Using React with the Godot Game Engine

Also, adding Javascript to your game for good and sometimes bad reasons

I’m DM!

- I make video games
- I like to make life difficult for myself

I’m DM!

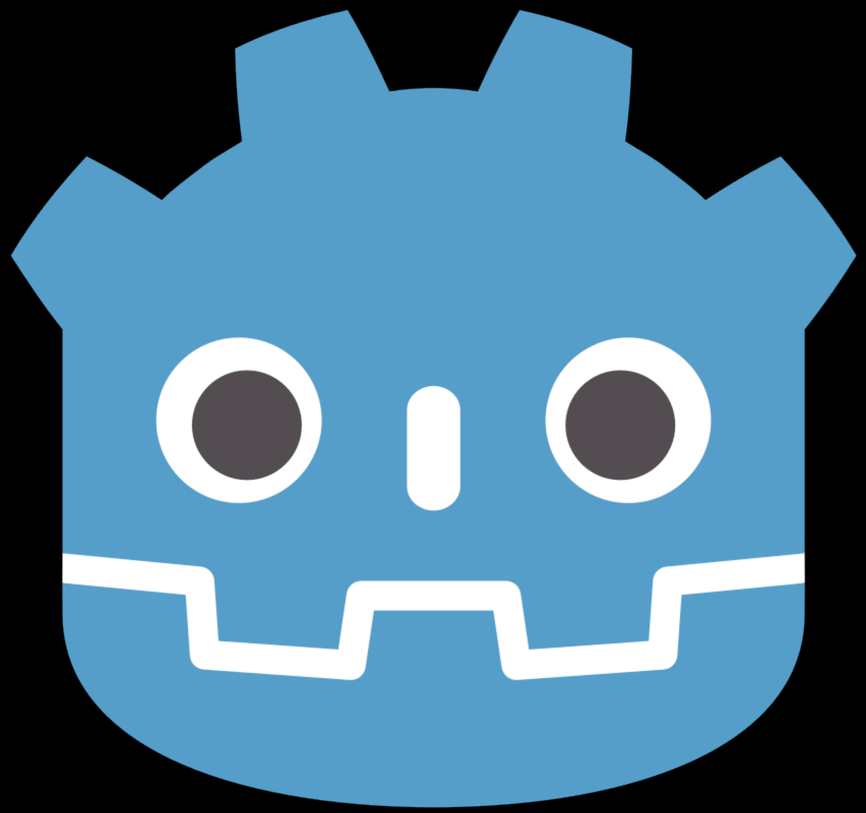
- I make video games
- **I like to make life difficult for myself**

Using React with the Godot Game Engine

React

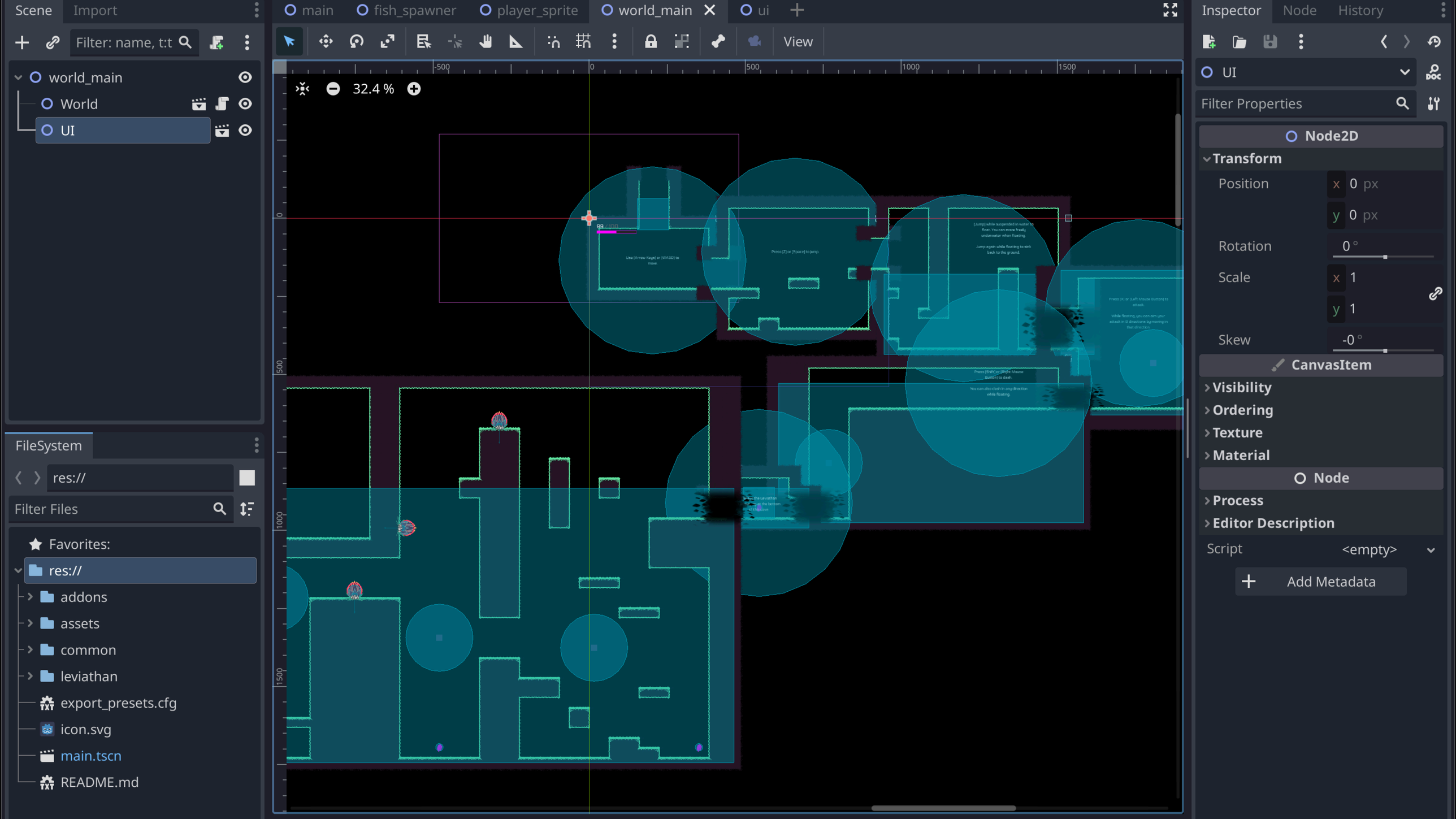
The library for web and native user interfaces

