@font-face {

font-family: "icomoon";

src: url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.eot?u8ckod");

src: url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.eot?u8ckod#iefix")

format("embedded-opentype"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.ttf?u8ckod")

format("truetype"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.woff?u8ckod")

format("woff"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.svg?u8ckod#icomoon")

format("svg");

font-weight: normal;

font-style: normal;

font-display: block;

}

[class^="icon-"],

[class\*=" icon-"] {

/\* use !important to prevent issues with browser extensions that change fonts \*/

font-family: "icomoon" !important;

speak: never;

font-style: normal;

font-weight: normal;

font-variant: normal;

text-transform: none;

line-height: 1;

/\* Better Font Rendering =========== \*/

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

.icon-back:before {

content: "\e900";

color: #827d7b;

}

.icon-next:before {

content: "\e901";

color: #827d7b;

}

.icon-pause:before {

content: "\e902";

color: #fff;

}

.icon-play:before {

content: "\e903";

color: #fff;

}

.icon-playlist:before {

content: "\e904";

color: #fff;

}

@font-face {

font-family: Avenir;

src: url(https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/font/AvenirNextRoundedProMedium.TTF);

}

html {

box-sizing: border-box;

--duration: 1s;

--ease-slider: cubic-bezier(0.4, 0, 0.2, 1);

--ease-timeline: cubic-bezier(0.71, 0.21, 0.3, 0.95);

}

html \*,

html \*::before,

html \*::after {

box-sizing: inherit;

scrollbar-width: none;

}

body {

margin: 0;

display: flex;

align-items: center;

justify-content: center;

height: calc(var(--vH) \* 100);

font-family: Avenir, sans-serif;

background-color: var(--body-bg, #fff);

-webkit-tap-highlight-color: transparent;

transition: var(--duration) background-color var(--ease-slider);

}

::-webkit-scrollbar {

width: 0;

height: 0;

}

/\* PUBLIC CLASSES \*/

.img {

width: 100%;

flex-shrink: 0;

display: block;

object-fit: cover;

}

.list {

margin: 0;

padding: 0;

list-style-type: none;

}

.text\_trsf-cap {

text-transform: capitalize;

}

.button {

all: unset;

cursor: pointer;

}

.center {

display: flex;

align-items: center;

justify-content: center;

}

.flex-row {

display: flex;

}

.flex-column {

display: flex;

flex-direction: column;

}

.\_align\_center {

align-items: center;

}

.\_align\_start {

align-items: flex-start;

}

.\_align\_end {

align-items: flex-end;

}

.\_justify\_center {

justify-content: center;

}

.\_justify\_start {

justify-content: flex-start;

}

.\_justify\_end {

justify-content: flex-end;

}

.\_justify\_space-btwn {

justify-content: space-between;

}

.text\_overflow {

width: 66%;

overflow: hidden;

white-space: nowrap;

display: inline-block;

text-overflow: ellipsis;

}

.loading {

gap: 0 0.5rem;

font-size: 5rem;

font-weight: bold;

}

/\* PUBLIC CLASSES \*/

.music-player {

--color-white: #fff;

--color-gray: #e5e7ea;

--color-blue: #78adfe;

--color-blue-dark: #5781bd;

--box-shadow: 0 2px 6px 1px #0000001f;

--color-text-1: #000;

--color-text-2: #0000006b;

--cover-size: 3.8125em;

--border-radius: 1.625em;

--music-player-height: 24.375em;

--offset-cover: 1.60125em;

width: 20.9375em;

overflow: hidden;

user-select: none;

color: var(--color-text-1);

height: var(--music-player-height);

border-radius: var(--border-radius);

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

}

.slider {

--shadow-opacity: 1;

z-index: 0;

flex-shrink: 0;

height: 7.125em;

position: relative;

border-radius: inherit;

transition: var(--duration) height var(--ease-timeline);

}

.slider.resize {

--shadow-opacity: 0;

height: var(--music-player-height);

}

.slider::after {

top: 0;

left: 0;

right: 0;

content: "";

width: 100%;

z-index: -1;

height: 100%;

position: absolute;

pointer-events: none;

border-radius: inherit;

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

opacity: var(--shadow-opacity);

transition: var(--duration) opacity;

}

.slider\_\_content {

top: 0;

left: 0;

overflow: hidden;

position: absolute;

border-radius: inherit;

width: var(--cover-size);

height: var(--cover-size);

transition: transform, width, height;

transition-duration: var(--duration);

transition-timing-function: var(--ease-timeline);

transform: translate3d(var(--offset-cover), var(--offset-cover), 0);

}

.slider.resize .slider\_\_content {

width: 100%;

height: 17.8125em;

transform: translate3d(0, 0, 0);

}

.slider\_\_content .button {

--size: 3em;

z-index: 1;

position: absolute;

width: var(--size);

height: var(--size);

}

.slider\_\_content i {

position: absolute;

pointer-events: none;

font-size: var(--size);

}

.music-player\_\_playlist-button {

top: 5.5%;

left: 5.5%;

transform: scale(0);

transition: calc(var(--duration) / 2) transform;

}

.slider.resize .music-player\_\_playlist-button {

transform: scale(1);

transition: 0.35s var(--duration) transform cubic-bezier(0, 0.85, 0.11, 1.64);

}

.music-player\_\_broadcast-guarantor .icon-pause,

.music-player\_\_broadcast-guarantor.click .icon-play {

opacity: 0;

}

.music-player\_\_broadcast-guarantor.click .icon-pause {

opacity: 1;

}

.slider\_\_imgs {

width: 100%;

height: 100%;

filter: brightness(75%);

transform: translate3d(calc(var(--index) \* 100%), 0, 0);

transition: var(--duration) transform var(--ease-slider);

}

.slider\_\_imgs > img {

pointer-events: none;

}

.slider\_\_controls {

--controls-y: 145%;

--controls-x: 17.3%;

--controls-width: 68.4%;

--controls-resize-width: 88%;

/\* Animation performance is better than transition \*/

gap: 0.375em 0;

flex-wrap: wrap;

position: absolute;

align-items: center;

padding-top: 0.375em;

width: var(--controls-width);

transform: translate3d(var(--controls-x), 0, 0);

animation: var(--controls-animate, "down paused") var(--duration)

var(--ease-timeline) forwards;

}

@keyframes down {

100% {

width: var(--controls-resize-width);

transform: translate3d(0, var(--controls-y), 0);

}

}

@keyframes up {

0% {

width: var(--controls-resize-width);

transform: translate3d(0, var(--controls-y), 0);

}

100% {

width: var(--controls-width);

transform: translate3d(var(--controls-x), 0, 0);

}

}

.slider\_\_switch-button {

font-size: 3em;

height: max-content;

}

.music-player\_\_info {

width: 56.3%;

cursor: pointer;

line-height: 1.8;

overflow: hidden;

font-weight: bold;

padding: 0 0.0625em;

white-space: nowrap;

}

.music-player\_\_info > \* {

margin: 0 auto;

pointer-events: none;

}

.music-player\_\_singer-name {

font-size: 1.25em;

width: max-content;

}

.music-player\_\_subtitle {

font-size: 0.85em;

font-weight: bold;

color: var(--color-text-2);

}

.slider\_\_controls .music-player\_\_subtitle {

width: max-content;

}

.music-player\_\_singer-name.animate,

.music-player\_\_subtitle.animate {

--subtitle-gap: 1.5625em;

display: flex;

gap: 0 var(--subtitle-gap);

animation: subtitle 12s 1.2s linear infinite;

}

@keyframes subtitle {

80%,

100% {

transform: translate3d(calc((100% + var(--subtitle-gap)) / -2), 0, 0);

}

}

.progress {

width: 90%;

height: 1.25em;

cursor: pointer;

transition: var(--duration) width var(--ease-timeline);

}

.slider.resize .progress {

width: 100%;

}

.progress\_\_wrapper {

width: 100%;

height: 0.3125em;

position: relative;

border-radius: 1em;

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

}

.progress\_\_bar {

top: 0;

left: 0;

bottom: 0;

position: absolute;

width: var(--width);

border-radius: inherit;

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

}

.progress\_\_bar::after {

--size: 0.4375em;

left: 98%;

content: "";

position: absolute;

width: var(--size);

height: var(--size);

border-radius: 100%;

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

}

.music-player\_\_playlist {

height: 100%;

overflow: hidden auto;

padding: 1.28125em 1.09375em 0 var(--offset-cover);

}

.music-player\_\_song {

--gap: 0.75em;

cursor: pointer;

margin-bottom: var(--gap);

padding-bottom: var(--gap);

border-bottom: 1.938px solid #d8d8d859;

}

.music-player\_\_song audio {

display: none;

}

.music-player\_\_song-img {

width: var(--cover-size);

height: var(--cover-size);

border-radius: var(--border-radius);

}

.music-player\_\_playlist-info {

width: 100%;

overflow: hidden;

line-height: 1.3;

font-size: 1.06875em;

margin-left: 0.7875em;

}

.music-player\_\_song-duration {

font-weight: bold;

font-size: 0.7875em;

color: var(--color-text-2);

}

@media screen and (min-width: 1366px) {

.music-player {

font-size: 1.17132vw;

}

}

@media screen and (max-width: 480px) {

.music-player {

font-size: 0.8rem;

}

}

@media screen and (max-width: 280px) {

.music-player {

font-size: 0.6rem;

}

}/\*\* @jsx dom \*/

let indexSong = 0;

let isLocked = false;

let songsLength = null;

let selectedSong = null;

let loadingProgress = 0;

let songIsPlayed = false;

let progress\_elmnt = null;

let songName\_elmnt = null;

let sliderImgs\_elmnt = null;

let singerName\_elmnt = null;

let progressBar\_elmnt = null;

let playlistSongs\_elmnt = [];

let loadingProgress\_elmnt = null;

let musicPlayerInfo\_elmnt = null;

let progressBarIsUpdating = false;

let broadcastGuarantor\_elmnt = null;

const root = querySelector("#root");

function App({ songs }) {

function handleChangeMusic({ isPrev = false, playListIndex = null }) {

if (isLocked || indexSong === playListIndex) return;

if (playListIndex || playListIndex === 0) {

indexSong = playListIndex;

} else {

indexSong = isPrev ? indexSong -= 1 : indexSong += 1;

}

if (indexSong < 0) {

indexSong = 0;

return;

} else if (indexSong > songsLength) {

indexSong = songsLength;

return;

}

selectedSong.pause();

selectedSong.currentTime = 0;

progressBarIsUpdating = false;

selectedSong = playlistSongs\_elmnt[indexSong];

if (selectedSong.paused && songIsPlayed) selectedSong.play();else

selectedSong.pause();

setBodyBg(songs[indexSong].bg);

setProperty(sliderImgs\_elmnt, "--index", -indexSong);

updateInfo(singerName\_elmnt, songs[indexSong].artist);

updateInfo(songName\_elmnt, songs[indexSong].songName);

}

setBodyBg(songs[0].bg);

return (

dom("div", { class: "music-player flex-column" },

dom(Slider, { slides: songs, handleChangeMusic: handleChangeMusic }),

dom(Playlist, { list: songs, handleChangeMusic: handleChangeMusic })));

}

function Slider({ slides, handleChangeMusic }) {

function handleResizeSlider({ target }) {

if (isLocked) {

return;

} else if (target.classList.contains("music-player\_\_info")) {

this.classList.add("resize");

setProperty(this, "--controls-animate", "down running");

return;

} else if (target.classList.contains("music-player\_\_playlist-button")) {

this.classList.remove("resize");

setProperty(this, "--controls-animate", "up running");

return;

}

}

function handlePlayMusic() {

if (selectedSong.currentTime === selectedSong.duration) {

handleChangeMusic({});

