Scaling D for Real-World Projects: Doing it fastly

What is this talk about

- ⚡ Refactoring old code
- ⚡ Underrated optimizations
- ⚡ Reducing output binary size
- ⚡ API design optimization

My Background

- ⚡ Game Developer
- ⚡ Author of Redub - Hipreme Engine
- ⚡ Maintainer of Objective-C Bindings
- ⚡ Co-authors minimal D custom runtime
- ⚡ Specialist in Rendering & Porting

Identifying Improvements

- -profile=gc
- Overall stats with GC.stats
- Running code in weaker devices

Looking into the past

⚡ DConf 23': If I Cannot Dissuade You from Using Atomics, at least Do It Safely

- Roy Margalit

⚡ DConf 23': Stack Memory is Awesome!

- Dennis Korpel

⚡ DConf 24': Good Fun: Creating a Data-Oriented Parser/AST/Visitor Generator.

- Robert Schadek

Profiling Binary Size

Shallow Bytes	Shallow %	Item
201891	28.02%	"function names" subsection
102786	14.27%	data[0]
32842	4.56%	data[1]
19539	2.71%	_D3hip4font6bmfont13HipBitmapFont9loadAtlasMFAyaQdZb
8766	1.22%	_D4arsd3ttf21stbtt__run_charstringFNbNiNePSQBpQBn14stbtt_font
8105	1.12%	_D3hip4data4json9JSONValue5parseFAyaZSQBkQBjQBhQBf
5226	0.73%	_D4arsd3ttf23stbtt_InitFont_internalFNbNiNePSQBrQBp14stbtt_fo
5171	0.72%	_D4arsd3ttf28stbtt__fill_active_edges_newFNbNiNePfQciPSQCbQBz
3320	0.46%	_D4arsd3ttf22stbtt__GetGlyphShapeTTFNbNiNePSQBqQBo14stbtt_fon
3302	0.46%	elem[0]
2862	0.40%	_D3hip6assets7tilemap10HipTilemap16parseObjectLayerFKCQCa3api
2685	0.37%	_D3hip6assets7tilemap10HipTilemap13readTiledJSONFAyaxAhDFCQCe
2646	0.37%	_D3hip12assetmanager15HipAssetManager10initializeFZv
1748	0.24%	_D3hip7systems7gamepad10HipGamepad6__ctorMFCQBqQBpQBk15IHipGa
1727	0.24%	_D3hip4util15to_string_range__T13toStringRangeTSQBuQBt6string
1691	0.23%	_D3hip6assets12textureatlas15HipTextureAtlas8readJSONFxAhAyaQ
1635	0.23%	_D3hip11hiprenderer7backend2gl14defaultshaders22getSpriteBatc
1612	0.22%	_D3hip4util15to_string_range__T13toStringRangeTSQBuQBt6string
1581	0.22%	_D3hip6assets8inputmap11HipInputMap13parseInputMapFxAhAyahZ22
1508	0.21%	_D3hip4data3ini6HipINI14loadFromMemoryMFAyaQdZb

Example of optimized output for wasm binary

Twiggy

https://github.com/AlexEne/twiggy

Rust tool used for WASM binary size profiling

Profiling Binary Size

Shallow Bytes	Shallow %	Item
201891	28.02%	"function names" subsection
102786	14.27%	data[0]
32842	4.56%	data[1]
19539	2.71%	_D3hip4font6bmfont13HipBitmapFont9loadAtlasMFAyaQdZb
8766	1.22%	_D4arsd3ttf21stbtt__run_charstringFNbNiNePSQBpQBn14stbtt_font
8105	1.12%	_D3hip4data4json9JSONValue5parseFAyaZSQBkQBjQBhQBf
5226	0.73%	_D4arsd3ttf23stbtt_InitFont_internalFNbNiNePSQBrQBp14stbtt_fo
5171	0.72%	_D4arsd3ttf28stbtt__fill_active_edges_newFNbNiNePfQciPSQCbQBz
3320	0.46%	_D4arsd3ttf22stbtt__GetGlyphShapeTTFNbNiNePSQBqQBo14stbtt_fon
3302	0.46%	elem[0]
2862	0.40%	_D3hip6assets7tilemap10HipTilemap16parseObjectLayerFKCQCa3api
2685	0.37%	_D3hip6assets7tilemap10HipTilemap13readTiledJSONFAyaxAhDFCQCe
2646	0.37%	_D3hip12assetmanager15HipAssetManager10initializeFZv
1748	0.24%	_D3hip7systems7gamepad10HipGamepad6__ctorMFCQBqQBpQBk15IHipGa
1727	0.24%	_D3hip4util15to_string_range__T13toStringRangeTSQBuQBt6string
1691	0.23%	_D3hip6assets12textureatlas15HipTextureAtlas8readJSONFxAhAyaQ
1635	0.23%	_D3hip11hiprenderer7backend2gl14defaultshaders22getSpriteBatc
1612	0.22%	_D3hip4util15to_string_range__T13toStringRangeTSQBuQBt6string
1581	0.22%	_D3hip6assets8inputmap11HipInputMap13parseInputMapFxAhAyahZ22
1508	0.21%	_D3hip4data3ini6HipINI14loadFromMemoryMFAyaQdZb

Example of optimized output for wasm binary

Twiggy

https://github.com/AlexEne/twiggy

Rust tool used for WASM binary size profiling

Reducing Binary Size

```
case Context.chars:
{
    //Advance "count"
    charactersCount = count = getNextInt(data, index);
    uint maxWidth = 0;
    for(int i = 0; i < count; i++)
    {
        HipFontChar ch;
        //Advance "char"
        index = nextToken(data, index);
        //id
        index = nextToken(data, index);
        ch.id = getNextInt(data, index);
        // x
        index = nextToken(data, index);
        ch.x = getNextInt(data, index);
        // y
        index = nextToken(data, index);
        ch.y = getNextInt(data, index);
        // width
        index = nextToken(data, index);
        ch.width = getNextInt(data, index);
        if(ch.width > maxWidth)
            maxWidth = ch.width;
        // height
        index = nextToken(data, index);
        ch.height = getNextInt(data, index);
        // xoffset
        index = nextToken(data, index);
        ch.xoffset = getNextInt(data, index);
        // yoffset
        index = nextToken(data, index);
        ch.yoffset = getNextInt(data, index);
        // xadvance
        index = nextToken(data, index);
        ch.xadvance = getNextInt(data, index);
        // page
        index = nextToken(data, index);
        ch.page = getNextInt(data, index);
        // chnl
        index = nextToken(data, index);
        ch.chnl = getNextInt(data, index);

        characters[ch.id] = ch;
    }
    auto space = ' ' in characters;
    if(space is null || (space.width == 0 && space.xadvance == 0))
        spaceWidth = maxWidth;
    else
        spaceWidth = space.xadvance > space.width ? space.xadvance : space.width;
    lineBreakHeight = lineHeight;
    context = Context.unknown;
    break;
}
case Context.kernings:
```

Check your parser:

