Intro

About

AthenaEnv is a project that seeks to facilitate and at the same time brings a complete kit for users to create homebrew software for PlayStation 2 using the JavaScript language. It has dozens of built-in Methods, both for creating games and apps.

The main advantage over using AthenaEnv project instead of the pure PS2SDK is above all the practicality, you will use one of the simplest possible languages to create what you have in mind, besides not having to compile, just script and test, fast and simple.

Prerequisites

Using AthenaEnv you only need one way to code and one way to test your code, that is, if you want, you can even create your code on PS2, but I'll leave some recommendations below.

PC

- Visual Studio Code (with JavaScript extension)
- PCSX2 (1.7.0 or above) or PS2Client for test.
- Enable HostFS on PCSX2

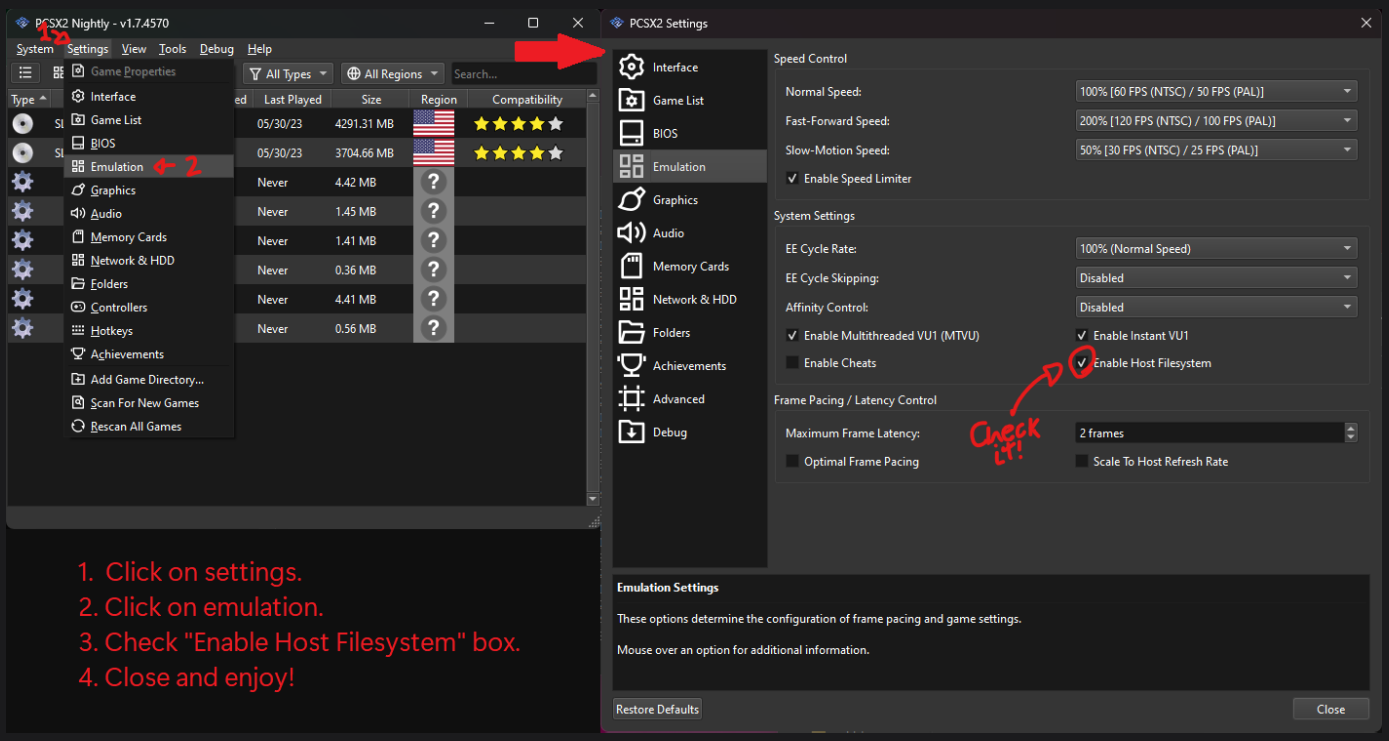
Android

- QuickEdit
- PS2 with wLE for test.

Oh, and I also have to mention that an essential prerequisite for using AthenaEnv is knowing how to code in JavaScript.