}

this.classList.toggle("click");

songIsPlayed = !songIsPlayed;

selectedSong.paused ? selectedSong.play() : selectedSong.pause();

}

return (

dom("div", { class: "slider center", onClick: handleResizeSlider },

dom("div", { class: "slider\_\_content center" },

dom("button", { class: "music-player\_\_playlist-button center button" },

dom("i", { class: "icon-playlist" })),

dom("button", {

onClick: handlePlayMusic,

class: "music-player\_\_broadcast-guarantor center button" },

dom("i", { class: "icon-play" }),

dom("i", { class: "icon-pause" })),

dom("div", { class: "slider\_\_imgs flex-row" },

slides.map(({ songName, files: { cover } }) =>

dom("img", { src: cover, class: "img", alt: songName })))),

dom("div", { class: "slider\_\_controls center" },

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: true }) },

dom("i", { class: "icon-back" })),

dom("div", { class: "music-player\_\_info text\_trsf-cap" },

dom("div", null,

dom("div", { class: "music-player\_\_singer-name" },

dom("div", null, slides[0].artist))),

dom("div", null,

dom("div", { class: "music-player\_\_subtitle" },

dom("div", null, slides[0].songName)))),

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: false }) },

dom("i", { class: "icon-next" })),

dom("div", {

class: "progress center",

onPointerdown: e => {

handleScrub(e);

progressBarIsUpdating = true;

} },

dom("div", { class: "progress\_\_wrapper" },

dom("div", { class: "progress\_\_bar center" }))))));

}

function Playlist({ list, handleChangeMusic }) {

function loadedAudio() {

const duration = this.duration;

const target = this.parentElement.querySelector(

".music-player\_\_song-duration");

let min = parseInt(duration / 60);

if (min < 10) min = "0" + min;

let sec = parseInt(duration % 60);

if (sec < 10) sec = "0" + sec;

target.appendChild(document.createTextNode(`${min}:${sec}`));

}

function updateTheProgressBar() {

const duration = this.duration;

const currentTime = this.currentTime;

const progressBarWidth = currentTime / duration \* 100;

setProperty(progressBar\_elmnt, "--width", `${progressBarWidth}%`);

if (songIsPlayed && currentTime === duration) {

handleChangeMusic({});

}

if (

indexSong === songsLength &&

this === selectedSong &&

currentTime === duration)

{

songIsPlayed = false;

broadcastGuarantor\_elmnt.classList.remove("click");

}

}

return (

dom("ul", { class: "music-player\_\_playlist list" },

list.map(({ songName, artist, files: { cover, song } }, index) => {

return (

dom("li", {

class: "music-player\_\_song",

onClick: () =>

handleChangeMusic({ isPrev: false, playListIndex: index }) },

dom("div", { class: "flex-row \_align\_center" },

dom("img", { src: cover, class: "img music-player\_\_song-img" }),

dom("div", { class: "music-player\_\_playlist-info text\_trsf-cap" },

dom("b", { class: "text\_overflow" }, songName),

dom("div", { class: "flex-row \_justify\_space-btwn" },

dom("span", { class: "music-player\_\_subtitle" }, artist),

dom("span", { class: "music-player\_\_song-duration" })))),

dom("audio", {

src: song,

onLoadeddata: loadedAudio,

onTimeupdate: updateTheProgressBar })));

})));

}

function Loading() {

return (

dom("div", { class: "loading flex-row" },

dom("span", { class: "loading\_\_progress" }, "0"),

dom("span", null, "%")));

}

function dom(tag, props, ...children) {

if (typeof tag === "function") return tag(props, ...children);

function addChild(parent, child) {

if (Array.isArray(child)) {

child.forEach(nestedChild => addChild(parent, nestedChild));

} else {

parent.appendChild(

child.nodeType ? child : document.createTextNode(child.toString()));

}

}

const element = document.createElement(tag);

Object.entries(props || {}).forEach(([name, value]) => {

if (name.startsWith("on") && name.toLowerCase() in window) {

element[name.toLowerCase()] = value;

} else if (name === "style") {

Object.entries(value).forEach(([styleProp, styleValue]) => {

element.style[styleProp] = styleValue;

});

} else {

element.setAttribute(name, value.toString());

}

});

children.forEach(child => {

addChild(element, child);

});

return element;

}

fetch(

"https://gist.githubusercontent.com/abxlfazl/37404417d17230a629683eb3f2f0a88a/raw/366ad64df645e94592847283a306fe2276de458e/music-info.json").

then(respone => respone).

then(data => data.json()).

then(result => {

const songs = result.songs;

function downloadTheFiles(media, input) {

return Promise.all(

input.map(song => {

const promise = new Promise(resolve => {

const url = song.files[media];

const req = new XMLHttpRequest();

req.open("GET", url, true);

req.responseType = "blob";

req.send();

req.onreadystatechange = () => {

if (req.readyState === 4) {

if (req.status === 200) {

const blob = req.response;

const file = URL.createObjectURL(blob);

song.files[media] = file;

resolve(song);

}

}

};

});

promise.then(() => {

loadingProgress++;

const progress = Math.round(

loadingProgress / (songs.length \* 2) \* 100);

loadingProgress\_elmnt.innerHTML = progress;

});

return promise;

}));

}

root.appendChild(dom(Loading, null));

loadingProgress\_elmnt = querySelector(".loading\_\_progress");

downloadTheFiles("cover", songs).then(respone => {

downloadTheFiles("song", respone).then(data => {

root.removeChild(querySelector(".loading"));

root.appendChild(dom(App, { songs: data }));

songsLength = data.length - 1;

progress\_elmnt = querySelector(".progress");

playlistSongs\_elmnt = querySelectorAll("audio");

sliderImgs\_elmnt = querySelector(".slider\_\_imgs");

songName\_elmnt = querySelector(".music-player\_\_subtitle");

musicPlayerInfo\_elmnt = querySelector(".music-player\_\_info");

singerName\_elmnt = querySelector(".music-player\_\_singer-name");

selectedSong = playlistSongs\_elmnt[indexSong];

progressBar\_elmnt = querySelector(".progress\_\_bar");

broadcastGuarantor\_elmnt = querySelector(

".music-player\_\_broadcast-guarantor");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

});

});

});

function controlSubtitleAnimation(parent, child) {

if (child.classList.contains("animate")) return;

const element = child.firstChild;

if (child.clientWidth > parent.clientWidth) {

child.appendChild(element.cloneNode(true));

child.classList.add("animate");

}

setProperty(child.parentElement, "width", `${element.clientWidth}px`);

}

function handleResize() {

const vH = window.innerHeight \* 0.01;

setProperty(document.documentElement, "--vH", `${vH}px`);

}

function querySelector(target) {

return document.querySelector(target);

}

function querySelectorAll(target) {

return document.querySelectorAll(target);

}

function setProperty(target, prop, value = "") {

target.style.setProperty(prop, value);

}

function setBodyBg(color) {

setProperty(document.body, "--body-bg", color);

}

function updateInfo(target, value) {

while (target.firstChild) {

target.removeChild(target.firstChild);

}

const targetChild\_elmnt = document.createElement("div");

targetChild\_elmnt.appendChild(document.createTextNode(value));

target.appendChild(targetChild\_elmnt);

target.classList.remove("animate");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, target);

}

function handleScrub(e) {

const progressOffsetLeft = progress\_elmnt.getBoundingClientRect().left;

const progressWidth = progress\_elmnt.offsetWidth;

const duration = selectedSong.duration;

const currentTime = (e.clientX - progressOffsetLeft) / progressWidth;

selectedSong.currentTime = currentTime \* duration;

}

handleResize();

window.addEventListener("resize", handleResize);

window.addEventListener("orientationchange", handleResize);

window.addEventListener("transitionstart", ({ target }) => {

if (target === sliderImgs\_elmnt) {

isLocked = true;

setProperty(sliderImgs\_elmnt, "will-change", "transform");

}

});

window.addEventListener("transitionend", ({ target, propertyName }) => {

if (target === sliderImgs\_elmnt) {

isLocked = false;

setProperty(sliderImgs\_elmnt, "will-change", "auto");

}

if (target.classList.contains("slider") && propertyName === "height") {

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

}

});

window.addEventListener("pointerup", () => {

if (progressBarIsUpdating) {

selectedSong.muted = false;

progressBarIsUpdating = false;

}

});

window.addEventListener("pointermove", e => {

if (progressBarIsUpdating) {

handleScrub(e, this);

selectedSong.muted = true;

}

});/\*\* @jsx dom \*/

let indexSong = 0;

let isLocked = false;

let songsLength = null;

let selectedSong = null;

let loadingProgress = 0;

let songIsPlayed = false;

let progress\_elmnt = null;

let songName\_elmnt = null;

let sliderImgs\_elmnt = null;

let singerName\_elmnt = null;

let progressBar\_elmnt = null;

let playlistSongs\_elmnt = [];

let loadingProgress\_elmnt = null;

let musicPlayerInfo\_elmnt = null;

let progressBarIsUpdating = false;

let broadcastGuarantor\_elmnt = null;

const root = querySelector("#root");

function App({ songs }) {

function handleChangeMusic({ isPrev = false, playListIndex = null }) {

if (isLocked || indexSong === playListIndex) return;

if (playListIndex || playListIndex === 0) {

indexSong = playListIndex;

} else {

indexSong = isPrev ? indexSong -= 1 : indexSong += 1;

}

if (indexSong < 0) {

indexSong = 0;

return;

} else if (indexSong > songsLength) {

indexSong = songsLength;

return;

}

selectedSong.pause();

selectedSong.currentTime = 0;

progressBarIsUpdating = false;

selectedSong = playlistSongs\_elmnt[indexSong];

if (selectedSong.paused && songIsPlayed) selectedSong.play();else

selectedSong.pause();

setBodyBg(songs[indexSong].bg);

setProperty(sliderImgs\_elmnt, "--index", -indexSong);

updateInfo(singerName\_elmnt, songs[indexSong].artist);

updateInfo(songName\_elmnt, songs[indexSong].songName);

}

setBodyBg(songs[0].bg);

return (

dom("div", { class: "music-player flex-column" },

dom(Slider, { slides: songs, handleChangeMusic: handleChangeMusic }),

dom(Playlist, { list: songs, handleChangeMusic: handleChangeMusic })));

}

function Slider({ slides, handleChangeMusic }) {

function handleResizeSlider({ target }) {

if (isLocked) {

return;

} else if (target.classList.contains("music-player\_\_info")) {

this.classList.add("resize");

setProperty(this, "--controls-animate", "down running");

return;

} else if (target.classList.contains("music-player\_\_playlist-button")) {

this.classList.remove("resize");

setProperty(this, "--controls-animate", "up running");

return;

}

}

function handlePlayMusic() {

if (selectedSong.currentTime === selectedSong.duration) {

handleChangeMusic({});

}

this.classList.toggle("click");

songIsPlayed = !songIsPlayed;

selectedSong.paused ? selectedSong.play() : selectedSong.pause();

}

return (

dom("div", { class: "slider center", onClick: handleResizeSlider },

dom("div", { class: "slider\_\_content center" },

dom("button", { class: "music-player\_\_playlist-button center button" },

dom("i", { class: "icon-playlist" })),

dom("button", {

onClick: handlePlayMusic,

class: "music-player\_\_broadcast-guarantor center button" },

dom("i", { class: "icon-play" }),

dom("i", { class: "icon-pause" })),

dom("div", { class: "slider\_\_imgs flex-row" },

slides.map(({ songName, files: { cover } }) =>

dom("img", { src: cover, class: "img", alt: songName })))),

dom("div", { class: "slider\_\_controls center" },

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: true }) },

dom("i", { class: "icon-back" })),

dom("div", { class: "music-player\_\_info text\_trsf-cap" },

dom("div", null,

dom("div", { class: "music-player\_\_singer-name" },

dom("div", null, slides[0].artist))),

dom("div", null,

dom("div", { class: "music-player\_\_subtitle" },

dom("div", null, slides[0].songName)))),

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: false }) },

dom("i", { class: "icon-next" })),

dom("div", {

class: "progress center",

onPointerdown: e => {

handleScrub(e);

progressBarIsUpdating = true;

