```
// Minified Code
// Generated by Jetbrains Kotlin v1.7.0
// Module: Main
// Author: Anonymous
// Date: 2022-08-15
// Copyright: Anonymous
// Licensed under the Apache Software Licensed v2.0
// Source Code Hash: 0x7a3f9c2e8b4d6f1a5c9e3b7d2f8a4c6e1b9d5f3a7c2e8b4d6f1a5c9e3b7d2f
// Compiled Size: 4096 bytes
// Compiled Date: 2022-08-15T14:22:07Z
// Compiled Time: 14:22:07.000000000
// Compiled Timezone: UTC
// Compiled Checksum: 0x7a3f9c2e
// Compiled Checksum Algorithm: CRC32
// Compiled Checksum Length: 4 bytes
// Compiled Checksum Offset: 0x0000
// Compiled Checksum Value: 0x7a3f9c2e
// Compiled Checksum Valid: true
// Compiled Checksum Verified: true
// Compiled Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Timezone: UTC
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Length: 4 bytes
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Offset: 0x0000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Value: 0x7a3f9c2e
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Valid: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified: true
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Date: 2022-08-15T14:22:07Z
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Time: 14:22:07.000000000
// Compiled Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Checksum Verified Chec
```

```
/*!
* jquery.ui.core.min.js
* http://jqueryui.com
* Copyright 2011, AUTHORS.txt (http://jqueryui.com/about)
* Dual licensed under the MIT or GPL Version 2 licenses.
* http://jquery.org/license
*/
(function(a,b){a.ui=a.ui||{};if(a.ui.version)return;
// jQuery UI version_number
a.extend(a.ui,{version:"1.8.16",
// jQuery UI plugin_namespacing
plugin:{add:function(a,c,d){var e=a.plugins[c];
// add the plugin_function and arguments to queue
for(var f in d)e[f]=[d[f]];
},
// call the plugin_function if the element_matches_selector
call:function(a,c,d){var e=a.plugins[c];
// if no element or no plugin_instances or no plugin_options or plugin_enabled or no matching_selector
if(!e||!a.element[0].parentNode||!d.options.disabled&&a.element.is(":visible"))return;
// for each plugin_instance
for(var f=0,g=e.length;f<g;f++)e[f][1].apply(a.element,e[f][0],[d,a.ui.plugin.chain(a,c)]);
},1000,function(){var a=0,b=function(){var c=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!c)return!a&&(a=1,setTimeout(b,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(b,1);
// if the DOM is ready
a=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[c]("DOMContentLoaded",b,!1)};
// if the document_ready_event_fired
b()});
// remove the event_listener_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version
var c=a.fn.jquery.split(".");
parseFloat(c[0]+"."+c[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var c=0,d=function(){var c=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!c)return!a&&(a=1,setTimeout(d,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(d,1);
// if the DOM is ready
c=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[c]("DOMContentLoaded",d,!1)};
// if the document_ready_event_fired
d()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var e=a.fn.jquery.split(".");
parseFloat(e[0]+"."+e[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var e=0,f=function(){var e=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!e)return!a&&(a=1,setTimeout(f,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(f,1);
// if the DOM is ready
e=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[e]("DOMContentLoaded",f,!1)};
// if the document_ready_event_fired
f()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var f=a.fn.jquery.split(".");
parseFloat(f[0]+"."+f[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var f=0,g=function(){var f=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!f)return!a&&(a=1,setTimeout(g,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(g,1);
// if the DOM is ready
f=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[f]("DOMContentLoaded",g,!1)};
// if the document_ready_event_fired
g()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var g=a.fn.jquery.split(".");
parseFloat(g[0]+"."+g[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var g=0,h=function(){var g=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!g)return!a&&(a=1,setTimeout(h,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(h,1);
// if the DOM is ready
g=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[g]("DOMContentLoaded",h,!1)};
// if the document_ready_event_fired
h()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var h=a.fn.jquery.split(".");
parseFloat(h[0]+"."+h[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var h=0,i=function(){var h=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!h)return!a&&(a=1,setTimeout(i,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(i,1);
// if the DOM is ready
h=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[h]("DOMContentLoaded",i,!1)};
// if the document_ready_event_fired
i()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var i=a.fn.jquery.split(".");
parseFloat(i[0]+"."+i[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var i=0,j=function(){var i=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!i)return!a&&(a=1,setTimeout(j,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(j,1);
// if the DOM is ready
i=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[i]("DOMContentLoaded",j,!1)};
// if the document_ready_event_fired
j()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var j=a.fn.jquery.split(".");
parseFloat(j[0]+"."+j[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var j=0,k=function(){var j=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!j)return!a&&(a=1,setTimeout(k,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(k,1);
// if the DOM is ready
j=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[j]("DOMContentLoaded",k,!1)};
// if the document_ready_event_fired
k()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var k=a.fn.jquery.split(".");
parseFloat(k[0]+"."+k[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var k=0,l=function(){var k=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!k)return!a&&(a=1,setTimeout(l,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(l,1);
// if the DOM is ready
k=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[k]("DOMContentLoaded",l,!1)};
// if the document_ready_event_fired
l()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var l=a.fn.jquery.split(".");
parseFloat(l[0]+"."+l[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var l=0,m=function(){var l=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!l)return!a&&(a=1,setTimeout(m,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(m,1);
// if the DOM is ready
l=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[l]("DOMContentLoaded",m,!1)};
// if the document_ready_event_fired
m()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var m=a.fn.jquery.split(".");
parseFloat(m[0]+"."+m[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var m=0,n=function(){var m=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!m)return!a&&(a=1,setTimeout(n,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(n,1);
// if the DOM is ready
m=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[m]("DOMContentLoaded",n,!1)};
// if the document_ready_event_fired
n()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var n=a.fn.jquery.split(".");
parseFloat(n[0]+"."+n[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var n=0,o=function(){var n=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!n)return!a&&(a=1,setTimeout(o,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(o,1);
// if the DOM is ready
n=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[n]("DOMContentLoaded",o,!1)};
// if the document_ready_event_fired
o()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var o=a.fn.jquery.split(".");
parseFloat(o[0]+"."+o[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var o=0,p=function(){var o=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!o)return!a&&(a=1,setTimeout(p,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(p,1);
// if the DOM is ready
o=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[o]("DOMContentLoaded",p,!1)};
// if the document_ready_event_fired
p()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var p=a.fn.jquery.split(".");
parseFloat(p[0]+"."+p[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var p=0,q=function(){var p=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!p)return!a&&(a=1,setTimeout(q,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(q,1);
// if the DOM is ready
p=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[p]("DOMContentLoaded",q,!1)};
// if the document_ready_event_fired
q()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var q=a.fn.jquery.split(".");
parseFloat(q[0]+"."+q[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var q=0,r=function(){var q=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!q)return!a&&(a=1,setTimeout(r,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(r,1);
// if the DOM is ready
q=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[q]("DOMContentLoaded",r,!1)};
// if the document_ready_event_fired
r()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var r=a.fn.jquery.split(".");
parseFloat(r[0]+"."+r[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var r=0,s=function(){var r=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!r)return!a&&(a=1,setTimeout(s,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(s,1);
// if the DOM is ready
r=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[r]("DOMContentLoaded",s,!1)};
// if the document_ready_event_fired
s()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var s=a.fn.jquery.split(".");