⚡ Optimizer can’t do much, so the gain is much bigger

⚡ Style is also easy to get wrong

⚡ Function calls piles up quickly

⚡ They are usually top offenders in the binary dump

⚡ On first implementation, hard to optimize for binary size

Reducing Binary Size

```
case Context.chars:
{
    //Advance "count"
    charactersCount = count = getNextInt(data, index);
    uint maxWidth = 0;


    for(int i = 0; i < count; i++)
    {
        HipFontChar ch;
        int*[10] fields = [cast(int*)&ch.id, &ch.x, &ch.y, &ch.width, &ch.height, &ch.xoffset, &ch.yoffset, &ch.xadvance, &ch.page, &ch.chnl];
        //Advance "char"
        index = nextToken(data, index);
        for(int fIndex = 0; fIndex < fields.length; fIndex++)
        {
            index = nextToken(data, index);
            *fields[fIndex] = getNextInt(data, index);
        }
        if(ch.width > maxWidth)
            maxWidth = ch.width;
        characters[ch.id] = ch;
    }
    auto space = ' ' in characters;
    if(space is null || (space.width == 0 && space.xadvance == 0))
        spaceWidth = maxWidth;
    else
        spaceWidth = space.xadvance > space.width ? space.xadvance : space.width;
    lineBreakHeight = lineHeight;
    context = Context.unknown;
    break;
}
```

7397	1.04%	_D3hip4font6bmfont13HipBitmapFont9loadAtlasMFAyaQdZb

⚡ With -Oz, size reduced by 13Kb with a single change

⚡ Also became easier to maintain

Reducing Binary Size

```d
enum HipMouseButton : ubyte
{
    left,
    middle,
    right,
    button1,
    button2,
    any,
    invalid
}

void main()
{
    int[HipMouseButton.button2+1] btns;

    foreach(mem; __traits(allMembers, HipMouseButton))
    {
        ubyte btn = __traits(getMember, HipMouseButton, mem);
        if(btn < HipMouseButton.any)
            btns[btn] = 50;
    }
}
```

Looping reflection members

Generated AST →

```d
void main()
{
    int[5] btns = 0;
    /*unrolled*/ {
        {
            enum string mem = "left";
            ubyte btn = cast(ubyte)0u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
        {
            enum string mem = "middle";
            ubyte btn = cast(ubyte)1u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
        {
            enum string mem = "right";
            ubyte btn = cast(ubyte)2u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
        {
            enum string mem = "button1";
            ubyte btn = cast(ubyte)3u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
        {
            enum string mem = "button2";
            ubyte btn = cast(ubyte)4u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
        {
            enum string mem = "any";
            ubyte btn = cast(ubyte)5u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
        {
            enum string mem = "invalid";
            ubyte btn = cast(ubyte)6u;
            if (cast(int)btn < 5)
                btns[cast(ulong)btn] = 50;
        }
    }
    return 0;
}
```

Reducing Binary Size

```
void main()
{
    int[HipMouseButton.button2+1] btns;

    foreach(i; 0..btns.length)
    {
        btns[i] = 50;
    }
}
```

Solution: **Use Known Information*****

Generated AST

```
void main()
{
    int[5] btns = 0;
    {
        ulong __key467 = 0LU;
        ulong __limit468 = 5LU;
        for (; __key467 < __limit468; __key467 += 1LU)
        {
            ulong i = __key467;
            btns[i] = 50;
        }
    }
    return 0;
}
```

Reducing Binary Size

USE KNOWN INFORMATION: TYPE ID

```
typedAssetFactory = [
    typeid(IHipAudioClip).toString : (string path, const(ubyte)[] extraData, string f, size_t l)=> new HipAudioLoadTask(path,path, nu
    typeid(IImage).toString :  (string path, const(ubyte)[] extraData, string f, size_t l) => new HipImageLoadTask(path,path,null, ex
    typeid(string).toString :  (string path, const(ubyte)[] extraData, string f, size_t l) => new HipFileLoadTask(path,path,null, extr
    typeid(IHipIniFile).toString :  (string path, const(ubyte)[] extraData, string f, size_t l) => new HipINILoadTask(path,path,null,e
    typeid(IHipCSV).toString :  (string path, const(ubyte)[] extraData, string f, size_t l) => new HipCSVLoadTask(path,path,null, extr
    typeid(IHipJSONC).toString :  (string path, const(ubyte)[] extraData, string f, size_t l) => new HipJSONCLoadTask(path,path,null,
    typeid(IHipTextureAtlas).toString :  (string path, const(ubyte)[] extraData, string f, size_t l) => new HipTextureAtlasLoadTask(pa
]
```

Note: Unfortunately .toString is required because typeid sharing on the dynamic library bridge has a bug and is not found

Old Style

```
IHipAssetLoadTask loadAsset(type)(string assetPath)
{
    static if(is(type == IHipCSV))
        return HipAssetManager.loadCSV(assetPath);
    else static if(is(type == IHipFont))
        return HipAssetManager.loadFont(assetPath);
    else static if(is(type == IImage))
        return HipAssetManager.loadImage(assetPath);
    else static if(is(type == IHipIniFile))
        return HipAssetManager.loadINI(assetPath);
    else static if(is(type == IHipJSONC))
        return HipAssetManager.loadJSONC(assetPath);
    else static if(is(type == IHipTexture))
        return HipAssetManager.loadTexture(assetPath);
    else static if(is(type == IHipTextureAtlas))
        return HipAssetManager.loadTextureAtlas(assetPath);
    else static if(is(type == IHipTilemap))
        return HipAssetManager.loadTilemap(assetPath);
    else static if(is(type == IHipTileset))
        return HipAssetManager.loadTileset(assetPath);
    else static if(is(type == IHipAudioClip))
        return HipAssetManager.loadAudio(assetPath);
    else
        return HipAssetManager.loadFile(assetPath);
}
```

```
IHipAssetLoadTask[] loadAssets(type)(string assetPath, const(ubyte)[] extraData, int start, int end)
{
    import hip.api;
    int sign = end - start >= 0 ? 1 : -1;
    ///Include 1 for the upper bounds
    int count = ((end - start) * sign) + 1;
    if(count == 1) return [HipAssetManager.loadAsset(type, assetPath, extraData)];
    IHipAssetLoadTask[] ret = new IHipAssetLoadTask[count];

    static string formatStr(string str, int number)
    {
        import hip.util.to_string_range;
        char[32] numSink = 0xff;
        toStringRange(numSink[], number);
        int charCount = 0;
        while(numSink[charCount++] != 0xff){} charCount--;
        //-1 for the $
        char[] formattedStr = new char[(cast(int)str.length)-1+charCount];
        int i = 0;
        foreach(ch; str)
        {
            if(ch == '$')
                formattedStr[i..i+=charCount] = numSink[0..charCount];
            else
                formattedStr[i++] = ch;
        }
        return formattedStr;
    }

    foreach(i; 0..count)
        ret[i] = HipAssetManager.loadAsset!type(formatStr(assetPath, start+i*sign), extraData);
    return ret;
}
```

