVisible = useState(false);: Tento stav bude true, jakmile je celá sekce viditelná.
   * **První useEffect (s prázdným polem závislostí):**
     + Nastaví IntersectionObserver, který sleduje statsSectionRef.
     + Když sekce vstoupí do viewportu (dle threshold), nastaví isStatsSectionVisible na true a **okamžitě zavolá observer.unobserve(sectionNode);**. Toto je klíčové, aby se isStatsSectionVisible změnilo na true jen jednou.
   * **Mapování statsData:**
     + Místo psaní každého .statItem zvlášť, vytvořil jsem pole statsData a procházím ho metodou map. To je čistší.
     + Každé komponentě CountUpNumber se předává startCondition={isStatsSectionVisible}. To znamená, že všechny číselné animace začnou ve stejný moment, kdy se celá sekce stane viditelnou.

**Proč by to mělo fungovat lépe:**

* **Jediný Observer:** Máš jen jeden IntersectionObserver pro celou sekci. Jakmile je viditelná, nastaví se isStatsSectionVisible na true a observer se odpojí. isStatsSectionVisible už se tedy nezmění zpět na false kvůli scrollu.
* **startCondition pro CountUpNumber:** Každá instance CountUpNumber začne animovat až tehdy, když je tato podmínka splněna.
* **hasAnimated v CountUpNumber:** Každá CountUpNumber si sama pamatuje, zda už animovala. I kdyby se startCondition (teoreticky) změnilo vícekrát (což by nemělo díky unobserve), hasAnimated zabrání opakované animaci.

**Důležité:**

* observer.unobserve(sectionNode); je kritické pro to, aby se isStatsSectionVisible nastavilo jen jednou.
* Ujisti se, že threshold v IntersectionObserver (např. 0.3) je hodnota, která ti vyhovuje pro spuštění animace (tj. když je vidět 30% sekce).

Tímto způsobem bys měl dosáhnout toho, že animace proběhne spolehlivě jen jednou pro každé číslo, když uživatel doscrolluje k sekci statistik. A kód je stále obsažen v aboutme.tsx.

Chtel bych, aby ta Navbar.tsx byla na mobilu (a klidne i na desktopu, kdyz uz jsme u toho) **vzdycky viditelna a prilepena nahore** (position: sticky nebo fixed?), i kdyz scrolluju dolu. Proste aby nezmizela.

V Navbar.module.css mam pro .navbar tohle:

.navbar {

position: relative; /\* Tohle je tam ted \*/

background-color: var(--color-navbar-background);

padding: 20px;

box-shadow: 0 2px 4px var(--color-navbar-shadow);

display: flex;

justify-content: space-between;

align-items: center;

width: 100%;

box-sizing: border-box;

z-index: 1000; /\* Z-index tam mam, to je dobre \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

A v Layout.tsx je Navbar jen takhle jednoduse vlozena:

// Layout.tsx

// ...

const Layout: React.FC<LayoutProps> = ({ children }) => {

return (

<div className={styles.layoutContainer}>

<Navbar /> {/\* TADY JE NAVBAR \*/}

<main className={styles.mainContent}>{children}</main>

<Footer />

</div>

);

};

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Co mam zmenit v tom .navbar CSS, aby byla furt nahore a nehejbala se pri scrollu? A musim neco menit i v Layout.tsx nebo jinde, aby to fungovalo spravne a neprekryvalo to treba kus obsahu pod tim?

Jasně, "přilepený" (sticky/fixed) navbar je častý požadavek. To "chytré" schovávání a objevování, které popisuješ, není standardní chování CSS pro position: relative;, takže buď:  
a) Máš tam nějaký JavaScript, který toto chování implementuje (což jsi nezmiňoval, že bys měl).  
b) Nebo se jedná o nějaké výchozí chování specifické pro mobilní prohlížeč v kombinaci s tím, jak je stránka strukturovaná (např. pokud je navbar uvnitř nějakého elementu, který se scrolluje jinak).  
c) Nebo je to jen dojem a ve skutečnosti se s position: relative normálně odscrolluje s obsahem.

Předpokládejme, že chceš standardní přilepený navbar.

**Způsob 1: Použití position: sticky; (doporučeno pro jednoduchost a lepší výkon)**

position: sticky; je moderní a obvykle nejlepší způsob, jak dosáhnout přilepeného efektu, protože element se chová jako relative dokud nedoscrolluješ na určitou pozici, a pak se stane fixed.

1. **Úprava CSS v Navbar.module.css:**
2. .navbar {
3. position: sticky; /\* ZMĚNA Z RELATIVE NA STICKY \*/
4. top: 0; /\* DŮLEŽITÉ: Určuje, kde se "přilepí" (0px od horního okraje viewportu) \*/
5. background-color: var(--color-navbar-background);
6. padding: 20px; /\* Můžeš upravit padding podle potřeby \*/
7. box-shadow: 0 2px 4px var(--color-navbar-shadow);
8. display: flex;
9. justify-content: space-between;
10. align-items: center;
11. width: 100%;
12. box-sizing: border-box;
13. z-index: 1000; /\* Velmi důležité, aby byl navbar nad ostatním obsahem \*/
14. /\* Můžeš přidat transition pro plynulejší přechod stínu, pokud by se měnil \*/
15. /\* transition: box-shadow 0.3s ease; \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Důležité pro position: sticky;:**

* + Musí být specifikována alespoň jedna z hodnot top, bottom, left, nebo right, aby prohlížeč věděl, kde se má element "přilepit". V tvém případě top: 0;.
  + Žádný z rodičovských elementů nesmí mít nastaveno overflow: hidden;, overflow: scroll; nebo overflow: auto; na ose, po které se sticky chování očekává. V tvém případě by to byl overflow-y. Tvůj .layoutContainer v Layout.module.css nemá overflow-y, takže by to mělo být v pořádku. Hlavní scroll by měl být na body nebo html.

**Způsob 2: Použití position: fixed;**

position: fixed; vyjme element z normálního toku dokumentu a pozicuje ho relativně k viewportu. Bude vždy na stejném místě, bez ohledu na scroll.

1. **Úprava CSS v Navbar.module.css:**
2. .navbar {
3. position: fixed; /\* ZMĚNA NA FIXED \*/
4. top: 0; /\* Přilepení k hornímu okraji viewportu \*/
5. left: 0; /\* Roztažení přes celou šířku viewportu \*/
6. right: 0; /\* Alternativně width: 100%; ale left:0 a right:0 je robustnější \*/
7. background-color: var(--color-navbar-background);
8. padding: 20px;
9. box-shadow: 0 2px 4px var(--color-navbar-shadow);
10. /\* display: flex; ... (zbytek zůstává stejný) ... \*/
11. width: 100%; /\* Zajistí plnou šířku, i když je vyjmut z toku \*/
12. box-sizing: border-box;
13. z-index: 1000;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

1. **Možná úprava v Layout.tsx nebo globálním CSS kvůli překrytí obsahu:**  
   Když použiješ position: fixed;, navbar je vyjmut z normálního toku dokumentu. To znamená, že obsah, který je v HTML za ním (<main>), začne hned odshora stránky a bude překrytý navbarem.  
   Musíš tedy přidat padding-top na tvůj hlavní obsahový kontejner (.mainContent v Layout.module.css), aby se obsah posunul dolů o výšku navbaru.
   * **Nejprve zjisti výšku navbaru.** Můžeš to udělat JavaScriptem nebo ji odhadnout/nastavit fixně. Pokud máš padding 20px nahoře a dole a výška písma/ikonek je cca 24px, tak výška navbaru bude zhruba 20 + 20 + 24 = 64px. Může se lišit.
   * **Úprava Layout.module.css:**
   * /\* Layout.module.css \*/
   * .mainContent {
   * flex: 1;
   * padding: 30px 40px; /\* Tvoje stávající paddingy \*/
   * max-width: 1440px;
   * margin: 0 auto;
   * width: 100%;
   * box-sizing: border-box;
   * /\* PŘIDÁNO POKUD POUŽÍVÁŠ position:fixed PRO NAVBAR \*/
   * /\* Výšku si uprav podle skutečné výšky tvého navbaru \*/
   * /\* padding-top: calc(30px + VYSKA\_NAVBARU\_PX); \*/
   * /\* Například, pokud má navbar výšku 70px: \*/
   * /\* padding-top: calc(30px + 70px); // Celkem 100px shora \*/
   * /\* Nebo jednodušeji, pokud padding-top 30px byl už dostatečný: \*/
   * /\* margin-top: VYSKA\_NAVBARU\_PX; \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Lepší způsob pro position: fixed; je nastavit padding-top na body nebo na hlavní obalovač stránky (.layoutContainer nebo #\_\_next).**

Pokud tvůj Navbar.tsx má dynamickou výšku (např. se mění padding na mobilu), je složitější nastavit správný padding-top. Můžeš:  
a) Použít JavaScript k dynamickému nastavení padding-top na mainContent podle aktuální výšky navbaru.  
b) Nastavit pevnou výšku navbaru (např. height: 70px;) a pak použít tuto hodnotu pro padding-top.

**Doporučení:**

* **Začni s position: sticky; a top: 0; na .navbar.** Je to modernější, často jednodušší na implementaci (obvykle nepotřebuješ řešit padding-top na obsahu) a lépe spolupracuje s přirozeným tokem dokumentu. Ověř, že žádný rodičovský element nemá overflow, který by to pokazil.
* Pokud position: sticky; z nějakého důvodu nefunguje (např. kvůli složité struktuře DOMu nebo overflow u rodičů, které nemůžeš změnit), pak sáhni po position: fixed; a nezapomeň vyřešit padding-top pro .mainContent, aby navbar nepřekrýval začátek tvého obsahu.
* z-index: 1000; je dobré mít, aby byl navbar vždy navrchu.

Vyzkoušej nejprve variantu s position: sticky;. Měla by být pro tvůj případ ideální.

Mam tam ted jenom jednoduchy pozadi (nejakej rgba(0,0,0,0.8)) pres obrazek a bily text. Chtel bych, aby to bylo takovy vic... no, aby to vic lakalo kliknout. Mozna jiny barvy, nebo nejaky efekt na to tlacitko .ctaButton?

Tady je ten kus JSX z index.tsx:

// index.tsx

// ... (ostatni sekce) ...

<section className={styles.ctaSection}> {/\* TUTO SEKCI CHCI VYLEPŠIT \*/}

<div className={styles.ctaOverlay}></div> {/\* Overlay pro ztmavení pozadí \*/}

<div className={styles.ctaContent}>

<h2 className={styles.ctaTitle}>Připraveni Změnit Svůj Život?</h2>

<p className={styles.ctaText}>

Udělejte první krok ke zdravějšímu a silnějšímu já. Kontaktujte mě ještě dnes!

</p>

<Link href="/kontakt" className={styles.ctaButton}> {/\* TOTO TLAČÍTKO \*/}

Chci Začít!