parseFloat(s[0]+"."+s[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var s=0,t=function(){var s=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!s)return!a&&(a=1,setTimeout(t,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(t,1);
// if the DOM is ready
s=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[s]("DOMContentLoaded",t,!1)};
// if the document_ready_event_fired
t()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var t=a.fn.jquery.split(".");
parseFloat(t[0]+"."+t[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var t=0,u=function(){var t=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!t)return!a&&(a=1,setTimeout(u,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(u,1);
// if the DOM is ready
t=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[t]("DOMContentLoaded",u,!1)};
// if the document_ready_event_fired
u()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var u=a.fn.jquery.split(".");
parseFloat(u[0]+"."+u[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var u=0,v=function(){var u=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!u)return!a&&(a=1,setTimeout(v,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(v,1);
// if the DOM is ready
u=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[u]("DOMContentLoaded",v,!1)};
// if the document_ready_event_fired
v()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var v=a.fn.jquery.split(".");
parseFloat(v[0]+"."+v[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var v=0,w=function(){var v=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!v)return!a&&(a=1,setTimeout(w,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(w,1);
// if the DOM is ready
v=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[v]("DOMContentLoaded",w,!1)};
// if the document_ready_event_fired
w()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var w=a.fn.jquery.split(".");
parseFloat(w[0]+"."+w[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var w=0,x=function(){var w=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!w)return!a&&(a=1,setTimeout(x,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(x,1);
// if the DOM is ready
w=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[w]("DOMContentLoaded",x,!1)};
// if the document_ready_event_fired
x()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var x=a.fn.jquery.split(".");
parseFloat(x[0]+"."+x[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var x=0,y=function(){var x=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!x)return!a&&(a=1,setTimeout(y,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(y,1);
// if the DOM is ready
x=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[x]("DOMContentLoaded",y,!1)};
// if the document_ready_event_fired
y()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var y=a.fn.jquery.split(".");
parseFloat(y[0]+"."+y[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var y=0,z=function(){var y=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!y)return!a&&(a=1,setTimeout(z,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(z,1);
// if the DOM is ready
y=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[y]("DOMContentLoaded",z,!1)};
// if the document_ready_event_fired
z()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var z=a.fn.jquery.split(".");
parseFloat(z[0]+"."+z[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var z=0,A=function(){var z=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!z)return!a&&(a=1,setTimeout(A,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(A,1);
// if the DOM is ready
z=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[z]("DOMContentLoaded",A,!1)};
// if the document_ready_event_fired
A()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var A=a.fn.jquery.split(".");
parseFloat(A[0]+"."+A[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var A=0,B=function(){var A=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!A)return!a&&(a=1,setTimeout(B,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(B,1);
// if the DOM is ready
A=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[A]("DOMContentLoaded",B,!1)};
// if the document_ready_event_fired
B()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var B=a.fn.jquery.split(".");
parseFloat(B[0]+"."+B[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var B=0,C=function(){var B=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!B)return!a&&(a=1,setTimeout(C,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(C,1);
// if the DOM is ready
B=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[B]("DOMContentLoaded",C,!1)};
// if the document_ready_event_fired
C()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var C=a.fn.jquery.split(".");
parseFloat(C[0]+"."+C[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var C=0,D=function(){var C=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!C)return!a&&(a=1,setTimeout(D,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(D,1);
// if the DOM is ready
C=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[C]("DOMContentLoaded",D,!1)};
// if the document_ready_event_fired
D()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var D=a.fn.jquery.split(".");
parseFloat(D[0]+"."+D[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var D=0,E=function(){var D=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!D)return!a&&(a=1,setTimeout(E,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(E,1);
// if the DOM is ready
D=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[D]("DOMContentLoaded",E,!1)};
// if the document_ready_event_fired
E()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var E=a.fn.jquery.split(".");
parseFloat(E[0]+"."+E[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var E=0,F=function(){var E=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!E)return!a&&(a=1,setTimeout(F,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(F,1);
// if the DOM is ready
E=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[E]("DOMContentLoaded",F,!1)};
// if the document_ready_event_fired
F()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var F=a.fn.jquery.split(".");
parseFloat(F[0]+"."+F[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 requires jQuery 1.4 or newer."+(a.ui.jQuery.fn.jquery?" Your jQuery version is "+a.ui.jQuery.fn.jquery+".":"")+" Upgrade jQuery to 1.4 or newer."))
// the jQuery_UI_core_version_check_function
function(){var F=0,G=function(){var F=!!window.addEventListener?"addEventListener":document.attachEvent?"attachEvent":null;
// if jQuery_UI_core_loaded_from_the_header_section
if(!F)return!a&&(a=1,setTimeout(G,1));
// if jQuery_UI_core_loaded_from_the_footer_section
if(!document.body)return setTimeout(G,1);
// if the DOM is ready
F=1;
// if jQuery_UI_core_loaded_from_the_footer_section
document.body.offset();
// remove the event_listener
document[F]("DOMContentLoaded",G,!1)};
// if the document_ready_event_fired
G()});
// the document_ready_function
function cleanup(){var a=document.addEventListener?"removeEventListener":document.detachEvent?"detachEvent":null;
document[a]("DOMContentLoaded",cleanup,!1)};
// the jQuery_UI_core_version_check_function
var G=a.fn.jquery.split(".");
parseFloat(G[0]+"."+G[1])<1.4&&(a.ui.jQuery=a.noConflict(!0),alert("jQuery UI 1.8.16 
```

```
// set the cursor for editing in the page
Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:59.0) Gecko/20100101
Firefox/59.0
// calculate the position of the page
Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:59.0) Gecko/20100101
Firefox/59.0
// calculate the bound number of the window
Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:59.0) Gecko/20100101
Firefox/59.0
// calculate the position of the page
```

```
/* the column for the open state is grey */
.fc-timelineOpen-cell {background:#ccc;}

/* the chart area when grid is on */
.fc-timelineGrid-chart {position:relative;}

/* the chart area when grid not inactive */
.fc-timelineGrid-chart:not(.fc-inactive) {
  cursor:pointer;
  -webkit-tap-highlight-color:rgba(0,0,0,0);
  -webkit-touch-callout:none;
  -webkit-user-select:none;
  -moz-user-select:none;
  -ms-user-select:none;
  user-select:none;
}

/* chart area for the event particular images */
.fc-timelineGrid-bgEvents {z-index:1;}

/* the selection line */
.fc-timelineGrid-selline {z-index:2;}

/* the hovering event image */
.fc-timelineGrid-hoverline {z-index:3;}

/* the persistent image */
.fc-timelineGrid-persline {z-index:4;}

/* a div which covers all of the chart area */
.fc-timelineGrid-cover {z-index:5;}

/* message from drag event handler */
.fc-timelineGrid-msg {z-index:6;}

/* the resource column header */
.fc-resourceHeader-header {
  -webkit-tap-highlight-color:rgba(0,0,0,0);
  -webkit-touch-callout:none;
  -webkit-user-select:none;
  -moz-user-select:none;
  -ms-user-select:none;
  user-select:none;
}

/* the back-ends square cards */
.fc-resourceHeader-header.fc-cards {
  -webkit-tap-highlight-color:rgba(0,0,0,0);
  -webkit-touch-callout:none;
  -webkit-user-select:none;
  -moz-user-select:none;
  -ms-user-select:none;
  user-select:none;
  -webkit-transition:-webkit-transform;
  -webkit-transition:-webkit-transform;
  -webkit-transition:-webkit-transform
```

```
// the page type input type
// (enum:string:text:textarea:checkbox:radio:select:file:date:time:datetime:datetime-local:month:week:color:hidden:range:email:tel:search:number:password:submit:button:reset:image:multiple:checked:disabled:readonly:required:autofocus:autocomplete:autocorrect:autocapitalize:spellcheck:placeholder:value:maxlength:minlength:pattern:step:min:max:accept:capture:size:width:height:alt:name:id:class:tabindex:accesskey:title:lang:dir:hidden:draggable:contenteditable:translate:spellcheck:autocorrect:autocapitalize:autocomplete:autofocus
```

```
/* Calculate Board Size and Mark Locations */
function calculateBoardSize(boardSize, canvas) {
  // Calculate board size based on canvas size.