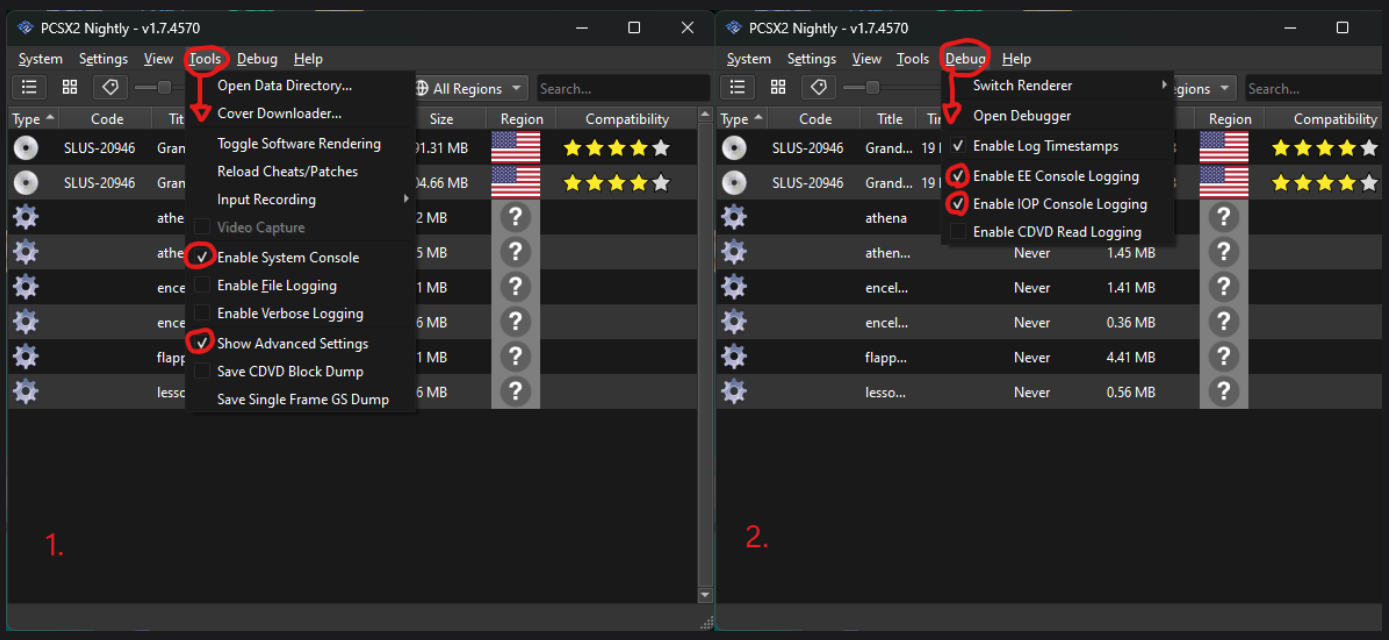
PCSX2

Qt version

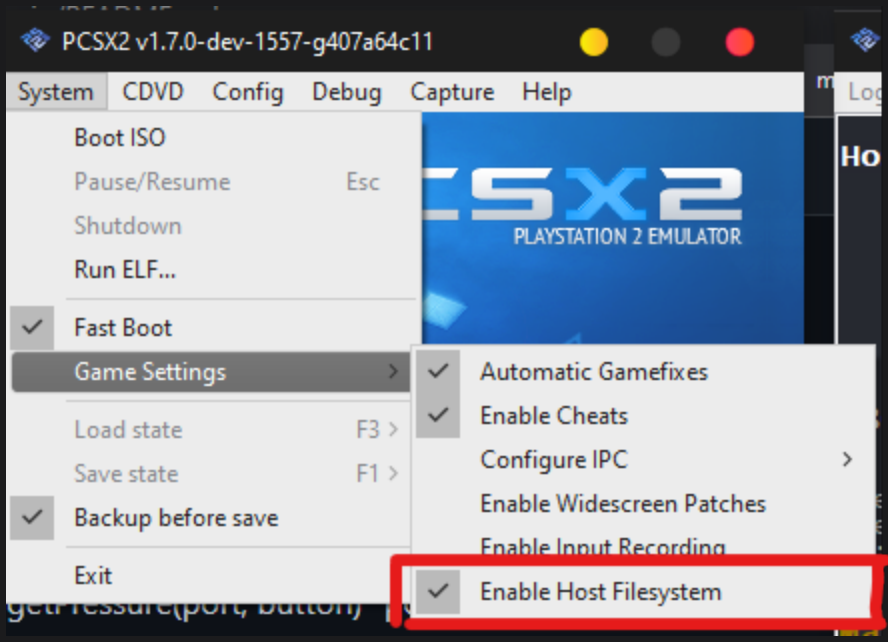


1. Click on settings.
2. Click on emulation.
3. Check "Enable Host Filesystem" box.
4. Close and enjoy!

> **TIP**
>
> Enable console output



(old) WxWidgets version



Getting Started

```
const font = new Font("default");

os.setInterval(() => { // Basically creates an infinite loop, similar
to while true(you can use it too).
  Screen.clear(); // Clear screen for the next frame.
  font.print(0, 0, "Hello World!"); // x, y, text
  Screen.flip(); // Updates the screen.
}, 0)
```

See more examples at AthenaEnv samples.

How to run it

Athena is basically a JavaScript loader, so it loads .js files. It runs "main.js" by default, but you can run other file names by passing it as the first argument when launching the ELF file.

Demo

If you try to just download it on releases tab from here and run it, that's what you will see:

That's the default dashboard, coded in default main.js file. It searchs JavaScript files with the first line containing the following structure:

```
// {"name": "App name", "author": "Who did it?", "version": "04012023",
"icon": "app_icon.png", "file": "my_app.js"}
// Now you can freely code below:
```

Once it was found, it will appear on the dashboard app list.

Features

Also it uses a slightly modified version of the QuickJS interpreter for JavaScript language, which means that it brings almost modern JavaScript features so far.

Float32

This project introduces a (old)new data type for JavaScript: single floats. Despite being less accurate than the classic doubles for number semantics, they are important for performance on the PS2, as the console only processes 32-bit floats on its FPU.

You can write single floats on AthenaEnv following the syntax below:

```
let test_float = 15.0f; // The 'f' suffix makes QuickJS recognizes it as a single float.
```

or

```
let test_float = Math.fround(15.0); // Math.fround returns real single floats on Athena.
```

Screen

The entire screen of your project (2D and 3D), being able to change the resolution, enable or disable parameters.

Methods

display

Makes the specified function behave like a main loop, when you don't need to clear or flip the screen because it's done automatically.

```
Screen.display(func);
```

clearColor

Sets a constant clear color for Screen.display function.

```
Screen.clearColor(color?);
```

clear

Clears screen with the specified color. If you don't specify any argument, it will use black as default.

```
Screen.clear(color?);
```

flip

Run the render queue and jump to the next frame, i.e.: Updates your screen.

```
Screen.flip();
```

getFreeVRAM

Returns the total of free Video Memory.

```
const freevram = Screen.getFreeVRAM();
```

setVSync

Toggles VSync, which makes the framerate stable in 15, 30, 60(depending on the mode) on screen.

```
Screen.setVSync(bool);
```

setFrameCounter

Toggles frame counting and FPS collecting.

```
Screen.setFrameCounter(bool);
```

waitVblankStart

Waits for a vertical sync.

```
Screen.waitVblankStart();
```

getFPS

Get Frames per second measure within the specified frame_interval in msec. Dependant on

Screen.setFrameCounter(true) to work.

```
const fps = Screen.getFPS(frame_interval);
```

getMode

Get actual video mode parameters. Returns an object.

```
const canvas = Screen.getMode();
```

Properties

- **canvas.width**: Screen width. Default: 640.
- **canvas.height**: Screen height. Default: 448 on NTSC consoles, 512 on PAL consoles.
- **canvas.psm**: Color mode. Available colormodes: CT16, CT16S, CT24, CT32.
- **canvas.interlace**: Available interlaces: INTERLACED, PROGRESSIVE.
- **canvas.field**: Available fields: FIELD, FRAME.
- **canvas.double_buffering**: Enable or disable double buffering(bool).
- **canvas.zbuffering**: Enable or disable Z buffering (3D buffering)(bool).
- **canvas.psmz**: ZBuffering color mode. Available zbuffer colormodes: Z16, Z16S, Z24, Z32.

setMode

Set the current video mode, get an video mode object as an argument.

```
Screen.setMode(canvas);
```

Font

Methods that control the texts that appear on the screen, loading texts, drawing and unloading from memory.

Construction

```
const font = new Font(path);
```

```
const osdFont = new Font();              // Load BIOS font, not
available for all console models
const font = new Font("Segoe UI.ttf");  // Load trueType font
```

INFO

- **path**: specify the location to a font file, E.g.: "images/atlas.png", "fonts/font.png".
- **formats**: png, bmp, jpg, otf and ttf.

Properties

- **color**: Define font tinting, default value is Color.new(255, 255, 255, 128).
- **scale**: Proportional scale, default: 1.0f

Methods

print

Draw text on screen (call it every frame).

```
print(x, y, text);
```

```
font.print(10.0, 10.0, "Hello world!");  // Example
```

getTextSize

Returns text absolute size in pixels (width, height).

```
getTextSize(text);
```

```
const size = font.getTextSize("Hello world!");  // Example
```

Draw

Shape drawing, triangles, circles etc.

