```
/* Minify Source code of Mario Kart Tracks 3.1.1 on prettydiffs.com */

var global,
module,
exports,
define,
sort,
parse,
Object.keys,Object.assign = Object.keys,Object.assign;
var lexer,
lexerOptions,
parseOptions,Object.assign(parseOptions,
{"parse_brace":"parse_brace",
"parse_brace_completion":"parse_brace_completion",
"parse_comment":"parse_comment",
"parse_comment_block":"parse_comment_block",
"parse_comment_line":"parse_comment_line",
"parse_function":"parse_function",
"parse_name":"parse_name",
"parse_operator":"parse_operator",
"parse_punctuator":"parse_punctuator",
"parse_regex":"parse_regex",
"parse_string":"parse_string",
"parse_template":"parse_template",
"parse_template_literal":"parse_template_literal",
"parse_template_literal_expression":"parse_template_literal_expression",
"parse_template_literal_string":"parse_template_literal_string",
"parse_template_literal_variable":"parse_template_literal_variable",
"parse_template_literal_variable_start":"parse_template_literal_variable_start"},
{"parse":"parse",
"lex":"lex"}),
parse,
lexer,
lexerOptions,
parseOptions,
Object.keys,
Object.assign,
global.parsejs = {parse: parse, lex: lex},
module.exports = global.parsejs,
define(function(){return global.parsejs});
var lex = function(source, lexerOptions){
var lexer = new Lexer(source, lexerOptions),
tokens = [],
token = lexer.lex();
while (token.type !== "EOF") {
tokens.push(token);
token = lexer.lex();
}
return tokens;
};
var Lexer = function(source, options){
var lexerOptions = options || {};
this.source = source.toLowerCase().replace(/\r\n?|[\n\u2028\u2029]/g, "\n").replace(/^[\u2028\u2029]+|\s+$/g, "");
if (lexerOptions.ignoreWhitespace !== false) {
this.ignoreWhitespace = true;
}
if (lexerOptions.ignoreComments !== false) {
this.ignoreComments = true;
}
if (lexerOptions.ignoreNewlines !== false) {
this.ignoreNewlines = true;
}
this.varMap = {};
this.tokens = [];
this.token = this.next();
this.peekCache = null;
this.braceStack = [];
this.templateLiteralStack = [];
this.templateLiteralExpressionStack = [];
this.parseBrace = function(){
var token = this.token;
if (token.value === "{") {
this.braceStack.push("brace");
}
else if (token.value === "}") {
this.braceStack.pop();
}
return token;
};
this.parseBraceCompletion = function(){
var token = this.token;
var braceStack = this.braceStack;
var templateLiteralStack = this.templateLiteralStack;
var templateLiteralExpressionStack = this.templateLiteralExpressionStack;
if (token.value === "}") {
var lastBrace = braceStack.pop();
if (lastBrace === "template") {
templateLiteralStack.push("template");
templateLiteralExpressionStack.pop();
}
else if (lastBrace === "template_expression") {
templateLiteralExpressionStack[templateLiteralExpressionStack.length - 1] = (templateLiteralExpressionStack[templateLiteralExpressionStack.length - 1] + 1);
}
else {
throw new Error("Unexpected '}' on line " + token.line + ", column " + token.column + ".");
}
}
else if (token.value === "${") {
var lastTemplateLiteralExpression = templateLiteralExpressionStack[templateLiteralExpressionStack.length - 1];
if (lastTemplateLiteralExpression === undefined) {
throw new Error("Unexpected '${' on line " + token.line + ", column " + token.column + ".");
}
else {
templateLiteralExpressionStack[templateLiteralExpressionStack.length - 1] = (lastTemplateLiteralExpression + 1);
}
}
return token;
};
this.parseComment = function(){
var token = this.token;
var source = this.source;
var ignoreComments = this.ignoreComments;
var ignoreNewlines = this.ignoreNewlines;
var lastChar = null;
var lastNewline = null;
var lastNewlineIndex = null;
var nextChar = null;
var nextNewline = null;
var nextNewlineIndex = null;
var nextNonWhitespace = null;
var nextNonWhitespaceIndex = null;
var nextTwoChars = null;
var nextWhitespace = null;
var nextWhitespaceIndex = null;
var offset = null;
var parseCommentBlock = function(){
var commentBlockEnd = null;
var commentBlockEndIndex = null;
var commentBlockStart = null;
var commentBlockStartIndex = null;
var offset = null;
var parseCommentBlockEnd = function(){
var token = this.token;
var source = this.source
```

```
  >> pseudo_ln_no = 0
  >> pseudo_ln_no = 0

>> prepare the moving piece to be displayed >>
displayPiece(TheBoardPosition,TheMovingPiece,NewPosition,NewLn,NewLnPos)

>> find the moving piece >>
pieceOnLnPos(Ln,LnPos,TheMovingPiece,_,_,_
```

```
// Generated by Dart SDK 2.19.6 (stable) on Wed Jan 11 2023 15:15:12 +0000
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// Converted by Kythe 0.0.72 (2022-11-17) from Dart 2.19.0 (stable) (Wed Jan
// 11 2023 15:15:12 +0000) to JSON.