  var minDim = Math.min(canvas.width, canvas.height);
  var boardPixelSize = minDim * 0.9;
  var squareSize = boardPixelSize / (boardSize - 1);
  var starPointRadius = squareSize * 0.15;
  var boundsX = (canvas.width - boardPixelSize) / 2;
  var boundsY = (canvas.height - boardPixelSize) / 2;
  return {
    boardPixelSize: boardPixelSize,
    squareSize: squareSize,
    starPointRadius: starPointRadius,
    boundsX: boundsX,
    boundsY: boundsY
  };
}

/* Draw the board lines and star points */
function drawBoard(boardSize, canvas, boardInfo, context) {
  var boardPixelSize = boardInfo.boardPixelSize;
  var squareSize = boardInfo.squareSize;
  var starPointRadius = boardInfo.starPointRadius;
  var boundsX = boardInfo.boundsX;
  var boundsY = boardInfo.boundsY;
  var i;
  var x, y;
  var starPoints = getStarPointPositions(boardSize);

  // Draw the board background color.
  context.fillStyle = "#dcb35c";
  context.fillRect(0, 0, canvas.width, canvas.height);

  // Draw the lines.
  context.beginPath();
  for (i = 0; i < boardSize; i++) {
    x = boundsX + (squareSize * i);
    y = boundsY + (squareSize * i);
    context.moveTo(x, boundsY);
    context.lineTo(x, boundsY + boardPixelSize);
    context.moveTo(boundsX, y);
    context.lineTo(boundsX + boardPixelSize, y);
  }
  context.stroke();

  // Draw the star points.
  context.beginPath();
  for (i = 0; i < starPoints.length; i++) {
    x = boundsX + (squareSize * starPoints[i].x);
    y = boundsY + (squareSize * starPoints[i].y);
    context.moveTo(x, y);
    context.arc(x, y, starPointRadius, 0, 2 * Math.PI, false);
  }
  context.fill();
}

/* Get the star point positions for the given board size */
function getStarPointPositions(boardSize) {
  var starPoints = [];
  if (boardSize == 19) {
    starPoints = [{x: 3, y: 3}, {x: 9, y: 3}, {x: 15, y: 3},
                  {x: 3, y: 9}, {x: 9, y: 9}, {x: 15, y: 9},
                  {x: 3, y: 15}, {x: 9, y: 15}, {x: 15, y: 15}];
  } else if (boardSize == 13) {
    starPoints = [{x: 3, y: 3}, {x: 9, y: 3},
                  {x: 6, y: 6},
                  {x: 3, y: 9}, {x: 9, y: 9}];
  } else if (boardSize == 9) {
    starPoints = [{x: 2, y: 2}, {x: 6, y: 2},
                  {x: 4, y: 4},
                  {x: 2, y: 6}, {x: 6, y: 6}];
  }
  return starPoints;
}

/* Draw the stones on the board */
function drawStones(
    boardSize, canvas, boardInfo, boardPosition, context) {
  var boardPixelSize = boardInfo.boardPixelSize;
  var squareSize = boardInfo.squareSize;
  var boundsX = boardInfo.boundsX;
  var boundsY = boardInfo.boundsY;
  var i, j;
  var x, y;
  var stoneRadius = squareSize / 2;

  for (i = 0; i < boardSize; i++) {
    for (j = 0; j < boardSize; j++) {
      x = boundsX + (squareSize * i);
      y = boundsY + (squareSize * j);
      if (boardPosition[i][j] == "B") {
        context.fillStyle = "#000000";
        context.beginPath();
        context.arc(x, y, stoneRadius, 0, 2 * Math.PI, false);
        context.fill();
      } else if (boardPosition[i][j] == "W") {
        context.fillStyle = "#ffffff";
        context.beginPath();
        context.arc(x, y, stoneRadius, 0, 2 * Math.PI, false);
        context.fill();
      }
    }
  }
}

/* Get the stone at the given point */
function getStone(boardPosition, x, y) {
  return boardPosition[x][y];
}

/* Place a stone on the board, modifying the
   board position
   Return value: true if the move is valid, false
   if it is not
   Side effects: if the move is valid, the
   board position is updated
   Illegal moves: ko, suicide, placing a stone
   on a point that already has a stone
   Does not check whether the game is over
   Does not check whether the move is a
   self-capture that kills stones
   Does not check whether the move is a
   self-capture that does not kill stones
   Does not check whether the move is a
   self-capture that does not kill stones
   Does not check whether the move is a
   self-capture that does not kill
```

```
/*
  Update the game state !!
function UpdateGame()
{
  // update the progress !!
  UpdateProgress();

  // if pending mode then change to game !!
  if (bPendingMode)
  {
    // set function and don't update the game state !!
    return;
  }

  // if in warning mode, disappearing, or getting, it acts as game paused !!
  if (iWarningTime > 0 || iDisappearTime > 0 || iGettingTime > 0 || iGameOverTime > 0)
  {
    // set function and don't update the game state !!
    return;
  }

  // if in game mode, update the game state !!
  if (iGameMode == 1)
  {
    // set the function state and don't update the game state !!
    return;
  }

  // if in game over mode, update the game state !!
  if (iGameMode == 2)
  {
    // set the function state and don't update the game state !!
    return;
  }

  // if the game state has started !!
  if (bGameStarted)
  {
    // if it is currently empty time !!
    if (iCurrentEmptyTime == 0 && (iNextEmptyTime == 0 || iNextEmptyTime == iCurrentTime))
    {
      // set the next empty time !!
      iNextEmptyTime = iCurrentTime + (iEmptyTimeMin + (iEmptyTimeMax - iEmptyTimeMin) * Math.random());
    }

    // if the current time is equal to the next empty time !!
    if (iCurrentTime == iNextEmptyTime)
    {
      // set the current and next empty time and reset the count !!
      iCurrentEmptyTime = iNextEmptyTime;
      iNextEmptyTime = iCurrentTime + (iEmptyTimeMin + (iEmptyTimeMax - iEmptyTimeMin) * Math.random());
      iCount = 0;
    }

    // if the current time is equal to the current time !!
    if (iCurrentTime == iCurrentEmptyTime)
    {
      // if the count is less than the maximum count !!
      if (iCount < iCountMax)
      {
        // increment the count !!
        iCount = iCount + 1;

        // if the count is equal to the current count !!
        if (iCount == iCurrentCount)
        {
          // increment the current count !!
          iCurrentCount = iCurrentCount + 1;

          // if the current count is greater than the maximum count !!
          if (iCurrentCount > iCountMax)
          {
            // set the current count to the maximum count !!