Methods

point

Draws a pixel on the specified color and position on the screen.

```
Draw.point(x, y, color);
```

rect

Draws a rectangle on the specified color, position and size on the screen.

```
Draw.rect(x, y, width, height, color);
```

line

Draws a line on the specified colors and position on the screen.

```
Draw.line(x, y, x2, y2, color);
```

circle

Draws a circle on the specified color, position, radius and fill on the screen.

```
Draw.circle(x, y, radius, color, filled?);
```

Draws a triangle on the specified points positions and colors on the screen.

```
Draw.triangle(x, y, x2, y2, x3, y3, color, color2?, color3?);
```

quad

Draws a quad on the specified points positions and colors on the screen.

```
Draw.quad(x, y, x2, y2, x3, y3, x4, y4, color, color2?, color3?,
color4?);
```

Color

Methods

new

Returns a color object from the specified RGB(A) parameters.

```
const col = Color.new(r, g, b, a?);
```

getR

Get red intensity of the color.

```
const r = Color.getR(col);
```

getG

Get green intensity of the color.

```
const g = Color.getG(col);
```

getB

Get blue intensity of the color.

```
const b = Color.getB(col);
```

getA

Get alpha intensity of the color.

```
const a = Color.getA(col);
```

setR

Set red intensity of the color.

```
Color.setR(col, r);
```

setG

Set green intensity of the color.

```
Color.setG(col, g);
```

setB

Set blue intensity of the color.

```
Color.setB(col, b);
```

setA

Set alpha intensity of the color.

```
Color.setA(col, a);
```

Image

Image drawing.

Construction

```
const image = new Image(path, mode?, async_list?);
```

```
const imageVRAM = new Image("owl.png", VRAM);
```

> **INFO**
>
> - **path:** Path to the file, E.g.: "images/test.png".
> - **mode:** Choose between storing the image between **RAM** or **VRAM**, default value is RAM.
> - **async_list:** Gets a ImageList object, which is a asynchronous image loading list, if you want to load images in the background.

Properties

- **width, height**: Image drawing size, default value is the original image size.
- **startx, starty**: Beginning of the area that will be drawn from the image, the default value is 0.0.
- **endx, endy**: End of the area that will be drawn from the image, the default value is the original image size.
- **angle**: Define image rotation angle, default value is 0.0.
- **color**: Define image tinting, default value is Color.new(255, 255, 255, 128).
- **filter**: Choose between **LINEAR** or **NEAREST**, default value is NEAREST.

- **size**: Returns image real size occupied in memory.
- **bpp**: Returns image bits per-pixel qantity.
- **delayed**: If true, your texture was loaded in RAM, else, VRAM.
- **pixels**: The image pixel ArrayBuffer.
- **palette**: If is a palette image, it has a palette ArrayBuffer right here.

Methods

draw

Draw loaded image onscreen(call it every frame). Example: image.draw(15.0, 100.0);

```
draw(x, y);
```

optimize

If your image has 24 bits per-pixel (aka RGB), you can use this to make it 16 bits per-pixel, saving some memory!

```
optimize();
```

ready

Returns true if an asynchronous image was successfully loaded in memory.

```
ready();
```

```
const loaded = image.ready();
```

ImageList

Load and manage multiple images while your code is running, multithreaded loading!

Construction

This constructor creates a new thread and a queue to load images in background, avoid building multiple ImageList objects.

```
const async_list = new ImageList();
```

Methods

process

This method starts the thread and loads added images on the queue.

```
async_list.process();
```

Sound

Sound Methods, supporting WAV, OGG and ADPCM.

Methods

setVolume

```
Sound.setVolume(volume, slot?);
```

*If slot is specified, it will change ADPCM slot volume, else it will change master volume.

load

```
const audio = Sound.load(path);
```

play

```
Sound.play(audio, slot?);
```

*ADPCM: If slot isn't specified, it will use 0.

free

```
Sound.free(audio);
```

isPlaying

```
const playing = Sound.isPlaying();
```

*Doesn't apply for ADPCM

duration

```
const msec = Sound.duration();
```

repeat

```
Sound.repeat(false);
```

*Doesn't apply for ADPCM

pause

```
Sound.pause(audio);
```

*Doesn't apply for ADPCM

resume

```
Sound.resume(audio);
```

*Doesn't apply for ADPCM

deinit

```
Sound.deinit();
```

Timer

Control the time precisely in your code, it contains several timing Methods.

Methods

new

```
const timer = Timer.new();
```

getTime

```
Timer.getTime(timer);
```

setTime

```
Timer.setTime(src, value);
```

destroy

```
Timer.destroy(timer);
```

pause

```
Timer.pause(timer);
```

resume

```
Timer.resume(timer);
```

reset

```
Timer.reset(timer);
```

isPlaying

```
Timer.isPlaying(timer);
```

Edit this page

OS

The os module provides Operating System specific functions:

- low level file access
- signals
- timers
- asynchronous I/O

> **NOTE**
>
> The OS Methods usually return 0 if OK or an OS specific negative error code.

Properties

- **os.platform**: Return a string representing the platform: "linux", "darwin", "win32", "ps2" or "js".

Methods

open

Open a file. Return a handle or < 0 if error.

```
let fd = os.open(filename, flags, mode = 0o666)
```

Flags

- os.O_RDONLY
- os.O_WRONLY

- os.O_RDWR
- os.O_APPEND
- os.O_CREAT
- os.O_EXCL
- os.O_TRUNC

> **INFO**
>
> POSIX open flags.

close

Close the file handle fd.

```
os.close(fd);
```

seek

Seek in the file. Use std.SEEK_* for whence. offset is either a number or a bigint. If offset is a bigint, a bigint is returned too.

```
os.seek(fd, offset, whence);
```

read

Read length bytes from the file handle fd to the ArrayBuffer buffer at byte position offset. Return the number of read bytes or < 0 if error.

```
os.read(fd, buffer, offset, length);
```

write

Write length bytes to the file handle fd from the ArrayBuffer buffer at byte position offset.

Return the number of written bytes or < 0 if error.

```
os.write(fd, buffer, offset, length);
```

remove

Remove a file. Return 0 if OK or -errno.

```
os.remove(filename);
```

rename

Rename a file. Return 0 if OK or -errno.

```
os.rename(oldname, newname);
```

realpath

Return [str, err] where str is the canonicalized absolute pathname of path and err the error code.

```
os.realpath(path);
```

getcwd

Return [str, err] where str is the current working directory and err the error code.

```
os.getcwd();
```

chdir

Change the current directory. Return 0 if OK or -errno.

```
os.chdir(path);
```

mkdir

Create a directory at path. Return 0 if OK or -errno.

```
os.mkdir(path, mode = 0o777);
```

stat

Return [obj, err] where obj is an object containing the file status of path. err is the error code.

The following fields are defined in obj: dev, ino, mode, nlink, uid, gid, rdev, size, blocks, atime, mtime, ctime.

The times are specified in milliseconds since 1970.

```
os.stat(path);
```

Constants to interpret the mode property returned by stat(). They have the same value as in the C system header sys/stat.h.