</Link>

</div>

</section>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A CSS z index.module.css:

.ctaSection {

position: relative;

padding: 80px 20px; /\* Puvodne 120px 40px \*/

background: url('/images/cta-background.jpg') center/cover no-repeat; /\* Jiny obrazek nez hero? \*/

/\* min-height: 350px; \*/

display: flex;

justify-content: center;

align-items: center;

text-align: center;

color: white;

/\* max-width: 1440px; \*/ /\* Toto je uz na obalovaci strance \*/

/\* margin: 0 auto; \*/

/\* width: 100%; \*/

}

.ctaOverlay {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.7); /\* Tmavsi overlay nez puvodni 0.8 \*/

}

.ctaContent {

position: relative;

z-index: 2;

max-width: 800px;

}

.ctaTitle {

font-size: 2.5rem; /\* Puvodne 3rem \*/

font-weight: 800; /\* Puvodne 900 \*/

margin-bottom: 20px; /\* Puvodne 25px \*/

letter-spacing: 1px; /\* Puvodne 3px \*/

text-transform: uppercase;

/\* text-shadow: 0 2px 10px rgba(0, 0, 0, 0.5); // Stin muze zustat \*/

}

.ctaText {

font-size: 1.1rem; /\* Puvodne 1.3rem \*/

line-height: 1.7; /\* Puvodne 1.6 \*/

margin-bottom: 30px; /\* Puvodne 40px \*/

color: rgba(255, 255, 255, 0.9); /\* Puvodne 0.95 \*/

}

.ctaButton {

display: inline-block;

padding: 15px 35px; /\* Puvodne 18px 45px \*/

background: var(--color-primary); /\* POUŽÍVÁ GLOBÁLNÍ PRIMÁRNÍ BARVU \*/

OK, ten/\* background: linear-gradient(90deg, var(--color-primary), var(--color-primary-hover)); // video podklad v `.heroSection` na `index.tsx` je pe Puvodni gradient \*/

color: white;

border-radius: 50px; /\* Plne zakulacene \*/

font-weight: 700;

font-size: 1cka! Ale teď se koukám na tu stránku "O mně" (`aboutme.tsx`) a na.1rem; /\* Puvodne 1.2rem \*/

transition: all 0.3s konci mám tu `.ctaSection` (Call to Action) s nadpisem "Připraveni Začít Svou ease;

text-align: center;

box-shadow: 0 4px 15px rgba(0, 123, 255, 0.2); /\* Jemnejsi stin Proměnu?" a tlačítkem "Kontaktujte Mě". To pozadí je teď jenom nějaká bar, barva z --color-primary \*/

text-transform: uppercase;

letter-spacing: 1.5px; /\* Puvodne 2px \*/

border: 2px solid transparent; /\* Prova (`var(--color-primary-hover)`).

Chtěl bych, aby ta CTA sekce byla vizuálně zajímavější a víc "táhla" k tomu kliknutí. Co kdybychom tam taky dali pripadny hover efekt s borderem \*/

}

.ctaButton:hover {

background: var(--color-primary-hover);

transform: translateY(-3px) scale(1.03); /\* Mirne vetsi scale nějaký obrázek na pozadí, třeba nějakou dynamickou fotku z fitka nebo tak něco? A ten text a tlačítko by přes to musely být dobře vidět. A to tlačítko `.ctaButton` by \*/

box-shadow: 0 6px 20px rgba(0, 123, 255, 0.35); /\* Vyraznejsi stin pri hoveru \*/

taky mohlo mít nějaký lepší hover efekt.

Tady je JSX z `aboutme.tsx` pro tu CTA sekci:

```typescript jsx

// aboutme.tsx

// ...

{/\* 7. CTA Sekce \*/}}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Ten .ctaButton uz pouziva var(--color-primary). Co kdybychom  
<section className={styles.ctaSection}>  
<div className={styles.ctaContainer}>  
<h2 className={styles.ctaTitle}>Připraveni Začít Svou Proměnu?</h2>  
< zkusili nejaky jiny styl pro to tlacitko, treba "ghost button" (jenom op className={styles.ctaText}>  
Neváhejte mě kontaktovat pro nezávaznou konzultaci a společně najdeme tu nejlepší cestu k vašim cílům.  
</p>  
<Linkhraniceni) a pri hoveru by se vyplnilo? Nebo nejakej jinej gradient na href="/kontakt" className={styles.ctaButton}>  
Kontaktujte Mě  
</Link>  
pozadi ty sekce? A ten text, je to dostatecne motivacni?

Jasně, CTA</div>  
</section>  
// ...

A CSS z `aboutme.module.css`:

```css

/\* aboutme.module.css \*/

/\* ... \*/

.ctaSection {

padding: 70px 20px;

/\* PUVODNI POZADI - TOHLE CHCI ZMENIT \*/

background (Call to Action) sekce je klíčová, měla by opravdu zaujmout a motivovat k akci. Pojďme ji vylepšit!

\*\*1. Pozadí sekce (`.ctaSection` a `.ctaOverlay`-color: var(--color-primary-hover);

/\* :global(.dark-mode) & { background-color:):\*\*

Místo statického obrázku s tmavým overlayem můžeš zkusit:

\* \*\*Dynamičtější gradient:\*\* Místo jednoduchého `rgba(0,0,0,0 var(--color-primary); } // Toto je pro dark mode \*/

color: white; /\* Text je b.7)` můžeš použít gradient, který lépe ladí s tvými primárními barvami, nebo vytváří zajímavější atmosféru.

\* \*\*Kombinace gradientu a vzoru:\*\* Přily \*/

text-align: center;

position: relative; /\* Pro případný overlay nebo pseudoelementy \*/

overflow: hidden; /\* Pokud by obrázek pozadí byl větší \*/

}

.ctaes obrázek můžeš dát nejen barevný overlay, ale i jemný SVG vzor.

Příklad proContainer {

max-width: 700px;

margin: 0 auto;

position: relative; /\* Aby byl obsah nad případným overlayem pozadí \*/

z-index: 2;

}

.ctaTitle {

font-size: 2.2rem;

font-weight `.ctaOverlay` (pokud stále chceš obrázek na pozadí `.ctaSection`):

```css

.ctaOverlay {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

/\* Příklad gradientu, který jde od tvé primární bar: 700;

margin-bottom: 15px;

color: white;

}

.ctaText {

font-size: 1.1rem;

margin-bottom: 30px;

line-height: 1.7;

opacity: 0.9vy do tmavšího odstínu,

můžeš použít i více barevných bodů. \*/

background: linear-gradient(135deg, rgba(var(--color-primary-rgb), ; /\* Lehce průhledný text pro lepší čitelnost na různých pozadích \*/

}

.ctaButton {

display: inline-block;

padding: 14px 35px;

background-color: white; /\* Tlačítko je bílé \*/

color: var(--color-primary); /\* Text tlačítka v primární barvě \*/

border-radius: 8px;

text-decoration0.85) 0%, rgba(var(--color-primary-hover-rgb), 0.9) 70%, rgba(0,0,0,0.95) 100%);

/\* Předpokládá, že máš v :root definované --color-primary-rgb a --color-primary-: none;

font-weight: 700;

font-size: 1rem;

transition: all 0.3s ease;

box-shadow: 0 4px 10px rgba(0,0,0,0.1);

}

.ctaButton:hover {

background-color: #f0f0f0;

color: var(--color-primary-hover);

hover-rgb jako R,G,B hodnoty

Např. --color-primary-rgb: 0, 123, 255;

Pokud ne, použij hex/rgba přímo:

background: linear-gradient(135deg, rgba(0, 123, 255, transform: translateY(-2px);

box-shadow: 0 6px 15px rgba(0,0,0,0.15);

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).

Jak tam nejlépe přidat 0.85) 0%, rgba(0, 86, 179, 0.9) 70%, rgba(0,0,0,0.95) 100%);  
*/  
z-index: 1; /* Overlay je mezi pozadím sekce a obsahem \*/  
}

Nebo pokud chceš úplně nahradit obrázkové pozadí jen gradientem na ten obrázek na pozadí s překryvem a jak vylepšit ten hover efekt na tlačítku `.ctaButton`, aby to bylo víc "klikací"?

---

Jasně, CTA (Call to Action) sekce by měla být vizuálně výrazná a motivující. Použití obrázku na pozadí a vylepšení tlačítka jsou skvělé nápady.

\*\*1. Přidání obráz `.ctaSection`:

```css

.ctaSection {

/\* ... (ostatní styly) ... \*/

background: linear-gradient(135deg, var(--color-primary-hover) 0%, var(--color-primary) 60%, #0a2540 100%); /\* Příklad, uprav barvy \*/

/\* #0a2540 je tmavě modrá, můžeš použít vlastní tmavou barvuku na pozadí s překryvem pro `.ctaSection`:\*\*

Podobně jako u hero sekce s videem, použijeme obrázek na pozadí a přes něj poloprůhlednou vrstvu (overlay) pro zajištění čitelnosti textu.

\* \*\*Příprava obrázku:\*\* Vyber si vhodný obrázek (dynamický, motivační). Měl by být dostatečně kvalitní, ale optimalizovaný pro web. Ulož ho např. do `public/images/cta-background.jpg`.

\* \*\*Úprava CSS v `aboutme.module.css`:\*\*

```css

/\* aboutme.module.css \*/

. z palety \*/

/\* Už nepotřebuješ .ctaOverlay, pokud je pozadí jen gradient \*/

}

/\* Pokud nepoužíváš .ctaOverlay, tak .ctaContent nepotřebuje z-index: 2, stačí position: relative \*/

.ctaContent {

position: relative; /\* Zůstává, pokud bys chtěl něco absolutně pozicovat uvnitř \*/

z-index: 1; /\* Pokud .ctactaSection {

padding: 80px 20px; /\* Mírně větší padding \*/

color: white;

text-align: center;

position: relative; /\* Nutné pro absolutní pozicování overlaye \*/

overflow: hidden; /\* Aby obrázek nepřetékal, pokud by byl větší \*/

background-image: url('/images/cta-background.jpg'); /\* Cesta k tvému obrázku \*/

background-size: cover; /\* Obrázek pokryje celou plochu \*/

background-position: center center; /\* Vycentrování obrázku \*/

background-attachment: fixed; /\* Volitelný parallax efekt (Overlay není, stačí 1 \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).

**2. Texty (.ctaTitle, .ctaText):**

* **.ctaTitle:** Zkus ho udělat ještě výraznější, možná větší, nebo s jiným letter-spacing.
* **.ctaText:** Můžeš přidat ještě jednu větu, která zdůrazní hlavní benefit nebo urgenci.

Příklad úpravy textů:

/\* Pokud nechceš parallax, dej 'scroll' nebo vynech \*/

}

/\* Přidání overlaye pro lepší čitelnost textu \*/

.ctaSection::before { /\* Použijeme pseudoelement pro overlay \*/

content: '';

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0css

.ctaTitle {

font-size: 2.8rem; /\* O něco větší \*/

font-weight: 800;

margin-bottom: 25px;

letter-spacing: 0.5px; /\* Menší pro tak velký nadpis může být čitelnější \*/

text-transform;

background-color: rgba(0, 0, 0, 0.65); /\* Tmavý overlay, uprav průhlednost (0.0 až 1.0) \*/

/\* Nebo můžeš použít gradientový overlay: \*/

/\* background: linear-gradient(to bottom, rgba(: uppercase;

line-height: 1.2;

/\* Můžeš zkusit textový gradient, pokud to podporují tvé cílové prohlížeče:

background: -webkit-linear-gradient(45deg, #eee, #fff);

-webkit-background-clip: text;

-webkit0,0,0,0.4) 0%, rgba(0,0,0,0.8) 100%); \*/

z-index: 1; /\* Pod obsahem (.ctaContainer) \*/

}

.ctaContainer { /\* Obsah musí být nad overlayem \*/

max-width:-text-fill-color: transparent;

\*/

}

.ctaText {

font-size: 1.2rem; /\* Zpět na trochu větší \*/

line-height: 1.8;

margin-bottom: 35px;

color: rgba(255, 2 750px; /\* Mírně širší \*/

margin: 0 auto;

position: relative; /\* Důležité \*/

z-index: 2; /\* Nad ::before pseudoelementem \*/

}

.ctaTitle {

font-size: 2.5rem; /\* Větší nadpis \*/

font-weight: 800; /\* Výraznější \*/

margin-bottom55, 255, 0.95); /\* Světlejší pro lepší kontrast \*/

max-width: 650px; /\* Mírné zúžení pro lepší čitelnost delšího textu \*/

margin-left: auto;

margin-right: auto;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Zvaž výkon na mobilech) \*/

V JSX pro `.: 20px;  
color: white;  
text-shadow: 1px 1px 3px rgba(0,0,0,0.3); /\* Jemný stín pro lepší čitelnost na obrázku \*/  
}

.ctaText {

font-size: 1.15rem; /\* Mírně větší text \*/

margin-bottom: 35px;

line-height: ctaText` můžeš zkusit:

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).

"Udělejte první krok ke zdravějšímu a silnějšímu já. Vaše cesta za lepší kondicí začíná právě teď. Kontaktujte mě ještě dnes!"

**3. Tlačítko (.ctaButton) – "Ghost Button" styl a vylepšený hover:**

1.8; /\* Větší řádkování */  
opacity: 0.95; /* Téměř neprůhledný, pokud je overlay dostatečný */  
text-shadow: 1px 1px 2px rgba(0,0,0,0.2); /* Jemný stín \*/  
}  
```  
**Vysvětlení:**  
\* .ctaSection má nyní nastavený `background"Ghost button" (průhledné pozadí, barevné ohraničení a text) může vypadat velmi elegantně.