Reducing Binary Size

```
///Binds texture to the specific slot
public void bind(int slot = 0)
{
    textureImpl.bind(slot);
}
public void unbind(int slot = 0)
{
    textureImpl.unbind(slot);
}
public void setWrapMode(TextureWrapMode mode){textureImpl.setWrapMode(mode);}
public void setTextureFilter(TextureFilter min, TextureFilter mag)
{
    this.min = min;
    this.mag = mag;
    textureImpl.setTextureFilter(min, mag);
}

Rect getBounds(){return Rect(0,0,width,height);}

/**
*   Returns whether the load was successful
*/
protected bool loadImpl(in IImage img)
{
    import hip.console.log;
    this.img = cast(IImage)img; //Promise it won't modify
    this.width = img.getWidth;
    this.height = img.getHeight;
    hiplog("Uploading Texture[",img.getName,"]", img.getWidth, "x", img.getHeight);
    this.textureImpl.load(img);
    setTextureFilter(TextureFilter.NEAREST, TextureFilter.NEAREST);
    return width != 0;
}
```

Wrapper of resource objects implementing an interface

```
class HipTexture : HipAsset
{
    IImage img;
    /**
    *   Make it available for implementors
    */
    IHipTexture textureImpl;
    alias textureImpl this;
}
```

Forwarding: Instead of tedious and error-prone rewriting

Better Performance: Less indirections in the code

Leaner TypeInfo: The less code inside a class, the better

Reducing Allocations

Example file of -profile=gc output

```
bytes allocated, allocations, type, function, file:line
     1159344             2 ubyte[] hip.util.array.uninitializedArray!(ubyte[]).uninitializedArray C:\Users\Marcelo\Documents\D\HipremeEngine\modules\util\source\hip\util
      139296             1 HipGeometryBatchVertex hip.graphics.g2d.geometrybatch.GeometryBatch.this C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\graphics\g2d\ge
      122912             1 HipSpriteVertex hip.graphics.g2d.spritebatch.HipSpriteBatch.this C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\graphics\g2d\spritebatc
       90144             1 HipTextRendererVertex hip.graphics.g2d.textrenderer.HipTextRenderer.this C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\graphics\g2d\te
       36864             1 ubyte hip.font.ttf.HipArsd_TTF_Font.generateImage C:\Users\Marcelo\Documents\D\HipremeEngine\modules\font\source\hip\font\ttf.d:214
       20480             1 ushort hip.graphics.g2d.geometrybatch.GeometryBatch.this C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\graphics\g2d\geometrybatch.d:79
       16384             1 ushort[] hip.util.array.uninitializedArray!(ushort[]).uninitializedArray C:\Users\Marcelo\Documents\D\HipremeEngine\modules\util\source\hip\ut
       12480           260 hip.event.handlers.button.HipButtonMetadata core.lifetime._d_newclassT!(HipButtonMetadata)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\drur
       12288           256 HipButtonMetadata hip.event.handlers.keyboard.KeyboardHandler.this C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\event\handlers\keyboa
        8448             1 RenderizedChar hip.font.ttf.HipArsd_TTF_Font.generateImage C:\Users\Marcelo\Documents\D\HipremeEngine\modules\font\source\hip\font\ttf.d:165
        8192             1 char[] hip.console.console.Console.this C:\Users\Marcelo\Documents\D\HipremeEngine\modules\console\source\hip\console\console.d:188
        7216             4 string hip.hiprenderer.backend.gl.defaultshaders.getSpriteBatchFragment C:\Users\Marcelo\Documents\D\HipremeEngine\modules\renderer\backends\g
        7168            33 string core.internal.array.concatenation._d_arraycatnTX!(string, string, string, string, string, string)._d_arraycatnTX C:\D\dmd2\windows\bin6
        4144             1 Vector!(2u, float) hip.event.handlers.mouse.HipMouse.this C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\event\handlers\mouse.d:21
        4096             8 HipWorkerThread hip.concurrency.thread.HipWorkerPool.this C:\Users\Marcelo\Documents\D\HipremeEngine\modules\concurrency\source\hip\concurrenc
        4096             8 hip.concurrency.thread.HipWorkerThread core.lifetime._d_newclassT!(HipWorkerThread)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\druntime\im
        4096             1 (string, string, string, string, string) hip.hiprenderer.backend.gl.defaultshaders.getSpriteBatchFragment C:\Users\Marcelo\Documents\D\Hipreme
        3072            32 (string, string, string, string, string) hip.hiprenderer.backend.gl.defaultshaders.getSpriteBatchFragment C:\Users\Marcelo\Documents\D\Hipreme
        2992             3 string hip.hiprenderer.backend.gl.glshader.Hip_GL_ShaderImpl.compileShader C:\Users\Marcelo\Documents\D\HipremeEngine\modules\renderer\backenc
        1824            21 string core.internal.array.concatenation._d_arraycatnTX!(string, string, string)._d_arraycatnTX C:\D\dmd2\windows\bin64\..\..\src\druntime\imp
        1056             6 hip.api.renderer.shadervar.ShaderVariablesLayout core.lifetime._d_newclassT!(ShaderVariablesLayout)._d_newclassT C:\D\dmd2\windows\bin64\..\..
         992             1 hip.api.data.commons.IHipAssetLoadTask delegate(immutable(char)[], const(ubyte)[], immutable(char)[], ulong)[immutable(char)[]] hip.assetmanag
         960            10 core.sync.mutex.Mutex core.lifetime._d_newclassT!(Mutex)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\druntime\import\core\lifetime.d:2763
         864             9 Mutex hip.concurrency.mutex.DebugMutex.this C:\Users\Marcelo\Documents\D\HipremeEngine\modules\concurrency\source\hip\concurrency\mutex.d:28
         864             9 hip.concurrency.mutex.DebugMutex core.lifetime._d_newclassT!(DebugMutex)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\druntime\import\core\l
         768             8 DebugMutex hip.concurrency.thread.HipWorkerThread.this C:\Users\Marcelo\Documents\D\HipremeEngine\modules\concurrency\source\hip\concurrency\t
         576             5 string core.internal.array.concatenation._d_arraycatnTX!(string, string, string, string)._d_arraycatnTX C:\D\dmd2\windows\bin64\..\..\src\drur
         544             4 HipVertexAttributeInfo[] hip.hiprenderer.vertex.HipVertexArrayObject.appendAttribute C:\Users\Marcelo\Documents\D\HipremeEngine\modules\render
         528            33 char hip.util.conv.toString C:\Users\Marcelo\Documents\D\HipremeEngine\modules\util\source\hip\util\conv.d:219
         528             3 Shader hip.hiprenderer.renderer.HipRendererImplementation.newShader C:\Users\Marcelo\Documents\D\HipremeEngine\modules\renderer\source\hip\hip
         528             3 hip.hiprenderer.shader.shader.Shader core.lifetime._d_newclassT!(Shader)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\druntime\import\core\l
         512             1 (string, string) hip.hiprenderer.backend.gl.defaultshaders.getBitmapTextFragment C:\Users\Marcelo\Documents\D\HipremeEngine\modules\renderer\t
         512             1 (string, string) hip.hiprenderer.backend.gl.defaultshaders.getGeometryBatchFragment C:\Users\Marcelo\Documents\D\HipremeEngine\modules\rendere
         512             1 WatcherThread hip.systems.compilewatcher.CompileWatcher.run C:\Users\Marcelo\Documents\D\HipremeEngine\dependencies\compilewatcher\source\hip\
         512             1 hip.systems.compilewatcher.WatcherThread core.lifetime._d_newclassT!(WatcherThread)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\druntime\im
         448             7 ShaderVar hip.api.renderer.shadervar.ShaderVar.createEmpty C:\Users\Marcelo\Documents\D\HipremeEngine\api\source\hip\api\renderer\shadervar.d:
         448             7 hip.api.renderer.shadervar.ShaderVar core.lifetime._d_newitemT!(ShaderVar)._d_newitemT C:\D\dmd2\windows\bin64\..\..\src\druntime\import\core\
         400             7 void hip.api.renderer.shadervar.ShaderVar.createEmpty C:\Users\Marcelo\Documents\D\HipremeEngine\api\source\hip\api\renderer\shadervar.d:252
         384             3 hip.assets.image.Image core.lifetime._d_newclassT!(Image)._d_newclassT C:\D\dmd2\windows\bin64\..\..\src\druntime\import\core\lifetime.d:2763
         368             1 HipArsd_TTF_Font hip.global.gamedef.loadDefaultAssets C:\Users\Marcelo\Documents\D\HipremeEngine\source\hip\global\gamedef.d:77
```