- os.S_IFMT
- os.S_IFIFO
- os.S_IFCHR
- os.S_IFDIR
- os.S_IFBLK
- os.S_IFREG
- os.S_IFSOCK
- os.S_IFLNK
- os.S_ISGID
- os.S_ISUID

lstat

lstat() is the same as stat() excepts that it returns information about the link itself.

```
os.lstat(path);
```

utimes

Change the access and modification times of the file path. The times are specified in milliseconds since 1970. Return 0 if OK or -errno.

```
os.utimes(path, atime, mtime);
```

readdir

Return [array, err] where array is an array of strings containing the filenames of the directory path. err is the error code.

```
os.readdir(path);
```

setReadHandler

Add a read handler to the file handle fd. func is called each time there is data pending for fd. A single read handler per file handle is supported. Use func = null to remove the handler.

```
os.setReadHandler(fd, func);
```

setWriteHandler

Add a write handler to the file handle fd. func is called each time data can be written to fd. A single write handler per file handle is supported. Use func = null to remove the handler.

```
os.setWriteHandler(fd, func);
```

Sleep during delay_ms milliseconds.

```
os.sleep(delay_ms);
```

setTimeout

Call the function func after delay ms. Return a handle to the timer.

```
os.setTimeout(func, delay);
```

setInterval

Call the function func at specified intervals (in milliseconds). Return a handle to the timer.

```
os.setInterval(func, interval);
```

setImmediate

Executes a given function immediately.

```
os.setImmediate(func);
```

clearTimeout

Cancel a timer.

```
os.clearTimeout(handle);
```

clearInterval

Cancel a interval.

```
os.clearInterval(handle);
```

clearImmediate

Cancel a immediate execution.

```
os.clearImmediate(handle);
```

Edit this page

System

Files, folders and system stuff.

Methods

listDir

```
const listdir = System.listDir(path?);
```

- **listdir[index].name**: return file name on indicated index(string)
- **listdir[index].size**: return file size on indicated index(integer)
- **listdir[index].directory**: return if indicated index is a file or a directory(bool)

removeDirectory

```
System.removeDirectory(path);
```

copyFile

```
System.copyFile(source, dest);
```

moveFile

```
System.moveFile(source, dest);
```

rename

```
System.rename(source, dest);
```

sleep

```
System.sleep(sec);
```

exitToBrowser

```
System.exitToBrowser();
```

setDarkMode

```
System.setDarkMode(value);
```

getTemperature

```
let temps = System.getTemperature();
```

> **INFO**
>
> It only works with SCPH-500XX and later models.

getMCInfo

```
const info = System.getMCInfo(slot);
```

- info.type
- info.freemem
- info.format

getCPUInfo

```
const ee_info = System.getCPUInfo();
```

- ee_info.implementation
- ee_info.revision
- ee_info.FPUimplementation
- ee_info.FPUrevision
- ee_info.ICacheSize
- ee_info.DCacheSize
- ee_info.RAMSize
- ee_info.MachineSize

getGPUInfo

```
const gs_info = System.getGPUInfo();
```

- gs_info.id
- gs_info.revision

getMemoryStats

```
const ram_usage = System.getMemoryStats();
```

- **ram_usage.core**: Kernel + Native code size in RAM
- **ram_usage.nativeStack**: Kernel + Native stack size
- **ram_usage.allocs**: Dynamic allocated memory tracking
- **ram_usage.used**: All above, but combined

Asynchronous Methods

threadCopyFile

```
System.threadCopyFile(source, dest);
```

getFileProgress

```
const progress = System.getFileProgress();
```

- progress.current
- progress.final

Archive

A simple compressed file extractor and manager.

Methods

open

```
const zip = Archive.open(fname);
```

list

```
const list = Archive.list(zip);
```

extractAll

```
Archive.extractAll(zip);
```

close

```
Archive.close(zip);
```

untar

```
Archive.untar(fname);
```

Edit this page

STD

The std module provides wrappers to the libc stdlib.h and stdio.h and a few other utilities.

Wrappers to the libc file stdin, stdout, stderr:

- std.in
- std.out
- std.err

Enumeration

object containing the integer value of common errors (additional error codes may be defined)

- std.EINVAL
- std.EIO
- std.EACCES
- std.EEXIST
- std.ENOSPC
- std.ENOSYS
- std.EBUSY
- std.ENOENT
- std.EPERM
- std.EPIPE

Methods

evalScript

Evaluate the string str as a script (global eval).

```
std.evalScript(str, options = undefined);
```

options is an optional object containing the following optional properties:

- std.backtrace_barrier - Boolean (default = false). If true, error backtraces do not list the stack frames below the evalScript.

loadScript

Evaluate the file filename as a script (global eval).

```
std.loadScript(filename);
```

exists

Returns a bool that determines whether the file exists or not.

```
const hasfile = std.exists(filename);
```

loadFile

Load the file filename and return it as a string assuming UTF-8 encoding. Return null in case of I/O error.

```
const fstr = std.loadFile(filename);
```

open

Open a file (wrapper to the libc fopen()). Return the FILE object or null in case of I/O error. If errorObj is not undefined, set its errno property to the error code or to 0 if no error occured.

```
const file = std.open(filename, flags, errorObj = undefined);
```

fdopen

Open a file from a file handle (wrapper to the libc fdopen()). Return the FILE object or null in case of I/O error. If errorObj is not undefined, set its errno property to the error code or to 0 if no error occured.

```
std.fdopen(fd, flags, errorObj = undefined);
```

tmpfile

Open a temporary file. Return the FILE object or null in case of I/O error. If errorObj is not undefined, set its errno property to the error code or to 0 if no error occured.

```
std.tmpfile(errorObj = undefined);
```

puts

Equivalent to std.out.puts(str).

```
std.puts(str);
```

printf

Equivalent to std.out.printf(fmt, ...args).

```
std.printf(fmt, ...args);
```

sprintf

Equivalent to the libc sprintf().

```
std.sprintf(fmt, ...args);
```

strerror

Return a string that describes the error errno.

```
std.strerror(errno);
```

gc

Manually invoke the cycle removal algorithm. The cycle removal algorithm is automatically started when needed, so this function is useful in case of specific memory constraints or for testing.

```
std.gc();
```

parseExtJSON

Parse str using a superset of JSON.parse.

```
std.parseExtJSON(str);
```

The following extensions are accepted:

- Single line and multiline comments
- unquoted properties (ASCII-only Javascript identifiers)
- trailing comma in array and object definitions
- single quoted strings
- \f and \v are accepted as space characters
- leading plus in numbers

- octal (0o prefix) and hexadecimal (0x prefix) numbers

FILE

open

```
const file = std.open(filename, flags);
```

```
const file = std.open("test.txt", "w");
```

> **INFO**
>
> - filename - Path to the file, E.g.: "samples/test.txt".
> - flags - File mode, E.g.: "w", "r", "wb", "rb", etc.

close

Close the file. Return 0 if OK or -errno in case of I/O error.

```
close();
```

puts

Outputs the string with the UTF-8 encoding.

```
puts(str);
```

printf

Formatted printf.

```
printf(fmt, ...args);
```

> The same formats as the standard C library printf are supported. Integer format types (e.g. %d) truncate the Numbers or BigInts to 32 bits. Use the l modifier (e.g. %ld) to truncate to 64 bits.