} },

dom("div", { class: "progress\_\_wrapper" },

dom("div", { class: "progress\_\_bar center" }))))));

}

function Playlist({ list, handleChangeMusic }) {

function loadedAudio() {

const duration = this.duration;

const target = this.parentElement.querySelector(

".music-player\_\_song-duration");

let min = parseInt(duration / 60);

if (min < 10) min = "0" + min;

let sec = parseInt(duration % 60);

if (sec < 10) sec = "0" + sec;

target.appendChild(document.createTextNode(`${min}:${sec}`));

}

function updateTheProgressBar() {

const duration = this.duration;

const currentTime = this.currentTime;

const progressBarWidth = currentTime / duration \* 100;

setProperty(progressBar\_elmnt, "--width", `${progressBarWidth}%`);

if (songIsPlayed && currentTime === duration) {

handleChangeMusic({});

}

if (

indexSong === songsLength &&

this === selectedSong &&

currentTime === duration)

{

songIsPlayed = false;

broadcastGuarantor\_elmnt.classList.remove("click");

}

}

return (

dom("ul", { class: "music-player\_\_playlist list" },

list.map(({ songName, artist, files: { cover, song } }, index) => {

return (

dom("li", {

class: "music-player\_\_song",

onClick: () =>

handleChangeMusic({ isPrev: false, playListIndex: index }) },

dom("div", { class: "flex-row \_align\_center" },

dom("img", { src: cover, class: "img music-player\_\_song-img" }),

dom("div", { class: "music-player\_\_playlist-info text\_trsf-cap" },

dom("b", { class: "text\_overflow" }, songName),

dom("div", { class: "flex-row \_justify\_space-btwn" },

dom("span", { class: "music-player\_\_subtitle" }, artist),

dom("span", { class: "music-player\_\_song-duration" })))),

dom("audio", {

src: song,

onLoadeddata: loadedAudio,

onTimeupdate: updateTheProgressBar })));

})));

}

function Loading() {

return (

dom("div", { class: "loading flex-row" },

dom("span", { class: "loading\_\_progress" }, "0"),

dom("span", null, "%")));

}

function dom(tag, props, ...children) {

if (typeof tag === "function") return tag(props, ...children);

function addChild(parent, child) {

if (Array.isArray(child)) {

child.forEach(nestedChild => addChild(parent, nestedChild));

} else {

parent.appendChild(

child.nodeType ? child : document.createTextNode(child.toString()));

}

}

const element = document.createElement(tag);

Object.entries(props || {}).forEach(([name, value]) => {

if (name.startsWith("on") && name.toLowerCase() in window) {

element[name.toLowerCase()] = value;

} else if (name === "style") {

Object.entries(value).forEach(([styleProp, styleValue]) => {

element.style[styleProp] = styleValue;

});

} else {

element.setAttribute(name, value.toString());

}

});

children.forEach(child => {

addChild(element, child);

});

return element;

}

fetch(

"https://gist.githubusercontent.com/abxlfazl/37404417d17230a629683eb3f2f0a88a/raw/366ad64df645e94592847283a306fe2276de458e/music-info.json").

then(respone => respone).

then(data => data.json()).

then(result => {

const songs = result.songs;

function downloadTheFiles(media, input) {

return Promise.all(

input.map(song => {

const promise = new Promise(resolve => {

const url = song.files[media];

const req = new XMLHttpRequest();

req.open("GET", url, true);

req.responseType = "blob";

req.send();

req.onreadystatechange = () => {

if (req.readyState === 4) {

if (req.status === 200) {

const blob = req.response;

const file = URL.createObjectURL(blob);

song.files[media] = file;

resolve(song);

}

}

};

});

promise.then(() => {

loadingProgress++;

const progress = Math.round(

loadingProgress / (songs.length \* 2) \* 100);

loadingProgress\_elmnt.innerHTML = progress;

});

return promise;

}));

}

root.appendChild(dom(Loading, null));

loadingProgress\_elmnt = querySelector(".loading\_\_progress");

downloadTheFiles("cover", songs).then(respone => {

downloadTheFiles("song", respone).then(data => {

root.removeChild(querySelector(".loading"));

root.appendChild(dom(App, { songs: data }));

songsLength = data.length - 1;

progress\_elmnt = querySelector(".progress");

playlistSongs\_elmnt = querySelectorAll("audio");

sliderImgs\_elmnt = querySelector(".slider\_\_imgs");

songName\_elmnt = querySelector(".music-player\_\_subtitle");

musicPlayerInfo\_elmnt = querySelector(".music-player\_\_info");

singerName\_elmnt = querySelector(".music-player\_\_singer-name");

selectedSong = playlistSongs\_elmnt[indexSong];

progressBar\_elmnt = querySelector(".progress\_\_bar");

broadcastGuarantor\_elmnt = querySelector(

".music-player\_\_broadcast-guarantor");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

});

});

});

function controlSubtitleAnimation(parent, child) {

if (child.classList.contains("animate")) return;

const element = child.firstChild;

if (child.clientWidth > parent.clientWidth) {

child.appendChild(element.cloneNode(true));

child.classList.add("animate");

}

setProperty(child.parentElement, "width", `${element.clientWidth}px`);

}

function handleResize() {

const vH = window.innerHeight \* 0.01;

setProperty(document.documentElement, "--vH", `${vH}px`);

}

function querySelector(target) {

return document.querySelector(target);

}

function querySelectorAll(target) {

return document.querySelectorAll(target);

}

function setProperty(target, prop, value = "") {

target.style.setProperty(prop, value);

}

function setBodyBg(color) {

setProperty(document.body, "--body-bg", color);

}

function updateInfo(target, value) {

while (target.firstChild) {

target.removeChild(target.firstChild);

}

const targetChild\_elmnt = document.createElement("div");

targetChild\_elmnt.appendChild(document.createTextNode(value));

target.appendChild(targetChild\_elmnt);

target.classList.remove("animate");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, target);

}

function handleScrub(e) {

const progressOffsetLeft = progress\_elmnt.getBoundingClientRect().left;

const progressWidth = progress\_elmnt.offsetWidth;

const duration = selectedSong.duration;

const currentTime = (e.clientX - progressOffsetLeft) / progressWidth;

selectedSong.currentTime = currentTime \* duration;

}

handleResize();

window.addEventListener("resize", handleResize);

window.addEventListener("orientationchange", handleResize);

window.addEventListener("transitionstart", ({ target }) => {

if (target === sliderImgs\_elmnt) {

isLocked = true;

setProperty(sliderImgs\_elmnt, "will-change", "transform");

}

});

window.addEventListener("transitionend", ({ target, propertyName }) => {

if (target === sliderImgs\_elmnt) {

isLocked = false;

setProperty(sliderImgs\_elmnt, "will-change", "auto");

}

if (target.classList.contains("slider") && propertyName === "height") {

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

}

});

window.addEventListener("pointerup", () => {

if (progressBarIsUpdating) {

selectedSong.muted = false;

progressBarIsUpdating = false;

}

});

window.addEventListener("pointermove", e => {

if (progressBarIsUpdating) {

handleScrub(e, this);

selectedSong.muted = true;

}

});@font-face {

font-family: "icomoon";

src: url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.eot?u8ckod");

src: url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.eot?u8ckod#iefix")

format("embedded-opentype"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.ttf?u8ckod")

format("truetype"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.woff?u8ckod")

format("woff"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.svg?u8ckod#icomoon")

format("svg");

font-weight: normal;

font-style: normal;

font-display: block;

}

[class^="icon-"],

[class\*=" icon-"] {

/\* use !important to prevent issues with browser extensions that change fonts \*/

font-family: "icomoon" !important;

speak: never;

font-style: normal;

font-weight: normal;

font-variant: normal;

text-transform: none;

line-height: 1;

/\* Better Font Rendering =========== \*/

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

.icon-back:before {

content: "\e900";

color: #827d7b;

}

.icon-next:before {

content: "\e901";

color: #827d7b;

}

.icon-pause:before {

content: "\e902";

color: #fff;

}

.icon-play:before {

content: "\e903";

color: #fff;

}

.icon-playlist:before {

content: "\e904";

color: #fff;

}

@font-face {

font-family: Avenir;

src: url(https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/font/AvenirNextRoundedProMedium.TTF);

}

html {

box-sizing: border-box;

--duration: 1s;

--ease-slider: cubic-bezier(0.4, 0, 0.2, 1);

--ease-timeline: cubic-bezier(0.71, 0.21, 0.3, 0.95);

}

html \*,

html \*::before,

html \*::after {

box-sizing: inherit;

scrollbar-width: none;

}

body {

margin: 0;

display: flex;

align-items: center;

justify-content: center;

height: calc(var(--vH) \* 100);

font-family: Avenir, sans-serif;

background-color: var(--body-bg, #fff);

-webkit-tap-highlight-color: transparent;

transition: var(--duration) background-color var(--ease-slider);

}

::-webkit-scrollbar {

width: 0;

height: 0;

}

/\* PUBLIC CLASSES \*/

.img {

width: 100%;

flex-shrink: 0;

display: block;

object-fit: cover;

}

.list {

margin: 0;

padding: 0;

list-style-type: none;

}

.text\_trsf-cap {

text-transform: capitalize;

}

.button {

all: unset;

cursor: pointer;

}

.center {

display: flex;

align-items: center;

justify-content: center;

}

.flex-row {

display: flex;

}

.flex-column {

display: flex;

flex-direction: column;

}

.\_align\_center {

align-items: center;

}

.\_align\_start {

align-items: flex-start;

}

.\_align\_end {

align-items: flex-end;

}

.\_justify\_center {

justify-content: center;

}

.\_justify\_start {

justify-content: flex-start;

}

.\_justify\_end {

justify-content: flex-end;

}

.\_justify\_space-btwn {

justify-content: space-between;

}

.text\_overflow {

width: 66%;

overflow: hidden;

white-space: nowrap;

display: inline-block;

text-overflow: ellipsis;

}

.loading {

gap: 0 0.5rem;

font-size: 5rem;

font-weight: bold;

}

/\* PUBLIC CLASSES \*/

.music-player {

--color-white: #fff;

--color-gray: #e5e7ea;

--color-blue: #78adfe;

--color-blue-dark: #5781bd;

--box-shadow: 0 2px 6px 1px #0000001f;

--color-text-1: #000;

--color-text-2: #0000006b;

--cover-size: 3.8125em;

--border-radius: 1.625em;

--music-player-height: 24.375em;

--offset-cover: 1.60125em;

width: 20.9375em;

overflow: hidden;

user-select: none;

color: var(--color-text-1);

height: var(--music-player-height);

border-radius: var(--border-radius);

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

}

.slider {

--shadow-opacity: 1;

z-index: 0;

flex-shrink: 0;

height: 7.125em;

position: relative;

border-radius: inherit;

transition: var(--duration) height var(--ease-timeline);

}

.slider.resize {

--shadow-opacity: 0;

height: var(--music-player-height);

}

.slider::after {

top: 0;

left: 0;

right: 0;

content: "";

width: 100%;

z-index: -1;

height: 100%;

position: absolute;

pointer-events: none;

border-radius: inherit;