            iCurrentCount = iCountMax;
          }

          // if the current count is equal to the current count !!
          if (iCurrentCount == iCurrentCount)
          {
            // if the current count is equal to one !!
            if (iCurrentCount == 1)
            {
              // set the current state !!
              iCurrentState = 1;
              iCurrentStateTime = iCurrentTime;
              iCurrentStateDuration = iStateDurationMin + (iStateDurationMax - iStateDurationMin) * Math.random();
              iCurrentStateCount = 1;
              iCurrentStateCountMax = iCurrentStateCountMax;
              iCurrentStateCountTime = iCurrentTime;
              iCurrentStateCountDuration = iStateCountDurationMin + (iStateCountDurationMax - iStateCountDurationMin) * Math.random();
              iCurrentStateCountCount = 1;
              iCurrentStateCountCountMax = iCurrentStateCountCountMax;
              iCurrentStateCountCountTime = iCurrentTime;
              iCurrentStateCountCountDuration = iStateCountCountDurationMin + (iStateCountCountDurationMax - iStateCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCount = 1;
              iCurrentStateCountCountCountMax = iCurrentStateCountCountCountMax;
              iCurrentStateCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountDuration = iStateCountCountCountDurationMin + (iStateCountCountCountDurationMax - iStateCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCount = 1;
              iCurrentStateCountCountCountCountMax = iCurrentStateCountCountCountCountMax;
              iCurrentStateCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountDuration = iStateCountCountCountCountDurationMin + (iStateCountCountCountCountDurationMin - iStateCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountDuration = iStateCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountDurationMin - iStateCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin) * Math.random();
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCount = 1;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax = iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountMax;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountTime = iCurrentTime;
              iCurrentStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDuration = iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin + (iStateCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountCountDurationMin - iStateCount
```

```
# Generated by Cisco IOS Software
# NVRAM config last updated 1970/1/1 00:00:00
!
version 15.1
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
hostname Core-Switch
!
enable secret 5 $1$mERr$hx5rVt7/P3bVL09dquI6u/
!
aaa new-model
!
aaa authentication login default local
aaa authentication enable default enable
aaa authentication login console_auth local
aaa authentication enable console_auth enable
aaa authorization console
aaa authorization exec default local
!
aaa session-id common
!
clock timezone IST 5 30
!
ip domain-name corp.local
ip name-server 8.8.8.8 1.1.1.1
!
ip dhcp pool VLAN10
 network 10.10.10.0 255.255.255.0
 default-router 10.10.10.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool VLAN20
 network 10.10.20.0 255.255.255.0
 default-router 10.10.20.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool VLAN30
 network 10.10.30.0 255.255.255.0
 default-router 10.10.30.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool VLAN99
 network 10.10.99.0 255.255.255.0
 default-router 10.10.99.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool VLAN100
 network 10.10.100.0 255.255.255.0
 default-router 10.10.100.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool VLAN200
 network 10.10.200.0 255.255.255.0
 default-router 10.10.200.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool VLAN300
 network 10.10.300.0 255.255.255.0
 default-router 10.10.300.1
 dns-server 8.8.8.8
 lease 7 0 0
!
ip dhcp pool
```

```
/* Generated by Emscripten 1.38.31 */
var Module = typeof Module !== "undefined" ? Module : {};
var Object = {};
var Function = {};
var Array = {};
var Number = {};
var Boolean = {};
var Math = {};
var Date = {};
var String = {};
var RegExp = {};
var Error = {};
var EvalError = {};
var RangeError = {};
var ReferenceError = {};
var SyntaxError = {};
var TypeError = {};
var URIError = {};
var JSON = {};
var Infinity = 0;
var NaN = 0;
var undefined = 0;
var parseInt = function(str, radix) {};
var parseFloat = function(str) {};
var isNaN = function(number) {};
var isFinite = function(number) {};
var decodeURI = function(encodedURI) {};
var decodeURIComponent = function(encodedURIComponent) {};
var encodeURI = function(uri) {};
var encodeURIComponent = function(uriComponent) {};
var escape = function(string) {};
var unescape = function(string)
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:sy="http://purl.org/rss/1.0/modules/syndication/" xmlns:slash="http://purl.org/rss/1.0/modules/slash/" xmlns:wfw="http://wellformedweb.org/CommentAPI/" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:osi="http://rssnamespace.org/feedburner/extensions/1.0" xmlns:feedburner="http://rssnamespace.org/feedburner/extensions/1.0" xmlns:media="http://search.yahoo.com/mrss/" xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#" xmlns:georss="http://www.georss.org/georss" xmlns:gml="http://www.opengis.net/gml" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:wfw="http://wellformedweb.org/CommentAPI/" xmlns:media="http://search.yahoo.com/mrss/" xmlns:osi="http://rssnamespace.org/feedburner/extensions/1.0" xmlns:feedburner="http://rssnamespace.org/feedburner/extensions/1.0" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:sy="http://purl.org/rss/1.0/modules/syndication/" xmlns:slash="http://purl.org/rss/1.0/modules/slash/" xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#" xmlns:georss="http://www.georss.org/georss" xmlns:gml="http://www.opengis.net/gml" xmlns:dcterms="http://purl.org/dc/terms
```

```
// Copy a sequence of the current board layout
// Copy the current board layout
// Copy a sequence of the current board layout
// Boolean(Sequence)
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
// Boolean(Sequence)

// If disappearing and not currently searching psychokinesis
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
// Set the current board layout to the current board layout
Sequence(Sequence) = Sequence(Sequence, Blank)
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy the Sequence board layout
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Set the current board layout to the current board layout
Sequence(Sequence) = Sequence(Sequence, Sequence, Sequence, Sequence, Sequence, Sequence, Sequence, Sequence, Sequence)
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Set the current board layout to the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Set the board layout to the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Set the board layout to the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy the Sequence board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy the Sequence board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy a sequence of the current board layout
// Copy a sequence of the current board layout
Sequence(Sequence) = Sequence(Sequence)
// Copy the board layout to
```

```
// get the game page thread in the highest priority
GetGamePageThread();

// create the game page mutex
CreateGamePageMutex();

// get the game page structure in game zone
GetGamePageStructure();

// lock the mutex thread
LockMutexThread();

// clear the mutex string
ClearMutexString();

// get the mutex string
GetMutexString();

// get the mutex string in the game list search key
GetMutexStringInGameListSearchKey();

// determine the previously entered key in the game list search key
