<style>

.keyboard {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

padding: 5px 0;

background: #004134;

box-shadow: 0 0 50px rgba(0, 0, 0, 0.5);

user-select: none;

transition: bottom 0.4s;

}

.keyboard--hidden {

bottom: -100%;

}

.keyboard\_\_keys {

text-align: center;

}

.keyboard\_\_key {

height: 45px;

width: 6%;

max-width: 90px;

margin: 3px;

border-radius: 4px;

border: none;

background: rgba(255, 255, 255, 0.2);

color: #ffffff;

font-size: 1.05rem;

outline: none;

cursor: pointer;

display: inline-flex;

align-items: center;

justify-content: center;

vertical-align: top;

padding: 0;

-webkit-tap-highlight-color: transparent;

position: relative;

}

.keyboard\_\_key:active {

background: rgba(255, 255, 255, 0.12);

}

.keyboard\_\_key--wide {

width: 12%;

}

.keyboard\_\_key--extra-wide {

width: 36%;

max-width: 500px;

}

.keyboard\_\_key--activatable::after {

content: '';

top: 10px;

right: 10px;

position: absolute;

width: 8px;

height: 8px;

background: rgba(0, 0, 0, 0.4);

border-radius: 50%;

}

.keyboard\_\_key--active::after {

background: #08ff00;

}

.keyboard\_\_key--dark {

background: rgba(0, 0, 0, 0.25);

}

</style>

<script>

const Keyboard = {

elements: {

main: null,

keysContainer: null,

keys: []

},

eventHandlers: {

oninput: null,

onclose: null

},

properties: {

value: "",

capsLock: false

},

init() {

// Create main elements

this.elements.main = document.createElement("div");

this.elements.keysContainer = document.createElement("div");

// Setup main elements

this.elements.main.classList.add("keyboard", "keyboard--hidden");

this.elements.keysContainer.classList.add("keyboard\_\_keys");

this.elements.keysContainer.appendChild(this.\_createKeys());

this.elements.keys = this.elements.keysContainer.querySelectorAll(".keyboard\_\_key");

// Add to DOM

this.elements.main.appendChild(this.elements.keysContainer);

document.body.appendChild(this.elements.main);

// Automatically use keyboard for elements with .use-keyboard-input

document.querySelectorAll(".use-keyboard-input").forEach(element => {

element.addEventListener("focus", () => {

this.open(element.value, currentValue => {

element.value = currentValue;

});

});

});

},

\_createKeys() {

const fragment = document.createDocumentFragment();

const keyLayout = [

"1", "2", "3", "4", "5", "6", "7", "8", "9", "0", "backspace",

"q", "w", "e", "r", "t", "y", "u", "i", "o", "p",

"caps", "a", "s", "d", "f", "g", "h", "j", "k", "l", "enter",

"done", "z", "x", "c", "v", "b", "n", "m", ",", ".", "?",

"space"

];

// Creates HTML for an icon

const createIconHTML = (icon\_name) => {

return <i class="material-icons">${icon\_name}</i>;

};

keyLayout.forEach(key => {

const keyElement = document.createElement("button");

const insertLineBreak = ["backspace", "p", "enter", "?"].indexOf(key) !== -1;

// Add attributes/classes

keyElement.setAttribute("type", "button");

keyElement.classList.add("keyboard\_\_key");

switch (key) {

case "backspace":

keyElement.classList.add("keyboard\_\_key--wide");

keyElement.innerHTML = createIconHTML("backspace");

keyElement.addEventListener("click", () => {

this.properties.value = this.properties.value.substring(0, this.properties.value.length - 1);

this.\_triggerEvent("oninput");

});

break;

case "caps":

keyElement.classList.add("keyboard\_key--wide", "keyboard\_key--activatable");

keyElement.innerHTML = createIconHTML("keyboard\_capslock");

keyElement.addEventListener("click", () => {

this.\_toggleCapsLock();

keyElement.classList.toggle("keyboard\_\_key--active", this.properties.capsLock);

});

break;

case "enter":

keyElement.classList.add("keyboard\_\_key--wide");

keyElement.innerHTML = createIconHTML("keyboard\_return");

keyElement.addEventListener("click", () => {

this.properties.value += "\n";

this.\_triggerEvent("oninput");

});

break;

case "space":

keyElement.classList.add("keyboard\_\_key--extra-wide");

keyElement.innerHTML = createIconHTML("space\_bar");

keyElement.addEventListener("click", () => {

this.properties.value += " ";

this.\_triggerEvent("oninput");

});

break;

case "done":

keyElement.classList.add("keyboard\_key--wide", "keyboard\_key--dark");

keyElement.innerHTML = createIconHTML("check\_circle");

keyElement.addEventListener("click", () => {

this.close();

this.\_triggerEvent("onclose");

});

break;

default:

keyElement.textContent = key.toLowerCase();

keyElement.addEventListener("click", () => {

this.properties.value += this.properties.capsLock ? key.toUpperCase() : key.toLowerCase();

this.\_triggerEvent("oninput");

});

break;

}

fragment.appendChild(keyElement);

if (insertLineBreak) {

fragment.appendChild(document.createElement("br"));

}

});

return fragment;

},

\_triggerEvent(handlerName) {

if (typeof this.eventHandlers[handlerName] == "function") {

this.eventHandlers[handlerName](this.properties.value);

}

},

\_toggleCapsLock() {

this.properties.capsLock = !this.properties.capsLock;

for (const key of this.elements.keys) {

if (key.childElementCount === 0) {

key.textContent = this.properties.capsLock ? key.textContent.toUpperCase() : key.textContent.toLowerCase();

}

}

},

open(initialValue, oninput, onclose) {

this.properties.value = initialValue || "";

this.eventHandlers.oninput = oninput;

this.eventHandlers.onclose = onclose;

this.elements.main.classList.remove("keyboard--hidden");

},

close() {

this.properties.value = "";

this.eventHandlers.oninput = oninput;

this.eventHandlers.onclose = onclose;

this.elements.main.classList.add("keyboard--hidden");

}

};

window.addEventListener("DOMContentLoaded", function () {

Keyboard.init();

});

</script>