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

opacity: var(--shadow-opacity);

transition: var(--duration) opacity;

}

.slider\_\_content {

top: 0;

left: 0;

overflow: hidden;

position: absolute;

border-radius: inherit;

width: var(--cover-size);

height: var(--cover-size);

transition: transform, width, height;

transition-duration: var(--duration);

transition-timing-function: var(--ease-timeline);

transform: translate3d(var(--offset-cover), var(--offset-cover), 0);

}

.slider.resize .slider\_\_content {

width: 100%;

height: 17.8125em;

transform: translate3d(0, 0, 0);

}

.slider\_\_content .button {

--size: 3em;

z-index: 1;

position: absolute;

width: var(--size);

height: var(--size);

}

.slider\_\_content i {

position: absolute;

pointer-events: none;

font-size: var(--size);

}

.music-player\_\_playlist-button {

top: 5.5%;

left: 5.5%;

transform: scale(0);

transition: calc(var(--duration) / 2) transform;

}

.slider.resize .music-player\_\_playlist-button {

transform: scale(1);

transition: 0.35s var(--duration) transform cubic-bezier(0, 0.85, 0.11, 1.64);

}

.music-player\_\_broadcast-guarantor .icon-pause,

.music-player\_\_broadcast-guarantor.click .icon-play {

opacity: 0;

}

.music-player\_\_broadcast-guarantor.click .icon-pause {

opacity: 1;

}

.slider\_\_imgs {

width: 100%;

height: 100%;

filter: brightness(75%);

transform: translate3d(calc(var(--index) \* 100%), 0, 0);

transition: var(--duration) transform var(--ease-slider);

}

.slider\_\_imgs > img {

pointer-events: none;

}

.slider\_\_controls {

--controls-y: 145%;

--controls-x: 17.3%;

--controls-width: 68.4%;

--controls-resize-width: 88%;

/\* Animation performance is better than transition \*/

gap: 0.375em 0;

flex-wrap: wrap;

position: absolute;

align-items: center;

padding-top: 0.375em;

width: var(--controls-width);

transform: translate3d(var(--controls-x), 0, 0);

animation: var(--controls-animate, "down paused") var(--duration)

var(--ease-timeline) forwards;

}

@keyframes down {

100% {

width: var(--controls-resize-width);

transform: translate3d(0, var(--controls-y), 0);

}

}

@keyframes up {

0% {

width: var(--controls-resize-width);

transform: translate3d(0, var(--controls-y), 0);

}

100% {

width: var(--controls-width);

transform: translate3d(var(--controls-x), 0, 0);

}

}

.slider\_\_switch-button {

font-size: 3em;

height: max-content;

}

.music-player\_\_info {

width: 56.3%;

cursor: pointer;

line-height: 1.8;

overflow: hidden;

font-weight: bold;

padding: 0 0.0625em;

white-space: nowrap;

}

.music-player\_\_info > \* {

margin: 0 auto;

pointer-events: none;

}

.music-player\_\_singer-name {

font-size: 1.25em;

width: max-content;

}

.music-player\_\_subtitle {

font-size: 0.85em;

font-weight: bold;

color: var(--color-text-2);

}

.slider\_\_controls .music-player\_\_subtitle {

width: max-content;

}

.music-player\_\_singer-name.animate,

.music-player\_\_subtitle.animate {

--subtitle-gap: 1.5625em;

display: flex;

gap: 0 var(--subtitle-gap);

animation: subtitle 12s 1.2s linear infinite;

}

@keyframes subtitle {

80%,

100% {

transform: translate3d(calc((100% + var(--subtitle-gap)) / -2), 0, 0);

}

}

.progress {

width: 90%;

height: 1.25em;

cursor: pointer;

transition: var(--duration) width var(--ease-timeline);

}

.slider.resize .progress {

width: 100%;

}

.progress\_\_wrapper {

width: 100%;

height: 0.3125em;

position: relative;

border-radius: 1em;

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

}

.progress\_\_bar {

top: 0;

left: 0;

bottom: 0;

position: absolute;

width: var(--width);

border-radius: inherit;

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

}

.progress\_\_bar::after {

--size: 0.4375em;

left: 98%;

content: "";

position: absolute;

width: var(--size);

height: var(--size);

border-radius: 100%;

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

}

.music-player\_\_playlist {

height: 100%;

overflow: hidden auto;

padding: 1.28125em 1.09375em 0 var(--offset-cover);

}

.music-player\_\_song {

--gap: 0.75em;

cursor: pointer;

margin-bottom: var(--gap);

padding-bottom: var(--gap);

border-bottom: 1.938px solid #d8d8d859;

}

.music-player\_\_song audio {

display: none;

}

.music-player\_\_song-img {

width: var(--cover-size);

height: var(--cover-size);

border-radius: var(--border-radius);

}

.music-player\_\_playlist-info {

width: 100%;

overflow: hidden;

line-height: 1.3;

font-size: 1.06875em;

margin-left: 0.7875em;

}

.music-player\_\_song-duration {

font-weight: bold;

font-size: 0.7875em;

color: var(--color-text-2);

}

@media screen and (min-width: 1366px) {

.music-player {

font-size: 1.17132vw;

}

}

@media screen and (max-width: 480px) {

.music-player {

font-size: 0.8rem;

}

}

@media screen and (max-width: 280px) {

.music-player {

font-size: 0.6rem;

}

}/\*\* @jsx dom \*/

let indexSong = 0;

let isLocked = false;

let songsLength = null;

let selectedSong = null;

let loadingProgress = 0;

let songIsPlayed = false;

let progress\_elmnt = null;

let songName\_elmnt = null;

let sliderImgs\_elmnt = null;

let singerName\_elmnt = null;

let progressBar\_elmnt = null;

let playlistSongs\_elmnt = [];

let loadingProgress\_elmnt = null;

let musicPlayerInfo\_elmnt = null;

let progressBarIsUpdating = false;

let broadcastGuarantor\_elmnt = null;

const root = querySelector("#root");

function App({ songs }) {

function handleChangeMusic({ isPrev = false, playListIndex = null }) {

if (isLocked || indexSong === playListIndex) return;

if (playListIndex || playListIndex === 0) {

indexSong = playListIndex;

} else {

indexSong = isPrev ? indexSong -= 1 : indexSong += 1;

}

if (indexSong < 0) {

indexSong = 0;

return;

} else if (indexSong > songsLength) {

indexSong = songsLength;

return;

}

selectedSong.pause();

selectedSong.currentTime = 0;

progressBarIsUpdating = false;

selectedSong = playlistSongs\_elmnt[indexSong];

if (selectedSong.paused && songIsPlayed) selectedSong.play();else

selectedSong.pause();

setBodyBg(songs[indexSong].bg);

setProperty(sliderImgs\_elmnt, "--index", -indexSong);

updateInfo(singerName\_elmnt, songs[indexSong].artist);

updateInfo(songName\_elmnt, songs[indexSong].songName);

}

setBodyBg(songs[0].bg);

return (

dom("div", { class: "music-player flex-column" },

dom(Slider, { slides: songs, handleChangeMusic: handleChangeMusic }),

dom(Playlist, { list: songs, handleChangeMusic: handleChangeMusic })));

}

function Slider({ slides, handleChangeMusic }) {

function handleResizeSlider({ target }) {

if (isLocked) {

return;

} else if (target.classList.contains("music-player\_\_info")) {

this.classList.add("resize");

setProperty(this, "--controls-animate", "down running");

return;

} else if (target.classList.contains("music-player\_\_playlist-button")) {

this.classList.remove("resize");

setProperty(this, "--controls-animate", "up running");

return;

}

}

function handlePlayMusic() {

if (selectedSong.currentTime === selectedSong.duration) {

handleChangeMusic({});

}

this.classList.toggle("click");

songIsPlayed = !songIsPlayed;

selectedSong.paused ? selectedSong.play() : selectedSong.pause();

}

return (

dom("div", { class: "slider center", onClick: handleResizeSlider },

dom("div", { class: "slider\_\_content center" },

dom("button", { class: "music-player\_\_playlist-button center button" },

dom("i", { class: "icon-playlist" })),

dom("button", {

onClick: handlePlayMusic,

class: "music-player\_\_broadcast-guarantor center button" },

dom("i", { class: "icon-play" }),

dom("i", { class: "icon-pause" })),

dom("div", { class: "slider\_\_imgs flex-row" },

slides.map(({ songName, files: { cover } }) =>

dom("img", { src: cover, class: "img", alt: songName })))),

dom("div", { class: "slider\_\_controls center" },

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: true }) },

dom("i", { class: "icon-back" })),

dom("div", { class: "music-player\_\_info text\_trsf-cap" },

dom("div", null,

dom("div", { class: "music-player\_\_singer-name" },

dom("div", null, slides[0].artist))),

dom("div", null,

dom("div", { class: "music-player\_\_subtitle" },

dom("div", null, slides[0].songName)))),

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: false }) },

dom("i", { class: "icon-next" })),

dom("div", {

class: "progress center",

onPointerdown: e => {

handleScrub(e);

progressBarIsUpdating = true;

} },

dom("div", { class: "progress\_\_wrapper" },

dom("div", { class: "progress\_\_bar center" }))))));

}

function Playlist({ list, handleChangeMusic }) {

function loadedAudio() {

const duration = this.duration;

const target = this.parentElement.querySelector(

".music-player\_\_song-duration");

let min = parseInt(duration / 60);

if (min < 10) min = "0" + min;

let sec = parseInt(duration % 60);

if (sec < 10) sec = "0" + sec;

target.appendChild(document.createTextNode(`${min}:${sec}`));

}

function updateTheProgressBar() {

const duration = this.duration;

const currentTime = this.currentTime;

const progressBarWidth = currentTime / duration \* 100;

setProperty(progressBar\_elmnt, "--width", `${progressBarWidth}%`);

if (songIsPlayed && currentTime === duration) {

handleChangeMusic({});

}

if (

indexSong === songsLength &&

this === selectedSong &&

currentTime === duration)

{

songIsPlayed = false;

broadcastGuarantor\_elmnt.classList.remove("click");

}

}

return (

dom("ul", { class: "music-player\_\_playlist list" },

list.map(({ songName, artist, files: { cover, song } }, index) => {

return (

dom("li", {

class: "music-player\_\_song",

onClick: () =>

handleChangeMusic({ isPrev: false, playListIndex: index }) },

dom("div", { class: "flex-row \_align\_center" },

dom("img", { src: cover, class: "img music-player\_\_song-img" }),

dom("div", { class: "music-player\_\_playlist-info text\_trsf-cap" },

dom("b", { class: "text\_overflow" }, songName),

dom("div", { class: "flex-row \_justify\_space-btwn" },

dom("span", { class: "music-player\_\_subtitle" }, artist),

dom("span", { class: "music-player\_\_song-duration" })))),

dom("audio", {

src: song,

onLoadeddata: loadedAudio,

onTimeupdate: updateTheProgressBar })));

})));

}

function Loading() {

return (

dom("div", { class: "loading flex-row" },

dom("span", { class: "loading\_\_progress" }, "0"),

dom("span", null, "%")));

}

function dom(tag, props, ...children) {

if (typeof tag === "function") return tag(props, ...children);

function addChild(parent, child) {

if (Array.isArray(child)) {

child.forEach(nestedChild => addChild(parent, nestedChild));

} else {

parent.appendChild(

child.nodeType ? child : document.createTextNode(child.toString()));

}

}

const element = document.createElement(tag);

Object.entries(props || {}).forEach(([name, value]) => {

if (name.startsWith("on") && name.toLowerCase() in window) {

element[name.toLowerCase()] = value;

} else if (name === "style") {

Object.entries(value).forEach(([styleProp, styleValue]) => {

element.style[styleProp] = styleValue;