.ctaButton {

display: inline-block;

padding: 16px 40px; /\* Větší padding pro výraznější tlačítko \*/

background-image`, `background-size: cover` a `background-position: center`.

\* Pseudoelement `.ctaSection::before` vytváří tmavý poloprůhledný overlay přes obrázek. `z-index: 1` ho umístí mezi obrázek pozadí a skutečný obsah.

\* `.ctaContainer` má: transparent; /\* Průhledné pozadí pro ghost efekt \*/

color: white; /\* Barva textu - musí být kontrastní s pozadím sekce \*/

border-radius: 8px; /\* Méně zakulacené pro modernější vzhled, nebo nechej 50px \*/

font-weight `z-index: 2`, aby byl text a tlačítko nad tímto overlayem.

\* Přidal jsem `text-shadow` k `.ctaTitle` a `.ctaText` pro ještě lepší čitelnost na komplexním pozadí.

\*\*2. Vylepšení hover efektu na tlačítku `.ctaButton`: 700;

font-size: 1.1rem;

transition: all 0.3s ease-out; /\* 'ease-out' pro plynulejší konec animace \*/

text-align: center;

text-transform: uppercase;

letter-spacing: 1.8px:\*\*

Chceme, aby tlačítko více "lákalo" ke kliknutí.

\* \*\*Úprava CSS v `aboutme.module.css`:\*\*

```css

.ctaButton {

display: inline-block;

padding: 16px 40px; /\* Větší padding pro "; /\* Větší prostrkání \*/

border: 2px solid white; /\* Ohraničení vmasitější" tlačítko \*/

background-color: var(--color-primary); /\* Změna pozadí na primární barvu pro větší kontrast s textem sekce \*/

/\* Původně bylo bílé \*/

color: white; /\* Text tlačítka bude bílý \*/

border-radius barvě textu (nebo var(--color-primary) pokud by text byl tmavý) \*/

cursor: pointer; /\* Explicitně nastavit kurzor \*/

box-shadow: none; /\* Ghost buttony obvykle nemají stín ve výchozím stavu \*/

}

.ctaButton:hover {

background: white; /\* Při hoveru se vyplní bílou (nebo var(--color-primary)) \*/

color: var(--: 50px; /\* Plně zakulacené rohy pro moderní vzhled \*/

text-decoration: none;

font-weight: 700;

font-size: 1.05rem; /\* Mírně větší písmo \*/

text-transform: uppercase; /\* Velká písmenacolor-primary); /\* Text změní barvu na primární (nebo na barvu pozadí sekce) \*/

transform: translateY(-4px) scale(1.05); /\* Výraznější "nadskočení" \*/

/\* box-shadow: 0 8px 25px rgba(255, pro CTA \*/

letter-spacing: 1px; /\* Lehké prostrkání \*/

border: 2px solid transparent; /\* Pro případný efekt borderu při hoveru \*/

transition: all 0.3s cubic-bezier(0.25, 0.8, 0.25, 1); /\* Plynulejší přechod s custom easing funkcí \*/

box-shadow: 0 4255, 255, 0.2); \*/ /\* Jemný světlý stín, pokud se vyplní světlou barvou \*/

border-color: white; /\* Ujisti se, že border zůstane stejný nebo zmizí, pokud je to žádoucí \*/

}

/\* Alternativní hover,px 15px rgba(0, 0, 0, 0.2); /\* Výraznější stín \*/

/\* Můžeš přidat ikonu, pokud chceš \*/

/\* display: inline-flex; align-items: center; gap: 8px; \*/

}

.ctaButton:hover,

pokud bys chtěl primární barvu jako výplň \*/

/\*

.ctaButton:hover {

background: var(--color-primary);

color: white;

border-color: var(--color-primary); .ctaButton:focus { /\* Přidán i :focus stav pro přístupnost \*/

background-color: var(--color-primary-hover); /\* Tmavší odstín primární barvy při hoveru \*/

/\* Nebo můžeš změnit barvu na bílou a text na primární: \*/

/\* background-color: white // Border se "schová" do barvy pozadí

transform: translateY(-4px) scale(1.05);

box-shadow: 0 8px 25px rgba(var(--color-primary-rgb), 0.3);

}

\*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Další nápady pro CTA sekci:**

; */  
/* color: var(--color-primary); */  
/* border-color: var(--color-primary); // Pokud máš transparentní border */  
transform: translateY(-3px) scale(1.03); /* Výraznější "nadzvednutí" a lehké zvětšení */  
box-shadow: 0 8px* **Ikonka před/za textem v nadpisu:**  
html <h 25px rgba(var(--color-primary-rgb, 0, 123, 255), 0.3); /\* Stín v barvě tlačítka, před2 className={styles.ctaTitle}><FaRocket style={{ marginRight: '15px' }} />Připraveni Změnit Svůj Život?</h2>

* **Animace při scrollu:** Můžeš použítpokládá --color-primary-rgb  
  v globals.css, např. --color-primary-rgb: 0,123,255; */  
  /* Pokud nemáš, použij např. rgba(0,0,0,0.3) */  
  outline: none; /* Odebrat výchozí focus outline, pokud je vlastní stín dostatečný \*/  
  }

.ctaButton:active { /\* Efekt při kliknutí \*/  
transform: translateY(-1px) scale(0.98);  
box-shadow: 0 2px 10px rgba(var(--color-primary-rgb, 0, IntersectionObserver (podobně jako u statistik), aby se prvky v CTA sekci (nadpis, text, tlačítko) objevily s jemnou animací (např. fade-in, slide-up), když uživatel na sekci doscrolluje. To by dodalo ještě větší "wow" efekt.

Například přidat třídu, když je sekce viditelná:

.ctaContent.isVisible .ctaTitle {

animation: fadeInUp 0.8s ease-out forwards;

}

.ctaContent.isVisible .ctaText {

animation: fadeInUp 0.8s ease-out 0.2123, 255), 0.25);

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Vysvětlení změn na tlačítku:**

* + Změnil jsem background-color na var(--color-primary) a color na white pro lepší kontrast s pozadím sekce. Původní bílé tlačítko by na světlém obrázku nemuselo být dobře vidět.  
    s forwards; /\* delay 0.2s */  
    }  
    .ctaContent.isVisible .ctaButton {  
    animation: fadeInUp 0.8s ease-out 0.4s forwards; /* delay 0.4s \*/  
    }

@keyframes fadeInUp {  
from {  
opacity: 0;  
transform: translateY(30px);  
}  
to {  
opacity: 1;  
transform: translateY(0);  
}  
}

A v JSX bys pak na \* `border-radius: 50px;` pro plně zakulacené rohy.

\* Přidán `text-transform: uppercase;` a `letter-spacing`.

\* `transition` má upravenou `cubic-bezier` funkci pro zajímavější animaci.

\* \*\*Hover efekt:\*\* Výraznější `transform` (posun nahoru a zvětšení) a dynamičtější `box-shadow`. `.ctaContent` přidával třídu `.isVisible` pomocí stavu a IntersectionObserveru.

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).

Experimentuj s těmito návrhy. Změna pozadí a stylu tlačítka může mít velký dopad na celkový pocit z CTA sekce. Kombinace tmavšího, bohatšího pozadí (třeba s gradientem využívajícím tvé primární barvy) a kontrastního "ghost" tlačítka, které se při hoveru vyplní, často vypadá velmi dobře.

Chtěl bych, aby to vypadalo víc jako opravdový formulář pro odběr novinek. Třeba aby ten input .newsletterInput měl vedle sebe tlačítko .newsletterButton s nějakou ikonkou (třeba šipka nebo obálka). A aby to celé bylo v jedné lajně a vypadalo kompaktně.

Tady je ten kus JSX z Footer.tsx, kde to mám (teď tam to tlačítko ani není, jen jsem o něm přemýšlel):

// Footer.tsx

// ... (ostatní sloupce) ...

<div className={styles.footerColumn}>

<h3 className={styles.footerTitle}>Novinky</h3>

<p className={styles.footerText}>

Přihlaste se k odběru našich novinek a získejte exkluzivní tipy a nabídky přímo do vaší schránky.

</p>

{/\* TADY CHCI TEN VYLEPŠENÝ FORMULÁŘ \*/}

<form className={styles.newsletterForm} onSubmit={(e) => e.preventDefault()}>

<input

type="email"

placeholder="Váš e-mail"

className={styles.newsletterInput}

required

/>

{/\* Tlačítko bych přidal sem \*/}

<button type="submit" className={styles.newsletterButton} aria-label="Přihlásit k odběru">

{/\* <FaPaperPlane /> Nebo nějaká šipka \*/}

Odebírat

</button>

</form>

<p className={styles.newsletterDisclaimer}>

Odesláním souhlasíte se zpracováním osobních údajů.

</p>

</div>

// ...

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

A v Footer.module.css pro to zatím nemám skoro nic specifického, jen obecný .footerText:

/\* Footer.module.css \*/

.footerColumn { /\* ... \*/ }

.footerTitle { /\* ... \*/ }

.footerText { /\* ... \*/ }

/\* SEM BY PŘIŠLY STYLY PRO NEWSLETTER \*/

.newsletterForm {

/\* Jak to zarovnat do jedné řádky? \*/

/\* display: flex; ? \*/

margin-bottom: 15px; /\* Odsazení od disclaimeru \*/

}

.newsletterInput {

/\* Jak to nastylovat, aby to ladilo s tlačítkem? \*/

padding: 10px 15px;

border: 1px solid rgba(255, 255, 255, 0.2); /\* Světlý border na tmavém pozadí footeru \*/

background-color: rgba(255, 255, 255, 0.05); /\* Velmi jemné pozadí \*/

color: white;

border-radius: 4px; /\* Aby ladilo s tlačítkem \*/

/\* width: 70%; ? Nebo flex-grow? \*/

margin-right: -1px; /\* Aby se border s tlačítkem hezky napojil, pokud jsou vedle sebe \*/

}

.newsletterInput::placeholder {

color: rgba(255, 255, 255, 0.5);

}

.newsletterButton {

/\* Jak to nastylovat? \*/

padding: 10px 15px;

background-color: var(--color-primary); /\* Použít primární barvu \*/

color: white;

border: none;

border-radius: 4px; /\* Aby ladilo s inputem \*/

cursor: pointer;

/\* transition: background-color 0.3s ease; \*/

}

/\* .newsletterButton:hover { background-color: var(--color-primary-hover); } \*/

.newsletterDisclaimer {

font-size: 0.8rem;

color: rgba(255,255,255,0.5);

margin-top: 10px;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Jak nejlépe nastylovat ten .newsletterForm, .newsletterInput a .newsletterButton, aby byly hezky vedle sebe, input zabíral většinu místa a tlačítko bylo na konci? A jakou ikonku bys doporučil pro to tlačítko?

Jasně, vylepšení formuláře pro odběr novinek ve footeru je dobrý nápad, aby to působilo profesionálněji a funkčněji.

**1. Zarovnání inputu a tlačítka vedle sebe (.newsletterForm):**

Nejjednodušší způsob, jak dostat input a tlačítko vedle sebe, je použít display: flex; na jejich rodičovském elementu (.newsletterForm).

/\* Footer.module.css \*/

.newsletterForm {

display: flex; /\* Zarovná input a tlačítko vedle sebe \*/

align-items: stretch; /\* Zajistí, že input i tlačítko budou stejně vysoké (pokud mají padding) \*/

margin-bottom: 15px;

border-radius: 6px; /\* Zaoblení pro celý formulářový prvek \*/

overflow: hidden; /\* Aby se border-radius aplikoval i na vnitřní prvky bez vlastního zaoblení na spojích \*/

box-shadow: 0 2px 5px rgba(0,0,0,0.2); /\* Jemný stín pro formulář na tmavém pozadí footeru \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

overflow: hidden; na .newsletterForm spolu s border-radius pomůže vytvořit dojem jednoho kompaktního prvku, pokud input a tlačítko nebudou mít vlastní zaoblení na stranách, kde se dotýkají.

**2. Stylování inputu (.newsletterInput):**

Input by měl zabrat většinu dostupného místa.

.newsletterInput {

flex-grow: 1; /\* Input zabere všechen dostupný prostor v flex kontejneru \*/

padding: 12px 18px; /\* Větší padding pro lepší klikací plochu a vzhled \*/

border: none; /\* Odstraníme výchozí border, protože celý form má teď radius a stín \*/

/\* border-right: 1px solid rgba(255, 255, 255, 0.15); Oddělovací linka, pokud chceš \*/

background-color: rgba(255, 255, 255, 0.08); /\* Mírně světlejší pozadí \*/

color: white;

font-size: 0.95rem;

/\* border-radius: 0; /\* Odstraníme radius, pokud je na .newsletterForm \*/

/\* Aby neměl radius na pravé straně, kde se napojuje na tlačítko: \*/

border-top-left-radius: 6px; /\* Stejný radius jako .newsletterForm \*/

border-bottom-left-radius: 6px; /\* Stejný radius jako .newsletterForm \*/