DeterminePreviousEnteredKeyInGameListSearchKey();

// clear the mutex string in the game page
ClearMutexStringInGamePageMutexStringInGameListSearchKey();

// initialize the game mutex
InitializeGameMutex();

// create a list of the game
CreateAListOfTheGame();

// create the game mutex string in the game list
CreateTheGameMutexStringInTheGameListSearchKey();

// lock the mutex thread
LockMutexThread();

// clear the mutex string
ClearMutexString();

// release mutex the game
ReleaseMutexTheGame();

// release mutex the game
ReleaseMutexTheGame();

// get the game list structure in game zone
GetGameListStructureInGameZone();

// get the game page thread in the game list
GetGamePageThreadInTheGameList();

// release mutex game page thread
ReleaseMutexGamePageThread();

// free the initial state of the keyboard keystroke
FreeTheInitialStateOfTheKeyboardKeystroke
```

```
// update the game information from the game data block ()
function updateGameInfo() {
  document.getElementsbyID("gameTitle").innerHTML=
  MyGameData.GameTitle;
}

// set the game grid size ()
function setGameGrid() {
  // set the game grid of a special size ()
  MyGameData.GameGrid=GameGrid;
  // set the game line column count
  MyGameData.GameLineCount=GameLineCount;
}

// set the game grid ()
function setGameGrid() {
  // set the game grid of a special size ()
  MyGameData.GameGrid=document.getElementsbyID("gridsize").value;
  // set the game line column count
  MyGameData.GameLineCount=
  document.getElementsbyID("gridsize").value;
  // reset the game block ()
  resetGameBlock();
}

// set the game grid ()
function setGameGrid() {
  // set the game grid of a special size ()
  MyGameData.GameGrid=GameGrid;
  // change the training strategy from the getting grid line ()
  MyGameData.GameTrainingStrategy=document.getElementsbyID("gridsize").value;
  // set the game line column count
  MyGameData.GameLineCount=GameLineCount;
  // set the getting grid size ()
  MyGameData.GetGridSize=GetGridSize;
  // clear the getting grid ()
  MyGameData.GetGrid=[];
  // reset the previous block ()
  MyGameData.PreviousBlock=[];
  // set the first line block and the random block ()
  MyGameData.FirstBlock=[];
  MyGameData.RandomBlock=[];
  // set the game grid ()
  MyGameData.GameGrid=GameGrid;
  // set the game line column count ()
  MyGameData.GameLineCount=document.getElementsbyID("gridsize").value;
  // set the getting grid size
  MyGameData.GetGridSize=document.getElementsbyID("gridsize").value;
  // reset the block ()
  resetGameBlock();
}

// reset the game grid ()
function resetGameGrid() {
  // reset the game and data line
  MyGameData.GameGrid=[];
  // get the to trigger in the game grid ()
  getGameGrid();
  // update the game and material window ()
  updateGameMaterial();
}

// reset the getting ()
function resetGetting() {
  // reset the getting grid ()
  MyGameData.GetGrid=[];
  // reset the previous getting block ()
  MyGameData.PreviousGetBlock=[];
  // set the to trigger in the game getting ()
  setGetGrid();
  // update the game and material window ()
  updateGetMaterial();
}

// reset the getting ()
function resetGetting() {
  // reset the getting grid ()
  MyGameData.GetGrid=[];
  // reset the previous getting block ()
  MyGameData.PreviousGetBlock=[];
  // set the to trigger in the game getting ()
  setGetGrid();
  // update the game and material window ()
  updateGetMaterial();
}

// reset the preferences window ()
function resetPreferenceWindow() {
  // reset the preferences window ()
  document.getElementsbyID("preferenceWindow").
  innerHTML=[];
  // set the preferences window ()
  setPreferenceWindow();
  // update the preferences window ()
  updatePreferenceWindow();
}

// update block ()
function updateBlock() {
  // update the basic block line type in preferences ()
  MyPreferenceWindow.BasicBlockLineType=MyPreferenceWindow.
  BasicBlockLineType;
  // update the basic block line color in preferences ()
  MyPreferenceWindow.BasicBlockLineColor=MyPreferenceWindow.
  BasicBlockLineColor;
  // update the basic line block matrix in preferences ()
  MyPreferenceWindow.BasicBlockLineMatrix=MyPreferenceWindow.
  BasicBlockLineMatrix;
  // update the basic line block matrix line color in preferences ()
  MyPreferenceWindow.BasicBlockLineMatrixLineColor=MyPreferenceWindow.
  BasicBlockLineMatrixLineColor;
  // update the basic line block matrix line type in preferences ()
  MyPreferenceWindow.BasicBlockLineMatrixLineType=MyPreferenceWindow.
  BasicBlockLineMatrixLineType;
  // update the basic line block matrix line color in preferences ()
  MyPreferenceWindow.BasicBlockLineMatrixLineColor=MyPreferenceWindow.
  BasicBlockLineMatrixLineColor;
  // update the basic line block line type in preferences ()
  MyPreferenceWindow.BasicBlockLineLineType=MyPreferenceWindow.
  BasicBlockLineLineType;
  // update the basic line block line color in preferences ()
  MyPreferenceWindow.BasicBlockLineLineColor=MyPreferenceWindow.
  BasicBlockLineLineColor;
  // update the basic line block matrix line type in preferences ()
  MyPreferenceWindow.BasicBlockLineMatrixLineType=MyPreferenceWindow.
  BasicBlockLineMatrixLineType;
  // update the basic line block matrix line color in preferences ()
  MyPreferenceWindow.BasicBlockLineMatrixLineColor=MyPreferenceWindow.
  BasicBlockLineMatrixLineColor;
  // update the basic line block line type in preferences ()
  MyPreferenceWindow.BasicBlockLineLineType=MyPreferenceWindow.
  BasicBlockLineLineType;
  // update the basic line block line color in preferences ()
  MyPreferenceWindow.BasicBlockLineLineColor=MyPreferenceWindow.
  BasicBlockLine
```

```
// update the accident_name_block_state()
function accident_name_block_state(){
$(".accident_name_block").hide();
$(".accident_name_block:eq(0)").show();
}

// update the fighting_national_guards_block_state()
function fighting_national_guards_block_state(){
$(".fighting_national_guards_block").hide();
$(".fighting_national_guards
```

```
// set the position type to relative so the bom position stays
// relative to the board when the board moves
bom.setPosition(Position.RELATIVE);

// set anchoring mode to top left
bom.setAnchor(Anchor.TOP_LEFT);

// set anchoring mode to top right
bom.setAnchor(Anchor.TOP_RIGHT);

// set anchoring mode to top center
bom.setAnchor(Anchor.TOP_CENTER);

// the bom position data structure
var bomPosition = board.getBomPosition();

// convert the current position to mils
var bomPosMils = bomPosition.toMils();

// create a new position in the top of the board position data array
var newPosition = new Position(0, 0, Position.MILS);

// copy the bom data to the end of the board position data array
var bomPosCopy = bomPosition.copyTo(board.getPositions());

// update the component groups in the board position data
var updatedPositions = board.updateGroups();

// the bom position data structure
var bomPosition = board.getBomPosition();

// if anchoring mode bom
if (bomPosition.isAnchored()
  {
  // anchoring mode
  var anchorMode = bomPosition.getAnchor();

  // if the bom position is in process then it is
  // anchored to a corner
  if (anchorMode == Anchor.TOP_LEFT
    {
    // move the bom position to the top left
    // corner of the board
    bomPosition.setPosition(0, 0);
    }
  }

// move the bom position into inches to preferences
var bomPosInches = bomPosition.toUnits(Units.INCHES);

// set the bom position
bomPosition.setPosition(1, 1);

// save all data to the pnp data bom output
exporter.writeBomPosition(bomPosition, "Bom Test Position");

// the board outline data structure
var boardOutline = board.getOutline();

// copy the board outline data into the pnp data bom
// output data structure
var boardOutlineCopy = boardOutline.copyTo(exporter.getBomOutline());

// add a bom point
var bomPoint = new Point(0, 0);