Learn React | API Reference

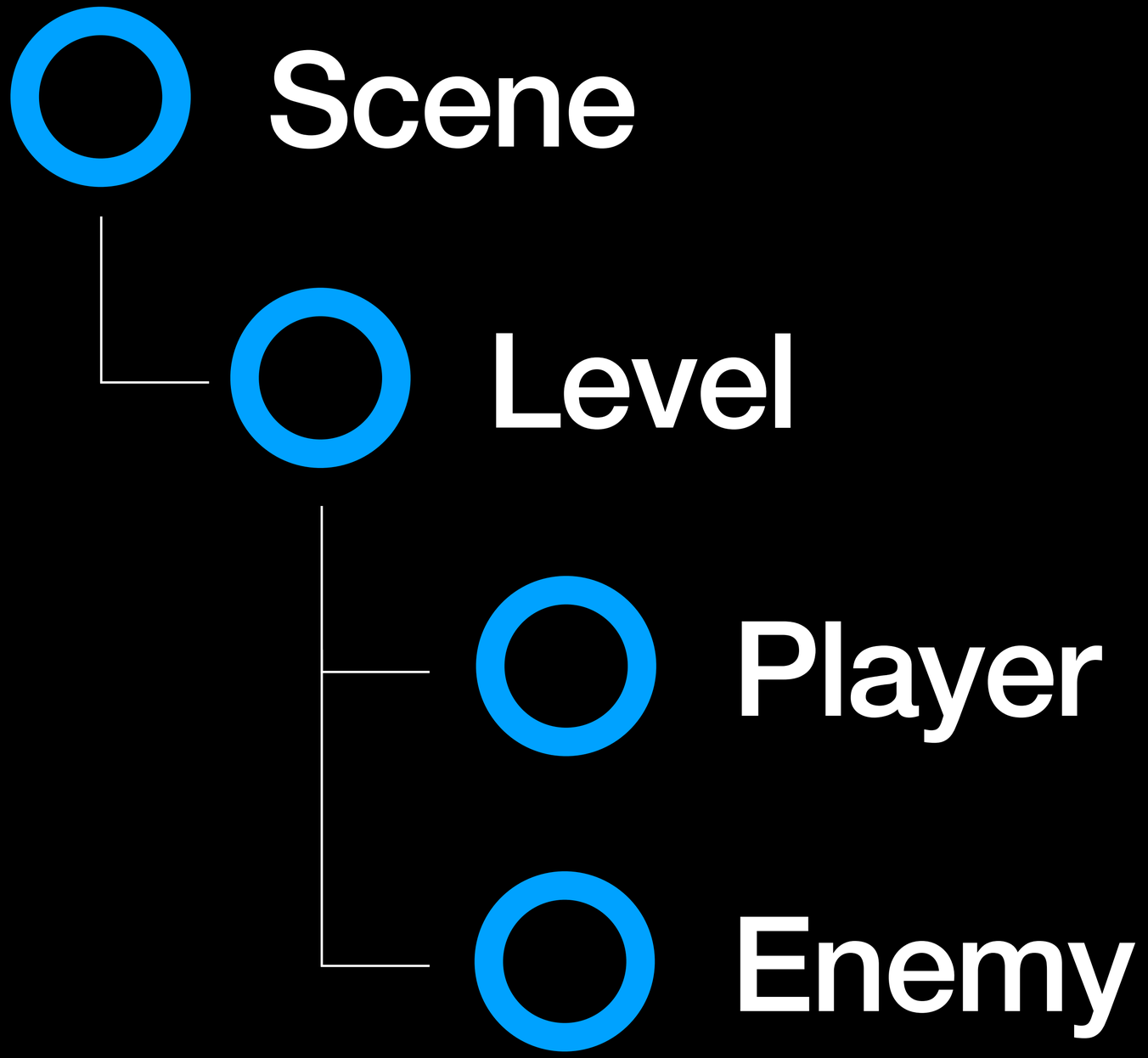


GODOT

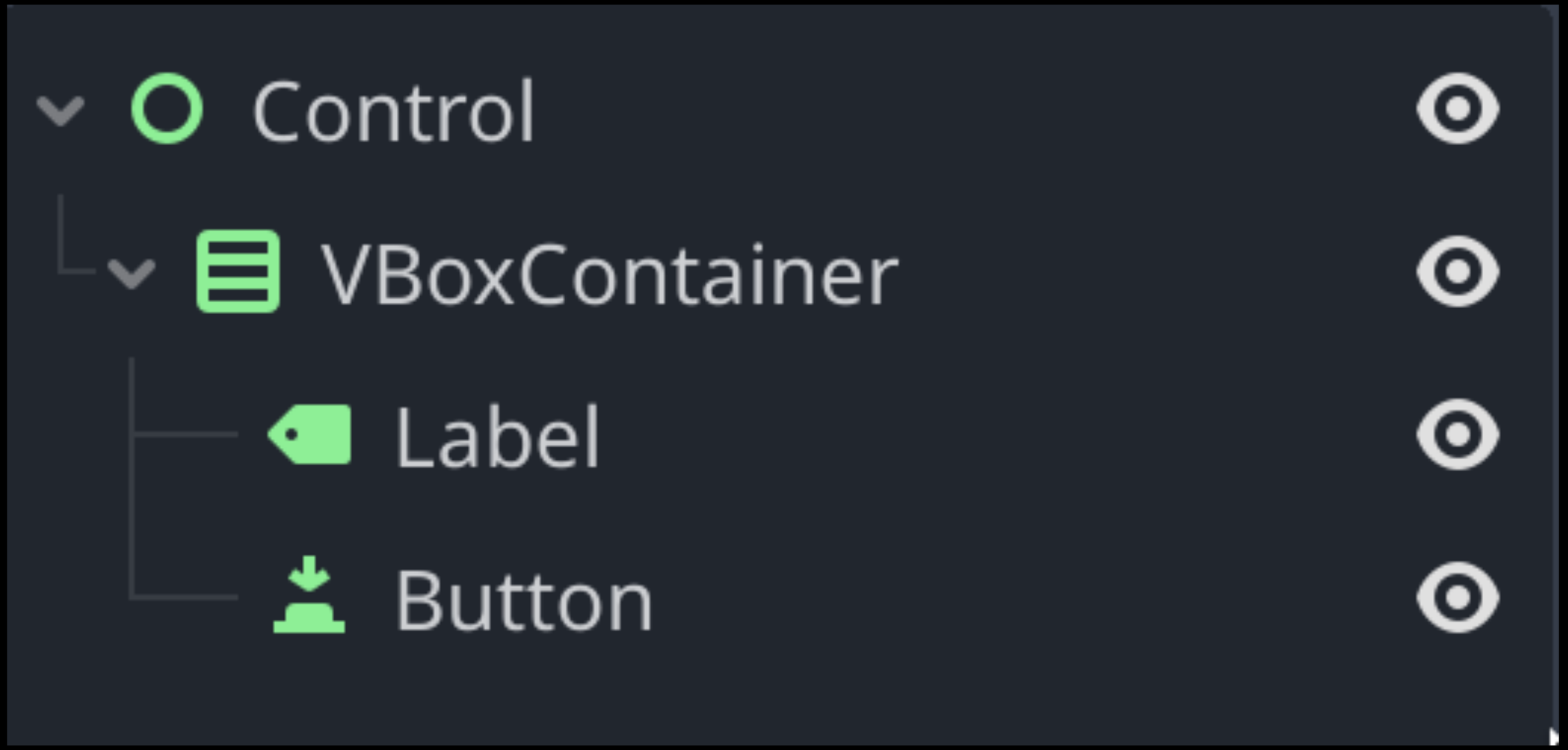
Game engine

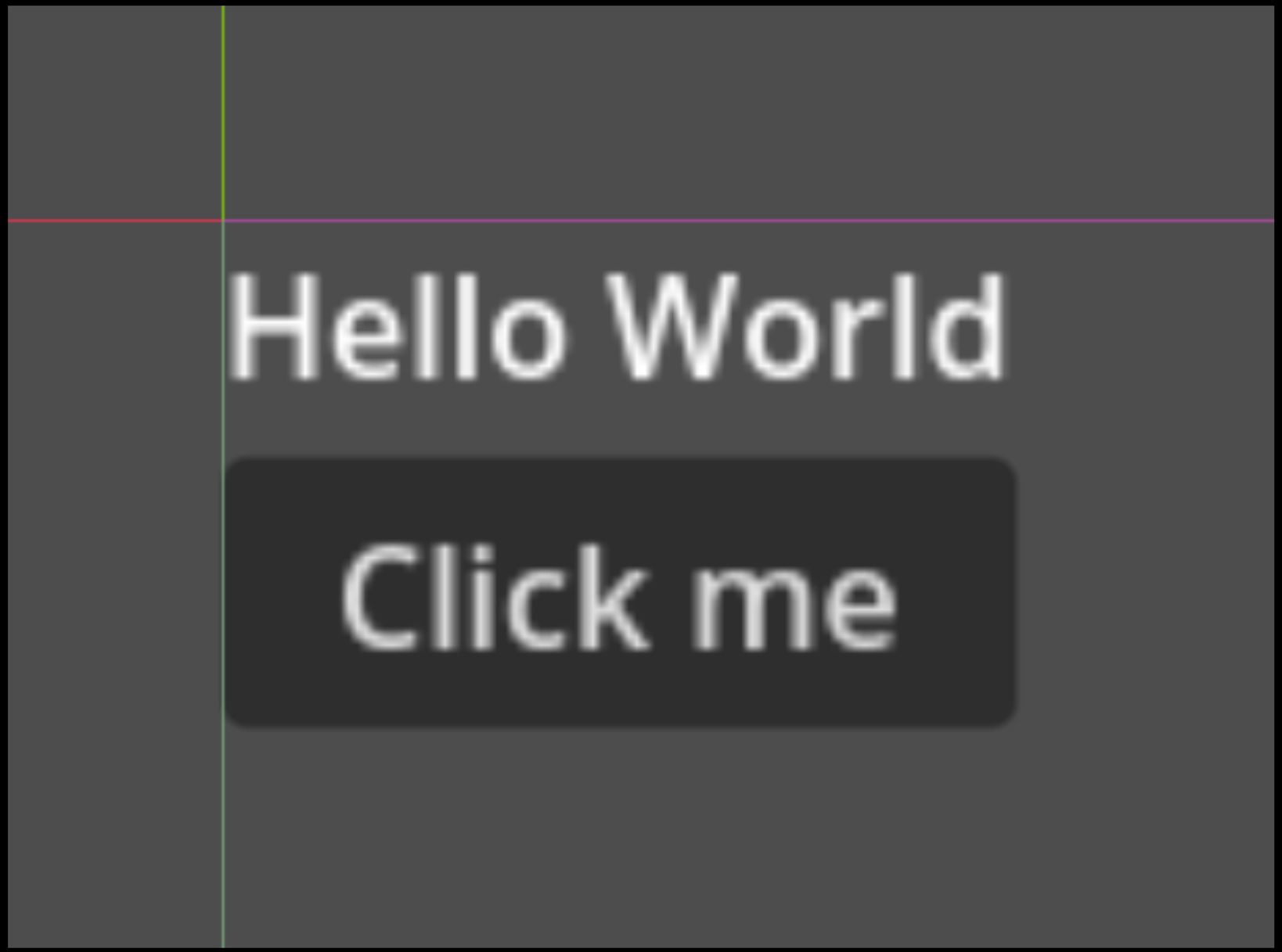

Scene
Import
Filter: name, t:t
world_main
World
UI
main
fish_spawner
player_sprite
world_main
ui
View
32.4 %
Inspector
Node
History
UI
Filter Properties
Node2D
Transform
Position
x 0 px
y 0 px
Rotation
0 °
Scale
x 1
y 1
Skew
-0 °
CanvasItem
Visibility
Ordering
Texture
Material
Node
Process
Editor Description
Script
<empty>
Add Metadata
FileSystem
res://
Filter Files
Favorites:
res://
addons
assets
common
leviathan
export_presets.cfg
icon.svg
main.tscn
README.md

(Yes, it’s not Javascript,
we’ll get there)