Reducing Allocations

TYPE YOUR ENUMS

```d
enum HipButtonType
{
    down = 0,
    up = 1
}

enum AutoRemove
{
    no = false,
    yes = true
}
alias HipInputAction = void delegate(const(AHipButtonMetadata) meta);
/**
 * Handler for any kind of button
 */
struct HipButton
{
    ushort id;
    HipButtonType type;
    AutoRemove isAutoRemove = AutoRemove.no;
    HipInputAction action;
    alias id this;
}
```

32 Bytes

```d
enum HipButtonType : ubyte
{
    down = 0,
    up = 1
}

enum AutoRemove : bool
{
    no = false,
    yes = true
}
alias HipInputAction = void delegate(const(AHipButtonMetadata) meta);
/**
 * Handler for any kind of button
 */
struct HipButton
{
    ushort id;
    HipButtonType type;
    AutoRemove isAutoRemove = AutoRemove.no;
    HipInputAction action;
    alias id this;
}
```

24 Bytes

⚡ **-profile=gc** makes it easier to find unoptimized bits in huge code bases

⚡ **Impossible to auto-optimize:** ABI compatibility exists , thus requiring developer awareness.

Reducing Allocations

SORT YOUR CLASSES MEMBERS

```
pragma(msg, __traits(classInstanceSize, HipButtonMetadata));
final class HipButtonMetadata
{
    bool _isNewState = false;
    float lastDownTime, downTimeStamp;
    ubyte clickCount = 0;
    float lastUpTime, upTimeStamp;
    bool _isPressed = false;
    float timeStartedCheckingRestart = 0;
    float timeUntilRestartMulticlick = 100;
}
```

52 bytes

```
pragma(msg, __traits(classInstanceSize, HipButtonMetadata));
final class HipButtonMetadata
{
    float lastDownTime, downTimeStamp;
    float lastUpTime, upTimeStamp;
    float timeStartedCheckingRestart = 0;
    float timeUntilRestartMulticlick = 0;
    bool _isPressed = false;
    bool _isNewState = false;
    ubyte clickCount = 0;
}
```

43 bytes

- **OOP** programmers commonly forgets the rules still applies to classes.
- **Make them final**. D final classes have special optimizations. Use them.
- **24 bytes are reserved.** They are always included in classes.

Reducing Allocations

FIND LEAKS

```
Object hipSaveRef(Object reference, int target = HipExportedTargets.nativeScript_D)
{
    for(uint i = 0; i < _hipExportedSharedUserData.length; i++)
    {
        if(i >= _hipExportedSharedUserData[target].length)
            break;
        // else if (_hipExportedSharedUserData[target][i] is reference)
            // return reference;
        if(_hipExportedSharedUserData[target][i] is null)
        {
            _hipExportedSharedUserData[target][i] = reference;
            return reference;
        }
    }
    _hipExportedSharedUserData[target] ~= reference;
    return reference;
}
```

```
const D3D_SHADER_MACRO[] defines =
[
    cast(D3D_SHADER_MACRO)null, cast(D3D_SHADER_MACRO)null
]; //.staticArray
```

```
import std.array:split;
foreach(l; capture.split("\n"))
//import std.algorithm.iteration:splitter;
// foreach(l; capture.splitter("\n"))
{
    l = l.trim;
    if(l == "")
        continue;
    string[] kv = l.split("=");
    l.splitRange("=").put(&k, &v);
    // string k, v;
    // l.splitRange("=").put(&k, &v);
    if(v == "")
    {
        errors~= "No value for key '"~k~"'";
        _noError = false;
        break;
    }
    string name = k.replaceAll(' ', "");
    block.vars[name] =  IniVar(name, formatValue(v));
}
```

⚡ **Your function may have a bug:** and it will show in **-profile=gc** with ridiculously large numbers.

⚡ **You could be allocating:** without even knowing.

⚡ **You know new techniques:** older code might allocate where there's no need

Reducing Allocations

IMPROVE YOUR PERFORMANCE

```
enum JSONType
{
    bool_,
    float_,
    int_, integer = int_, uinteger = int_,
    string_, string = string_,
    array,
    object,
    null_
}
struct JSONValue
{
    union JSONData{

        double _float;
        long _int;
        bool _bool;
        string _string;
        JSONObject* object;
        JSONArray* array;
    }
    JSONData data;
    string key;
    string error;
    JSONType type = JSONType.object;
}
```

UNOPTIMIZED VERSION

- ⚡ Untyped enum
- ⚡ Redundant information
- ⚡ 56 bytes structure

5 tests in 50MB dictionary jsons

- ⚡ MB Per Second: 29.654
- ⚡ Free: 238MB
- ⚡ Allocated: 6569MB
- ⚡ Used: 3863MB

30K tests in 110KB string dictionary jsons

- ⚡ MB Per Second: 236.565
- ⚡ Free: 8MB
- ⚡ Allocated: 57845MB
- ⚡ Used: 14MB

Reducing Allocations

IMPROVE YOUR PERFORMANCE

```d
enum JSONType : ubyte
{
    bool_ = 0,
    float_ = 1,
    int_ = 2, integer = int_, uinteger = int_,
    string_ = 3, string = string_,
    array = 4,
    object = 5,
    error = 6,
    null_ = 7 //0b111
}
private union JSONData
{
    double _float;
    long _int;
    bool _bool;
    immutable(char)* _string;
    JSONObject object;
    JSONArray* array;
}
struct JSONValue
{
    JSONData data;
    static if(size_t.sizeof == uint.sizeof)
    {
        private static enum bitOffset = 29;
        private static enum lengthMask = 0x1FFFFFFF;
    }
    else
    {
        ///Bit offset on where the type information is stored
        private static enum bitOffset = 61;
        ///All the bits that defines where length is.
        private static enum lengthMask = 0x1FFFFFFFFFFFFFFF;
    }
    ///Used only for the string.
    private size_t _length;

    pragma(inline, true) JSONType type(JSONType t)
    {
        _length = (_length & lengthMask) | (cast(size_t)t << bitOffset);
        return t;
    }
    pragma(inline, true) JSONType type() const
    {
        return cast(JSONType)(_length >> bitOffset);
    }
```