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// Schema: gs3 (node_id_v2)
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// Corpus: dart.pub
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// Root: lib/src/utilities/extensions/element.dart
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[{"kind":"anchor","start":0,"end":0,"snippet":{"begin":0,"end":0,"
  // Returns the field declaration that is private to this library.
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  // If the field initializer has no private id, or
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  // if the private id is
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  // not the same name as this private id,
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  // then the same field as this field is
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  // not private.
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
// Add service to create a work in a game //
POST /api/v1/game/{gameId}/work/create
{
  // Id the game owner //
  "ownerId": 1,
  // Array that each element qualify this service a game in this service //
  "requirementsGameServices": [{"requirementsGameServiceId": 1,
    "requirementsGameServiceDescription": "Requirements for the work",
    "requirementsGameServiceCreationDate": "2020-12-31T00:00:00.000Z",
    "requirementsGameServiceLastUpdate": "2020-12-31T00:00:00.000Z",
    "requirementsGameServiceCreationUser": "Administrator",
    "requirementsGameServiceLastUpdateUser": "Administrator"
  }]
}

// Add the service answer //
POST /api/v1/game/{gameId}/work/{workId}/answer
{
  // Id the game //
  "gameId": 1,
  // Id the work //
  "workId": 1,
  // Array that each element is a work requirement //
  "workRequirementsAnswers": [{"workRequirementsAnswerId": 1,
    "workRequirementsAnswerDescription": "Requirements for the work",
    "workRequirementsAnswerCreationDate": "2020-12-31T00:00:00.000Z",
    "workRequirementsAnswerLastUpdate": "2020-12-31T00:00:00.000Z",
    "workRequirementsAnswerCreationUser": "Administrator",
    "workRequirementsAnswerLastUpdateUser": "Administrator",
    "workRequirementsAnswerId": 1,
    "workRequirementsAnswerDate": "2020-12-31T00:00:00.000Z",
    "workRequirementsAnswerStatus": "Pendiente",
    "workRequirementsAnswerComment": "Comentario"
  }]
}

// Get the header image //
GET /api/v1/game/{gameId}/header/image
{
  // Content-Type header //
  "Content-Type": "image/jpeg",
  // Array of bytes of the image //
  "Content-Disposition": "inline; filename=\"header.jpeg\"",
  "Cache-Control": "max-age=604800, must-revalidate",
  "Expires": "Thu, 01 Jan 1970 00:00:00 GMT",
  "Last-Modified": "Wed, 23 Jun 2021 00:00:00 GMT",
  "ETag": "\"1234567890abcdef\"",
  "Accept-Ranges": "bytes",
  "Content-Length": "1234",
  "Vary": "Accept-Encoding",
  "Connection": "keep-alive",
  "Date": "Wed, 23 Jun 2021 00:00:00 GMT",
  "Server": "nginx/1.18.0",
  "X-Content-Type-Options": "nosniff",
  "X-Frame-Options": "SAMEORIGIN",
  "X-XSS-Protection": "1; mode=block",
  "Strict-Transport-Security": "max-age=31536000; includeSubDomains; preload",
  "Content-Security-Policy": "default-src 'self'; image-src 'self' data:",
  "Referrer-Policy": "strict-origin-when-cross-origin",
  "Permissions-Policy": "geolocation=(), microphone=(), camera=()",
  "Cross-Origin-Resource-Policy": "same-origin",
  "Cross-Origin-Opener-Policy": "same-origin",
  "Cross-Origin-Embedder-Policy": "require-corp",
  "Content-Encoding": "gzip",
  "Transfer-Encoding": "chunked",
  "Trailer": "X-Checksum-SHA256",
  "X-Checksum-SHA256": "abcdef1234567890abcdef1234567890abcdef1234567890abcdef1234567890",
  "X-Cache": "HIT",
  "X-Cache-Hits": "1",
  "X-Cache-Expiry": "2021-06-23T00:00:00.000Z",
  "X-Cache-Key": "/api/v1/game/1/header/image",
  "X-Cache-Region": "us-east-1",
  "X-Cache-Node": "cache-node-01",
  "X-Cache-Version": "1.0",
  "X-Cache-Size": "1234",
  "X-Cache-Compression": "gzip",
  "X-Cache-Encoding": "utf-8",
  "X-Cache-Content-Type": "image/jpeg",
  "X-Cache-Content-Length": "1234",
  "X-Cache-Content-Encoding": "gzip
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?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:
media="http://search.yahoo.com/mrss/" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:gr="http://
www.google.com/ns/reader" xmlns:dc="http://purl.org/dc/elements/1.1
```