border-top-right-radius: 0;

border-bottom-right-radius: 0;

outline: none; /\* Odebrat výchozí focus outline \*/

transition: background-color 0.3s ease;

}

.newsletterInput::placeholder {

color: rgba(255, 255, 255, 0.6);

}

.newsletterInput:focus {

background-color: rgba(255, 255, 255, 0.15); /\* Zvýraznění při focusu \*/

/\* box-shadow: inset 0 0 0 2px var(--color-primary); Jemný vnitřní stín při focusu \*/

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

Tím, že nastavíme border-top-right-radius: 0; a border-bottom-right-radius: 0;, se pravá strana inputu hezky napojí na tlačítko.

**3. Stylování tlačítka (.newsletterButton):**

Tlačítko by mělo být vizuálně odlišné a obsahovat ikonu.

* **Výběr ikony:**
  + FaPaperPlane (papírové letadlo) z react-icons/fa je klasika.
  + FaEnvelope (obálka) nebo FaRegEnvelope.
  + FaArrowRight (šipka doprava) nebo FaChevronRight.
* **Úprava JSX v Footer.tsx (příklad s FaPaperPlane):**  
  Nezapomeň na import: import { FaPaperPlane } from 'react-icons/fa';
* <button type="submit" className={styles.newsletterButton} aria-label="Přihlásit k odběru novinek">
* <FaPaperPlane />
* {/\* Pokud chceš i text: \*/}
* {/\* <span className={styles.buttonText}>Odebírat</span> \*/}

</button>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Typescript jsx

Pokud chceš jen ikonu, text můžeš vynechat a použít aria-label pro přístupnost. Pokud chceš text i ikonu, můžeš text obalit do spanu a skrýt ho vizuálně pro menší obrazovky, pokud by bylo málo místa.

* **CSS pro .newsletterButton:**
* .newsletterButton {
* padding: 12px 18px; /\* Stejný vertikální padding jako input \*/
* background-color: var(--color-primary);
* color: white;
* border: none;
* /\* border-radius: 0; /\* Odstraníme radius, pokud je na .newsletterForm \*/
* /\* Aby neměl radius na levé straně: \*/
* border-top-left-radius: 0;
* border-bottom-left-radius: 0;
* border-top-right-radius: 6px; /\* Stejný radius jako .newsletterForm \*/
* border-bottom-right-radius: 6px; /\* Stejný radius jako .newsletterForm \*/
* cursor: pointer;
* transition: background-color 0.3s ease, transform 0.2s ease;
* display: flex; /\* Pro zarovnání ikony, pokud ji máš \*/
* align-items: center;
* justify-content: center;
* font-size: 1.1rem; /\* Velikost ikony (pokud používáš react-icons) \*/
* }
* .newsletterButton:hover {
* background-color: var(--color-primary-hover);
* transform: scale(1.05); /\* Jemné zvětšení při hoveru \*/
* }
* .newsletterButton:active {
* transform: scale(0.98); /\* Efekt při kliknutí \*/
* }
* /\* Pokud máš text v tlačítku, který chceš na mobilu skrýt a nechat jen ikonu \*/
* .buttonText {
* /\* margin-left: 8px; /\* Mezera od ikony \*/
* }
* /\* @media (max-width: např. 480px) {
* .buttonText {
* display: none; // Skryje text na malých obrazovkách
* }
* .newsletterButton {
* padding: 12px 15px; // Menší padding, když je jen ikona
* }

} \*/

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

**Celkový vzhled:**  
Kombinací těchto stylů bys měl dosáhnout kompaktního formuláře, kde input a tlačítko tvoří jeden vizuální celek. overflow: hidden a specifické border-radius na inputu a tlačítku zajistí, že vnější zaoblení bude z .newsletterForm a vnitřní hrany budou ostré.

**Příklad Footer.module.css s těmito úpravami:**

/\* Footer.module.css \*/

/\* ... (tvé ostatní styly) ... \*/

.footerColumn { /\* Jen pro kontext \*/

/\* ... \*/

}

.footerTitle { /\* ... \*/ }

.footerText { /\* ... \*/ }

.newsletterForm {

display: flex;

align-items: stretch;

margin-bottom: 15px;

border-radius: 6px;

overflow: hidden;

box-shadow: 0 1px 3px rgba(0,0,0,0.3); /\* Jemnější stín \*/

border: 1px solid rgba(255, 255, 255, 0.1); /\* Velmi jemný border pro definici na tmavém pozadí \*/

}

.newsletterInput {

flex-grow: 1;

padding: 12px 18px;

border: none;

background-color: rgba(0, 0, 0, 0.2); /\* Tmavší pozadí inputu pro kontrast s footerem \*/

color: white;

font-size: 0.95rem;

border-top-left-radius: 5px; /\* Radius z vnitřní strany, pokud má form vnější border \*/

border-bottom-left-radius: 5px;

outline: none;

transition: background-color 0.3s ease;

}

.newsletterInput::placeholder {

color: rgba(255, 255, 255, 0.5);

}

.newsletterInput:focus {

background-color: rgba(0, 0, 0, 0.35);

}

.newsletterButton {

padding: 12px 18px;

background-color: var(--color-primary);

color: white;

border: none;

border-top-right-radius: 5px; /\* Radius z vnitřní strany \*/

border-bottom-right-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease, transform 0.2s ease;

display: flex;

align-items: center;

justify-content: center;

font-size: 1.1rem; /\* Pro ikonu \*/

}

.newsletterButton:hover {

background-color: var(--color-primary-hover);

transform: scale(1.05);

}

.newsletterButton:active {

transform: scale(0.98);

}

.newsletterDisclaimer {

font-size: 0.8rem;

color: rgba(255,255,255,0.6); /\* O něco světlejší pro lepší čitelnost \*/

margin-top: 10px;

line-height: 1.4;

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Css

V tomto upraveném příkladu jsem dal border-radius i na vnitřní strany inputu/tlačítka a vnější border na .newsletterForm, což může vypadat také dobře. Experimentuj, co se ti líbí víc – zda má být zaoblený celý formulář jako jeden blok, nebo každý prvek zvlášť. Verze s overflow: hidden na formě a nulovým radiusem na spojích inputu/tlačítka je asi čistší pro ten "jednolitý" vzhled.

chci u toho checkboxu u kontaktniho formulare link na ty obchodni podminky a GDPR. celé kódy. cele cesty

Edit

Analyzoval přidání odkazů na obchodní podmínky a GDPR

6s

The user wants to modify the contact form checkbox for privacy consent to include links to the business terms (obchodni podminky) and GDPR (privacy policy). They specifically mention wanting the "whole code" and "whole paths."

Based on the files shared and the directory structure shown, I can see:

1. The contact form is in src/pages/kontakt.tsx
2. There's a CSS file for styling at src/pages/kontakt.module.css
3. There appears to be business terms files:
   * obchodni-podminky.module.css
   * obchodni-podminky.tsx

From what I can see in the code, currently the checkbox label for privacy consent is:

jsx

<label htmlFor="privacyConsent" className={styles.privacyLabel}>

Souhlasím se smluvními podmínkami a zásadami ochrany osobních údajů.\*

</label>

The user wants to modify this to add links to the business terms and GDPR pages. Let's create a solution that adds these links.

Based on the directory structure shown, the paths would be:

* Business terms page (obchodni-podminky): /obchodni-podminky
* GDPR page: This isn't explicitly shown in the directory tree, but it could potentially be part of the business terms or a separate page.

I'll provide an implementation that links to the business terms page at /obchodni-podminky and I'll assume there's a GDPR page at /gdpr (the user can adjust this path as needed).

I'll help you modify the checkbox to add links to the business terms and GDPR. Here's the implementation you need for the checkbox in the contact form:

Contact Form Checkbox with Links

Code

// In src/pages/kontakt.tsx // Import Link from Next.js at the top of the file with other imports import Link from 'next/link'; // Then replace the existing checkbox group with this code: <div className={styles.formGroup}> <div className={styles.c

Now, let's add the necessary CSS style for the links. You'll need to add this to your kontakt.module.css file:

CSS for Privacy Links

Code

/\* Add this to src/pages/kontakt.module.css \*/ .privacyLink { color: #0088cc; text-decoration: underline; transition: color 0.2s ease; } .privacyLink:hover { color: #006bb3; } :global(.dark-mode) .privacyLink { color: #00a0cc; } :global(

**Explanation:**

1. I've added the Next.js Link component to create proper navigation links to your pages.
2. The paths used are:
   * /obchodni-podminky for business terms
   * /gdpr for privacy policy/GDPR
3. If your GDPR page has a different path, you should update the second link accordingly.
4. The CSS styling makes the links stand out with the site's blue color scheme and adds hover effects.
5. Dark mode support is included to ensure the links remain visible in both light and dark themes.

If the GDPR page doesn't exist yet or has a different path than /gdpr, just adjust the path in the Link component to match your site structure.

Retry

D

ty mě nepochopil. chci celý kód co vložím přímo do toho souboru.

Edit

Připravil kompletní kód pro přímé vložení do souboru.

2s

I apologize for misunderstanding. The user wants me to provide the complete code that they can copy and paste directly into their file, not just the specific changes. I'll provide the entire code for the kontakt.tsx file with the checkbox modification included.

Based on the provided code snippets, I'll create a complete version of the file with the checkbox modification integrated. I'll make sure to include:

1. The full kontakt.tsx file with the checkbox modification
2. The CSS additions needed for the kontakt.module.css file

I should provide code that can be directly copied and pasted.

Omlouvám se za nepochopení. Zde je celý kód pro oba soubory, který můžete přímo vložit:

Kompletní kód pro src/pages/kontakt.tsx

Code

// src/pages/kontakt.tsx import React, { useState } from 'react'; import styles from './kontakt.module.css'; import Head from 'next/head'; import Link from 'next/link'; const KontaktPage: React.FC = () => { const [formData, setFormData] = useState

Kompletní kód pro src/pages/kontakt.module.css

Code

/\* src/pages/kontakt.module.css \*/ .kontaktContainer { max-width: 100%; overflow-x: hidden; } /\* Hero Section - stylizace podobná jako na stránce O mně \*/ .heroSection { position: relative; min-height: 300px; display: flex; justify-conte

Tyto dva soubory představují kompletní kód s přidanými odkazy na obchodní podmínky a GDPR. Stačí je vložit přímo do příslušných souborů:

1. Kód pro src/pages/kontakt.tsx - nahraďte celý současný soubor
2. Kód pro src/pages/kontakt.module.css - nahraďte celý současný soubor

Odkazy směřují na cesty:

* /obchodni-podminky pro obchodní podmínky
* /gdpr pro zásady ochrany osobních údajů

Pokud potřebujete upravit cesty odkazů, stačí změnit hodnoty href v Link komponentách.

Retry

D

ja tam chci jen jeden odkaz na ty obchodni podminky, nejak to vymysli

Edit

Crafted single link solution for business terms.

2s

Aha, teď rozumím. Chcete pouze jeden odkaz na obchodní podmínky, ne dva oddělené odkazy. Můžeme to vyřešit tak, že vytvoříme jediný odkaz, který povede na stránku s obchodními podmínkami, kde budou zároveň obsaženy i informace o GDPR/ochraně osobních údajů.

Upravím tedy kód tak, aby tam byl jen jeden odkaz na stránku obchodní podmínky.

Chápu, upravím kód tak, aby tam byl pouze jeden odkaz na obchodní podmínky. Zde je kompletní kód pro kontaktní formulář:

Kompletní kód pro src/pages/kontakt.tsx s jedním odkazem

Code

// src/pages/kontakt.tsx import React, { useState } from 'react'; import styles from './kontakt.module.css'; import Head from 'next/head'; import Link from 'next/link'; const KontaktPage: React.FC = () => { const [formData, setFormData] = useState

Kompletní kód pro src/pages/kontakt.module.css

Code

/\* src/pages/kontakt.module.css \*/ .kontaktContainer { max-width: 100%; overflow-x: hidden; } /\* Hero Section - stylizace podobná jako na stránce O mně \*/ .heroSection { position: relative; min-height: 300px; display: flex; justify-conte

Nyní jsem upravil kód tak, aby obsahoval pouze jeden odkaz na obchodní podmínky, který zahrnuje i zmínku o zásadách ochrany osobních údajů. Text odkazu je nyní "obchodními podmínkami a zásadami ochrany osobních údajů" a vede na cestu "/obchodni-podminky".

Toto řešení předpokládá, že na této stránce budou obsaženy jak obchodní podmínky, tak i informace o ochraně osobních údajů (GDPR).

Kód je připraven k přímému vložení do příslušných souborů:

1. src/pages/kontakt.tsx
2. src/pages/kontakt.module.css

Retry

D

jinak u toho email senderu. existuje nějaký jiný (lepší) způsob jak si odeslat data na email než to jak to mam použito já? napiš mi a pak si kdyžtak vyberu

Edit

Analyzoval alternativní způsoby odesílání formulářových dat e-mailem.

19s

Vidím, že aktuálně používáte FormSubmit.co (action="<https://formsubmit.co/zidomination@gmail.com>"). Existuje několik alternativních a případně lepších způsobů, jak odesílat formulářová data na e-mail. Popíšu nejběžnější z nich:

**1. Vlastní serverové řešení s Next.js API Routes**

Next.js poskytuje API Routes, které umožňují vytvářet serverové endpointy přímo ve vaší aplikaci. To vám dává větší kontrolu nad zpracováním formuláře, validací dat a odesíláním e-mailů.

javascript

*// pages/api/send-email.js*

import nodemailer from 'nodemailer';

export default async function handler(req, res) {

if (req.method === 'POST') {

try {

const transporter = nodemailer.createTransport({

host: process.env.EMAIL\_HOST, *// např. 'smtp.gmail.com'*

port: 587,

secure: false,

auth: {

user: process.env.EMAIL\_USER, *// váš e-mail*

pass: process.env.EMAIL\_PASS, *// heslo nebo app password*

},

});

const { name, email, phone, message, */\* další pole \*/* } = req.body;

const mailOptions = {

from: process.env.EMAIL\_USER,

to: 'vas@email.cz', *// kam se mají e-maily odesílat*

subject: `Nová zpráva z kontaktního formuláře od ${name}`,

text: `Jméno: ${name}\nE-mail: ${email}\nTelefon: ${phone}\n\nZpráva: ${message}\n\n...`, *// zformátujte podle potřeby*

html: `<p><strong>Jméno:</strong> ${name}</p>

<p><strong>E-mail:</strong> ${email}</p>

<p><strong>Telefon:</strong> ${phone}</p>

...

<p><strong>Zpráva:</strong> ${message}</p>`,

};

await transporter.sendMail(mailOptions);

res.status(200).json({ success: true });

} catch (error) {

console.error('Chyba při odesílání e-mailu:', error);

res.status(500).json({ success: false, error: 'Nepodařilo se odeslat e-mail' });

}

} else {

res.setHeader('Allow', ['POST']);

res.status(405).end(`Method ${req.method} Not Allowed`);

}

}

A pak byste ve formuláři místo atributu action použili vlastní funkci pro odeslání dat:

jsx

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await fetch('/api/send-email', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(formData),

});

const data = await response.json();

if (data.success) {

*// Zobrazit zprávu o úspěchu*

} else {

*// Zobrazit chybu*

}

} catch (error) {

console.error('Chyba při odesílání formuláře:', error);

*// Zobrazit chybu*

}

};

**2. Emailjs**

Emailjs je populární služba, která umožňuje odesílat e-maily přímo z klientského JavaScriptu bez nutnosti vlastního serveru.

jsx

*// Nejprve nainstalujte: npm install @emailjs/browser*

import emailjs from '@emailjs/browser';

import { useRef } from 'react';

const KontaktPage = () => {

const form = useRef();

const handleSubmit = (e) => {

e.preventDefault();

emailjs.sendForm(

'YOUR\_SERVICE\_ID', *// ze služby na emailjs.com*

'YOUR\_TEMPLATE\_ID', *// z šablony na emailjs.com*

form.current,

'YOUR\_PUBLIC\_KEY' *// veřejný klíč z vašeho účtu*

)

.then((result) => {

console.log('E-mail odeslán!', result.text);

*// Zobrazit zprávu o úspěchu*

}, (error) => {

console.error('Chyba při odesílání:', error.text);

*// Zobrazit chybu*

});

};

return (

<form ref={form} onSubmit={handleSubmit}>

{*/\* Vaše formulářová pole \*/*}

</form>

);

};

**3. Formspree**

Formspree je další populární služba podobná FormSubmit, ale nabízí více funkcí v bezplatné i placené verzi.

jsx

<form

action="https://formspree.io/f/your\_form\_id"

method="POST"

>

{*/\* Vaše formulářová pole \*/*}

</form>

**4. Netlify Forms**

Pokud hostujete svou stránku na Netlify, můžete využít Netlify Forms.

jsx

<form

name="kontakt"

method="POST"

data-netlify="true"

data-netlify-honeypot="bot-field"

>

<input type="hidden" name="form-name" value="kontakt" />

{*/\* Vaše formulářová pole \*/*}

</form>

**5. Google Forms**

Můžete také vytvořit formulář v Google Forms a přesměrovat na něj, nebo ho vložit do stránky.

**6. Zapier Integration**

Zapier umožňuje propojit formulář s e-mailem a mnoha dalšími službami.

**7. SendGrid API nebo Mailchimp API**

Tyto služby poskytují robustní API pro odesílání e-mailů a řízení jejich doručování.

Popíšu je všechny a můžete si vybrat, které by pro vás mohlo být nejlepší.

Existuje několik lepších alternativ k FormSubmit.co pro odesílání dat na e-mail. Každá má své výhody:

Alternativní způsoby odesílání e-mailů z formuláře

Code

# Alternativy pro odesílání e-mailů z kontaktního formuláře ## 1. Next.js API Routes s Nodemailer \*\*Výhody:\*\* - Plná kontrola nad zpracováním dat - Vlastní formátování e-mailů - Možnost validace před odesláním - Možnost ukládat data do databáze - L

Zde jsou čtyři nejlepší alternativy k vašemu současnému řešení s FormSubmit:

**1. Vlastní API v Next.js s Nodemailer**

**Výhody:**

* Plná kontrola nad zpracováním dat
* Možnost formátování e-mailů podle vašich potřeb
* Data se nikdy nedostanou do rukou třetí strany
* Můžete jednoduše implementovat ochranu proti spamu
* Možnost ukládat data současně do databáze

**Nevýhody:**

* Potřebujete nakonfigurovat SMTP server
* Musíte implementovat ošetření chyb a validaci

**2. EmailJS**

**Výhody:**

* Snadná implementace bez vlastního backendu
* Profesionální doručování e-mailů
* Vlastní šablony a možnost přizpůsobení
* Bezplatný plán (100 e-mailů měsíčně)

**Nevýhody:**

* Omezení v bezplatné verzi
* API klíč je viditelný na klientovi

**3. Formspree**

**Výhody:**

* Podobné jako FormSubmit, ale s lepší spolehlivostí
* Lepší funkce pro ochranu proti spamu
* Příjemnější uživatelské rozhraní
* Jednoduchá implementace

**Nevýhody:**

* Omezení v bezplatné verzi (50 e-mailů měsíčně)

**4. React-Hook-Form + vlastní API**

**Výhody:**

* Nejlepší validace na straně klienta
* Lepší uživatelská zkušenost
* Kombinuje se s libovolným backendem pro odesílání
* Snižuje počet vykreslování formuláře

**Nevýhody:**

* Složitější implementace než statické řešení

Pro menší projekty bych doporučil **Formspree** jako přímou náhradu za FormSubmit, nebo **EmailJS** pro trochu více funkcí. Pro profesionální a škálovatelné řešení je nejlepší vlastní **Next.js API endpoint s Nodemailer**.

Podrobnější implementační kódy najdete v přiloženém dokumentu. Mohu vám pomoci s implementací vámi vybraného řešení.

Retry

D

tak chci udělat tu varianti č.1 řekni mi co vše musím udělat. Můj email, který chci kam to chodit je zidomination@gmail.com. Co víc po mně potřebuješ??? Vše ti řeknu. Budu chtít celý kód co muzu rovnou použít atd.

Edit

Naplánoval kompletní řešení e-mailového formuláře v Next.js.

6s

Uživatel chce implementovat variantu 1, tedy Next.js API Route s Nodemailer. Potřebuji vytvořit kompletní řešení pro odesílání e-mailů z kontaktního formuláře. Potřebuji mu poskytnout:

1. Kód pro API endpoint (pages/api/send-email.ts)
2. Úpravu kontaktního formuláře (pages/kontakt.tsx)
3. Instrukce pro nastavení proměnných prostředí
4. Instrukce pro konfiguraci a instalaci potřebných balíčků

Uživatel mi sdělil, že chce, aby e-maily chodily na [zidomination@gmail.com](mailto:zidomination@gmail.com).

Pro Gmail budu potřebovat:

* Vytvořit heslo pro aplikaci v Google účtu, nebo použít klasické heslo (méně bezpečné)
* Nastavit proměnné prostředí v projektu

Budu muset také zohlednit formát dat z kontaktního formuláře, který už uživatel má, a zajistit, že všechna pole budou správně přenesena do e-mailu.

Ukážu, jak nastavit proměnné prostředí v Next.js a jak nakonfigurovat SMTP pro Gmail.

Podívejme se na strukturu kompletního řešení:

1. Instalace Nodemailer
2. Vytvoření API endpointu
3. Úprava formuláře pro použití API endpointu
4. Přidání zpracování odeslání a UI zpětné vazby
5. Nastavení proměnných prostředí

Poskytnu uživateli kompletní kód, který může použít, a vysvětlím vše, co potřebuje vědět.

Rád vám pomůžu s implementací řešení pomocí Next.js API Route a Nodemailer. Pojďme to udělat krok za krokem s kompletním kódem, který můžete rovnou použít.

**Krok 1: Instalace potřebných balíčků**

Nejprve musíte do vašeho projektu nainstalovat Nodemailer:

bash

npm install nodemailer

**Krok 2: Vytvoření API endpointu**

pages/api/send-email.ts

Code

// pages/api/send-email.ts import type { NextApiRequest, NextApiResponse } from 'next'; import nodemailer from 'nodemailer'; // Definice typu pro odpověď API type ResponseData = { success: boolean; message?: string; error?: string; }; export

**Krok 3: Úprava kontaktního formuláře**

src/pages/kontakt.tsx (upravený)

Code

// src/pages/kontakt.tsx import React, { useState } from 'react'; import styles from './kontakt.module.css'; import Head from 'next/head'; import Link from 'next/link'; const KontaktPage: React.FC = () => { const [formData, setFormData] = useState

**Krok 4: Přidání CSS stylů pro nové prvky**

Přidání nových CSS stylů (přidat na konec kontakt.module.css)

Code

/\* Přidejte tyto styly na konec souboru src/pages/kontakt.module.css \*/ /\* Styly pro chybové a úspěšné zprávy \*/ .errorMessage { background-color: rgba(255, 87, 87, 0.1); border-left: 4px solid #ff5757; color: #d32f2f; padding: 15px; borde

**Krok 5: Nastavení proměnných prostředí**

Pro nastavení proměnných prostředí vytvořte soubor .env.local v kořenovém adresáři vašeho projektu:

.env.local

Code

# .env.local EMAIL\_USER=zidomination@gmail.com EMAIL\_APP\_PASSWORD=your-app-password-here

**Krok 6: Vytvoření hesla pro aplikaci v Google**

Pro Gmail potřebujete vytvořit heslo pro aplikaci, protože Google vyžaduje dvoufaktorové ověření a speciální heslo pro aplikace třetích stran:

1. Přejděte na <https://myaccount.google.com/security>
2. Pokud ještě nemáte aktivní dvoufaktorové ověření, aktivujte ho
3. Přejděte na "Aplikační hesla" (v sekci "Přihlášení do Google")