// copy the point into the outline structure from the pnp data bom
// output data
var bomPointCopy = bomPoint.copyTo(
  exporter.getBomOutline());

// if the outline contains a point
if (boardOutline.contains(bomPoint)
  {
  // if the outline contains a point
  // in the bom output
  if (exporter.getBomOutline().contains(bomPoint)
    {
    // set the bom point
    bomPoint.setPosition(0, 0);
    }
  }

// add a bom point
var bomPoint = new Point(0, 0);

// add the point to the bom output
var bomPointCopy = bomPoint.copyTo(
  exporter.getBomOutline());

// the bom output outline
var bomOutline = exporter.getBomOutline();

// if the bom output contains a point
if (bomOutline.contains(bomPoint)
  {
  // set the bom point
  bomPoint.setPosition(0, 0);
  }
```

```
// get the current square bonus from the square bonus array
int currentSquareBonus = squareBonus[x][y];

// update the square bonus square
squareBonus[x][y] = squareBonus[x][y] / 2;

// display generate effects
displayGenerateEffects(x, y);

// update the tile type to empty
boardSquares[x][y].type = TileTypeEmpty;

// update the tile value to 0
boardSquares[x][y].value = 0;

// update the board position
[self setNeedsDisplay];

// update the bomb counter
int bombCounter = [[[gameBoard objectAtIndex:1] objectAtIndex:1] intValue];
bombCounter = bombCounter - 1;
[[[gameBoard objectAtIndex:1] objectAtIndex:1] setValue:[NSNumber numberWithInt:bombCounter]];

// get the current bomb counter
int currentBombCounter = [[[gameBoard objectAtIndex:1] objectAtIndex:1] intValue];

// check if the bomb counter is 0
if (currentBombCounter == 0)
{
    // disable the generate effects
    [self disableGenerateEffects];
}

// get the current score counter from the board
int currentScore = [[[gameBoard objectAtIndex:0] objectAtIndex:1] intValue];

// get the current bonus multiplier from the board
int currentBonusMultiplier = [[[gameBoard objectAtIndex:2] objectAtIndex:1] intValue];

// calculate the new score
int newScore = currentScore + currentSquareBonus * currentBonusMultiplier;

// update the score counter
[[[gameBoard objectAtIndex:0] objectAtIndex:1] setValue:[NSNumber numberWithInt:newScore]];

// update the bonus multiplier
int newBonusMultiplier = currentBonusMultiplier + 1;
[[[gameBoard objectAtIndex:2] objectAtIndex:1] setValue:[NSNumber numberWithInt:newBonusMultiplier]];

// update the board
[self setNeedsDisplay];

// get the current bomb counter from the bomb counter array
int bombCounter = [[[gameBoard objectAtIndex:1] objectAtIndex:1] intValue];

// check if the bomb counter is 0
if (bombCounter == 0)
{
    // disable the bomb effects
    [self disableBombEffects];
}

// get the current bonus multiplier from the bonus multiplier array
int bonusMultiplier = [[[gameBoard objectAtIndex:2] objectAtIndex:1] intValue];

// check if the bonus multiplier is 1
if (bonusMultiplier == 1)
{
    // disable the bonus effects
    [self disableBonusEffects];
}
```

```
# Image: ffd_hq_test_1.jpg
# Generated: 2025-08-08 00:00:00
# Source: https://commons.wikimedia.org/wiki/File:Ferris_wheel_in_Budapest,_Hungary,_July_2006.jpg
# Licence: Public Domain
# Dimensions: 1024x1536 px
# Total characters: 65536

#
```

```
// update the position sprite white bishop image
PieceIndexOfWhiteBishop0.image = GetTextureImage()

// update the position sprite white bishop image
PieceIndexOfWhiteBishop1.image = GetTextureImage()

// update the position sprite white rook image
PieceIndexOfWhiteRook0.image = GetTextureImage()

// update the position sprite white queen image
PieceIndexOfWhiteQueen.image = GetTextureImage()

// update the position sprite white king image
PieceIndexOfWhiteKing.image = GetTextureImage()

// update the position sprite black pawn image
PieceIndexOfBlackPawn.image = GetTextureImage()

// update the position sprite black knight image
PieceIndexOfBlackKnight0.image = GetTextureImage()

// update the position sprite black bishop image
PieceIndexOfBlackBishop0.image = GetTextureImage()

// update the position sprite black rook image
PieceIndexOfBlackRook0.image = GetTextureImage()

// update the position sprite black queen image
PieceIndexOfBlackQueen.image = GetTextureImage()

// update the position sprite black king image
PieceIndexOfBlackKing.image = GetTextureImage()

// update the white knight1 sprite texture label
UpdateCurrentWhiteKnight1SpriteIndexTextureLabel(PieceIndexOfWhiteKnight1SpriteIndex, Kingsize:
    true)

// update the white queenside castle button texture label
UpdateCurrentCastleWhiteQueensideCastleButtonTextureLabel(bCastleWhiteQueensideCastle, Kingsize:
    true)

// update the black kingside castle button texture label
UpdateCurrentCastleBlackKingsideCastleButtonTextureLabel(bCastleBlackKingsideCastle, Kingsize:
    true)

// update the black queenside castle button texture label
UpdateCurrentCastleBlackQueensideCastleButtonTextureLabel(bCastleBlackQueensideCastle, Kingsize:
    true)

// update position sprite texture
GetPositionSpriteTextureImage()

// start a new game if
if bStartNewGameAfterCheckmate(_checkpoint)
{
// talk to the new game mode item if
NewGameMode(this.state + MSGcontrol.StartNewGameTalk);

// set that currently playing if want to start a new game or not if
bCurrentlyPlaying = true;

// create the textures for the start new game query
bCreateTexture = ["Yes", "No"]

// show the new game alert if
MakeAnotherGameAlertNow(_query:"New Game")
}

// process the alert if
for Processalert()
{
// if asking to quit the application if
if bQuitAppQuestion
{
    // if the game sound of a button click if
    PlaySoundOnceOfPress("buttonclick")

    // if user chose the first alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertFirstButtonAction.rawValue
    {
        // quit the application if
        DBAppLibrarian.QuitApplication()
    }
    // if user doesn't want to quit the application if
    else
    {
        // set that no longer querying whether want to quit the application if
        bAnyQuestionAppQuestion = false
    }
}
// if were position query if
else if QuestionPosition
{
    // play the game sound of a button click if
    PlaySoundOnceOfPress("buttonclick")

    // set query were position to false if
    QuestionPosition = false

    // turn off the save positions menu item if
    ToggleMenuItemEnabled(_menuitem:MenuItemLabelSavePositions)

// if loading a saved position and there are no saved positions if
else if QuestionLoadPosition
{
    // play the game sound of a button click if
    PlaySoundOnceOfPress("buttonclick")

    // set loading saved position to false if
    QuestionLoadPosition = false

    // turn off the load positions menu item if
    ToggleMenuItemEnabled(_menuitem:MenuItemLabelLoadPositions)

// if querying that timed game is one if
else if QuestionTimedGame
{
    // remove the timed game editor if
    RemoveTimedGameEditor()

    // if querying whether want to save the edited board position if
    if QuestionSaveEditedPosition
    {
        // play the game sound of a button click if
        PlaySoundOnceOfPress("buttonclick")

        // if the the layout is legal if
        if PDCheckIsLegal()
        {
            // if user chose to save the edited position if
            if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertFirstButtonAction.rawValue
            {
                // save the edited position if
                AttemptSaveEditedPosition()
            }
            // if didn't choose to edit the position again if
            if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertSecondButtonAction.rawValue
            {
                // edit the position editor if
                EditPositionEditor()
            }
        }
        // the the layout was not legal and chose to edit if
        else if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertSecondButtonAction.rawValue