flush

Flush the buffered file.

```
flush();
```

seek

Seek to a give file position (whence is std.SEEK_*). offset can be a number or a bigint. Return 0 if OK or -errno in case of I/O error.

```
seek(offset, whence);
```

Constants:

- std.SEEK_SET
- std.SEEK_CUR
- std.SEEK_END

tell

Return the current file position.

```
tell();
```

tello

Return the current file position as a bigint.

```
tello();
```

eof

Return true if end of file.

```
eof();
```

fileno

Return the associated OS handle.

```
fileno();
```

error

Return true if there was an error.

```
error();
```

clearerr

Clear the error indication.

```
clearerr();
```

read

Read length bytes from the file to the ArrayBuffer buffer at byte position position (wr
to the libc fread).

```
read(buffer, position, length);
```

write

Write length bytes to the file from the ArrayBuffer buffer at byte position position (wrapper to the libc fwrite).

```
write(buffer, position, length);
```

getline

Return the next line from the file, assuming UTF-8 encoding, excluding the trailing line feed.

```
getline();
```

readAsString

Read max_size bytes from the file and return them as a string assuming UTF-8 encoding. If max_size is not present, the file is read up its end.

```
readAsString(max_size = undefined);
```

getByte

Return the next byte from the file. Return -1 if the end of file is reached.

```
getByte();
```

putByte

Write one byte to the file.

```
putByte(c);
```

Pads

Above being able to draw and everything else, A human interface is important. Supports rumble and pressure sensitivity.

Buttons List

Type	Buttons
Directional	`Pads.UP`, `Pads.DOWN`, `Pads.LEFT`, `Pads.RIGHT`
Action	`Pads.TRIANGLE`, `Pads.CIRCLE`, `Pads.CROSS`, `Pads.SQUARE`
System	`Pads.SELECT`, `Pads.START`
Shoulder	`Pads.L1`, `Pads.R1`, `Pads.L2`, `Pads.R2`
Stick	`Pads.L3`, `Pads.R3`

Methods

get

Returns a pad object

```
const pad = Pads.get(port?);
```

Properties

- **pad.btns**: Button state on the current check.
- **pad.old_btns**: Button state on the last check.
- **pad.lx**: Left analog horizontal position (left = -127, default = 0, right = 128).
- **pad.ly**: Left analog vertical position (up = -127, default = 0, down = 128).
- **pad.rx**: Right analog horizontal position (left = -127, default = 0, right = 128).
- **pad.ry**: Right analog vertical position (up = -127, default = 0, down = 128).

update

Updates all pads pressed and stick positions data.

```
update();
```

pressed

Checks if a button is being pressed (continuously).

```
pressed(button);
```

justPressed

Checks if a button was pressed only once.

```
justPressed(button);
```

Sets the pad object to listen events defined by Pads.newEvent, so it doesn't need to be updated.

```
setEventHandler();
```

newEvent

Creates an asynchronous pad event, returns the event id.

```
const event_id = Pads.newEvent(button, kind, function);
```

> **WARNING**
>
> Remember to set the pad object event handler first!

> **PAD EVENTS**
>
> - Pads.PRESSED
> - Pads.JUST_PRESSED
> - Pads.NON_PRESSED

deleteEvent

Deletes the event created by Pads.newEvent.

```
Pads.deleteEvent(event_id);
```

getType

Gets gamepad type in the specified port.

```
const type = Pads.getType(port?);
```

> **PAD TYPES**
>
> - Pads.DIGITAL
> - Pads.ANALOG
> - Pads.DUALSHOCK

getPressure

Get button pressure level.

```
const press = Pads.getPressure(port?, button);
```

rumble

Rumble your gamepad.

```
Pads.rumble(port, big, small);
```

Keyboard

Basic USB keyboard support.

Methods

init

Initialize keyboard routines.

```
Keyboard.init();
```

get

Get keyboard current char.

```
const c = Keyboard.get();
```

setRepeatRate

Set keyboard repeat rate.

```
Keyboard.setRepeatRate(msec);
```

setBlockingMode

Sets keyboard to block (or not) the thread waiting for the next key to be pressed.

```
Keyboard.setBlockingMode(mode);
```

Destroy keyboard routines.

```
Keyboard.deinit();
```

Edit this page

Mouse

Basic USB mouse support.

Methods

init

Initialize mouse routines.

```
Mouse.init();
```

get

```
const mouse = Mouse.get();
```

Returns mouse actual properties on the object format below:

- mouse.x
- mouse.y
- mouse.wheel
- mouse.buttons

setBoundary

Set mouse x and y bounds.

```
Mouse.setBoundary(minX, maxX, minY, maxY);
```

getMode

Get mouse mode (absolute or relative).

```
const mode = Mouse.getMode();
```

setMode

Set mouse mode.

```
Mouse.setMode(mode);
```

getAccel

Get mouse acceleration.

```
const accel = Mouse.getAccel();
```

setAccel

Set mouse acceleration.

```
Mouse.setAccel(val);
```

setPosition

Set mouse pointer position.

```
Mouse.setPosition(x, y);
```

IOP

The PlayStation 2 has an I/O processor to deal with drivers and modules. Take control of it!

Properties

- **IOP.keyboard**: USB Keyboard
- **IOP.mouse**: USB Mouse
- **IOP.freeram**: IOP RAM Info
- **IOP.ds34bt**: Bluetooth DualShock 3/4 pads
- **IOP.ds34usb**: USB DualShock 3/4 pads
- **IOP.network**: Network drivers
- **IOP.pads**: DualShock 1/2 pads
- **IOP.memcard**: Memory Card
- **IOP.audio**: Audio driver
- **IOP.usb_mass**: USB Mass storage, supports FAT32 and exFAT
- **IOP.cdfs**: Disc driver
- **IOP.hdd**: Internal HDD driver
- **IOP.boot_device**: Storage device used to boot Athena

Methods

loadModule

```
const result = IOP.loadModule(fname, arg_len?, args?);
```

loadModuleBuffer

```
const result = IOP.loadModuleBuffer(mod_buf, arg_len?, args?);
```

loadDefaultModule

```
IOP.loadDefaultModule(mod_id);
```

reset

```
IOP.reset();
```

getMemoryStats

```
const stats = IOP.getMemoryStats();
```

> ⚠ **WARNING**
>
> Requires **IOP.loadDefaultModule(IOP.freeram)** first!

Properties

- stats.free
- stats.used

Render

Basic 3D support powered by a VU1 renderer.

Methods

setMode

```
const canvas = Screen.getMode();
canvas.zbuffering = true;
canvas.psmz = Z16S;

Screen.setMode(canvas);
```

> **INFO**
>
> - Remember to enable zbuffering on screen mode
> - Default NTSC mode(3D enabled)

setView

Initializes rendering routines.

```
Render.setView(aspect, fov?);
```

> **INFO**
>
> - default aspect is 4/3, widescreen is 16/9.
> - default fov: 0.2

vertex

Returns a vertex to build a 3D mesh. It should be used to create vertex arrays.

```
Render.vertex(x, y, z, n1, n2, n3, s, t, r, g, b, a);
```

- **x, y, z**: Vertex position on 3D world.
- **n1, n2, n3**: Vertex normal.
- **s, t**: Vertex texture coordinates.
- **r, g, b, a**: Vertex color.