});

} else {

element.setAttribute(name, value.toString());

}

});

children.forEach(child => {

addChild(element, child);

});

return element;

}

fetch(

"https://gist.githubusercontent.com/abxlfazl/37404417d17230a629683eb3f2f0a88a/raw/366ad64df645e94592847283a306fe2276de458e/music-info.json").

then(respone => respone).

then(data => data.json()).

then(result => {

const songs = result.songs;

function downloadTheFiles(media, input) {

return Promise.all(

input.map(song => {

const promise = new Promise(resolve => {

const url = song.files[media];

const req = new XMLHttpRequest();

req.open("GET", url, true);

req.responseType = "blob";

req.send();

req.onreadystatechange = () => {

if (req.readyState === 4) {

if (req.status === 200) {

const blob = req.response;

const file = URL.createObjectURL(blob);

song.files[media] = file;

resolve(song);

}

}

};

});

promise.then(() => {

loadingProgress++;

const progress = Math.round(

loadingProgress / (songs.length \* 2) \* 100);

loadingProgress\_elmnt.innerHTML = progress;

});

return promise;

}));

}

root.appendChild(dom(Loading, null));

loadingProgress\_elmnt = querySelector(".loading\_\_progress");

downloadTheFiles("cover", songs).then(respone => {

downloadTheFiles("song", respone).then(data => {

root.removeChild(querySelector(".loading"));

root.appendChild(dom(App, { songs: data }));

songsLength = data.length - 1;

progress\_elmnt = querySelector(".progress");

playlistSongs\_elmnt = querySelectorAll("audio");

sliderImgs\_elmnt = querySelector(".slider\_\_imgs");

songName\_elmnt = querySelector(".music-player\_\_subtitle");

musicPlayerInfo\_elmnt = querySelector(".music-player\_\_info");

singerName\_elmnt = querySelector(".music-player\_\_singer-name");

selectedSong = playlistSongs\_elmnt[indexSong];

progressBar\_elmnt = querySelector(".progress\_\_bar");

broadcastGuarantor\_elmnt = querySelector(

".music-player\_\_broadcast-guarantor");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

});

});

});

function controlSubtitleAnimation(parent, child) {

if (child.classList.contains("animate")) return;

const element = child.firstChild;

if (child.clientWidth > parent.clientWidth) {

child.appendChild(element.cloneNode(true));

child.classList.add("animate");

}

setProperty(child.parentElement, "width", `${element.clientWidth}px`);

}

function handleResize() {

const vH = window.innerHeight \* 0.01;

setProperty(document.documentElement, "--vH", `${vH}px`);

}

function querySelector(target) {

return document.querySelector(target);

}

function querySelectorAll(target) {

return document.querySelectorAll(target);

}

function setProperty(target, prop, value = "") {

target.style.setProperty(prop, value);

}

function setBodyBg(color) {

setProperty(document.body, "--body-bg", color);

}

function updateInfo(target, value) {

while (target.firstChild) {

target.removeChild(target.firstChild);

}

const targetChild\_elmnt = document.createElement("div");

targetChild\_elmnt.appendChild(document.createTextNode(value));

target.appendChild(targetChild\_elmnt);

target.classList.remove("animate");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, target);

}

function handleScrub(e) {

const progressOffsetLeft = progress\_elmnt.getBoundingClientRect().left;

const progressWidth = progress\_elmnt.offsetWidth;

const duration = selectedSong.duration;

const currentTime = (e.clientX - progressOffsetLeft) / progressWidth;

selectedSong.currentTime = currentTime \* duration;

}

handleResize();

window.addEventListener("resize", handleResize);

window.addEventListener("orientationchange", handleResize);

window.addEventListener("transitionstart", ({ target }) => {

if (target === sliderImgs\_elmnt) {

isLocked = true;

setProperty(sliderImgs\_elmnt, "will-change", "transform");

}

});

window.addEventListener("transitionend", ({ target, propertyName }) => {

if (target === sliderImgs\_elmnt) {

isLocked = false;

setProperty(sliderImgs\_elmnt, "will-change", "auto");

}

if (target.classList.contains("slider") && propertyName === "height") {

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

}

});

window.addEventListener("pointerup", () => {

if (progressBarIsUpdating) {

selectedSong.muted = false;

progressBarIsUpdating = false;

}

});

window.addEventListener("pointermove", e => {

if (progressBarIsUpdating) {

handleScrub(e, this);

selectedSong.muted = true;

}

});/\*\* @jsx dom \*/

let indexSong = 0;

let isLocked = false;

let songsLength = null;

let selectedSong = null;

let loadingProgress = 0;

let songIsPlayed = false;

let progress\_elmnt = null;

let songName\_elmnt = null;

let sliderImgs\_elmnt = null;

let singerName\_elmnt = null;

let progressBar\_elmnt = null;

let playlistSongs\_elmnt = [];

let loadingProgress\_elmnt = null;

let musicPlayerInfo\_elmnt = null;

let progressBarIsUpdating = false;

let broadcastGuarantor\_elmnt = null;

const root = querySelector("#root");

function App({ songs }) {

function handleChangeMusic({ isPrev = false, playListIndex = null }) {

if (isLocked || indexSong === playListIndex) return;

if (playListIndex || playListIndex === 0) {

indexSong = playListIndex;

} else {

indexSong = isPrev ? indexSong -= 1 : indexSong += 1;

}

if (indexSong < 0) {

indexSong = 0;

return;

} else if (indexSong > songsLength) {

indexSong = songsLength;

return;

}

selectedSong.pause();

selectedSong.currentTime = 0;

progressBarIsUpdating = false;

selectedSong = playlistSongs\_elmnt[indexSong];

if (selectedSong.paused && songIsPlayed) selectedSong.play();else

selectedSong.pause();

setBodyBg(songs[indexSong].bg);

setProperty(sliderImgs\_elmnt, "--index", -indexSong);

updateInfo(singerName\_elmnt, songs[indexSong].artist);

updateInfo(songName\_elmnt, songs[indexSong].songName);

}

setBodyBg(songs[0].bg);

return (

dom("div", { class: "music-player flex-column" },

dom(Slider, { slides: songs, handleChangeMusic: handleChangeMusic }),

dom(Playlist, { list: songs, handleChangeMusic: handleChangeMusic })));

}

function Slider({ slides, handleChangeMusic }) {

function handleResizeSlider({ target }) {

if (isLocked) {

return;

} else if (target.classList.contains("music-player\_\_info")) {

this.classList.add("resize");

setProperty(this, "--controls-animate", "down running");

return;

} else if (target.classList.contains("music-player\_\_playlist-button")) {

this.classList.remove("resize");

setProperty(this, "--controls-animate", "up running");

return;

}

}

function handlePlayMusic() {

if (selectedSong.currentTime === selectedSong.duration) {

handleChangeMusic({});

}

this.classList.toggle("click");

songIsPlayed = !songIsPlayed;

selectedSong.paused ? selectedSong.play() : selectedSong.pause();

}

return (

dom("div", { class: "slider center", onClick: handleResizeSlider },

dom("div", { class: "slider\_\_content center" },

dom("button", { class: "music-player\_\_playlist-button center button" },

dom("i", { class: "icon-playlist" })),

dom("button", {

onClick: handlePlayMusic,

class: "music-player\_\_broadcast-guarantor center button" },

dom("i", { class: "icon-play" }),

dom("i", { class: "icon-pause" })),

dom("div", { class: "slider\_\_imgs flex-row" },

slides.map(({ songName, files: { cover } }) =>

dom("img", { src: cover, class: "img", alt: songName })))),

dom("div", { class: "slider\_\_controls center" },

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: true }) },

dom("i", { class: "icon-back" })),

dom("div", { class: "music-player\_\_info text\_trsf-cap" },

dom("div", null,

dom("div", { class: "music-player\_\_singer-name" },

dom("div", null, slides[0].artist))),

dom("div", null,

dom("div", { class: "music-player\_\_subtitle" },

dom("div", null, slides[0].songName)))),

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: false }) },

dom("i", { class: "icon-next" })),

dom("div", {

class: "progress center",

onPointerdown: e => {

handleScrub(e);

progressBarIsUpdating = true;

} },

dom("div", { class: "progress\_\_wrapper" },

dom("div", { class: "progress\_\_bar center" }))))));

}

function Playlist({ list, handleChangeMusic }) {

function loadedAudio() {

const duration = this.duration;

const target = this.parentElement.querySelector(

".music-player\_\_song-duration");

let min = parseInt(duration / 60);

if (min < 10) min = "0" + min;

let sec = parseInt(duration % 60);

if (sec < 10) sec = "0" + sec;

target.appendChild(document.createTextNode(`${min}:${sec}`));

}

function updateTheProgressBar() {

const duration = this.duration;

const currentTime = this.currentTime;

const progressBarWidth = currentTime / duration \* 100;

setProperty(progressBar\_elmnt, "--width", `${progressBarWidth}%`);

if (songIsPlayed && currentTime === duration) {

handleChangeMusic({});

}

if (

indexSong === songsLength &&

this === selectedSong &&

currentTime === duration)

{

songIsPlayed = false;

broadcastGuarantor\_elmnt.classList.remove("click");

}

}

return (

dom("ul", { class: "music-player\_\_playlist list" },

list.map(({ songName, artist, files: { cover, song } }, index) => {

return (

dom("li", {

class: "music-player\_\_song",

onClick: () =>

handleChangeMusic({ isPrev: false, playListIndex: index }) },

dom("div", { class: "flex-row \_align\_center" },

dom("img", { src: cover, class: "img music-player\_\_song-img" }),

dom("div", { class: "music-player\_\_playlist-info text\_trsf-cap" },

dom("b", { class: "text\_overflow" }, songName),

dom("div", { class: "flex-row \_justify\_space-btwn" },

dom("span", { class: "music-player\_\_subtitle" }, artist),

dom("span", { class: "music-player\_\_song-duration" })))),

dom("audio", {

src: song,

onLoadeddata: loadedAudio,

onTimeupdate: updateTheProgressBar })));

})));

}

function Loading() {

return (

dom("div", { class: "loading flex-row" },

dom("span", { class: "loading\_\_progress" }, "0"),

dom("span", null, "%")));

}

function dom(tag, props, ...children) {

if (typeof tag === "function") return tag(props, ...children);

function addChild(parent, child) {

if (Array.isArray(child)) {

child.forEach(nestedChild => addChild(parent, nestedChild));

} else {

parent.appendChild(

child.nodeType ? child : document.createTextNode(child.toString()));

}

}

const element = document.createElement(tag);

Object.entries(props || {}).forEach(([name, value]) => {

if (name.startsWith("on") && name.toLowerCase() in window) {

element[name.toLowerCase()] = value;

} else if (name === "style") {

Object.entries(value).forEach(([styleProp, styleValue]) => {

element.style[styleProp] = styleValue;

});

} else {

element.setAttribute(name, value.toString());

}

});

children.forEach(child => {

addChild(element, child);

});

return element;

}

fetch(

"https://gist.githubusercontent.com/abxlfazl/37404417d17230a629683eb3f2f0a88a/raw/366ad64df645e94592847283a306fe2276de458e/music-info.json").

then(respone => respone).

then(data => data.json()).

then(result => {

const songs = result.songs;

function downloadTheFiles(media, input) {

return Promise.all(

input.map(song => {

const promise = new Promise(resolve => {

const url = song.files[media];

const req = new XMLHttpRequest();

req.open("GET", url, true);

req.responseType = "blob";

req.send();

req.onreadystatechange = () => {

if (req.readyState === 4) {

if (req.status === 200) {

const blob = req.response;

const file = URL.createObjectURL(blob);

song.files[media] = file;

resolve(song);