```
   return Graphics.CGImageToNSImage(FrameCGImage(Index, Size:Size * 2,
          UpdateBoard:UpdateBoard, Table:Table), Size:Size)
}

// get the board and pieces frame as a cgimage //
func FrameCGImage(_ Index:Int, Size:CGFloat, UpdateBoard:Bool, Table:Bool) ->
                CGImage
{
   // return the board and pieces frame as a cgimage //
   return GCUI.BPComp.Recomposite(Size, BPData:((GCUI.Attributes.Inf
          .GetCount() == Index) && !Table) ? BPData : GCUI.PGNMovesData
          .GetData(Index), InPositionEditor:Inf.Vals("InPositionEditor").VB,
          FENEnpassantSquare:GCUI.Attributes.Inf.Vals("FENEnpassantSquare")
          .VS, EnpassantSquare:EnpassantSquare, RankAndFile:MovingPiece
          .Location, UpdateBoard:UpdateBoard, Info:false)
}

// get a board preview image //
func GetPreview(_ Size:CGFloat) -> CGImage
{
   // return the board preview image //
   return FrameCGImage(GCUI.Attributes.Inf.GetCount(), Size:Size,
                     UpdateBoard:false, Table:false)
}

// update the board preview images //
func UpdatePreviews(_ RefreshBoard:Bool, Board:Bool, Diagram:Bool, Texture:
                  Bool)
{
   // refresh the board if need to //
   if RefreshBoard {Refresh("Board")}

   // update the board preview image if need to //
   if Board {GCUI.Objs["BoardPreview"]!.Preview = GetPreview(800)}

   // update the diagram preview image if need to //
   if Diagram
   {
      // update the diagram preview image //
      GCUI.Objs["DiagramPreview"]!.Preview = GetPreview(Inf
                                             .VF("IllustrationSize") * 2)
   }

   // update the textured window background preview if need to //
   if Texture {GCUI.Objs["WindowPreview"]!.Preview = WindowTexture.Preview}
}

// update the board preview images //
func UpdatePreviews(_ OnMain:Bool, RefreshBoard:Bool, Board:Bool, Diagram:
                  Bool, Texture:Bool)
{
   // if need to run on the main thread //
   if OnMain {DispatchQueue.main.async
   {
      // update the board preview images //
      self.UpdatePreviews(RefreshBoard, Board:Board, Diagram:Diagram,
                          Texture:Texture)
   }}

   // update the board preview images on the current thread if need to //
   if !OnMain
   {
      // update the board preview images //
      UpdatePreviews(RefreshBoard, Board:Board, Diagram:Diagram, Texture:
                     Texture)
   }
}

// update the capture and threaten highlights //
func UpdateCaptureThreatenHighlightLists()
{
   // set the board so can check for squares where pieces are threatened //
   BoardCopy(ThreatenedBoard, Board)

   // update the capture and threaten highlights //
   BPData.UpdateCaptureThreatLists(String(cString:
          BoardGetListOfCaptureSquares(Board)).components(separatedBy:"*"),
          ThreatSquares:String(cString:
          BoardGetListOfThreatenedSquares(ThreatenedBoard))
          .components(separatedBy:"*"))
}

// update the illustration data //
func UpdateIllustrationData(_ Index:Int)
{
   // update the illustration data //
   GCUI.PGNMovesData.Update(Index, BPData:BPData)
}

// refresh the graphics //
func Refresh(_ Tasks:String)
{
   // if need to replay the moves //
   if Tasks.contains("Replay") || Tasks.contains("All") {ReplayMoves()}

   // if need to recomposite the board and refresh the view //
   if Tasks.contains("Board") {RefreshView(true)}

   // if just need to refresh the view //
   if Tasks.contains("View") || Tasks.contains("All") {RefreshView(false)}

   // if need to refresh all the table views //
   if Tasks.contains("Tables") || Tasks.contains("All") {RefreshTableViews()}
}

// refresh the main view //
func RefreshView(_ UpdateBoard:Bool)
{
   // set the main view image //
   MainViewImage = FrameCGImage(GCUI.Attributes.Inf.GetCount(), Size:1320,
                                UpdateBoard:UpdateBoard, Table:false)

   // update the sketch if diagramming //
   if Inf.Vals("Diagramming").VB
   {
      // set the sketch background to be the main view image //
      Sketch.SetBackground(MainViewImage)

      // refresh the sketch //
      Sketch.Refresh(false)
   }

   // update the main view if not diagramming //
   if !Inf.Vals("Diagramming").VB {display(bounds)}
}
}

// get the application value //
let ReturnValue:Int32 = NSApplicationMain(CommandLine.argc, CommandLine
                                          .unsafeArgv)
```