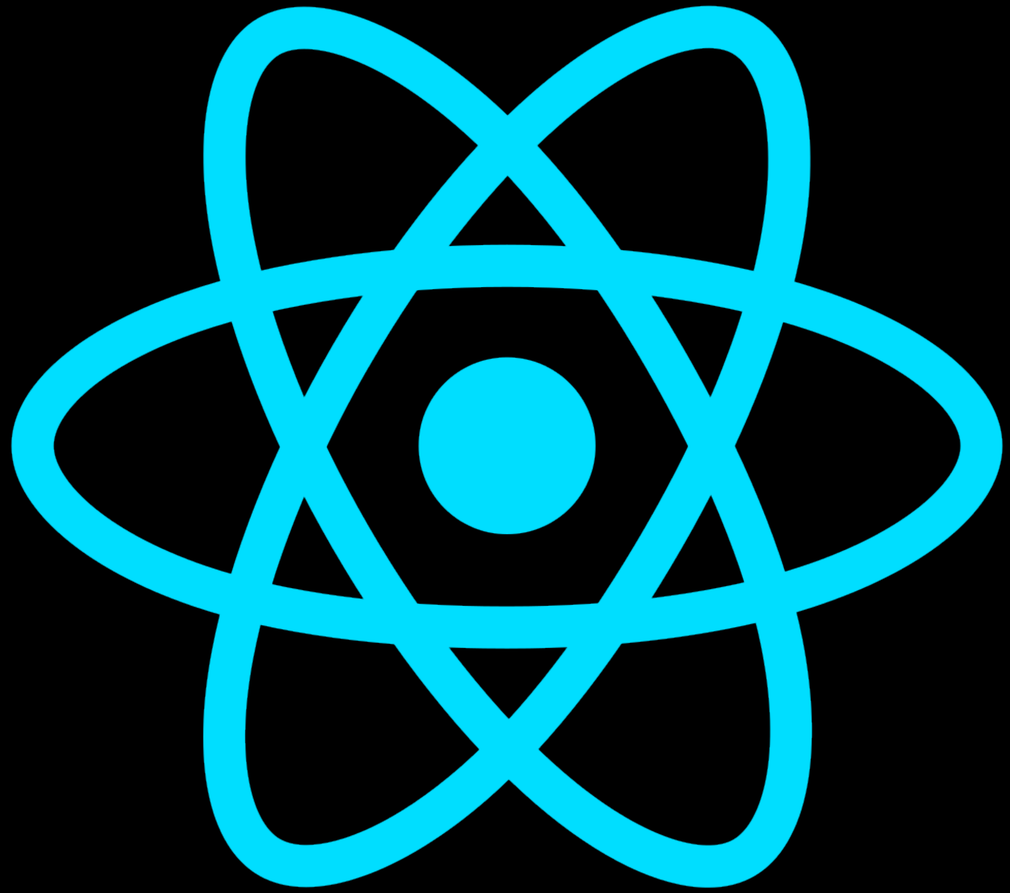

Scene
Level
Player
Enemy

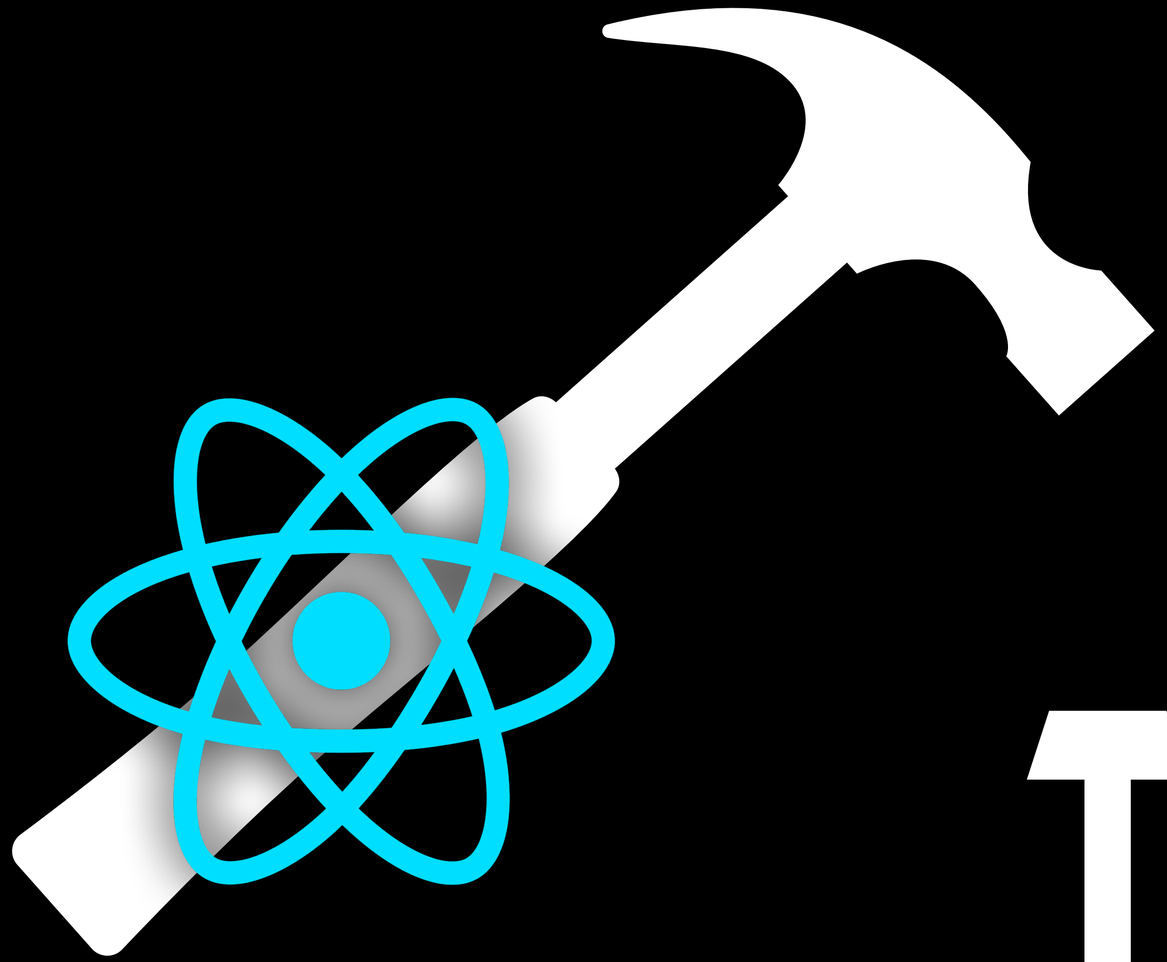
Battle
UnderLayer
Sprite2D
InputManager
PlayerNode
EnemyManager
Enemy1
Enemy2