OPTIMIZED VERSION

- ⚡ Multi-purpose enum
- ⚡ Reuse and encode fields
- ⚡ 300% faster at string parsing dictionary
- ⚡ 37% faster at dictionary-only json
- ⚡ Memory consumption reduced by 82% on string dictionary
- ⚡ Memory consumption reduced by 20% on dictionary of dictionaries
- ⚡ 16 bytes structure [possibly reducible to 8]

5 tests in 50MB dictionary jsons

- ⚡ MB Per Second: 40.23
- ⚡ Free: 360MB
- ⚡ Allocated: 5284MB
- ⚡ Used: 3076MB

30K tests in 110KB string dictionary jsons

- ⚡ MB Per Second: 753.97
- ⚡ Free: 2MB
- ⚡ Allocated: 10577MB
- ⚡ Used: 9MB

Improving Performance

```d
import std;
import std.datetime.stopwatch;
void function(ref Test) breaker;
void breakOptimization(ref Test t)
{
    t.otherField++;
}

struct Test
{
    float[1024] data = void;
    int otherField;
}

void test1()
{
    Test t;
    breaker(t);

}

void test2()
{
    Test t = void;
    t.otherField = 0;
    breaker(t);
}

void main()
{
    auto fnTest = &test1;
    auto fnTest2 = &test2;
    breaker = &breakOptimization;
    writeln = benchmark!(()(*fnTest)();}, ()(*fnTest2)();})(10_000_000);
}
```

Takes 642 ms

Takes 23 ms

Timings took with LDC -O3 on run.dlang.org

NEVER DO IT

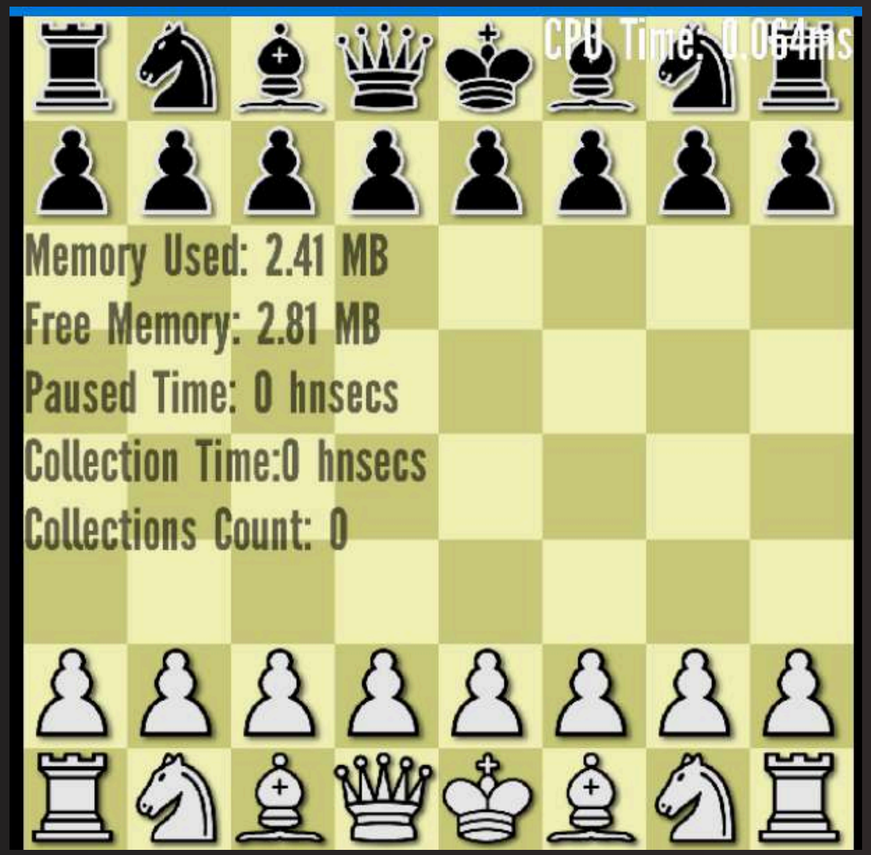
⚡ Changes the init value of a data to 0
⚡ Does nothing on optimization
⚡ Might be double initializing the variable
⚡ Might mislead you into believing the code has zero-cost initialization

DO IT

Actual void initialization

Reducing Allocations

CONSTANT AWARENESS



⚡ **GC.Stats:** Used/Free memory
⚡ **GC.ProfileStats:** Timings for GC
⚡ **`extern(C) __gshared string[] rt_options = [ "gcopt=profile:1" ]`:** Prints stats in the program end
⚡ **`--DRT-gcopt=profile:1` :** Every D program accepts this command line argument

Reducing Allocations

REFCOUNT AND BUMP ALLOCATOR ISSUES

- ⚡ Bump allocator for WebAssembly runtime
- ⚡ RefCount for strings
- ⚡ Special handling to avoid fragmentation

Reducing Allocations

SOLUTION: STACK MEMORY IS AWESOME

1. **Low overhead** with void initialization
2. **No fragmentation** as CPU keep tracks of it
3. **Use-Case specialization.** Choose best size for the job
4. **No allocator needed** every platform already has their implementation
5. **Faster.** Profiling shown ⅓ of CPU usage compared to RefCount
6. **Best for fire-and-forget API**

```d
struct StringBuffer(size_t capacity)
{
    @nogc:
    private char[capacity] chars;
    private size_t _length;

    pragma(inline, true)
    size_t length() @safe pure const nothrow { return _length; }

    /**
     * Returns: An empty StringBuffer which avoids initialization on its buffer
     */
    static StringBuffer!(capacity) get()
    {
        StringBuffer!(capacity) ret = void;
        ret._length = 0;
        return ret;
    }

    static auto opCall(Args...)(Args args)
    {
        auto ret = StringBuffer!(capacity).get();
        static foreach(a; args)
            ret~= a;
        return ret;
    }

    void preAllocate(size_t howMuch)
    {
        assert(length + howMuch < capacity, "Can't preallocate more to string buffer.");
    }
    void put(char c){chars[_length++] = c;}
    void put(const(char)[] s){chars[_length.._length+s.length] = s[]; _length+= s.length;}
    void put(immutable(char)* s){put(s[0..strlen(s)]);}
    void put(String s){put(s.toString());}

    StringBuffer opSlice(size_t start, size_t end)
    {
        assert(end >= start, "Slice end must be greater or equal than start.");
        StringBuffer ret = void;
        ret.chars[0..end-start] = chars[start..end];
        ret._length = end - start;
        return ret;
    }
}
alias BigString = StringBuffer!(8192);
alias PathString = StringBuffer!(2048);
alias SmallString = StringBuffer!(256);
```

Into Scalability

Redub Plugins

A plugin system that scales without adding complexity

⚡ **Extend features** using an optimized tool
⚡ **Optional** and easy to use
⚡ **Unifies** everything into a single tool
⚡ **Ensure** clear interaction with core functionality
⚡ **Share** functionality across dependencies