Camera

Methods

position

```
Camera.position(x, y, z);
```

rotation

```
Camera.rotation(x, y, z);
```

Edit this page

RenderObject

Construction

```
const model = new RenderObject(mesh, texture?);
```

> **INFO**
>
> - Load simple WaveFront OBJ files or vertex arrays.
> - MTL is supported on OBJs (including per-vertex colors and multi-texturing).
> - If you don't have a MTL file but you want to bind a texture on it, just pass the image as a second argument if you want to use it.

Properties

- **vertices**: A Render.vertex array that can be modified and read.
- **size**: Vertex quantity.

Methods

draw

Draws the object on screen.

```
draw(pos_x, pos_y, pos_z, rot_x, rot_y, rot_z);
```

drawBounds

Draws object bounding box.

```
drawBounds(pos_x, pos_y, pos_z, rot_x, rot_y, rot_z);
```

getTexture

Gets the nth texture object from the model.

```
getTexture(id);
```

setTexture

Changes or sets the nth texture on models.

```
setTexture(id, texture, range?);
```

getPipeline

Returns the current rendering pipeline loaded for the model.

```
getPipeline();
```

setPipeline

Sets the current pipeline for the model.

```
setPipeline(pipeline);
```

Available pipelines

- **Render.PL_NO_LIGHTS_COLORS**: Colors and lights disabled.
- **Render.PL_NO_LIGHTS_COLORS_TEX**: Colors, lights and textures disabled.

- **Render.PL_NO_LIGHTS**: Lights disabled, colors still working.
- **Render.PL_NO_LIGHTS_TEX**: Textures and lights disabled, colors still working.
- **Render.PL_DEFAULT**: Default for textured models. Lights and colors enabled.
- **Render.PL_DEFAULT_NO_TEX**: Default for non-textured models. Lights and colors enabled.

Lights

You have 4 lights to use in 3D scenes, use **set** to configure them.

Atributes

- Lights.DIRECTION
- Lights.AMBIENT
- Lights.DIFFUSE

Methods

set

```
Lights.set(id, attribute, x, y, z);
```

Edit this page

Network

Net basics and web requests :D.

Methods

init

```
Network.init(ip?, netmask?, gateway?, dns?);
```

```
Network.init("192.168.0.10", "255.255.255.0", "192.168.0.1",
"192.168.0.1"); // Static mode
Network.init(); // DHCP Mode, dynamic.
```

getConfig

```
const conf = Network.getConfig();
```

Returns

- conf.ip
- conf.netmask
- conf.gateway
- conf.dns

deinit

Shutdown network.

```
Network.deinit();
```

WebSocket

Construction

```
new WebSocket(url);
```

```
const socket = new WebSocket("wss://example.com");
```

Methods

send

Send data with Buffer

```
send(data);
```

recv

Receive data to a buffer

```
recv();
```

Edit this page

Socket

Well, sockets.

Construction

```
const s = new Socket(domain, type);
```

```
const s = new Socket(AF_INET, SOCK_STREAM);
```

Methods

connect

```
connect(host, port);
```

bind

```
bind(host, port);
```

listen

```
listen();
```

send

Send data with Buffer

```
send(data);
```

recv

Receive data to a buffer

```
recv(size);
```

close

```
close();
```

Edit this page

Request

Construction

```
const r = new Request();
```

Properties

- **keepalive** (boolean)
- **useragent** (string)
- **userpwd** (string)
- **headers** (string[])

Methods

get

```
get(url);
```

head

```
head(url);
```

post

```
post(url, data);
```

```
download(url, fname);
```

Asynchronous methods

asyncGet

```
asyncGet(url);
```

asyncDownload

```
asyncDownload(url, fname);
```

ready

```
ready(timeout?, conn_timeout?);
```

getAsyncData

```
getAsyncData();
```

getAsyncSize

```
getAsyncSize();
```

Modules

list of usable functions of AthenaEnv project currently, this list is constantly being updated.

Parameters followed by "?" (question mark) are optional, example:

```
Color.new(r, g, b, a?);
```