4. Vytvořte nové heslo pro aplikaci - vyberte "Jiné" a pojmenujte ho např. "Fitness Web"
5. Zkopírujte vygenerované heslo a vložte ho do .env.local jako hodnotu EMAIL\_APP\_PASSWORD

**Důležité**: Toto heslo je citlivé, nikdy ho neukládejte do veřejných repozitářů a vždy ho držte v bezpečí!

**Přehled implementace**

Co jsme vytvořili:

1. **API endpoint** (pages/api/send-email.ts) pro zpracování odesílání e-mailů
2. **Upravený formulář** (src/pages/kontakt.tsx) s logikou pro odeslání dat na API
3. **Nové CSS styly** pro zobrazení stavů odesílání a chybových zpráv
4. **Konfiguraci e-mailu** pomocí proměnných prostředí

**Bezpečnostní poznámky**

1. Váš e-mail a heslo pro aplikaci jsou vždy uloženy pouze na serveru a nikdy se neposlají do prohlížeče.
2. API endpoint přijímá pouze POST požadavky, čímž se snižuje riziko zneužití.
3. Validace formuláře probíhá jak na straně klienta, tak na straně serveru.

**Co potřebujete dodat**

* Heslo pro aplikaci z vašeho Google účtu (viz Krok 6)

Potřebujete ještě nějaké další informace nebo úpravy? Například přizpůsobit formát e-mailu nebo přidat další funkce?

Retry

D

dej mi cele CSS prosím upravene

Edit

Připravil kompletní CSS formulář s novými styly.

3s

Uživatel chce kompletní CSS kód pro kontaktní formulář, který by zahrnoval všechny původní styly plus nově přidané styly pro zpracování odesílání formuláře a zobrazení stavových zpráv.

Původní CSS kód z sdíleného souboru je v dokumentu paste-2.txt a obsahuje styling pro formulář. Přidal jsem několik nových CSS tříd pro zpracování úspěšných a chybových zpráv, které jsem umístil do samostatného artefaktu.

Nyní potřebuji vytvořit kompletní CSS soubor, který bude obsahovat vše z původního CSS souboru plus nově přidané styly. Budu proto kombinovat původní CSS z paste-2.txt s novými styly, které jsem navrhl.

Zde je kompletní aktualizovaný CSS soubor s přidanými styly:

Kompletní kód pro src/pages/kontakt.module.css

Code

/\* src/pages/kontakt.module.css \*/ .kontaktContainer { max-width: 100%; overflow-x: hidden; } /\* Hero Section - stylizace podobná jako na stránce O mně \*/ .heroSection { position: relative; min-height: 300px; display: flex; justify-conte

Nyní máte kompletní implementaci emailového odesílání formuláře s využitím Next.js API Routes a Nodemailer. Zde je shrnutí všech souborů, které jsem vám připravil:

**1. API Endpoint**

* **Soubor**: pages/api/send-email.ts
* **Účel**: Zpracování odesílání e-mailů na serveru

**2. Upravený kontaktní formulář**

* **Soubor**: src/pages/kontakt.tsx
* **Účel**: Formulář s React logikou pro odesílání dat na API

**3. Kompletní CSS**

* **Soubor**: src/pages/kontakt.module.css
* **Účel**: Styly pro formulář a všechny stavy

**4. Proměnné prostředí**

* **Soubor**: .env.local
* **Účel**: Uložení citlivých údajů (e-mail a heslo pro aplikaci)

Pro správné fungování potřebujete:

1. Nainstalovat Nodemailer: npm install nodemailer
2. Vytvořit heslo pro aplikaci Google:
   * Přejděte na <https://myaccount.google.com/security>
   * Aktivujte dvoufaktorové ověření, pokud ho ještě nemáte
   * Vytvořte heslo pro aplikaci a vložte ho do .env.local

Toto řešení je bezpečné, profesionální a plně pod vaší kontrolou. Nemusíte se spoléhat na externí služby a máte možnost si e-maily formátovat přesně podle vašich potřeb.

Retry

D

Krok 6: Vytvoření hesla pro aplikaci v Google Pro Gmail potřebujete vytvořit heslo pro aplikaci, protože Google vyžaduje dvoufaktorové ověření a speciální heslo pro aplikace třetích stran: 1. Přejděte na <https://myaccount.google.com/security> 2. Pokud ještě nemáte aktivní dvoufaktorové ověření, aktivujte ho 3. Přejděte na "Aplikační hesla" (v sekci "Přihlášení do Google") 4. Vytvořte nové heslo pro aplikaci - vyberte "Jiné" a pojmenujte ho např. "Fitness Web" 5. Zkopírujte vygenerované heslo a vložte ho do .env.local jako hodnotu EMAIL\_APP\_PASSWORD mám google v angličtině, tak prosím naviguj anglicky tento krok

Edit

Navigoval kroky pro vytvoření hesla aplikace v Google.

2s

I need to provide the instructions for creating an app password in Google when the user's Google interface is in English. Let me translate those instructions to English:

1. Go to <https://myaccount.google.com/security>
2. If you don't have two-factor authentication (2FA) active yet, activate it
3. Go to "App passwords" (in the "Signing in to Google" section)
4. Create a new app password - select "Other" and name it e.g. "Fitness Web"
5. Copy the generated password and paste it into .env.local as the value for EMAIL\_APP\_PASSWORD

**Step 6: Creating an App Password in Google (English Instructions)**

For Gmail, you need to create an app password because Google requires two-factor authentication and special passwords for third-party applications:

1. Go to <https://myaccount.google.com/security>
2. If you don't have two-factor authentication (2FA) activated yet, enable it first
3. Find and click on "App passwords" (located in the "Signing in to Google" section)
4. Create a new app password:
   * Select "Other" from the dropdown menu
   * Name it something like "Fitness Web"
   * Click "Create"
5. Google will generate a 16-character password without spaces
6. Copy this generated password and paste it into your .env.local file as the value for EMAIL\_APP\_PASSWORD

**Important**: This password gives access to your email account, so never share it or commit it to public repositories. Google only shows this password once, so make sure you copy it correctly before closing the window.

Retry

D

je toto ono ? nebo ne

Edit

Odhalil rozdíl mezi správcem hesel a hesly aplikací

2s

The image shows a Google "Password Manager" interface. This shows that the user has 39 passwords saved in their Google Account, with a button to "Manage passwords."

However, this is NOT the App Passwords section we're looking for. The App Passwords feature is a separate feature specifically for creating special passwords that allow third-party applications to access your Google account when you have 2-Step Verification turned on.

I need to clarify that this is not the correct section, and direct them to the specific "App passwords" section in their Google Account security settings.

No, this is not the App Passwords section we need. This is the general Google Password Manager, which stores passwords for websites you log into.

For sending emails from your application, you need to find the "App passwords" section, which is specifically for creating special access passwords for applications when you have 2-Step Verification enabled.

Here's how to find it:

1. Go to <https://myaccount.google.com/security>
2. Scroll down to the "Signing in to Google" section
3. Look for "2-Step Verification" - you need to have this enabled first
4. After enabling 2-Step Verification, you should see "App passwords" as an option below it
5. Click on "App passwords" to create a new application-specific password

The App passwords section will let you generate a 16-character password without spaces that your application can use to authenticate with Google's servers.

If you don't see the "App passwords" option, make sure 2-Step Verification is fully enabled on your account first.

Retry

D

ja tam nic takove ho nevidim

Edit

Analyzoval možné důvody, proč uživatel nenachází hesla aplikací.

1s

If the user can't find the "App passwords" section, there could be a few reasons:

1. They might not have 2-Step Verification fully enabled yet. This is a prerequisite for App passwords to appear.
2. The Google interface might have changed, or there might be regional/language differences.
3. They might be looking in the wrong section.

Let me provide alternative instructions to help them find it or offer an alternative solution.

Rozumím, že nemůžete najít sekci "App passwords". Toto se může stát z několika důvodů:

1. Možná ještě nemáte plně aktivované dvoufaktorové ověřování (2-Step Verification). To je nutné předtím, než se zobrazí možnost App passwords.
2. Google občas mění své rozhraní.

Zkusme alternativní postup:

1. Přejděte přímo na stránku App passwords pomocí tohoto odkazu: <https://myaccount.google.com/apppasswords>
2. Pokud vás stránka vyzve k přihlášení, přihlaste se.
3. Pokud ani tento odkaz nefunguje nebo se stránka nenačte, existuje alternativní řešení:

**Alternativní řešení - použití "Less secure app access":**

1. Google nedávno ukončil podporu pro "Less secure app access", což byla dříve jednodušší alternativa.
2. Místo toho můžeme použít službu jako SendGrid nebo Mailjet, které nepotřebují App password:

Alternativní řešení pomocí SendGrid

Code

// 1. Nejprve nainstalujte SendGrid: npm install @sendgrid/mail // 2. Nahraďte pages/api/send-email.ts tímto kódem: import type { NextApiRequest, NextApiResponse } from 'next'; import sgMail from '@sendgrid/mail'; type ResponseData = { success: b

Pro použití SendGrid byste museli:

1. Vytvořit účet na SendGrid (mají bezplatný plán s 100 e-maily denně): <https://signup.sendgrid.com/>
2. Ověřit vaši e-mailovou doménu nebo adresu
3. Získat API klíč ze sekce Settings > API Keys
4. Vytvořit .env.local soubor s těmito proměnnými:

SENDGRID\_API\_KEY=váš\_api\_klíč\_zde

SENDGRID\_VERIFIED\_SENDER=vaše\_ověřená\_emailová\_adresa

Chcete-li použít toto řešení se SendGrid místo původního s Nodemailer a Google App Passwords? Je to jednodušší na nastavení a má vyšší spolehlivost doručování.

Retry

D

už to mám, heslo ti mám sdelit?

Edit

Varoval před sdílením hesla a doporučil bezpečnostní postup.

2s

The user is indicating they've successfully gotten the app password for their Google account, and they're asking whether they should share this password with me.

I should be very clear that they should absolutely NOT share their app password with me or anyone else. App passwords provide access to the account they're generated for, and sharing them would be a security risk.

I should:

1. Clearly advise against sharing the password
2. Explain that they should keep it confidential and only enter it in their own .env.local file
3. Provide guidance on the next steps now that they have the password

Ne, v žádném případě mi nesdělujte své heslo!

Heslo pro aplikaci je citlivý údaj, který byste nikdy neměli sdílet s nikým - ani se mnou ani s jinou osobou nebo službou. Toto heslo poskytuje přístup k vašemu e-mailovému účtu a mělo by zůstat důvěrné.

Co byste měli udělat s tímto heslem:

1. Vložte ho pouze do souboru .env.local ve vašem projektu jako hodnotu pro EMAIL\_APP\_PASSWORD
2. Ujistěte se, že soubor .env.local je přidán do vašeho .gitignore, aby se nedostal do verzovacího systému

Když už máte heslo, můžete pokračovat k dokončení implementace:

1. Vytvořte/upravte soubor .env.local v kořenovém adresáři projektu:

EMAIL\_USER=zidomination@gmail.com

EMAIL\_APP\_PASSWORD=vase\_app\_heslo

1. Spusťte server pomocí npm run dev
2. Otestujte formulář - měl by nyní odesílat e-maily na váš e-mail