Control
VBoxContainer
Label
Button


Hello World
Click me

```
<section>
    <div>
        <h1>Hello World</h1>
        <button>Click</button>
    </div>
</section>
```

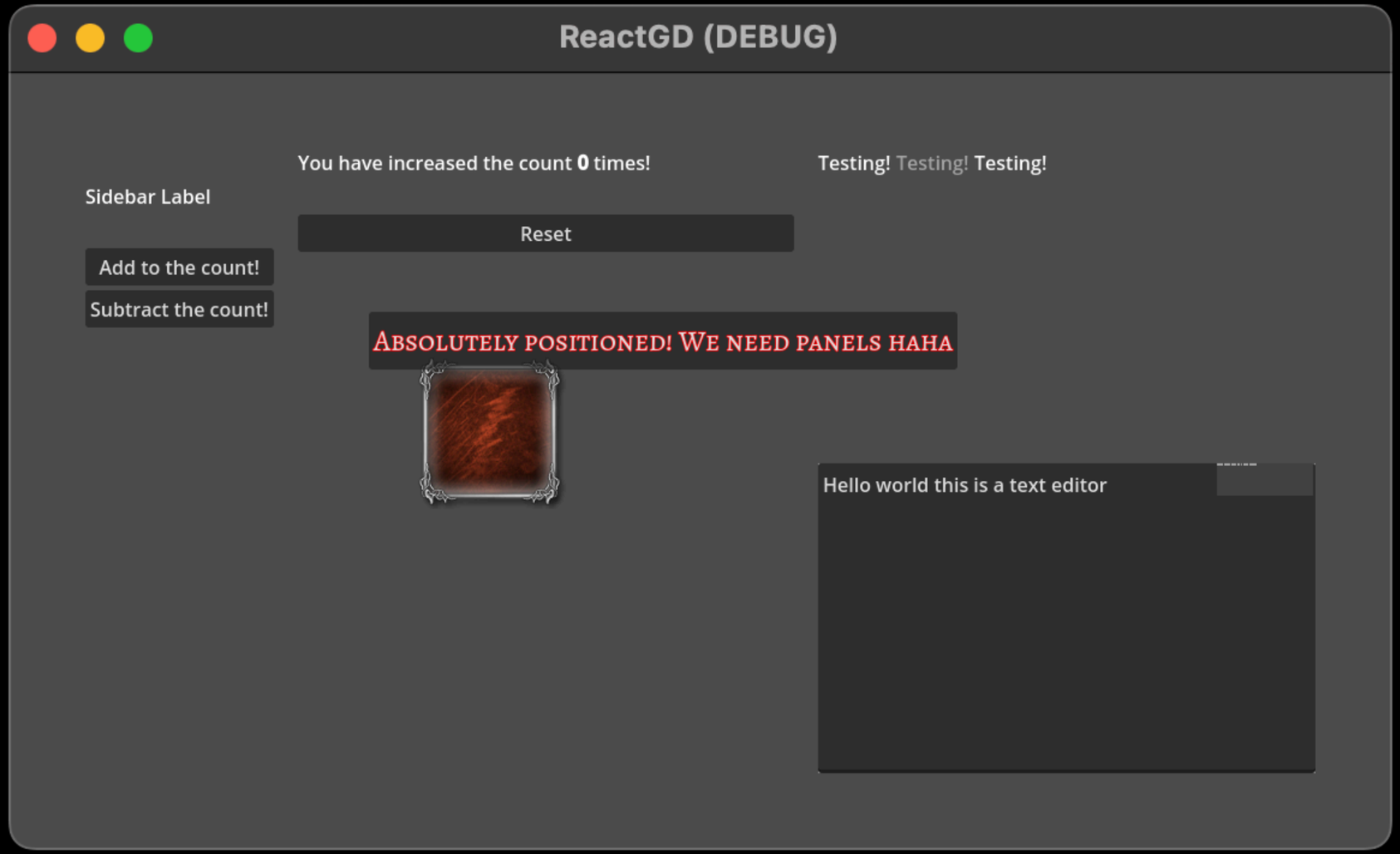




T T T

But we really can just do that

I'll do an actual demo later, right now I'm just saying you can



Problem: Godot code is written in C#

Solution: Just add Javascript to your C#!

V8
Original
100%
Vegetable Juice
FROM CONCENTRATE WITH
ADDED INGREDIENTS
60
CALORIES
PER CAN
2
FULL SERVINGS OF
Vegetables
NET 11.5 FL.OZ. (340mL)

ClearScript 7

Add scripting to your .NET applications quickly and easily.

```
_engine = new V8ScriptEngine();

_engine.AddHostType("GD", typeof(GD));
...

using var file = Godot.FileAccess.Open(
    "res://app/dist/index.js",
    Godot.FileAccess.ModeFlags.Read
);
_engine.Execute(file.GetAsText());
```

Ok great! But how does React talk to the engine?

React

The library for web and native user interfaces

Learn React | API Reference

It’s actually two libraries!