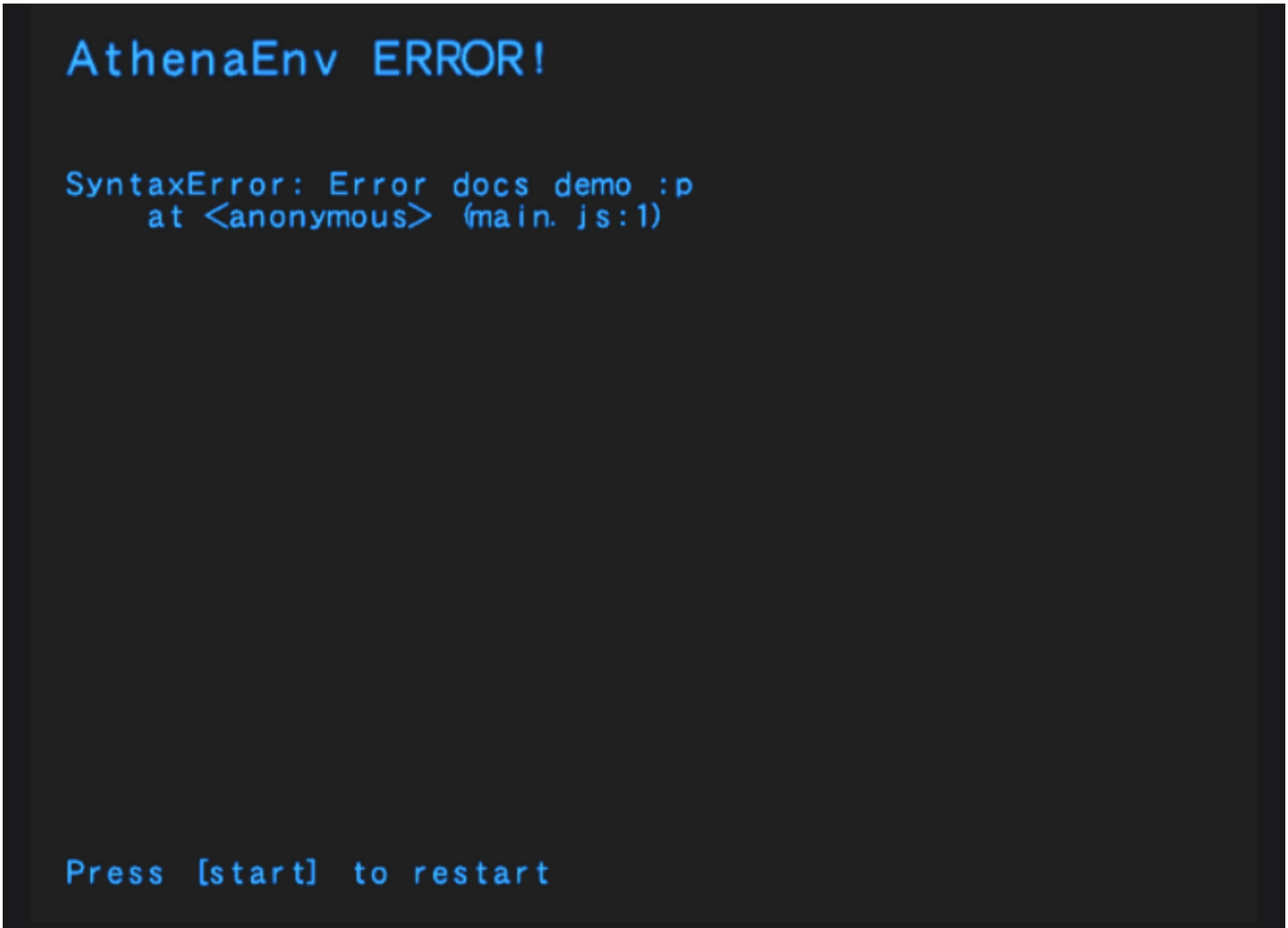
Edit this page

Error Report System

Athena has a consistent error system, which is capable of pointing the error type, custom message, files, lines and it even has a color code.

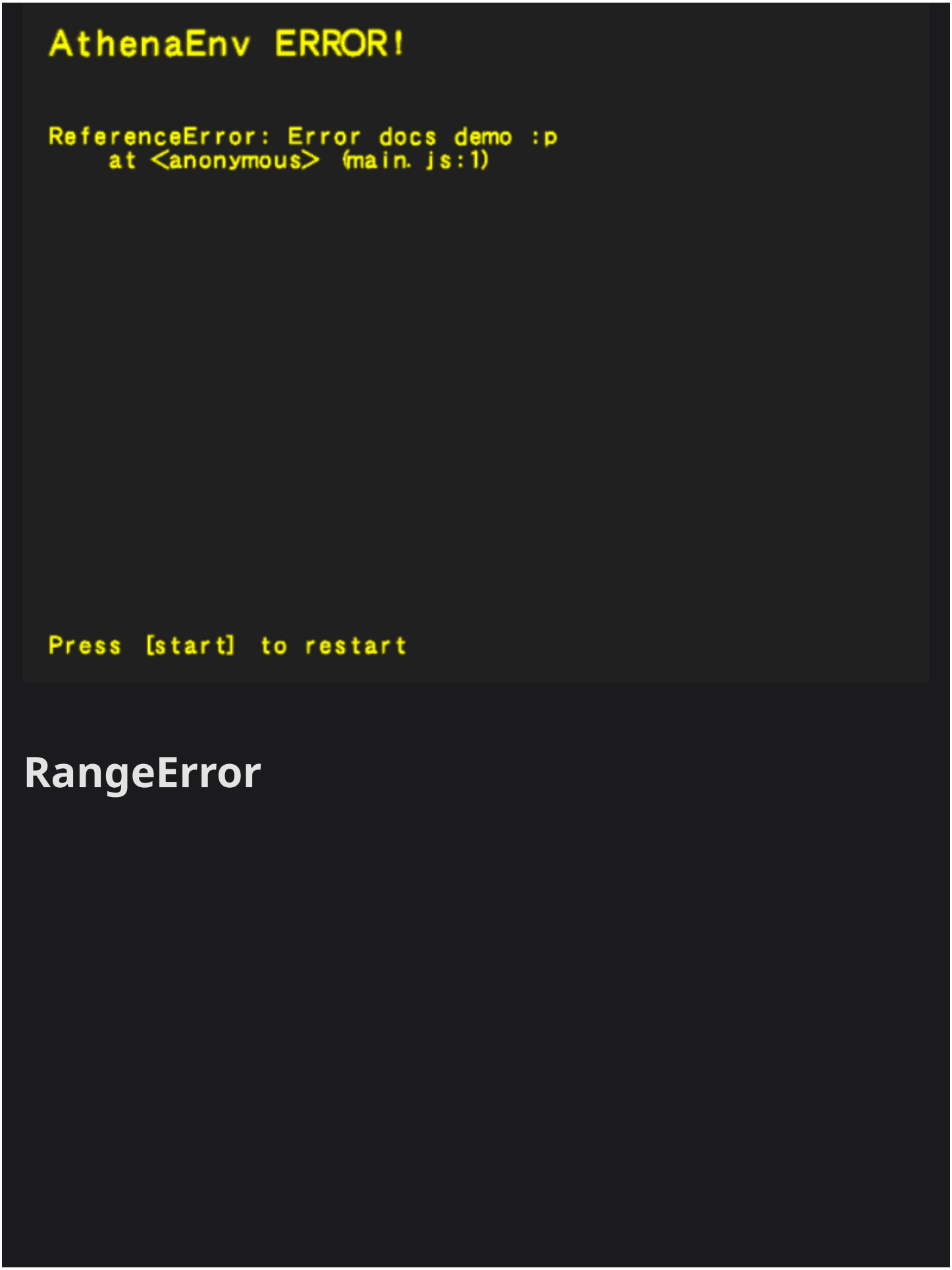
EvalError

SyntaxError


AthenaEnv ERROR!
SyntaxError: Error docs demo :p
at <anonymous> (main.js:1)
Press [start] to restart