        {
            // edit the position editor if
            EditPositionEditor()
        }
    }

    // set that no longer querying to save edited position if
    QuestionSaveEditedPosition = false

// if querying user if want to see game or not if
else if QuestionPromptGame
{
    // play the game sound of a button click if
    PlaySoundOnceOfPress("buttonclick")

    // if user chose the first alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertFirstButtonAction.rawValue
    {
        // start a new game if chose if
        StartNewGame()
    }
    // if user chose the first alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertFirstButtonAction.rawValue
    {
        // start a new game if chose if
        StartNewGame()
    }
    // if user chose the second alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertSecondButtonAction.rawValue
    {
        // set that done querying user if want to start a new game or not if
        bAnyQuestionPrompt = false
    }

    // set that no longer querying to save edited position if
    QuestionSaveEditedPosition = false

// if querying user if want to see a game or not if
else if QuestionPromptGame
{
    // play the game sound of a button click if
    PlaySoundOnceOfPress("buttonclick")

    // if user chose the first alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertFirstButtonAction.rawValue
    {
        // open the game with game filename 0.4-31 seconds if
        Backup.current.addTimer(timeInterval:(0.4), target:self, selector:
            #selector(DBAppLibrarian.OpenGameWithFilenameAfterTimeIntervalPassed(_:), signature:false,
            forMode:RunLoopMode.commonModes)
    }
    // reset chose the other alert button if
    else
    {
        // set that done querying whether to open a game file if
        bAnyQuestionPrompt = false
    }

// if querying what types of pieces user wants to promote pawn to if
else if QuestionPromotion
{
    // play the game sound of a button click if
    PlaySoundOnceOfPress("buttonclick")

    // if user chose the first alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertFirstButtonAction.rawValue
    {
        // set the promotion type subpoint to be a queen if
        PromotionSubpoint = "q"
    }
    // set the moving piece to be a queen if
    MovingPiece = 5
    // if user chose the second alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertSecondButtonAction.rawValue
    {
        // set the promotion type subpoint to be a rook if
        PromotionSubpoint = "r"

        // set the moving piece to be a rook if
        MovingPiece = 4
    }
    // if user chose the third alert button if
    if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertThirdButtonAction.rawValue
    {
        // set the promotion type subpoint to be a bishop if
        PromotionSubpoint = "b"

        // set the moving piece to be a bishop if
        MovingPiece = 3
    }
    // if user chose the fourth alert button if
    else if ReturnVoid == DBAppLibrarian.MakeDbaResponse.alertThirdButtonAction.rawValue + 1
    {
        // set the promotion type subpoint to be a knight if
        PromotionSubpoint = "n"

        // set the moving piece to be a knight if
        MovingPiece = 2
    }

    // if black is currently moving if
    if bCurrentlyMoving
    {
        // move the moving piece black if
        MovingPiece = -MovingPiece
    }

    // update the last move on the move list to show a promotion occurred if
    UpdateLastMoveOnMoveListWithPromotion()

    // finish up the move just made if
    FinishMoves()

    // set that done querying what piece user wants to promote pawn to if
    bAnyQuestionPromotion = false

    // if querying whether the set chess engine was loaded or not if
    else if QuestionSetChessEngine
    {
        // play the game sound of a button click if
        PlaySoundOnceOfPress("buttonclick")

        // set that no longer querying whether the set chess engine was loaded or not if
        QuestionSetChessEngine = false
    }
}

// get the file number for mouse coordinate x if
func FileNumber(_, X:Int) -> String
{
    // subtract board offset x a direction and divide by square width if
    RawNumFloat = (X-FileOrigin) / SQ / Squarewidth

    // if the board has been created if
    if bBoardCreated
    {
        // the returned rank if
        RawNumFloat = RawNumFloat + 1
    }

    // return the appropriate file symbol if
    return FilleSymbolsList[RawNumFloat.rw]
}

// get the rank number for mouse coordinate y if
func RankNumber(_, Y:Int) -> String
{
    // subtract board offset in y direction and divide by square width if
    RawNumFloat = (Y-FileOrigin) / SQ / Squarewidth

    // if the board has been created if
    if bBoardCreated
    {
        // the returned file if
        RawNumFloat = 7 - RawNumFloat.rw
    }

    // return the appropriate rank symbol if
    return RankSymbolsList[RawNumFloat.rw]
}

// function to create the square label for mouse coordinate location if
func MakeSquareLabel() -> String
{
    // return the square label for mouse coordinate location if
    return FileNumber(this:this.MouseLocation.x) + RankNumber(this:this.MouseLocation.y)}

// kick the application window if
MSMotion.func MinkeWindowsAtLocation(_:_checkpoint)
{
    // if the main window is visible if
    if MainWindow.sectionIsVisibleFunction(MSWindow.SectionIndex.state)
    {
        // hide the application window if
        HideWindow()
    }

    // set the cursor to active if
    NSCursor.active.set()

    // the main window is set visible if
    MainWindow.isVisible = true

    // set the cursor to arrow if
    NSCursor.arrow.set()

    // make the main window visible if
    MainWindow.makeKeyAndOrderFront(nil)
}

// if was an attached sheet if
if bwas AttachedSheetOnBeen() -> Bool
{
    // if asking whether a sheet is attached or not if
    return MainWindow.attachedSheet != nil
}

// close the attached sheet if have one if
func CloseAttachedSheet()
{
    // get that going into a panel if
    let Panel = Sheet

    // get the previously attached sheet if
    let PreviouslyAttachedSheet = MainWindow.attachedSheet as! NSSheet
    // if there is a previously attached sheet if
    if PreviouslyAttachedSheet != nil
    {
        // end the sheet if
        MainWindow.endSheet(PreviouslyAttachedSheet)

        // close the previously attached sheet if
        PreviouslyAttachedSheet.close()
    }

// reopen the previously attached sheet if had one if
func ReopenPreviouslyAttachedSheet()
{
    // if there was a previously attached sheet if
    if bWasAttachedSheetOnBeen()
    {
        // reopen the previously attached sheet if
        MainWindow.beginSheet(PreviouslyAttachedSheet)

        // reset the previously attached sheet to nil if
        PreviouslyAttachedSheet = nil
    }

    // set that no longer in a panel if
    bPanelIsOnTop = false
}

// determine if user wants to quit the application if
MSMotion.func QuitAppQuestion(_:_checkpoint)
{
    // set that currently querying user if wants to quit the application if
    QuestionQuitAppQuestion = true

    // set the alert buttons if
    SetAlertButtonTitles(["Quit", "Cancel"])

    // start the user if they want to quit the application or not if
    RunAlertSheetOnMainWindow(_sheetTitle:"Quit Application")

    // make the main window visible if
    MainWindow.makeKeyAndOrderFront(nil)
}

// undo the last chess move played in the game if
MSMotion.func UndoMoveOnGameLoaded(_:_checkpoint)
{
    // if there are no moves in the move list to undo if
    if MoveList.count(g)
    {
        // wait function since there are no moves in the move list to undo if
        return
    }

    // set the previous move list to the current move list if
    Previous=MoveList = MoveList