- React
- React-DOM

react -> core tools to create components and manage UI state

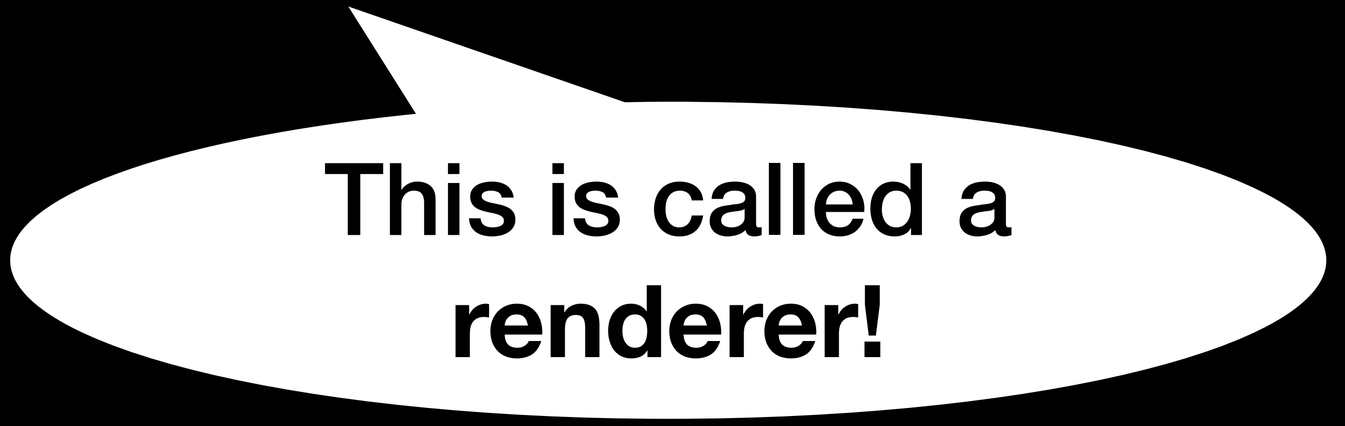
- Built-in components (Fragment, Suspense)
- Hooks (useEffect, useState)
- Functions (memo, forwardRef)

react-dom -> contains components and methods that can be rendered in a web DOM view

- HTML components (div, p, img)
- Hooks (useFormState)
- Functions (I’m not sure, I’ve never used them)

React -> Data management

React-dom -> Turns state into UI



More renderers!

- React-native
- React-three-fiber
- React-email
- Redocx

Let’s make a renderer for Godot UI elements!

react-godot !!!

Ok great! But how does React talk to the engine?

react-reconciler

react-reconciler

This is an experimental package for creating custom React renderers.

Its API is not as stable as that of React, React Native, or React DOM, and does not follow the common versioning scheme.

Use it at your own risk.

Ok that is way too long to fit in a slide but it’s actually documented in code

https://www.npmjs.com/package/@types/react-reconciler

```
const CustomReconciler = Reconciler({
    ...
    createInstance(type, props, root) {
      ...
    },
    // insert many more functions here
})
```

What do we actually need to implement?

- **createInstance**(type, props, rootContainer)
- **appendChild**(parentInstance, child)
- **removeChild**(parentInstance, child)
- **prepareUpdate**(instance, type, prevProps, nextProps)
- **commitUpdate**(instance, updatePayload, type, prevProps, nextProps, internalHandle)

Step 1: Create a DOM

```
public interface IDom
{
    public Node getNode();
    public void updateProps(ScriptObject newProps);

    public void clearChildren();
    public void appendChild(IDom node);
    public void removeChild(IDom node);
}
```

```
public partial class DomNode<T> : IDom
    where T : Godot.Node, new() {
    protected List<IDom> _children;
    protected T _instance;

    public void appendChild(IDom node) {
        _children.Add(node);
        _instance.AddChild(node.getNode());
    }
    public void removeChild(IDom node) {
        _children.Remove(node);
        _instance.RemoveChild(node.getNode());
        node.getNode().QueueFree();
    }
}
```

Step 2: Create the nodes for the DOM

```
public static IDom createElement(string type,
ScriptObject props, Document rootContainer) {
    IDom newNode;
    switch (type.ToLower()) {
        case "button":
            newNode = new DomNode<Button>();
            break;
        ...
    }
    return newNode;
}
```

```
const CustomReconciler = Reconciler({
    ...
    createInstance(type: string, props:
    ComponentProps<any>, rootContainer?: Document) {
        let element: IDom =
Document.createElement(type, props, rootContainer)
        return element
    },
})
```

Step 3: Hook it up to the reconciler API

```
const CustomReconciler = Reconciler({
    appendChild(parentInstance: IDom, child: IDom) {
        parentInstance.appendChild(child)
    },
    removeChild(parentInstance: IDom, child: IDom) {
        parentInstance.removeChild(child)
    },
  …
}
```

Now `<button>` will create a new Button node!

Step 4: Render with the custom reconciler

```
export const render = (element: any, container:
Document = root) => {
    const node = CustomReconciler.createContainer(
        container)
    CustomReconciler.updateContainer(element, node)
}
```

```
render(<button>Click</button>)
```

Ok so it all works now, right?

Nope!