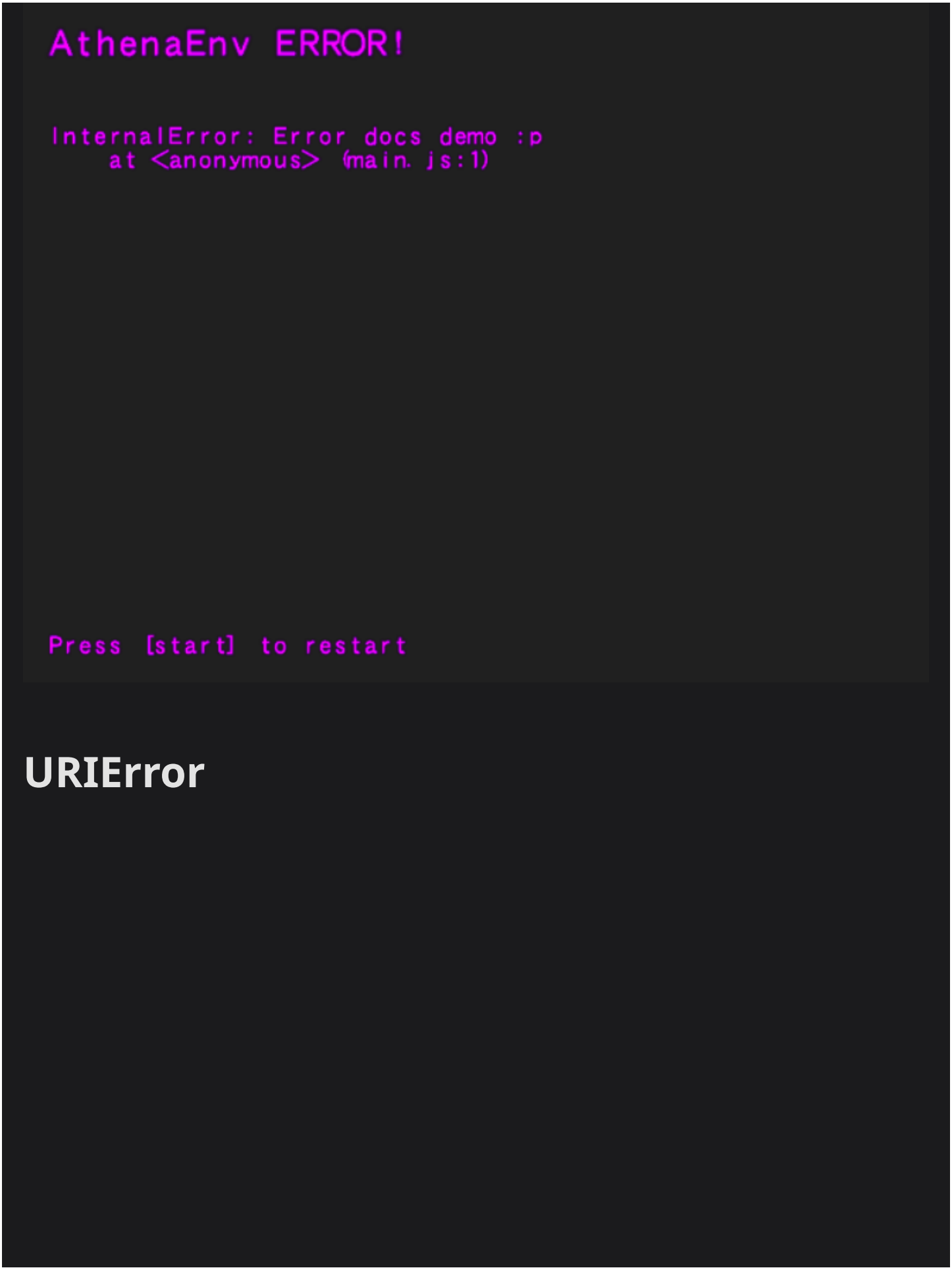
TypeError

ReferenceError

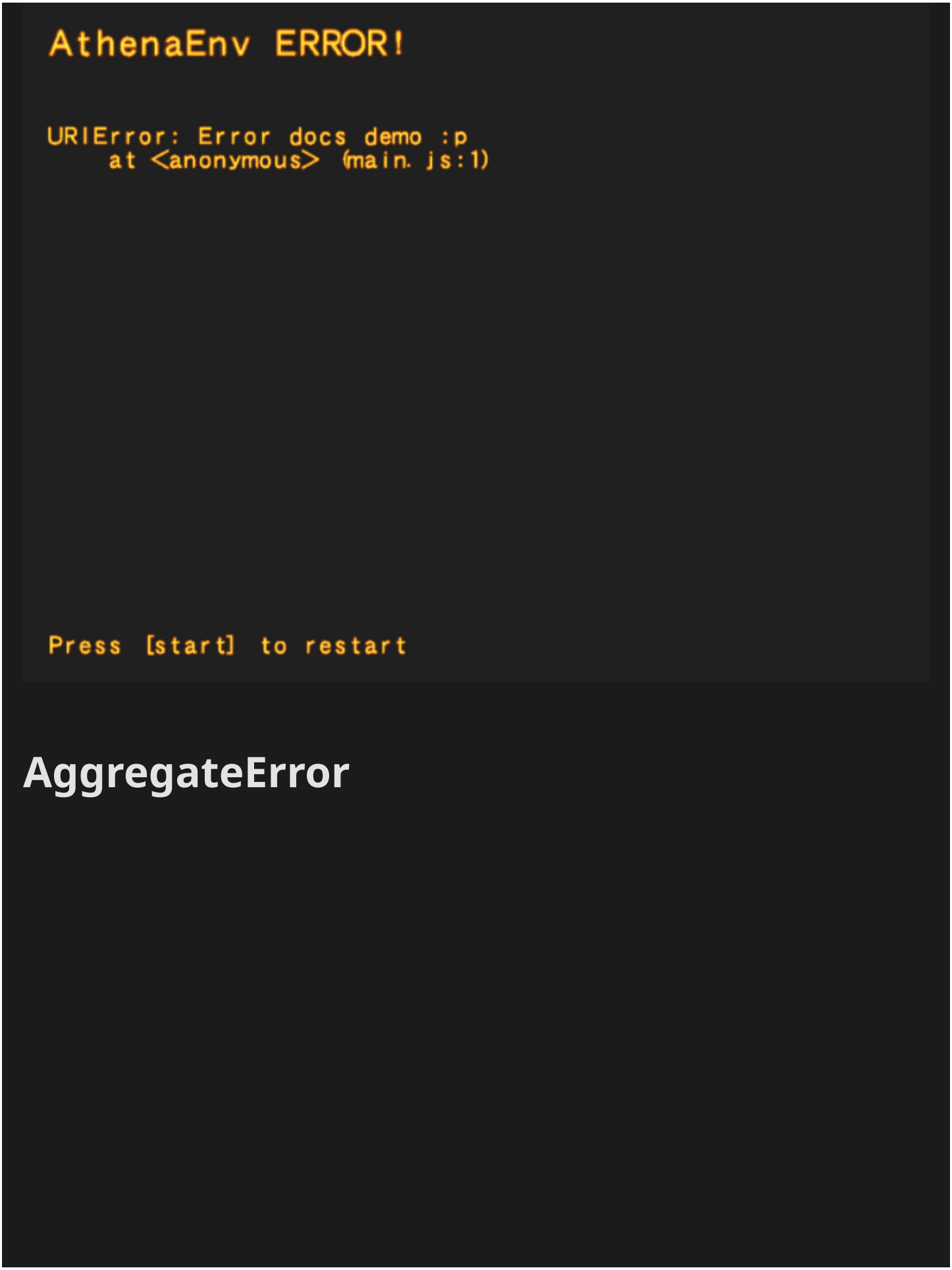

AthenaEnv ERROR!
ReferenceError: Error docs demo :p
at <anonymous> (main.js:1)
Press [start] to restart

RangeError

InternalError


AthenaEnv ERROR!
InternalError: Error docs demo :p
at <anonymous> (main.js:1)
Press [start] to restart

URIError



AggregateError

Edit this page

Contribute

Contributions are what make the open source community such an amazing place to be learn, inspire, and create. Any contributions you make are **greatly appreciated**.

AthenaEnv

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AwesomeFeature`)
3. Commit your Changes (`git commit -m 'Add some AwesomeFeature'`)
4. Push to the Branch (`git push origin feature/AwesomeFeature`)
5. Open a Pull Request

Website

See the instructions here

Edit this page

Community

List

A collection of resources built by the community.

Title	Author
athenaenv-pkgs	terremoth
athenaenv-starter-js	Wellinator
athenaenv-starter-ts	Wellinator
Hot Reload	Daniel Abrante
MercurySM-AthenaPS2	GustavoFurtad2
Modules-athena-Env	j0sedavi
Outline-in-Text-for-AthenaENV-PS2	PauloDevv

Edit this page