}

}

};

});

promise.then(() => {

loadingProgress++;

const progress = Math.round(

loadingProgress / (songs.length \* 2) \* 100);

loadingProgress\_elmnt.innerHTML = progress;

});

return promise;

}));

}

root.appendChild(dom(Loading, null));

loadingProgress\_elmnt = querySelector(".loading\_\_progress");

downloadTheFiles("cover", songs).then(respone => {

downloadTheFiles("song", respone).then(data => {

root.removeChild(querySelector(".loading"));

root.appendChild(dom(App, { songs: data }));

songsLength = data.length - 1;

progress\_elmnt = querySelector(".progress");

playlistSongs\_elmnt = querySelectorAll("audio");

sliderImgs\_elmnt = querySelector(".slider\_\_imgs");

songName\_elmnt = querySelector(".music-player\_\_subtitle");

musicPlayerInfo\_elmnt = querySelector(".music-player\_\_info");

singerName\_elmnt = querySelector(".music-player\_\_singer-name");

selectedSong = playlistSongs\_elmnt[indexSong];

progressBar\_elmnt = querySelector(".progress\_\_bar");

broadcastGuarantor\_elmnt = querySelector(

".music-player\_\_broadcast-guarantor");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

});

});

});

function controlSubtitleAnimation(parent, child) {

if (child.classList.contains("animate")) return;

const element = child.firstChild;

if (child.clientWidth > parent.clientWidth) {

child.appendChild(element.cloneNode(true));

child.classList.add("animate");

}

setProperty(child.parentElement, "width", `${element.clientWidth}px`);

}

function handleResize() {

const vH = window.innerHeight \* 0.01;

setProperty(document.documentElement, "--vH", `${vH}px`);

}

function querySelector(target) {

return document.querySelector(target);

}

function querySelectorAll(target) {

return document.querySelectorAll(target);

}

function setProperty(target, prop, value = "") {

target.style.setProperty(prop, value);

}

function setBodyBg(color) {

setProperty(document.body, "--body-bg", color);

}

function updateInfo(target, value) {

while (target.firstChild) {

target.removeChild(target.firstChild);

}

const targetChild\_elmnt = document.createElement("div");

targetChild\_elmnt.appendChild(document.createTextNode(value));

target.appendChild(targetChild\_elmnt);

target.classList.remove("animate");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, target);

}

function handleScrub(e) {

const progressOffsetLeft = progress\_elmnt.getBoundingClientRect().left;

const progressWidth = progress\_elmnt.offsetWidth;

const duration = selectedSong.duration;

const currentTime = (e.clientX - progressOffsetLeft) / progressWidth;

selectedSong.currentTime = currentTime \* duration;

}

handleResize();

window.addEventListener("resize", handleResize);

window.addEventListener("orientationchange", handleResize);

window.addEventListener("transitionstart", ({ target }) => {

if (target === sliderImgs\_elmnt) {

isLocked = true;

setProperty(sliderImgs\_elmnt, "will-change", "transform");

}

});

window.addEventListener("transitionend", ({ target, propertyName }) => {

if (target === sliderImgs\_elmnt) {

isLocked = false;

setProperty(sliderImgs\_elmnt, "will-change", "auto");

}

if (target.classList.contains("slider") && propertyName === "height") {

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

}

});

window.addEventListener("pointerup", () => {

if (progressBarIsUpdating) {

selectedSong.muted = false;

progressBarIsUpdating = false;

}

});

window.addEventListener("pointermove", e => {

if (progressBarIsUpdating) {

handleScrub(e, this);

selectedSong.muted = true;

}

});@font-face {

font-family: "icomoon";

src: url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.eot?u8ckod");

src: url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.eot?u8ckod#iefix")

format("embedded-opentype"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.ttf?u8ckod")

format("truetype"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.woff?u8ckod")

format("woff"),

url("https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/icomoon/fonts/icomoon.svg?u8ckod#icomoon")

format("svg");

font-weight: normal;

font-style: normal;

font-display: block;

}

[class^="icon-"],

[class\*=" icon-"] {

/\* use !important to prevent issues with browser extensions that change fonts \*/

font-family: "icomoon" !important;

speak: never;

font-style: normal;

font-weight: normal;

font-variant: normal;

text-transform: none;

line-height: 1;

/\* Better Font Rendering =========== \*/

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

.icon-back:before {

content: "\e900";

color: #827d7b;

}

.icon-next:before {

content: "\e901";

color: #827d7b;

}

.icon-pause:before {

content: "\e902";

color: #fff;

}

.icon-play:before {

content: "\e903";

color: #fff;

}

.icon-playlist:before {

content: "\e904";

color: #fff;

}

@font-face {

font-family: Avenir;

src: url(https://raw.githubusercontent.com/abxlfazl/music-player-widget/main/src/assets/font/AvenirNextRoundedProMedium.TTF);

}

html {

box-sizing: border-box;

--duration: 1s;

--ease-slider: cubic-bezier(0.4, 0, 0.2, 1);

--ease-timeline: cubic-bezier(0.71, 0.21, 0.3, 0.95);

}

html \*,

html \*::before,

html \*::after {

box-sizing: inherit;

scrollbar-width: none;

}

body {

margin: 0;

display: flex;

align-items: center;

justify-content: center;

height: calc(var(--vH) \* 100);

font-family: Avenir, sans-serif;

background-color: var(--body-bg, #fff);

-webkit-tap-highlight-color: transparent;

transition: var(--duration) background-color var(--ease-slider);

}

::-webkit-scrollbar {

width: 0;

height: 0;

}

/\* PUBLIC CLASSES \*/

.img {

width: 100%;

flex-shrink: 0;

display: block;

object-fit: cover;

}

.list {

margin: 0;

padding: 0;

list-style-type: none;

}

.text\_trsf-cap {

text-transform: capitalize;

}

.button {

all: unset;

cursor: pointer;

}

.center {

display: flex;

align-items: center;

justify-content: center;

}

.flex-row {

display: flex;

}

.flex-column {

display: flex;

flex-direction: column;

}

.\_align\_center {

align-items: center;

}

.\_align\_start {

align-items: flex-start;

}

.\_align\_end {

align-items: flex-end;

}

.\_justify\_center {

justify-content: center;

}

.\_justify\_start {

justify-content: flex-start;

}

.\_justify\_end {

justify-content: flex-end;

}

.\_justify\_space-btwn {

justify-content: space-between;

}

.text\_overflow {

width: 66%;

overflow: hidden;

white-space: nowrap;

display: inline-block;

text-overflow: ellipsis;

}

.loading {

gap: 0 0.5rem;

font-size: 5rem;

font-weight: bold;

}

/\* PUBLIC CLASSES \*/

.music-player {

--color-white: #fff;

--color-gray: #e5e7ea;

--color-blue: #78adfe;

--color-blue-dark: #5781bd;

--box-shadow: 0 2px 6px 1px #0000001f;

--color-text-1: #000;

--color-text-2: #0000006b;

--cover-size: 3.8125em;

--border-radius: 1.625em;

--music-player-height: 24.375em;

--offset-cover: 1.60125em;

width: 20.9375em;

overflow: hidden;

user-select: none;

color: var(--color-text-1);

height: var(--music-player-height);

border-radius: var(--border-radius);

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

}

.slider {

--shadow-opacity: 1;

z-index: 0;

flex-shrink: 0;

height: 7.125em;

position: relative;

border-radius: inherit;

transition: var(--duration) height var(--ease-timeline);

}

.slider.resize {

--shadow-opacity: 0;

height: var(--music-player-height);

}

.slider::after {

top: 0;

left: 0;

right: 0;

content: "";

width: 100%;

z-index: -1;

height: 100%;

position: absolute;

pointer-events: none;

border-radius: inherit;

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

opacity: var(--shadow-opacity);

transition: var(--duration) opacity;

}

.slider\_\_content {

top: 0;

left: 0;

overflow: hidden;

position: absolute;

border-radius: inherit;

width: var(--cover-size);

height: var(--cover-size);

transition: transform, width, height;

transition-duration: var(--duration);

transition-timing-function: var(--ease-timeline);

transform: translate3d(var(--offset-cover), var(--offset-cover), 0);

}

.slider.resize .slider\_\_content {

width: 100%;

height: 17.8125em;

transform: translate3d(0, 0, 0);

}

.slider\_\_content .button {

--size: 3em;

z-index: 1;

position: absolute;

width: var(--size);

height: var(--size);

}

.slider\_\_content i {

position: absolute;

pointer-events: none;

font-size: var(--size);

}

.music-player\_\_playlist-button {

top: 5.5%;

left: 5.5%;

transform: scale(0);

transition: calc(var(--duration) / 2) transform;

}

.slider.resize .music-player\_\_playlist-button {

transform: scale(1);

transition: 0.35s var(--duration) transform cubic-bezier(0, 0.85, 0.11, 1.64);

}

.music-player\_\_broadcast-guarantor .icon-pause,

.music-player\_\_broadcast-guarantor.click .icon-play {

opacity: 0;

}

.music-player\_\_broadcast-guarantor.click .icon-pause {

opacity: 1;

}

.slider\_\_imgs {

width: 100%;

height: 100%;

filter: brightness(75%);

transform: translate3d(calc(var(--index) \* 100%), 0, 0);

transition: var(--duration) transform var(--ease-slider);

}

.slider\_\_imgs > img {

pointer-events: none;

}

.slider\_\_controls {

--controls-y: 145%;

--controls-x: 17.3%;

--controls-width: 68.4%;

--controls-resize-width: 88%;

/\* Animation performance is better than transition \*/

gap: 0.375em 0;

flex-wrap: wrap;

position: absolute;

align-items: center;

padding-top: 0.375em;

width: var(--controls-width);

transform: translate3d(var(--controls-x), 0, 0);

animation: var(--controls-animate, "down paused") var(--duration)

var(--ease-timeline) forwards;

}

@keyframes down {

100% {

width: var(--controls-resize-width);

transform: translate3d(0, var(--controls-y), 0);

}

}

@keyframes up {

0% {

width: var(--controls-resize-width);

transform: translate3d(0, var(--controls-y), 0);

}

100% {

width: var(--controls-width);

transform: translate3d(var(--controls-x), 0, 0);

}

}

.slider\_\_switch-button {

font-size: 3em;

height: max-content;

}

.music-player\_\_info {

width: 56.3%;

cursor: pointer;

line-height: 1.8;

overflow: hidden;

font-weight: bold;

padding: 0 0.0625em;

white-space: nowrap;

}

.music-player\_\_info > \* {

margin: 0 auto;

pointer-events: none;

}

.music-player\_\_singer-name {

font-size: 1.25em;

width: max-content;

}

.music-player\_\_subtitle {

font-size: 0.85em;

font-weight: bold;

color: var(--color-text-2);

}

.slider\_\_controls .music-player\_\_subtitle {

width: max-content;

}

.music-player\_\_singer-name.animate,

.music-player\_\_subtitle.animate {

--subtitle-gap: 1.5625em;

display: flex;

gap: 0 var(--subtitle-gap);

animation: subtitle 12s 1.2s linear infinite;

}

@keyframes subtitle {

80%,

100% {

transform: translate3d(calc((100% + var(--subtitle-gap)) / -2), 0, 0);

}

}

.progress {

width: 90%;

height: 1.25em;

cursor: pointer;

transition: var(--duration) width var(--ease-timeline);

}

.slider.resize .progress {

width: 100%;

}

.progress\_\_wrapper {

width: 100%;

height: 0.3125em;

position: relative;

border-radius: 1em;

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

}

.progress\_\_bar {

top: 0;

left: 0;

bottom: 0;

position: absolute;

width: var(--width);

border-radius: inherit;