Existuje něco dalšího, s čím vám mohu pomoci při implementaci?

Retry

D

Edit

Diagnosed TypeScript type errors for nodemailer module.

2s

The error message shows that there's an issue with the TypeScript types for the nodemailer module. This is a common issue when working with TypeScript and nodemailer. Let's fix this by installing the type definitions for nodemailer.

The error messages are:

1. "Could not find a declaration file for module 'nodemailer'..." - This means TypeScript doesn't know the types for nodemailer
2. "Unexpected any. Specify a different type." - This is an ESLint error related to TypeScript's "noImplicitAny" rule

To fix this, we need to install the @types/nodemailer package, which provides TypeScript definitions for nodemailer.

Vidím chybu s TypeScript typy pro nodemailer. Tento problém je běžný, když pracujete s TypeScript a nodemailer. Potřebujete nainstalovat typové definice:

bash

npm install --save-dev @types/nodemailer

Tato chyba nastává, protože TypeScript nemůže najít definiční soubor pro modul nodemailer, což způsobuje, že implicitně používá typ 'any', a vaše ESLint pravidla to zakazují.

Po instalaci @types/nodemailer by měla chyba zmizet. Pokud by se to nestalo, je také možné upravit váš tsconfig.json a/nebo .eslintrc.js soubor, ale nejprve zkuste nainstalovat typové definice - to obvykle problém vyřeší.

Pokud by chyba přetrvávala, dejte mi vědět a můžeme zkusit další řešení.

Retry

D

Edit

Resolved TypeScript ESLint error with precise error typing.

2s

There's still an ESLint error about using 'any' for the error parameter in the catch block. In TypeScript, it's better to type the error parameter properly instead of using 'any', which ESLint is warning about with the rule "no-explicit-any".

In the code, I can see on line 286:

typescript

} catch (error: any) {

The error is that you're explicitly typing the error as 'any' which is against the ESLint rule. Let's fix this by using a more specific type for the error.

Vidím, že stále máte ESLint chybu související s explicitním použitím any v bloku catch. Tuto chybu můžeme vyřešit dvěma způsoby:

**Řešení 1: Použijte konkrétnější typ chyby**

typescript

} catch (error: Error) {

console.error('Chyba při odesílání e-mailu:', error);

*// Odpověď při chybě*

res.status(500).json({

success: false,

error: 'Chyba při odesílání e-mailu. Zkuste to prosím později nebo nás kontaktujte jiným způsobem.'

});

}

**Řešení 2: Vypněte pravidlo ESLint pro tento řádek**

typescript

} catch (error: any) { *// eslint-disable-line @typescript-eslint/no-explicit-any*

console.error('Chyba při odesílání e-mailu:', error);

*// Odpověď při chybě*

res.status(500).json({

success: false,

error: 'Chyba při odesílání e-mailu. Zkuste to prosím později nebo nás kontaktujte jiným způsobem.'

});

}

**Řešení 3: Použijte neznámý typ (unknown)**

typescript

} catch (error: unknown) {

console.error('Chyba při odesílání e-mailu:', error);

*// Odpověď při chybě*

res.status(500).json({

success: false,

error: 'Chyba při odesílání e-mailu. Zkuste to prosím později nebo nás kontaktujte jiným způsobem.'

});

}

Doporučuji řešení č. 3, protože unknown je typově bezpečnější než any a nutí vás provést typovou kontrolu před použitím hodnoty. Je to moderní přístup preferovaný v TypeScriptu.

Retry

D

dobře. můj projekt je asi ted hotový. jak ho hostnu na Vercelu?

Edit

Připravil postup nasazení Next.js projektu na Vercel.

1s

The user is asking how to host their Next.js project on Vercel. Let me provide them with comprehensive steps for deploying a Next.js project to Vercel.

Vercel is an excellent platform for hosting Next.js applications because:

1. It's created by the same team behind Next.js
2. It has excellent integration and optimization for Next.js projects
3. It offers a generous free tier
4. It has automatic deployments from Git

Here are the steps the user needs to follow:

**Jak hostovat Next.js projekt na Vercel**

Vercel je ideální platforma pro hostování Next.js aplikací, protože je vytvořená stejným týmem, který stojí za Next.js. Zde je jednoduchý postup:

**Krok 1: Příprava projektu**

1. **Ujistěte se, že máte Git repozitář**:
   * Pokud ještě nemáte iniciovaný Git, proveďte:

bash

git init

git add .

git commit -m "Initial commit"

1. **Nahrajte projekt na GitHub/GitLab/Bitbucket**:
   * Vytvořte repozitář na některé z těchto platforem
   * Nahrajte svůj kód do repozitáře

**Krok 2: Nastavení na Vercel**

1. **Vytvořte účet na Vercel**:
   * Jděte na [vercel.com](https://vercel.com/) a zaregistrujte se
   * Můžete se přihlásit pomocí GitHub, GitLab nebo Bitbucket (doporučuji stejnou platformu, kde máte kód)
2. **Importujte projekt**:
   * Po přihlášení klikněte na tlačítko "Add New" a vyberte "Project"
   * Vyberte svůj GitHub/GitLab/Bitbucket účet a autorizujte Vercel
   * Vyberte váš repozitář ze seznamu
3. **Konfigurace projektu**:
   * Vercel automaticky detekuje, že jde o Next.js projekt
   * Nastavte proměnné prostředí (Environment Variables) - jsou to ty samé, které máte v .env.local:
     + Přidejte EMAIL\_USER s hodnotou zidomination@gmail.com
     + Přidejte EMAIL\_APP\_PASSWORD s hodnotou vašeho hesla pro aplikaci (z Google)
   * Nechte ostatní nastavení výchozí (Build and Output Settings)
   * Klikněte na "Deploy"

**Krok 3: Po nasazení**

1. **Otestujte svůj web**:
   * Vercel vám poskytne URL (např. vase-aplikace.vercel.app)
   * Otevřete tuto URL a zkontrolujte, že vše funguje správně
   * Otestujte kontaktní formulář
2. **Připojte vlastní doménu (volitelné)**:
   * V nastavení projektu klikněte na "Domains"
   * Přidejte svou vlastní doménu a postupujte podle pokynů pro nastavení DNS

**Další výhody Vercel**

* **CI/CD**: Každý push do vašeho repozitáře automaticky spustí nové nasazení
* **Preview deployments**: Pro každý pull request se vytvoří náhledová verze
* **Analytika**: Základní analytika návštěvnosti je k dispozici
* **Serverless Functions**: Vaše API endpoints v /pages/api budou automaticky nasazeny jako serverless funkce

**Důležité poznámky**

1. **Omezení na bezplatném plánu**:
   * Maximální velikost nasazení: 100MB (bez node\_modules)
   * Omezené množství vyžádání (requestů) za den
   * Omezený počet serverless invokací
2. **Proměnné prostředí**:
   * Nezapomeňte, že proměnné prostředí nastavené v Vercel dashboardu jsou ekvivalentní vašemu lokálnímu .env.local souboru
   * Můžete mít různé proměnné pro Production, Preview a Development prostředí

Potřebujete další pomoc s nasazením na Vercel?

Retry

D

co tu??

Edit

Analyzoval postup nasazení projektu na Vercel.

5s

The image shows the Vercel deployment interface where you're setting up a new project. This is exactly what I described in my previous instructions about deploying to Vercel. Let me break down what we're seeing:

1. At the top, it says "New Project" and shows it's importing from GitHub (specifically from DavidUdic-P1A/rp with branch "main" and it shows "fitness-web").
2. Under "Vercel Team" it shows "Dave's projects" with "Hobby" tier selected.
3. The "Project Name" field has "rp" entered.
4. Framework Preset is correctly detected as "Next.js".
5. Root Directory is set to "fitness-web".
6. There are collapsible sections for "Build and Output Settings" and "Environment Variables" that are currently collapsed.
7. At the bottom is a "Deploy" button.

The user is asking "co tu??" which in Czech means "what's this??" or "what now??".

I should explain what they need to do next, which is to set the environment variables (EMAIL\_USER and EMAIL\_APP\_PASSWORD) before deploying.

Na tomto místě potřebujete nastavit proměnné prostředí pro odesílání e-mailů. Klikněte na šipku vedle "Environment Variables" pro rozbalení této sekce a přidejte tyto dvě proměnné:

1. Klikněte na šipku ▶️ vedle "Environment Variables" (aby se sekce rozbalila)
2. Přidejte tyto proměnné:
   * Jméno: EMAIL\_USER
   * Hodnota: zidomination@gmail.com

A pak druhou proměnnou:

* + Jméno: EMAIL\_APP\_PASSWORD
  + Hodnota: [vložte vaše Google App heslo]

1. Po nastavení těchto proměnných klikněte na tlačítko "Deploy"

Tyto proměnné prostředí jsou nezbytné, aby váš kontaktní formulář fungoval s odesíláním e-mailů. Bez nich by API endpoint nemohl odesílat e-maily.

Zbytek nastavení vypadá správně:

* Framework je správně detekován jako Next.js
* Root adresář je správně nastaven na "fitness-web"
* Vercel automaticky rozpozná build nastavení

Retry

D

Edit

Mapped out Vercel environment variables for email setup.

3s

This image shows the Environment Variables section in the Vercel deployment interface, which is expanded now. There's an example environment variable already shown (EXAMPLE\_NAME with a value of I9JU23NF394R6HH).

The user needs to:

1. Delete the example environment variable (by clicking the "-" button on the right) or modify it
2. Add their actual environment variables for the email functionality

They need to add two environment variables:

* EMAIL\_USER: [zidomination@gmail.com](mailto:zidomination@gmail.com)
* EMAIL\_APP\_PASSWORD: (their Google App password)

There's also an "Add More" button to add additional environment variables.

Teď vidím, že máte otevřenou sekci Environment Variables. Potřebujete:

1. Buď smažte ukázkovou proměnnou EXAMPLE\_NAME (kliknutím na tlačítko "-" vpravo) nebo ji přepište na vaši první proměnnou.
2. Přidejte tyto dvě proměnné:
   * EMAIL\_USER: [zidomination@gmail.com](mailto:zidomination@gmail.com)
   * EMAIL\_APP\_PASSWORD: [vaše Google App heslo]

Konkrétně:

* Do pole "Key" napište "EMAIL\_USER" a do pole "Value" vložte "[zidomination@gmail.com](mailto:zidomination@gmail.com)"
* Potom klikněte na "Add More"
* Do nového pole "Key" napište "EMAIL\_APP\_PASSWORD" a do pole "Value" vložte vaše Google App heslo

Po přidání obou proměnných klikněte na tlačítko "Deploy" dole.

Tyto proměnné prostředí jsou nezbytné pro správnou funkci odesílání e-mailů ve vaší aplikaci.

Retry

D

Edit

Diagnosed JSX quote escaping issue in TypeScript file.

5s

I can see there's an error with the build. The compiler is complaining about errors in the file ./src/pages/obchodni-podminky.tsx. The errors seem to be related to escape sequences for single quotes (').

The error messages repeatedly state: Error: "'" can be escaped with "&quot;", "&ldquo;", which suggests there may be unescaped quote marks in your JSX/TSX code in the obchodni-podminky (business terms) page.

This is a common issue in JSX when you have text content with apostrophes (single quotes) that aren't properly escaped in JSX strings. In JSX, you need to be careful with quotes and special characters.

To fix this issue, I would need to:

1. Look at the content of the obchodni-podminky.tsx file
2. Find where the single quotes are causing problems
3. Replace them with properly escaped versions

Here are a few ways to fix this:

1. Replace all instances of ' with &apos; in your JSX
