```
<template>
  <div class="flex flex-col items-center p-4">
    <h1 class="text-2xl font-bold mb-4">History Puzzle</h1>

    <div class="grid grid-cols-3 gap-1 w-60 h-60">
      <div
        v-for="(tile, index) in tiles"
        :key="index"
        @click="moveTile(index)"
        class="flex items-center justify-center bg-blue-300 border text-xl font-
bold cursor-pointer"
        :class="{ 'bg-white': tile === null, 'cursor-default': !canMove(index) }"
      >
        {{ tile !== null ? tile + 1 : '' }}
      </div>
    </div>

    <button @click="shuffleTiles" class="mt-4 px-4 py-2 bg-green-500 text-
white rounded">
      Shuffle
    </button>

    <div v-if="isSolved" class="mt-4 text-green-600 font-semibold">
      🎉 You solved it! Fun Fact: [Insert historical figure fact here]
    </div>
  </div>
</template>

<script setup>
import { ref, computed, onMounted } from 'vue'

const size = 3
const totalTiles = size * size

const tiles = ref([])
const emptyIndex = ref(totalTiles - 1)

function initTiles() {
  tiles.value = Array.from({ length: totalTiles }, (_, i) => (i < totalTiles - 1 ? i :
null))
  emptyIndex.value = totalTiles - 1
}

function shuffleTiles() {
  const shuffled = [...tiles.value]
  let currentIndex = shuffled.length
  while (currentIndex > 1) {
    const randomIndex = Math.floor(Math.random() * currentIndex)
    currentIndex--
    ;[shuffled[currentIndex], shuffled[randomIndex]] = [shuffled[randomIndex],
shuffled[currentIndex]]
  }

  tiles.value = shuffled
  emptyIndex.value = tiles.value.findIndex((tile) => tile === null)
}

function moveTile(index) {
  if (!canMove(index)) return
  ;[tiles.value[index], tiles.value[emptyIndex.value]] =
[tiles.value[emptyIndex.value], tiles.value[index]]
  emptyIndex.value = index
}

function canMove(index) {
  const row = Math.floor(index / size)
  const col = index % size
  const emptyRow = Math.floor(emptyIndex.value / size)
  const emptyCol = emptyIndex.value % size

  return (Math.abs(row - emptyRow) === 1 && col === emptyCol) ||
(Math.abs(col - emptyCol) === 1 && row === emptyRow)
}

const isSolved = computed(() => {
  return tiles.value.every((tile, i) => tile === (i < totalTiles - 1 ? i : null))
})

onMounted(() => {
  initTiles()
  shuffleTiles()
})
</script>

<style scoped>
.grid > div {
  width: 100%;
  height: 100%;
}
</style>
```