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

}

.progress\_\_bar::after {

--size: 0.4375em;

left: 98%;

content: "";

position: absolute;

width: var(--size);

height: var(--size);

border-radius: 100%;

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

}

.music-player\_\_playlist {

height: 100%;

overflow: hidden auto;

padding: 1.28125em 1.09375em 0 var(--offset-cover);

}

.music-player\_\_song {

--gap: 0.75em;

cursor: pointer;

margin-bottom: var(--gap);

padding-bottom: var(--gap);

border-bottom: 1.938px solid #d8d8d859;

}

.music-player\_\_song audio {

display: none;

}

.music-player\_\_song-img {

width: var(--cover-size);

height: var(--cover-size);

border-radius: var(--border-radius);

}

.music-player\_\_playlist-info {

width: 100%;

overflow: hidden;

line-height: 1.3;

font-size: 1.06875em;

margin-left: 0.7875em;

}

.music-player\_\_song-duration {

font-weight: bold;

font-size: 0.7875em;

color: var(--color-text-2);

}

@media screen and (min-width: 1366px) {

.music-player {

font-size: 1.17132vw;

}

}

@media screen and (max-width: 480px) {

.music-player {

font-size: 0.8rem;

}

}

@media screen and (max-width: 280px) {

.music-player {

font-size: 0.6rem;

}

}/\*\* @jsx dom \*/

let indexSong = 0;

let isLocked = false;

let songsLength = null;

let selectedSong = null;

let loadingProgress = 0;

let songIsPlayed = false;

let progress\_elmnt = null;

let songName\_elmnt = null;

let sliderImgs\_elmnt = null;

let singerName\_elmnt = null;

let progressBar\_elmnt = null;

let playlistSongs\_elmnt = [];

let loadingProgress\_elmnt = null;

let musicPlayerInfo\_elmnt = null;

let progressBarIsUpdating = false;

let broadcastGuarantor\_elmnt = null;

const root = querySelector("#root");

function App({ songs }) {

function handleChangeMusic({ isPrev = false, playListIndex = null }) {

if (isLocked || indexSong === playListIndex) return;

if (playListIndex || playListIndex === 0) {

indexSong = playListIndex;

} else {

indexSong = isPrev ? indexSong -= 1 : indexSong += 1;

}

if (indexSong < 0) {

indexSong = 0;

return;

} else if (indexSong > songsLength) {

indexSong = songsLength;

return;

}

selectedSong.pause();

selectedSong.currentTime = 0;

progressBarIsUpdating = false;

selectedSong = playlistSongs\_elmnt[indexSong];

if (selectedSong.paused && songIsPlayed) selectedSong.play();else

selectedSong.pause();

setBodyBg(songs[indexSong].bg);

setProperty(sliderImgs\_elmnt, "--index", -indexSong);

updateInfo(singerName\_elmnt, songs[indexSong].artist);

updateInfo(songName\_elmnt, songs[indexSong].songName);

}

setBodyBg(songs[0].bg);

return (

dom("div", { class: "music-player flex-column" },

dom(Slider, { slides: songs, handleChangeMusic: handleChangeMusic }),

dom(Playlist, { list: songs, handleChangeMusic: handleChangeMusic })));

}

function Slider({ slides, handleChangeMusic }) {

function handleResizeSlider({ target }) {

if (isLocked) {

return;

} else if (target.classList.contains("music-player\_\_info")) {

this.classList.add("resize");

setProperty(this, "--controls-animate", "down running");

return;

} else if (target.classList.contains("music-player\_\_playlist-button")) {

this.classList.remove("resize");

setProperty(this, "--controls-animate", "up running");

return;

}

}

function handlePlayMusic() {

if (selectedSong.currentTime === selectedSong.duration) {

handleChangeMusic({});

}

this.classList.toggle("click");

songIsPlayed = !songIsPlayed;

selectedSong.paused ? selectedSong.play() : selectedSong.pause();

}

return (

dom("div", { class: "slider center", onClick: handleResizeSlider },

dom("div", { class: "slider\_\_content center" },

dom("button", { class: "music-player\_\_playlist-button center button" },

dom("i", { class: "icon-playlist" })),

dom("button", {

onClick: handlePlayMusic,

class: "music-player\_\_broadcast-guarantor center button" },

dom("i", { class: "icon-play" }),

dom("i", { class: "icon-pause" })),

dom("div", { class: "slider\_\_imgs flex-row" },

slides.map(({ songName, files: { cover } }) =>

dom("img", { src: cover, class: "img", alt: songName })))),

dom("div", { class: "slider\_\_controls center" },

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: true }) },

dom("i", { class: "icon-back" })),

dom("div", { class: "music-player\_\_info text\_trsf-cap" },

dom("div", null,

dom("div", { class: "music-player\_\_singer-name" },

dom("div", null, slides[0].artist))),

dom("div", null,

dom("div", { class: "music-player\_\_subtitle" },

dom("div", null, slides[0].songName)))),

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: false }) },

dom("i", { class: "icon-next" })),

dom("div", {

class: "progress center",

onPointerdown: e => {

handleScrub(e);

progressBarIsUpdating = true;

} },

dom("div", { class: "progress\_\_wrapper" },

dom("div", { class: "progress\_\_bar center" }))))));

}

function Playlist({ list, handleChangeMusic }) {

function loadedAudio() {

const duration = this.duration;

const target = this.parentElement.querySelector(

".music-player\_\_song-duration");

let min = parseInt(duration / 60);

if (min < 10) min = "0" + min;

let sec = parseInt(duration % 60);

if (sec < 10) sec = "0" + sec;

target.appendChild(document.createTextNode(`${min}:${sec}`));

}

function updateTheProgressBar() {

const duration = this.duration;

const currentTime = this.currentTime;

const progressBarWidth = currentTime / duration \* 100;

setProperty(progressBar\_elmnt, "--width", `${progressBarWidth}%`);

if (songIsPlayed && currentTime === duration) {

handleChangeMusic({});

}

if (

indexSong === songsLength &&

this === selectedSong &&

currentTime === duration)

{

songIsPlayed = false;

broadcastGuarantor\_elmnt.classList.remove("click");

}

}

return (

dom("ul", { class: "music-player\_\_playlist list" },

list.map(({ songName, artist, files: { cover, song } }, index) => {

return (

dom("li", {

class: "music-player\_\_song",

onClick: () =>

handleChangeMusic({ isPrev: false, playListIndex: index }) },

dom("div", { class: "flex-row \_align\_center" },

dom("img", { src: cover, class: "img music-player\_\_song-img" }),

dom("div", { class: "music-player\_\_playlist-info text\_trsf-cap" },

dom("b", { class: "text\_overflow" }, songName),

dom("div", { class: "flex-row \_justify\_space-btwn" },

dom("span", { class: "music-player\_\_subtitle" }, artist),

dom("span", { class: "music-player\_\_song-duration" })))),

dom("audio", {

src: song,

onLoadeddata: loadedAudio,

onTimeupdate: updateTheProgressBar })));

})));

}

function Loading() {

return (

dom("div", { class: "loading flex-row" },

dom("span", { class: "loading\_\_progress" }, "0"),

dom("span", null, "%")));

}

function dom(tag, props, ...children) {

if (typeof tag === "function") return tag(props, ...children);

function addChild(parent, child) {

if (Array.isArray(child)) {

child.forEach(nestedChild => addChild(parent, nestedChild));

} else {

parent.appendChild(

child.nodeType ? child : document.createTextNode(child.toString()));

}

}

const element = document.createElement(tag);

Object.entries(props || {}).forEach(([name, value]) => {

if (name.startsWith("on") && name.toLowerCase() in window) {

element[name.toLowerCase()] = value;

} else if (name === "style") {

Object.entries(value).forEach(([styleProp, styleValue]) => {

element.style[styleProp] = styleValue;

});

} else {

element.setAttribute(name, value.toString());

}

});

children.forEach(child => {

addChild(element, child);

});

return element;

}

fetch(

"https://gist.githubusercontent.com/abxlfazl/37404417d17230a629683eb3f2f0a88a/raw/366ad64df645e94592847283a306fe2276de458e/music-info.json").

then(respone => respone).

then(data => data.json()).

then(result => {

const songs = result.songs;

function downloadTheFiles(media, input) {

return Promise.all(

input.map(song => {

const promise = new Promise(resolve => {

const url = song.files[media];

const req = new XMLHttpRequest();

req.open("GET", url, true);

req.responseType = "blob";

req.send();

req.onreadystatechange = () => {

if (req.readyState === 4) {

if (req.status === 200) {

const blob = req.response;

const file = URL.createObjectURL(blob);

song.files[media] = file;

resolve(song);

}

}

};

});

promise.then(() => {

loadingProgress++;

const progress = Math.round(

loadingProgress / (songs.length \* 2) \* 100);

loadingProgress\_elmnt.innerHTML = progress;

});

return promise;

}));

}

root.appendChild(dom(Loading, null));

loadingProgress\_elmnt = querySelector(".loading\_\_progress");

downloadTheFiles("cover", songs).then(respone => {

downloadTheFiles("song", respone).then(data => {

root.removeChild(querySelector(".loading"));

root.appendChild(dom(App, { songs: data }));

songsLength = data.length - 1;

progress\_elmnt = querySelector(".progress");

playlistSongs\_elmnt = querySelectorAll("audio");

sliderImgs\_elmnt = querySelector(".slider\_\_imgs");

songName\_elmnt = querySelector(".music-player\_\_subtitle");

musicPlayerInfo\_elmnt = querySelector(".music-player\_\_info");

singerName\_elmnt = querySelector(".music-player\_\_singer-name");

selectedSong = playlistSongs\_elmnt[indexSong];

progressBar\_elmnt = querySelector(".progress\_\_bar");

broadcastGuarantor\_elmnt = querySelector(

".music-player\_\_broadcast-guarantor");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

});

});

});

function controlSubtitleAnimation(parent, child) {

if (child.classList.contains("animate")) return;

const element = child.firstChild;

if (child.clientWidth > parent.clientWidth) {

child.appendChild(element.cloneNode(true));

child.classList.add("animate");

}

setProperty(child.parentElement, "width", `${element.clientWidth}px`);

}

function handleResize() {

const vH = window.innerHeight \* 0.01;

setProperty(document.documentElement, "--vH", `${vH}px`);

}

function querySelector(target) {

return document.querySelector(target);

}

function querySelectorAll(target) {

return document.querySelectorAll(target);

}

function setProperty(target, prop, value = "") {

target.style.setProperty(prop, value);

}

function setBodyBg(color) {

setProperty(document.body, "--body-bg", color);

}

function updateInfo(target, value) {

while (target.firstChild) {

target.removeChild(target.firstChild);

}

const targetChild\_elmnt = document.createElement("div");

targetChild\_elmnt.appendChild(document.createTextNode(value));

target.appendChild(targetChild\_elmnt);

target.classList.remove("animate");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, target);

}

function handleScrub(e) {

const progressOffsetLeft = progress\_elmnt.getBoundingClientRect().left;

const progressWidth = progress\_elmnt.offsetWidth;

const duration = selectedSong.duration;

const currentTime = (e.clientX - progressOffsetLeft) / progressWidth;

selectedSong.currentTime = currentTime \* duration;

}

handleResize();

window.addEventListener("resize", handleResize);

window.addEventListener("orientationchange", handleResize);

window.addEventListener("transitionstart", ({ target }) => {

if (target === sliderImgs\_elmnt) {

isLocked = true;

setProperty(sliderImgs\_elmnt, "will-change", "transform");