Into Scalability - Plugins

```d
module getmodules;
import redub.plugin.api;
class GetModulePlugin : RedubPlugin
{
    void preGenerate(){}
    void postGenerate(){}
    /**
     * Utility to generate a list of the modules found in all sourcePaths from the current dependency.
     * Use it with:
     ```json
     "preBuildPlugins": {
        "getmodules": "ct_assets/game_modules.txt"
     }
     ```
     */
    extern(C) ref RedubPluginStatus preBuild(RedubPluginData input, out RedubPluginData output, const ref string[] args, ref return RedubPluginStatus status)
    {
        import std.file;
        import std.path;
        import std.array:replace;

        if(args.length != 1)
            return status = RedubPluginStatus(RedubPluginExitCode.error, "Usage: \"getmodules\": [\"outputFileName\"]");
        string outputPath = args[0];
        if(exists(outputPath) && isDir(outputPath))
            return status = RedubPluginStatus(RedubPluginExitCode.error, "Invalid output path '"~outputPath~"', the output path is a directory");
        if(outputPath.length == 0)
            return status = RedubPluginStatus(RedubPluginExitCode.error, "Invalid output path '"~outputPath~"', the output path is empty.");

        string getModulesFile;

        foreach(string inputPath; input.sourcePaths)
        {
            import std.algorithm.searching;
            foreach(DirEntry e; dirEntries(inputPath, "*.d", SpanMode.depth))
            {
                if(countUntil(e.name, "gamescript") == -1)
                    continue;
                string file = e.name;
                if(getModulesFile != "")
                    getModulesFile~="\n";
                //Remove .d, change / or \ to .

                file = relativePath(file, inputPath)[0..$-2];
                getModulesFile~= file.replace(dirSeparator[0], '.');
            }
        }
        string outDir = dirName(outputPath);
        if(!std.file.exists(outDir))
            std.file.mkdirRecurse(outDir);

        std.file.write(outputPath, getModulesFile);
        return status = RedubPluginStatus(RedubPluginExitCode.success, "getModule plugin generated file "~outputPath);
    }
    void postBuild(){}
}
mixin PluginEntrypoint!(GetModulePlugin);
```

Basic structure of a plugin

Into Scalability - Refactor

ACCEPT THAT YOU DON’T KNOW EVERYTHING

```
version(DesktopRelease)
{
    import app;
    __gshared auto _keepMain = &main; ///TODO: Find some other way to avoid main being stripped.
}
```

A bug in either linker/compiler didn’t allow `main` (in DMD) to be inside a library.

UGLY SOLUTION: UNNAMED DEPENDENCY

```
"unnamedDependencies": [
    "#HIPREME_ENGINE_PATH"
],
```

⚡ **Custom dub preprocessor** As one would need a special case issue

⚡ **Additional tool needed:** A tool which created a new project description using engine as main

⚡ **Redundant** As that doesn’t solve any other issue

⚡ **Hacks doesn’t scale:** That tool solved the problem at that time, but made me refactor every single time

Into Scalability - Refactor

ACCEPT THAT YOU DON'T KNOW EVERYTHING

```
void referenceExported()
{
    auto a = &getNetworkInterface;
    auto b = &HipFileSystemAPI;
    auto c = &HipAudioPlayerAPI;
    auto d = &HipRendererAPI;
}
```

```
export extern(C) IHipAudioPlayer HipAudioPlayerAPI()
{
    return player;
}
```

```
export extern(C) IHipFS HipFileSystemAPI()
{
    return HipFileSystem();
}
```

Another bug would also strip some exported functions when inside library and not referenced by the main package

```
mixin HipExportDFunctions!(hip.graphics.g2d.animation);
mixin HipExportDFunctions!(hip.game.utils);
mixin HipExportDFunctions!(hip.assetmanager);
mixin HipExportDFunctions!(hip.systems.timer_manager);
```

⚡ **Reflection** for making the hack bearable

⚡ **Dynamic Library bridge hack** as dmd would collect memory sent to the dynamic library

```
// expansion at C:\Users\Marcelo\Documents\D\HipremeEngine\modules\util\source\hip\util\reflection.d(295)
export extern(System) IHipAssetLoadTask HipAssetManager_loadAsset(getParams!(sym)){
            import hip.util.lifetime;
            return cast(typeof(return))hipSaveRef(cast(Object)
        HipAssetManager.loadAsset(__traits(parameters)));}

// expansion at C:\Users\Marcelo\Documents\D\HipremeEngine\modules\util\source\hip\util\reflection.d(295)
export extern(System) IHipTilemap HipAssetManager_createTilemap(getParams!(sym)){
            import hip.util.lifetime;
            return cast(typeof(return))hipSaveRef(cast(Object)
        HipAssetManager.createTilemap(__traits(parameters)));}
```

Into Scalability - Refactor

EXCLUDE UNUSED CODE AS FAST AS POSSIBLE

⚡ **Lua Support Added in 2021**: That was when most base progress of the engine had been done

⚡ **The API exporting was completely based on it**


```
void initG2D()
{
    import hip.api.internal;
    import hip.api.console;
    loadClassFunctionPointers!HipG2DBinding;
    log("HipengineAPI: Initialized G2D");
}

class HipG2DBinding
{
    extern(System) __gshared //All functions there will be loaded
    {
        ///Use this only when you're sure you don't need!
        void function(bool enable = true) setRendererErrorCheckingEnabled;

        ///Will change the color for the next calls to drawPixel, drawRectangle, drawTriangle, fillRectangle, fillTriangle
        void function(HipColor color) setGeometryColor;
        ///Draw a pixel at (x, y) with the color specified at setGeometryColor
        void function(int x, int y, HipColor color = HipColor.no) drawPixel;
    }
}
```

As shown in the following code, the API turned into a function pointer holder, since that was how it made it exposable to Lua

Into Scalability - Refactor

CHECK MULTITHREADING EVERY TIME YOU LEARN

```
}
/**
*   This thread goes into an invalid st
*/
void finish()
{
    mutex.lock();
    isAlive = false;
    semaphore.notify;
    mutex.unlock();
}
bool isIdle()
{
    mutex.lock();
    bool ret = isIdleImpl();
    mutex.unlock();
    return ret;
}
private bool isIdleImpl()
{
    return jobsQueue.length == 0;
}
```

Locks for simple code

```
void run()
{
    while(isAlive)
    {
        mutex.lock();
        if(!isIdleImpl)
        {
            WorkerJob job = jobsQueue[0];
            mutex.unlock();
            try
            {
                mutex.lock();
                job.task();
                if(job.onTaskFinish != null)
                {
                    job.onTaskFinish(job.name);
                }
                mutex.unlock();
            }
            catch(Error e)
            {
                onAnyException(true, e.toString());
                return;
            }
            catch(Exception e)
            {
                onAnyException(false, e.toString());
                return;
            }
            mutex.lock();
            jobsQueue = jobsQueue[1..$];
            mutex.unlock();
        }
        else
            mutex.unlock();
        semaphore.wait;
    }
}
```

I see DEAD LOCKS

Into Scalability - Refactor

CHECK MULTITHREADING EVERY TIME YOU LEARN

```
/**
*   This thread goes into an invalid state after finis
*/
void finish()
{
    isAlive.atomicStore = false;
    semaphore.notify;
}
bool isIdle()
{
    return atomicLoad(jobsCount) == 0;
}
```