`setTimeout()`

^ This is a problem

`setTimeout()`

- It’s not part of EMCAScript! It’s a browser API!
- But React relies on it!
- So we need to make a custom version! In C#!

Oh, all the setTimeout calls are with a time of 0, so I can just remove them right?

Nope!

Whyyyyyyyyyy?!

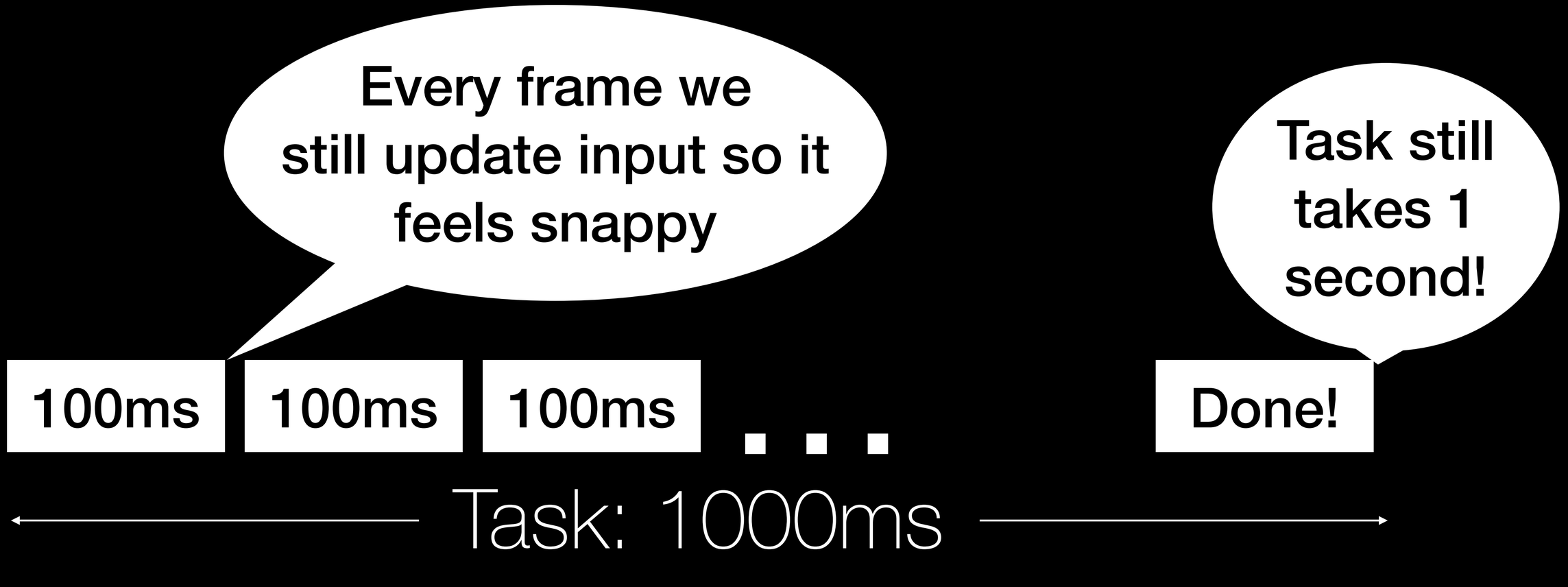
Time slicing!

What the heck is time slicing?

- Problem: long renders can cause the app to feel laggy

What the heck is time slicing?

- Solution: if a task has taken too long, **pause** and create a new frame and continue next frame!



setImmediate()

“This method is used to break up long running operations and run a callback function immediately after the browser has completed other operations such as events and display updates.”

IE and
Node.JS only

~~setImmediate()~~

“This method is used to break up long running operations and run a callback function immediately after the browser has completed other operations such as events and display updates.”

```
setTimeout(task, 0)
```

^ Ok but we never actually fixed the problem???

This is also a bit too long to fit on a slide, oops.

But someone wrote a SetTimeout implementation in the ClearScript issues so I didn’t have to do it myself: **https://github.com/microsoft/ClearScript/issues/475**

```
let queue = [], nextId = 0, maxId = 100000000000
const getNextId = () => (nextId = (nextId % maxId) + 1)
const add = (entry) => {
    // add a new entry to the queue
}
function set(periodic, func, delay) {
    add({ id, periodic, func, delay, due: now + delay })

    // called in C#
    impl.Schedule(queue[0].due - now)
    return id
}
global.setTimeout = set.bind(undefined, false)
```

```
public async void Schedule(double delay) {
    if (delay < 0) {
        if (_token != null) {
            _token.Cancel();
            _token = null;
        }
    } else if (delay == 0) {
        Schedule(_callback());
    } else {
        _token ??= new System.Threading.CancellationTokenSource();
        try {
            await Task.Delay((int)delay, _token.Token);
            Schedule(_callback());
        } catch (TaskCanceledException) {
            GD.Print("we canceled a task!");
        }
    }
}
```

Ok after all that does it work now?

Yes!!

“So, DM, did you actually make a game with all this?”

No!

(Remember what I said about making my life difficult?)

This code is 100% not ready for production

Thank you!

Demo!

…do I even have any time left for this