2. Use double quotes (") to wrap strings that contain single quotes
3. Use template literals (backticks) if you need both single and double quotes

I would need to see the code to provide a specific fix, but this is definitely a syntax error with quote escaping in your JSX content.

Vidím, že máte chyby při kompilaci v souboru ./src/pages/obchodni-podminky.tsx. Jde o problém s apostrofy/uvozovkami v textu.

Chyba říká: Error: "'" can be escaped with "&quot;", "&ldquo;" - což znamená, že v souboru máte nesprávně escapované apostrofy (jednoduché uvozovky).

**Jak to opravit:**

1. Otevřete soubor src/pages/obchodni-podminky.tsx
2. Nahraďte všechny apostrofy (') v textu jednou z těchto možností:
   * Použijte HTML entity: &apos; místo '
   * Nebo použijte JS escape sekvenci: \' místo '
   * Nebo obalte texty s apostrofy do dvojitých uvozovek nebo do backticks (`)

**Příklad:**

tsx

*// Problematický kód*

<p>Toto je text s 'apostrofem' který způsobuje problémy.</p>

*// Opravená verze 1 (HTML entity)*

<p>Toto je text s &apos;apostrofem&apos; který způsobuje problémy.</p>

*// Opravená verze 2 (JavaScript escape sekvence)*

<p>Toto je text s \'apostrofem\' který způsobuje problémy.</p>

*// Opravená verze 3 (Template literals)*

<p>{`Toto je text s 'apostrofem' který způsobuje problémy.`}</p>

Pokud nemáte přístup k tomuto souboru v běžném editoru, můžete:

1. Stáhnout projekt zpět lokálně
2. Opravit soubor
3. Udělat nový commit a push
4. Nechat Vercel znovu nasadit projekt

Retry

// src/pages/obchodni-podminky.tsx import React from 'react'; import styles from './obchodni-podminky.module.css'; import Head from 'next/head'; import Link from 'next/link'; const TermsPage: React.FC = () => { return ( <> <Head> <title>Obchodní podmínky a GDPR | Fitne

pasted

D

Edit

Diagnosed and proposed solutions for JSX apostrophe escaping issues.

8s

Tento soubor obsahuje obchodní podmínky pro fitness coaching web. Chyba, kterou jsme viděli v build logu, se týká apostrofů (') v souboru src/pages/obchodni-podminky.tsx.

Chyba říká: Error: "'" can be escaped with "&quot;", "&ldquo;" - což znamená, že v souboru jsou nesprávně escapované apostrofy (jednoduché uvozovky).

Podívejme se na obsah souboru a najděme všechny problematické apostrofy. V JSX je potřeba apostrofy v textu escapovat, protože JSX používá apostrofy pro ohraničení textových atributů.

Řádek s prvním výskytem chyby je:

./src/pages/obchodni-podminky.tsx

32:60 Error: "'" can be escaped with "&quot;", "&ldquo;"

Projdu kód a najdu všechny apostrofy, které mohou způsobovat problémy. Následně navrhnu řešení.

Při procházení kódu vidím hodně apostrofů v textu, například v:

1. "...povinnosti mlčenlivosti nejméně v rozsahu stanoveném těmito obchodními podmínkami."
2. "V opačném případě poskytované služby nemohou přinést očekávaný výsledek."
3. "...z celkové předplacené částky odečteny absolvované lekce + 40% srážka ze zbývající částky ve prospěch Poskytovatele."

V JSX můžeme apostrofy escapovat buď:

1. Použitím HTML entity: &apos;
2. Použitím JavaScript escape sekvence: \'
3. Nebo použitím backticks (`) místo uvozovek pro ohraničení textů s apostrofy

Moje doporučení bude nahradit všechny apostrofy v textu HTML entitou &apos;, což je nejčistší a nejlépe čitelné řešení pro JSX. Případně můžeme zabalit celé bloky textu do template literals (backticks) a pak nechat apostrofy jak jsou.

Vidím soubor s obchodními podmínkami. Nyní je třeba upravit všechny apostrofy ('), které způsobují chyby při kompilaci.

Opravený src/pages/obchodni-podminky.tsx

Code

// src/pages/obchodni-podminky.tsx import React from 'react'; import styles from './obchodni-podminky.module.css'; import Head from 'next/head'; import Link from 'next/link'; const TermsPage: React.FC = () => { return ( <> <Head>

Problém je v apostrofech (jednoduché uvozovky) v textu, které způsobují chyby při kompilaci v JSX. Vytvořil jsem opravenou verzi souboru, kde jsem:

1. Nahradil všechny apostrofy/jednoduché uvozovky (') HTML entitou &quot; (místo původních ')
2. Toto řešení používá dvojité uvozovky jako HTML entity, což je v JSX doporučený způsob pro escapování apostrofů

Jak použít opravenou verzi:

1. Otevřete soubor src/pages/obchodni-podminky.tsx ve vašem editoru
2. Přesuňte původní verzi do zálohy nebo ji smažte
3. Vložte celý kód z mé upravené verze do souboru
4. Uložte soubor

Po této změně by měla chyba zmizet a Vercel by měl být schopen úspěšně sestavit váš projekt. Pokud ještě narazíte na nějaké problémy, dejte mi vědět.

Retry

D

jinak mám menší dotaz, šlo by něco v tomto projektu ještě zmodernizovat? Reducer nebo tak? Teď chci jen odpovědi. nechci praktické změny

Edit

Analyzoval modernizační strategie pro React projekt bez praktických změn.

13s

Dobrá otázka. Podívejme se na možnosti, jak by šel projekt technicky zmodernizovat bez nutnosti praktických změn. Budu vycházet z kódu, který jsem viděl (kontaktní formulář a obchodní podmínky stránky).

1. **State Management**
   * Současně projekt používá React useState hook pro správu formulářových dat
   * Modernizace: Implementace React Context API nebo Redux Toolkit pro centralizovanou správu stavu, zejména pokud web obsahuje více formulářů nebo sdílený stav napříč komponentami
2. **React Hooks**
   * Modernizace: Rozšíření používání jiných React hooks jako useReducer (pro komplexnější stavovou logiku), useCallback (pro optimalizaci výkonu), useMemo (pro memoizaci výpočetně náročných operací), custom hooks (pro extrakci opakující se logiky)
3. **Formulářové řešení**
   * Současně: základní React formulář s manuální správou stavu a validací
   * Modernizace: Implementace React Hook Form nebo Formik pro lepší správu formulářů, validaci a handling chyb
4. **TypeScript**
   * Projekt již používá TypeScript, ale mohla by být zvýšena jeho úroveň, například:
   * Implementace Zod nebo Yup pro validaci schémat
   * Využívání více pokročilých typových funkcí (discriminated unions, utility types, apod.)
5. **API Volání**
   * Současně: Základní fetch API
   * Modernizace: Použití React Query, SWR nebo Apollo Client pro lépe kontrolované API požadavky s cacheováním, invalidací cache, opakováním požadavků, atd.
6. **Architektura**
   * Modernizace: Implementace Next.js App Router (pokud ještě není použit) místo Pages Router
   * Refaktorování na komponentní architekturu s lepším rozdělením zodpovědností (Atomic Design nebo podobný přístup)
7. **CSS Řešení**
   * Současně: CSS moduly
   * Modernizace: Implementace Tailwind CSS, Styled Components nebo Emotion pro modernější přístup ke stylování
8. **Testování**
   * Přidání unit testů (Jest, React Testing Library)
   * Přidání E2E testů (Cypress, Playwright)
9. **Performance Optimalizace**
   * Lazy loading komponent
   * Image optimalizace (next/image)
   * Preloading klíčových assets
10. **Progressive Web App (PWA)**
    * Přidání service workeru pro offline funkcionalitu
    * Web App Manifest pro instalaci jako aplikace
11. **Optimalizace pro SEO**
    * Strukturovaná data (JSON-LD)
    * Lepší meta tagy a více optimalizované Next.js getServerSideProps/getStaticProps
12. **Monitorování a Analytics**
    * Integrace nástrojů jako Sentry pro monitorování chyb
    * Detailnější webové analytiky

Navrhnu čtyři nejvýznamnější vylepšení, které by nejvíce zvýšily kvalitu kódu a developer experience.

**Možnosti modernizace projektu**

Na základě kódu, který jsem viděl, existuje několik způsobů, jak by se projekt dal zmodernizovat:

**1. State Management vylepšení**

* **useReducer místo useState** - zejména pro formulář, který má více polí a stavů (loading, error), by byl přehlednější useReducer
* **React Context API** - pokud máte state, který potřebujete sdílet mezi více komponentami (např. informace o uživateli, dark/light mode)

**2. Modernější formulářové řešení**

* **React Hook Form** - výrazně snížíte množství kódu a zlepšíte výkon formulářů
* **Zod nebo Yup** - typově bezpečná validace formulářů
* **Formik** - alternativa k React Hook Form

**3. API handling**

* **React Query / SWR** - pro správu cache, stavu načítání, opakování požadavků a invalidaci dat
* **Axios** místo nativního fetch - konzistentnější API, lepší práce s interceptory

**4. Architektura a struktura**

* **Next.js App Router** místo Pages Router (pokud jej ještě nepoužíváte)
* **Atomic Design** - rozdělení komponent do atomů, molekul, organismů pro lepší znovupoužitelnost
* **Custom hooks** - extrakce opakující se logiky do vlastních hooks

**5. Stylování**

* **Tailwind CSS** - utility-first CSS framework
* **Styled-components / Emotion** - CSS-in-JS řešení

**6. Testování**

* **Jest a React Testing Library** - unit a integrační testy
* **Cypress nebo Playwright** - E2E testy

**7. Developer Experience**

* **Husky** - pre-commit a pre-push hooks
* **ESLint a Prettier** v přísnější konfiguraci
* **Storybook** - pro vývoj a dokumentaci UI komponent

Tyto modernizace by zlepšily kvalitu kódu, jeho udržitelnost, a zrychlily by vývoj nových funkcí v budoucnu.

Retry

D

nesmím používat Tailwind, musím mít své vlastní CSS- jen říkam abys věděl

Edit

Adapted styling strategy without using Tailwind CSS.

1s

Ahá, děkuji za tuto informaci. Je důležité vědět omezení a požadavky projektu. Když nemůžete používat Tailwind CSS a musíte mít své vlastní CSS, upravím svá doporučení pro modernizaci projektu v oblasti stylování.

Díky za upřesnění. V tom případě pro oblast stylování doporučuji tyto alternativy:

**Modernizace CSS při zachování vlastních stylů:**

1. **CSS Modules** - pokračovat s tímto přístupem, ale systematičtěji (zdá se, že je už používáte)
2. **SASS/SCSS** - rozšířit současné CSS o preprocesor:
   * Vnořené selektory pro přehlednější hierarchii
   * Mixiny pro znovupoužitelné kusy kódu
   * Proměnné pro barvy, rozměry, apod.
3. **CSS Custom Properties (CSS Variables)** - modernější nativní alternativa:

css

:root {

--primary-color: #0088cc;

--secondary-color: #00a0cc;

}

1. **CSS-in-JS řešení**:
   * styled-components
   * emotion
   * Tyto knihovny vám umožní psát CSS přímo v JavaScript komponentách
2. **Utility třídy** - vytvořit vlastní sadu utility tříd podobnou Tailwindu, ale jen pro často používané vlastnosti:

css

.mb-1 { margin-bottom: 0.25rem; }

.mb-2 { margin-bottom: 0.5rem; }

.flex { display: flex; }

.items-center { align-items: center; }

1. **BEM metodika** (Block Element Modifier) - systematičtější způsob pojmenování CSS tříd:

css

.button { }

.button--primary { }

.button\_\_icon { }

1. **CSS Grid a Flexbox** - plně využít moderní layoutové systémy pro responsivní design
2. **Design system** - definovat jednotné komponenty, mezery, typografii pro konzistentní vzhled
3. **CSS Animations API** - modernější použití animací místo starších přístupů

Všechny ostatní dříve zmíněné modernizace (useReducer, React Hook Form, React Query, atd.) jsou stále platné a nezávisí na přístupu ke stylování.