⚡ **Lockless code:** State read should not need a lock, which makes it much faster and easier to read

⚡ **Separate length field:** For guaranteeing atomic reads, you might do it

```
void run()
{
    while(isAlive)
    {
        if(!isIdle)
        {
            mutex.lock();
            WorkerJob job = jobsQueue[0];
            jobsQueue = jobsQueue[1..$];
            mutex.unlock();
            try
            {
                job.task();
                if(job.onTaskFinish != null)
                    job.onTaskFinish(job.name);
                atomicFetchSub(jobsCount, 1);
            }
            catch(Error e)
            {
                onAnyException(true, job.name, e.toString());
                return;
            }
            catch(Exception e)
            {
                onAnyException(false, job.name, e.toString());
                return;
            }
        }
        semaphore.wait;
    }
}
```

Single Lock: Locking is done only for syncing jobs queue

Into Scalability - Refactor

DON’T FOLLOW A SINGLE DESIGN STYLE

```
@ExportD static IHipAssetLoadTask loadTexture(string texturePath, string f = __FILE__, size_t l = __LINE__)
{
    import hip.util.memory;
    hiplog("AssetManager: Loading Texture: ", texturePath);
    void delegate(string, void delegate(void[]), void delegate(string err = "")) assetLoadFunc =
    (pathOrLocation, onFirstStepComplete, onFailure)
    {
        import hip.filesystem.hipfs;
        HipFS.read(pathOrLocation).addOnSuccess((in ubyte[] data)
        {
            new Image(pathOrLocation, cast(ubyte[])data,
            (IImage img)
            {
                onFirstStepComplete(toHeapSlice(img));
            }, (){onFailure();});
            return FileReadResult.free;
        }).addOnError((string err)
        {
            ErrorHandler.showErrorMessage("Could not read file ", err);
        });
    };

    void delegate(void[], void delegate(HipAsset)) onPartialDataLoaded =
    (partialData, onSuccess)
    {
        Image img = cast(Image)(cast(IImage)partialData.ptr);
        HipTexture ret = new HipTexture(img);
        hiplog("AssetManager: Texture: Loaded ", texturePath, " ", ret.toHipString);
        onSuccess(ret);
        void* gcObjCopy = cast(void*)img;
        freeGCMemory(gcObjCopy);
    };
    IHipAssetLoadTask task = loadComplex("Load Texture", texturePath, assetLoadFunc, onPartialDataLoaded, f, l);
    return task;
}
```

```
final class HipImageLoadTask : HipAssetLoadTask
{
    private IHipFSPromise fs;
    this(string path, string name, HipAsset asset, const(ubyte)[] extraData, string fileRequesting, size_t lineR
    {
        super(path,name,asset,extraData, fileRequesting,lineRequesting);
    }

    override void update()
    {
        final switch(result) with (HipAssetResult)
        {
            case waiting:
                result = loading;
                worker = HipAssetManager.loadWorker("Load Image", ()
                {
                    fs = HipFS.read(path)
                    .addOnError((string error){result = cantLoad; this.error = error;})
                    .addOnSuccess((in ubyte[] data)
                    {
                        _asset = new Image(path, data, (IImage img){result = loaded;}, (){result = cantLoad;});
                        return FileReadResult.keep;
                    });
                });
                break;
            case loading, mainThreadLoading:
                break;
            case cantLoad: goto case loaded;
            case loaded:
                if(fs !is null)
                {
                    fs.dispose();
                    fs = null;
                }
                break;
        }
    }
```

Before

⚡ **Delegates:** No control on allocation

⚡ **Locks Needed:** Sharing threads data

After

⚡ **Classes:** Recycle memory, deallocate on demand

⚡ **State Machine:** Remove the usage of locks as state handles synchronization

Into Scalability - Refactor

PROBLEM: PROVIDE FAST BUILD TIMES AND FLEXIBLE API

Alternatives

- ❌ **Extern Function:** Unable to use since API the implementation has link-time dependency
- ❌**PImpl:** C interface hiding. Requires function API and needs reflection to make it scalable
- ⚠️**Dependency Injection:** Pass an object which holds the implementation, but has a verbose API and requires many refactors
- ⚠️ **Service Locator:** Create a global object which provides functionality. Not very flexible
- ✔️**Service Locator + Dependency Injection:** Both engine and users have the same API with explicit initialization in the engine. Locator uses interfaces to implementation

Into Scalability - Refactor

SERVICE LOCATOR EXAMPLE

```
///Dependency injection interface for HipFS
private __gshared IHipFS _fs;
void setIHipFS(IHipFS fsInstance)
{
    _fs = fsInstance;
}
IHipFS HipFileSystem()
{
    return _fs;
}
alias HipFS = .HipFileSystem;
```

Common code living in API

```
/**
 * Provides the interface for the filesystem singleton
 */
interface IHipFS
{
    ///Gets a path from the installed path
    string getPath(string path);

    ///Uses the only extra verifications to check if the path is valid
    bool isPathValidExtra(string path);

    /**
     * Encapsulates both the sync and async in the same API for reading a
     * Params:
     *   path = The path to read
     * Returns: A task/promise which will output the file data. It return
     */
    IHipFSPromise read(string path);
}
```

Interface code example

```
static void initEngine(bool audio3D = false)
{
    import hip.internal_configuration;
    Console.install(ActivePlatform, getPlatformPrintFunction());
    loglnInfo("Console installed for ", ActivePlatform);
    HipFS.initializeAbsolute();
    version(HandleArguments)
        HipremeHandleArguments();

    string fsInstallPath = getFSInstallPath(projectToLoad);
    HipFS.install(fsInstallPath, getFilesystemValidations());
    setIHipFS(HipFS);
```

Main executable DI startup

```
public import hip.api.filesystem.hipfs;
void initFS()
{
    import hip.api.internal;
    alias fs = extern(C) IHipFS function();
    setIHipFS((cast(fs)_loadSymbol(_dll, "HipFileSystemAPI"))());
    import hip.api.console;
    log("HipengineAPI: Initialized FS");
}
```

DLL DI startup

Into Scalability - Refactor

COMPARISON WITH OLD SOLUTION

```d
void initFS()
{
    import hip.api.internal;
    loadClassFunctionPointers!(HipFSBinding, UseExportedClass.Yes, "HipFileSystem");
    import hip.api.console;
    log("HipengineAPI: Initialized FS");
}
import hip.api.internal;
class HipFSBinding
{
    @disable this();
    extern(System) __gshared
    {
        string function (string path) getPath;
        bool function (string path, bool expectsFile = true, bool shouldVerify = true) isPathValid;
        bool function (string path) isPathValidExtra;
        bool function (string path) setPath;
        IHipFSPromise function (string path, out ubyte[] output) read;
        IHipFSPromise function (string path, out string output) readText;
        bool function (string path, void[] data) write;
        bool function (string path) exists;
        bool function (string path) remove;
        string function () getcwd;
        bool function (string path) absoluteExists;
        bool function (string path) absoluteIsDir;
        bool function (string path) absoluteIsFile;
        bool function (string path) isDir;
        bool function (string path) isFile;
        string function (string cacheName, void[] data) writeCache;
    }
}
mixin ExpandClassFunctionPointers!(HipFSBinding);
```