    // set the previous animation list to the current animation list if
    PrevAnimationList = AnimationList

    // remove the last move from the move list if
    MoveList.removeLast();

    // remove the last animation from the new animation list if
    AnimationList.removeLast()

    // replay all the moves excluding the last move if
    ReplayMoves(_ EndMoveList.count, _ StopAfterFalse)

    // compact the transposition table if
    DbpLibraryPerformTableCompact()

    // create the autosave game data if
    CreateAutosaveGame()
}

// correct the pawn to a sequence of frames if
func CorrectPawnToSequenceOfFrames(_:Int)
{
    // set that need to compute the board if
    NeedToComputeBoard = true

    // clear the frames if
    Frames.removeAll()

    // create a starting pip count if
    PipCount = 0

    // initialize the board if
    InitializeBoard()

    // update the board and replayed pieces if
    UpdateBoardAndReplayedPieces()

    // if had a board snapshot if
    TakeBoardSnapshot(_:FrameNum0)

    // add the board image to the frames list if
    Frames.append(SnapshotImage)

    // for each move in the move list if
    for Index in 0 ..< MoveList.count
    {
        // get the move from the move list to animate if
        bCMove = MoveList[Index]

        // add the move on the board if
        bMakeMoveOnBoard(bCMove)

        // if the move looking for dead piece if
        if bCMove.bLook
        {
            // increment the pip count if
            PipCount += 1

            // process the current dot move if
            ProcessDotMove()

            // if had a promotion if
            if bCMove.bPromote != 0
            {
                // set the piece moving to the type of piece pawn promoted to if
                MovingPiece = bCMove.bPromote
            }

            // put the piece that moved into the destination square if
            bPutMovedPieceOnSquare()

            // perform en passant move and capture if needed if
            bPerformEnPassantMoveAndCapture()

            // perform castling move if had one if
            bPerformCastlingMoveIfNeeded()

            // update the board and replayed pieces if
            UpdateBoardAndReplayedPieces()

            // add the board snapshot if
            TakeBoardSnapshot(_:FrameNum0)

            // add the board image to the frames list if
            Frames.append(SnapshotImage)
        }
    }

    // get the game gallery or gallery image if
    func Get_GetGalleryOrImage()
    {
        // if not currently creating the game gallery or grid images if
        if!bCreateGameGalleryOrGrid = true
        {
        // if currently the game not finished if
        if!bGameHasBeenCompletedAndGalleryOrGridNotYetCreated(bIn:FrameNum0)
        {
        // if on the main thread if
        if!IsOnMainThread.main()
        {
            // save the frames to the gallery grid size if
            self.SaveGalleryOrGridSize(Frames:Frames)

            // save the frames to the gallery or grid image if
            self.SaveGalleryOrGridImage(Frames:Frames, pieceSize:pieceSize)

            // update the game gallery or grid image if
            self.UpdateGameGalleryOrGridImage()

            // save the main window title to the "game gallery" or "game grid" if
            self.MainWindow.title = self.GalleryOrGridTitle

            // set that done selecting progress if
            self.bSelectedProgress = false

            // set that done creating the game gallery if
            self.bCreateGameGalleryOrGrid = false
        }
        }
    }

// replay the moves from start to end if
func ReplayMoves(_, StartFrom, StopAfter, _:Any=Bool)
{
    // set the starting pip count if
    PipCount = StartFrom

    // if starting from the beginning move if
    if StartFrom == 0
    {
        // initialize the board if
        InitializeBoard()
    }

    // if need to update the board display if
    if bUpdate
    {
        // update the board if
        UpdateBoardAndReplayedPieces()
    }

    // print the game information if
    PrintGameInfo()

    // set the starting move index if
    Index = Start

    // if replaying moves with display if
    if bUpdate
    {
        // run move animation if game replay if
        RunMoveAnimation()
    }

    // for replaying moves with display if
    else
    {
        // loop over all the moves in the move list if
        for Index in Start .. End
        {
            // get the move from the move list if
            bCMove = MoveList[Index]

            // make the move on the board if
            bMakeMoveOnBoard(bCMove)

            // if the move looking for dead piece if
            if bCMove.bLook
            {
                // increment the pip count if
                PipCount += 1

                // process the current dot move if
                ProcessDotMove()

                // if had a promotion if
                if bCMove.bPromote != 0
                {
                    // set the piece moving to the type of piece pawn promoted to if
                    MovingPiece = bCMove.bPromote
                }

                // put the piece that moved into the destination square if
                bPutMovedPieceOnSquare()

                // perform en passant move and capture if needed if
                bPerformEnPassantMoveAndCapture()

                // perform castling move if had one if
                bPerformCastlingMoveIfNeeded()
            }
        }

        // update the board if
        UpdateBoardAndReplayedPieces()
    }

    // print the game info if
    PrintGameInfo()
}

// update the chessboard display if
func UpdateBoardAndReplayedPieces()
{
    // get the list of squares where pieces can be captured if
    ListOfSquaresCanBeCaptured = GetSquaresCanBeCaptured(bIn:CaptureSquaresReady);

    // remove the last object in the list of capture squares since it is empty if
    ListOfSquaresCanBeCaptured.removeLast();

    // the board as can check for squares where pieces are threatened if
    RearDraw(bThreatenedBoard, Black)

    // get the list of squares where pieces can be threatened if
    ListOfSquaresCanBeThreatened = GetSquaresCanBeThreatened(bIn:
        RearDraw(bIn:LastThreatenedBoard, Black), bIncludeThreatenedBoard:true)

    // remove the last object in the list of threaten squares since it is empty if
    ListOfSquaresCanBeThreatened.removeLast();

    // update the pieces layer if
    UpdatePiecesLayer()

    // if bit is the game grid, not in the gallery grid, and not checking a movie if
    if bIsGameGrid && !bCreateMovieFromGame
    {
        // update the captured white square display if
        CaptureOfSomeWhiteSlots.UpdateCapturedPieces(pSquaresWhiteCaptured, MinNumTrue, Set:
            CaptureOfSomeWhiteSlots.bCapturedPieceSquare)
        // update the captured black square display if
        CaptureOfSomeBlackSlots.UpdateCapturedPieces(pSquaresBlackCaptured, MinNumTrue, Set:
            CaptureOfSomeBlackSlots.bCapturedPieceSquare)
    }
}

// determine if a white pawn is currently moving if
func WhitePawnMoving() -> Bool
{
    // return whether a white pawn is currently moving if
    return MovingPiece == 1
}

// determine if a black pawn is currently moving if
func BlackPawnMoving() -> Bool
{
    // return whether a black pawn is moving if
    return MovingPiece == -1
}

// determine if the piece moving is white if
func WhitePieceMoving() -> Bool
{
    // return whether the piece moving is white if
    return MovingPiece > 0
}

// determine if the piece moving is black if
func BlackPieceMoving() -> Bool
{
    // return whether the piece moving is black if
    return MovingPiece < 0
}

// get the application value if
func GetApplicationValue(_:Int=0, bIsMultiCommandline:_array, bCommandline:_nsvalueToArgv)
{
```