Using the Gamepad API for a better gaming experience on the web

Arun Michael Dsouza

arunmichaeldsouza.com
@amdsouza92

JS GameDev summit

arunmichaeldsouza.com

The Web Platform

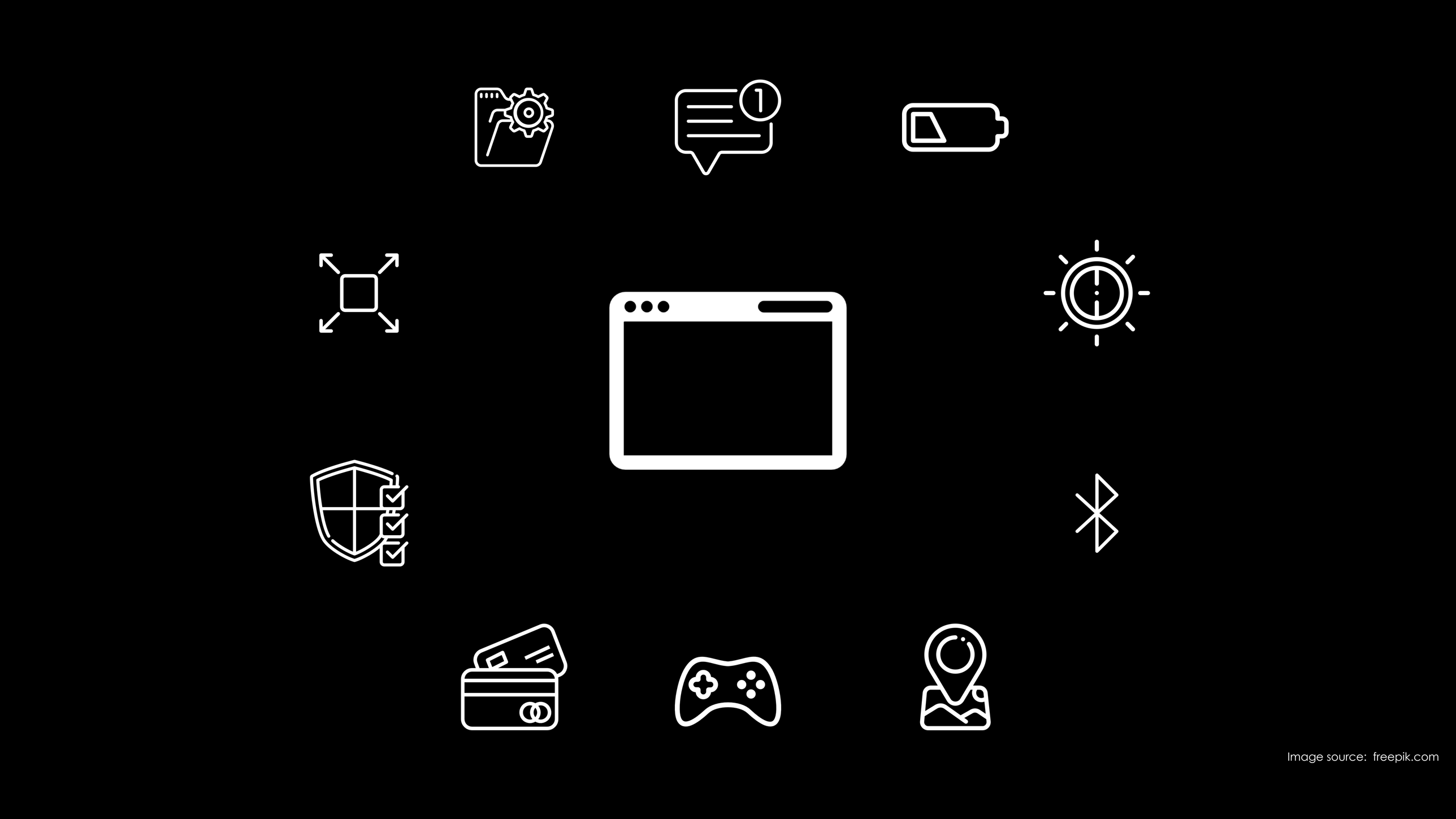
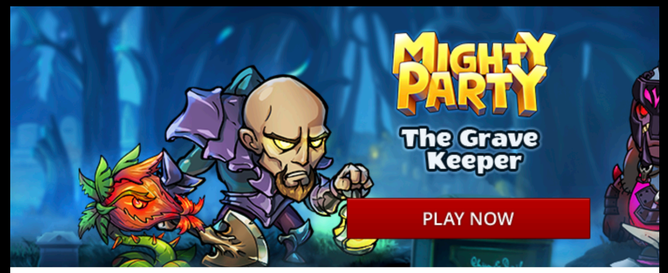
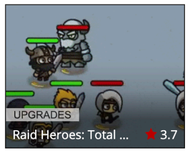
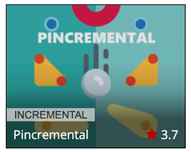