}

});

window.addEventListener("transitionend", ({ target, propertyName }) => {

if (target === sliderImgs\_elmnt) {

isLocked = false;

setProperty(sliderImgs\_elmnt, "will-change", "auto");

}

if (target.classList.contains("slider") && propertyName === "height") {

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

}

});

window.addEventListener("pointerup", () => {

if (progressBarIsUpdating) {

selectedSong.muted = false;

progressBarIsUpdating = false;

}

});

window.addEventListener("pointermove", e => {

if (progressBarIsUpdating) {

handleScrub(e, this);

selectedSong.muted = true;

}

});/\*\* @jsx dom \*/

let indexSong = 0;

let isLocked = false;

let songsLength = null;

let selectedSong = null;

let loadingProgress = 0;

let songIsPlayed = false;

let progress\_elmnt = null;

let songName\_elmnt = null;

let sliderImgs\_elmnt = null;

let singerName\_elmnt = null;

let progressBar\_elmnt = null;

let playlistSongs\_elmnt = [];

let loadingProgress\_elmnt = null;

let musicPlayerInfo\_elmnt = null;

let progressBarIsUpdating = false;

let broadcastGuarantor\_elmnt = null;

const root = querySelector("#root");

function App({ songs }) {

function handleChangeMusic({ isPrev = false, playListIndex = null }) {

if (isLocked || indexSong === playListIndex) return;

if (playListIndex || playListIndex === 0) {

indexSong = playListIndex;

} else {

indexSong = isPrev ? indexSong -= 1 : indexSong += 1;

}

if (indexSong < 0) {

indexSong = 0;

return;

} else if (indexSong > songsLength) {

indexSong = songsLength;

return;

}

selectedSong.pause();

selectedSong.currentTime = 0;

progressBarIsUpdating = false;

selectedSong = playlistSongs\_elmnt[indexSong];

if (selectedSong.paused && songIsPlayed) selectedSong.play();else

selectedSong.pause();

setBodyBg(songs[indexSong].bg);

setProperty(sliderImgs\_elmnt, "--index", -indexSong);

updateInfo(singerName\_elmnt, songs[indexSong].artist);

updateInfo(songName\_elmnt, songs[indexSong].songName);

}

setBodyBg(songs[0].bg);

return (

dom("div", { class: "music-player flex-column" },

dom(Slider, { slides: songs, handleChangeMusic: handleChangeMusic }),

dom(Playlist, { list: songs, handleChangeMusic: handleChangeMusic })));

}

function Slider({ slides, handleChangeMusic }) {

function handleResizeSlider({ target }) {

if (isLocked) {

return;

} else if (target.classList.contains("music-player\_\_info")) {

this.classList.add("resize");

setProperty(this, "--controls-animate", "down running");

return;

} else if (target.classList.contains("music-player\_\_playlist-button")) {

this.classList.remove("resize");

setProperty(this, "--controls-animate", "up running");

return;

}

}

function handlePlayMusic() {

if (selectedSong.currentTime === selectedSong.duration) {

handleChangeMusic({});

}

this.classList.toggle("click");

songIsPlayed = !songIsPlayed;

selectedSong.paused ? selectedSong.play() : selectedSong.pause();

}

return (

dom("div", { class: "slider center", onClick: handleResizeSlider },

dom("div", { class: "slider\_\_content center" },

dom("button", { class: "music-player\_\_playlist-button center button" },

dom("i", { class: "icon-playlist" })),

dom("button", {

onClick: handlePlayMusic,

class: "music-player\_\_broadcast-guarantor center button" },

dom("i", { class: "icon-play" }),

dom("i", { class: "icon-pause" })),

dom("div", { class: "slider\_\_imgs flex-row" },

slides.map(({ songName, files: { cover } }) =>

dom("img", { src: cover, class: "img", alt: songName })))),

dom("div", { class: "slider\_\_controls center" },

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: true }) },

dom("i", { class: "icon-back" })),

dom("div", { class: "music-player\_\_info text\_trsf-cap" },

dom("div", null,

dom("div", { class: "music-player\_\_singer-name" },

dom("div", null, slides[0].artist))),

dom("div", null,

dom("div", { class: "music-player\_\_subtitle" },

dom("div", null, slides[0].songName)))),

dom("button", {

class: "slider\_\_switch-button flex-row button",

onClick: () => handleChangeMusic({ isPrev: false }) },

dom("i", { class: "icon-next" })),

dom("div", {

class: "progress center",

onPointerdown: e => {

handleScrub(e);

progressBarIsUpdating = true;

} },

dom("div", { class: "progress\_\_wrapper" },

dom("div", { class: "progress\_\_bar center" }))))));

}

function Playlist({ list, handleChangeMusic }) {

function loadedAudio() {

const duration = this.duration;

const target = this.parentElement.querySelector(

".music-player\_\_song-duration");

let min = parseInt(duration / 60);

if (min < 10) min = "0" + min;

let sec = parseInt(duration % 60);

if (sec < 10) sec = "0" + sec;

target.appendChild(document.createTextNode(`${min}:${sec}`));

}

function updateTheProgressBar() {

const duration = this.duration;

const currentTime = this.currentTime;

const progressBarWidth = currentTime / duration \* 100;

setProperty(progressBar\_elmnt, "--width", `${progressBarWidth}%`);

if (songIsPlayed && currentTime === duration) {

handleChangeMusic({});

}

if (

indexSong === songsLength &&

this === selectedSong &&

currentTime === duration)

{

songIsPlayed = false;

broadcastGuarantor\_elmnt.classList.remove("click");

}

}

return (

dom("ul", { class: "music-player\_\_playlist list" },

list.map(({ songName, artist, files: { cover, song } }, index) => {

return (

dom("li", {

class: "music-player\_\_song",

onClick: () =>

handleChangeMusic({ isPrev: false, playListIndex: index }) },

dom("div", { class: "flex-row \_align\_center" },

dom("img", { src: cover, class: "img music-player\_\_song-img" }),

dom("div", { class: "music-player\_\_playlist-info text\_trsf-cap" },

dom("b", { class: "text\_overflow" }, songName),

dom("div", { class: "flex-row \_justify\_space-btwn" },

dom("span", { class: "music-player\_\_subtitle" }, artist),

dom("span", { class: "music-player\_\_song-duration" })))),

dom("audio", {

src: song,

onLoadeddata: loadedAudio,

onTimeupdate: updateTheProgressBar })));

})));

}

function Loading() {

return (

dom("div", { class: "loading flex-row" },

dom("span", { class: "loading\_\_progress" }, "0"),

dom("span", null, "%")));

}

function dom(tag, props, ...children) {

if (typeof tag === "function") return tag(props, ...children);

function addChild(parent, child) {

if (Array.isArray(child)) {

child.forEach(nestedChild => addChild(parent, nestedChild));

} else {

parent.appendChild(

child.nodeType ? child : document.createTextNode(child.toString()));

}

}

const element = document.createElement(tag);

Object.entries(props || {}).forEach(([name, value]) => {

if (name.startsWith("on") && name.toLowerCase() in window) {

element[name.toLowerCase()] = value;

} else if (name === "style") {

Object.entries(value).forEach(([styleProp, styleValue]) => {

element.style[styleProp] = styleValue;

});

} else {

element.setAttribute(name, value.toString());

}

});

children.forEach(child => {

addChild(element, child);

});

return element;

}

fetch(

"https://gist.githubusercontent.com/abxlfazl/37404417d17230a629683eb3f2f0a88a/raw/366ad64df645e94592847283a306fe2276de458e/music-info.json").

then(respone => respone).

then(data => data.json()).

then(result => {

const songs = result.songs;

function downloadTheFiles(media, input) {

return Promise.all(

input.map(song => {

const promise = new Promise(resolve => {

const url = song.files[media];

const req = new XMLHttpRequest();

req.open("GET", url, true);

req.responseType = "blob";

req.send();

req.onreadystatechange = () => {

if (req.readyState === 4) {

if (req.status === 200) {

const blob = req.response;

const file = URL.createObjectURL(blob);

song.files[media] = file;

resolve(song);

}

}

};

});

promise.then(() => {

loadingProgress++;

const progress = Math.round(

loadingProgress / (songs.length \* 2) \* 100);

loadingProgress\_elmnt.innerHTML = progress;

});

return promise;

}));

}

root.appendChild(dom(Loading, null));

loadingProgress\_elmnt = querySelector(".loading\_\_progress");

downloadTheFiles("cover", songs).then(respone => {

downloadTheFiles("song", respone).then(data => {

root.removeChild(querySelector(".loading"));

root.appendChild(dom(App, { songs: data }));

songsLength = data.length - 1;

progress\_elmnt = querySelector(".progress");

playlistSongs\_elmnt = querySelectorAll("audio");

sliderImgs\_elmnt = querySelector(".slider\_\_imgs");

songName\_elmnt = querySelector(".music-player\_\_subtitle");

musicPlayerInfo\_elmnt = querySelector(".music-player\_\_info");

singerName\_elmnt = querySelector(".music-player\_\_singer-name");

selectedSong = playlistSongs\_elmnt[indexSong];

progressBar\_elmnt = querySelector(".progress\_\_bar");

broadcastGuarantor\_elmnt = querySelector(

".music-player\_\_broadcast-guarantor");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

});

});

});

function controlSubtitleAnimation(parent, child) {

if (child.classList.contains("animate")) return;

const element = child.firstChild;

if (child.clientWidth > parent.clientWidth) {

child.appendChild(element.cloneNode(true));

child.classList.add("animate");

}

setProperty(child.parentElement, "width", `${element.clientWidth}px`);

}

function handleResize() {

const vH = window.innerHeight \* 0.01;

setProperty(document.documentElement, "--vH", `${vH}px`);

}

function querySelector(target) {

return document.querySelector(target);

}

function querySelectorAll(target) {

return document.querySelectorAll(target);

}

function setProperty(target, prop, value = "") {

target.style.setProperty(prop, value);

}

function setBodyBg(color) {

setProperty(document.body, "--body-bg", color);

}

function updateInfo(target, value) {

while (target.firstChild) {

target.removeChild(target.firstChild);

}

const targetChild\_elmnt = document.createElement("div");

targetChild\_elmnt.appendChild(document.createTextNode(value));

target.appendChild(targetChild\_elmnt);

target.classList.remove("animate");

controlSubtitleAnimation(musicPlayerInfo\_elmnt, target);

}

function handleScrub(e) {

const progressOffsetLeft = progress\_elmnt.getBoundingClientRect().left;

const progressWidth = progress\_elmnt.offsetWidth;

const duration = selectedSong.duration;

const currentTime = (e.clientX - progressOffsetLeft) / progressWidth;

selectedSong.currentTime = currentTime \* duration;

}

handleResize();

window.addEventListener("resize", handleResize);

window.addEventListener("orientationchange", handleResize);

window.addEventListener("transitionstart", ({ target }) => {

if (target === sliderImgs\_elmnt) {

isLocked = true;

setProperty(sliderImgs\_elmnt, "will-change", "transform");

}

});

window.addEventListener("transitionend", ({ target, propertyName }) => {

if (target === sliderImgs\_elmnt) {

isLocked = false;

setProperty(sliderImgs\_elmnt, "will-change", "auto");

}

if (target.classList.contains("slider") && propertyName === "height") {

controlSubtitleAnimation(musicPlayerInfo\_elmnt, songName\_elmnt);

controlSubtitleAnimation(musicPlayerInfo\_elmnt, singerName\_elmnt);

}

});

window.addEventListener("pointerup", () => {

if (progressBarIsUpdating) {

selectedSong.muted = false;

progressBarIsUpdating = false;

}

});

window.addEventListener("pointermove", e => {

if (progressBarIsUpdating) {

handleScrub(e, this);

selectedSong.muted = true;

}

});