⚡ **Required Reflection:** So it made defining API easier, which also increased compilation times and code complexity

⚡ **Functionality not shareable:** Could not import resources since they needed unexposed functionality

```d
@ExportD public static bool setPath(string path);
@ExportD public static bool isPathValid(string path, bool expectsFile = true, bool shouldVerify = true);
@ExportD public static string getPath(string path);
```

Every function was static and required a special UDA

Into Scalability - Refactor

NEW SOLUTION USAGE

```
public class Image : HipAsset, IImage
{
    IHipImageDecoder decoder;
    string imagePath;
    int width, height;
    ubyte bytesPerPixel;
    ushort bitsPerPixel;

    ubyte[] pixels;
    this(string path = "")
    {
        import hip.util.system : sanitizePath;
        path = sanitizePath(path);
        super("Image_"~path);
        imagePath = path;
        decoder = getDecoder(path);
    }
}
```

⚡ **Use getDecoder API:** set at startup of main and dll

```
void initGlobalAssets()
{
    import hip.api.data.image;
    import hip.api.internal;
    loadClassFunctionPointers!HipGlobalAssetsBinding;
    import hip.api.console;
    setImageDecoderProvider(getDecoder);
    log("HipEngineAPI: Initialized Global Assets");
}
```

DLL DI Initialization

⚡ **No partial functionality:** So the resource can be directly used by the user

⚡ **API defined once:** No more manual definition on what the developer can use

```
private __gshared extern(System) IHipImageDecoder function(string path)
void setImageDecoderProvider(typeof(getDecoderFn) provider)
{
    getDecoderFn = provider;
}
IHipImageDecoder getDecoder(string path){return getDecoderFn(path);}
```

Another dependency injection

Into Scalability - Reflection

NETWORK RELATED CODE SHARING

```d
/**
 * Those are the reserved type IDs that are found in every MarkNetData instance.
 * Since a new instance of this enum is created, the reserved types are written
 * snake-cased.
 * Custom type members will have the exact same name as the type they intend to use.
 *
 *  connect: void send_connect(INetwork)
 *  disconnect: void send_disconnect(INetwork)
 *  get_connected_clients: send_get_connected_clients(INetwork)
 *  client_connect: send_client_connect(INetwork, uint targetID)
 */
enum MarkedNetReservedTypes : ubyte
{
    invalid,
    ///Send that message so NetController can identify that a network connection was established.
    @NetBinding!(void, void) connect,
    ///Send that message so NetController can identify that a network interface was disconnected
    @NetBinding!(void, void) disconnect,
    ///Sends a message to the server requesting for the available connection IDs
    @NetBinding!(void, ConnectedClientsResponse) get_connected_clients,
    ///The ID to connect to. Must be a valid ID received from get_connected_clients
    @NetBinding!(uint, ConnectToClientResponse) client_connect
}

template Attributes(T, string mem)
{
    alias Attributes = __traits(getAttributes, __traits(getMember, T, mem));
}

static foreach(m; __traits(allMembers, MarkedNetReservedTypes))
{
    static if(Attributes!(MarkedNetReservedTypes, m).length)
    {
        static if(is(Attributes!(MarkedNetReservedTypes, m)[0].Request == void))
        {
            mixin("void send_",m,"(INetwork net){",
            "net.sendDataToServer(__traits(getMember, MarkedNetReservedTypes, m));}");
        }
        else
        {
            mixin("void send_",m,"(INetwork net, Attributes!(MarkedNetReservedTypes, m)[0].Reques
            "pragma(LDC_no_typeinfo) static struct Data { align(1): MarkedNetReservedTypes t; Att
            "net.sendDataToServer(Data(__traits(getMember, MarkedNetReservedTypes, m), d));}");
        }
    }
}
```

⚡ **Shared Independent API:** As it needs to run outside the engine . Uses the same code inside the server

⚡ **Reserve Functionality:** The more it is implemented on both, the better. This guarantees stability and scalability

Into Scalability - Reflection

NETWORK RELATED CODE SHARING

```d
template MarkNetData(T...)
{
    enum PredefinedTypesCount = __traits(allMembers, MarkedNetReservedTypes).length;

    static if(T.length <= ubyte.max)
        alias idType = ubyte;
    else static if(T.length <= ushort.max)
        alias idType = ushort;

    mixin(enumFromTypes!(T)("Types", "idType"));

    string getTypeName(idType v)
    {
        static foreach(t; T)
        {
            if(v == __traits(getMember, Types, t.stringof))
                return t.stringof;
        }
        return "Unknown";
    }

    bool isInvalid(idType v)
    {
        return v == 0 || v >= T.length;
    }

    static foreach(i, t; T)
    {
        void sendData(INetwork net, t data)
        {
            align(1)
            static struct TypedData
            {
                t actualData;
                idType typeID = i + PredefinedTypesCount;
            }
            net.sendData(TypedData(data, i+PredefinedTypesCount));
        }
    }
        /**
     * To its data removing the type id.
     * Params:
     *   buffer = The buffer that is used to take the type id and slices it to the actua
     * Returns: The type id from that buffer. Also enforces it is not invalid
     */
    Types getDataFromBuffer(ref ubyte[] buffer)
    {
        idType typeID = *cast(idType*)(buffer.ptr + buffer.length - idType.sizeof);
        buffer = buffer[0..buffer.length - idType.sizeof];
        return cast(Types)typeID;
    }
```

⚡ **Standardized Serialization and Deserialization:** Create a common way to the data identification on both client and server

⚡ **Optimize on lower bandwidth cost:** Use D functionality to generate tightly packed data to be shared on the network

Into Scalability - Reflection

NETWORK RELATED CODE SHARING

```
struct Action
{
    ubyte fromX, fromY, toX, toY;
}

struct BoardState
{
    Action[] actions;
}
enum AssignColor
{
    black,
    white
}
alias ChessNetController = NetController!(MarkNetData!(
    Action,
    AssignColor,
    BoardState
));

ChessNetController c = new ChessNetController(getNetworkInterface);
c.connect(NetIPAddress("127.0.0.1", 10_000));
c.on_connect(()
{
    c.sendData([BoardState()]);
});
c.on_disconnect(()
{
    logg("Waiting for other player to connect...");
});
with(c.poll)
{
    switch(typeID)
    {
        case Types.disconnect: //Disconnect event has no data
            break;
        case Types.Action:
            Action act = getAction();
            break;
        case Types.AssignColor:
            AssignColor c = getAssignColor();
            break;
        default:
            //Received an invalid typeID.
            break;
    }
```

⚡ **Auto code generation:** Create a class which wraps the MarkNetData so it becomes easier to use the data

⚡ **Work in a way where the protocol is completely abstracted:** Using reflection alongside code sharing and abstraction can create a powerful, easy to use and performant API

CONCLUSION

1. **Mix different techniques:** This will actually reduce the complexity of the code and reduce workarounds and hacks
2. **You can’t escape from refactor:** As your experience as a developer grows, you’ll start doing it
3. **Refactor as soon as you don’t understand:** Chances are that you have already outgrown yourself
4. **Create scope challenges:** Be it on binary size, memory consumption or CPU. You’ll always learn something from it and make your library much better

QUESTIONS?