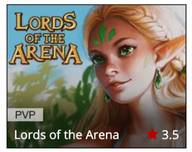
Image source: freepik.com

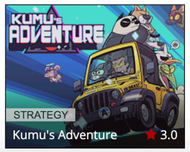


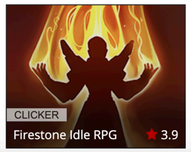
FEATURED GAMES (see all)











Menu

- Main Page
- Game Consoles
- Online Emulators
- Offline Emulators
- Games Control
- Random Game

Games

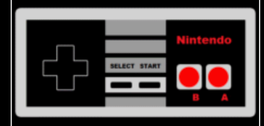
- Adventure (79)
- Action (933)
- Fighting (88)
- Puzzle (139)
- Platform (387)
- Adult (47)
- RPG (68)
- Simulation (48)
- Sport (115)
- Strategy (40)
- Shooter (294)
- Racing (84)
- Others (25)
- All games (1388)

Others

- TOP 100
- Videos
- Glossary
- Help/FAQ

Contra - Nintendo NES system

NES gamepad:



Gamepad control:

	Player 1:	Player 2:
↑	↑	-
↓	↓	-
←	←	-
→	→	-
A	X	-
B	Z	-
SELECT	Shift	-
START	Enter	-

Emulator selection:

The following emulators are available for this game: NeptunJS (JavaScript), Nesbox (Flash), Retro-Games (JS) and vNES (Java).

Rate this game:

Average game rating: 72% Voted: 7180x Played: 972426x

Other platforms:

Unfortunately, this game is cur-

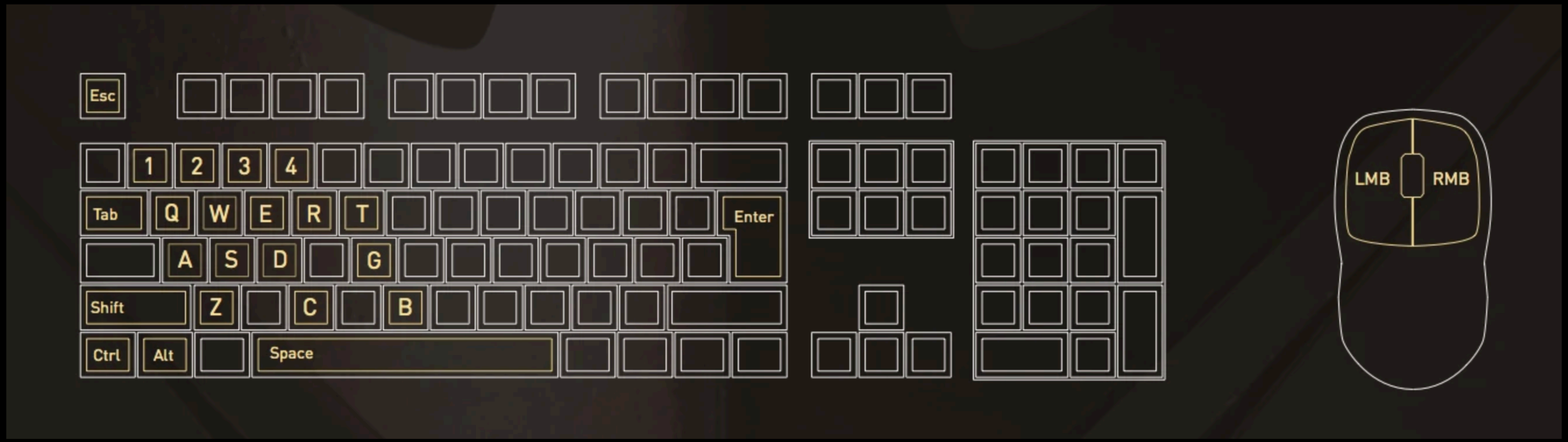
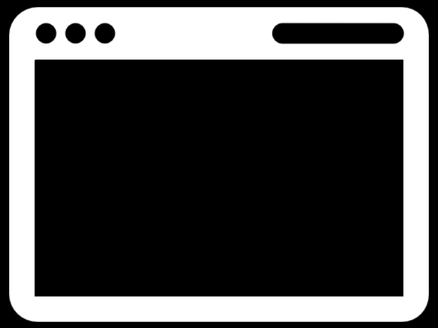


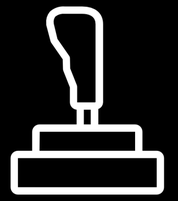
Image source: nocookie.net

Image source: mobilefreetoplay.com, windowscentral.com

Gamepad API

bit.ly/gamepad_api





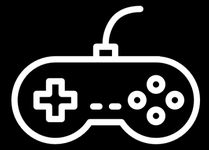


Image source: freepik.com


Y
X
B
A


X
Y
A
B


SHARE
OPTIONS

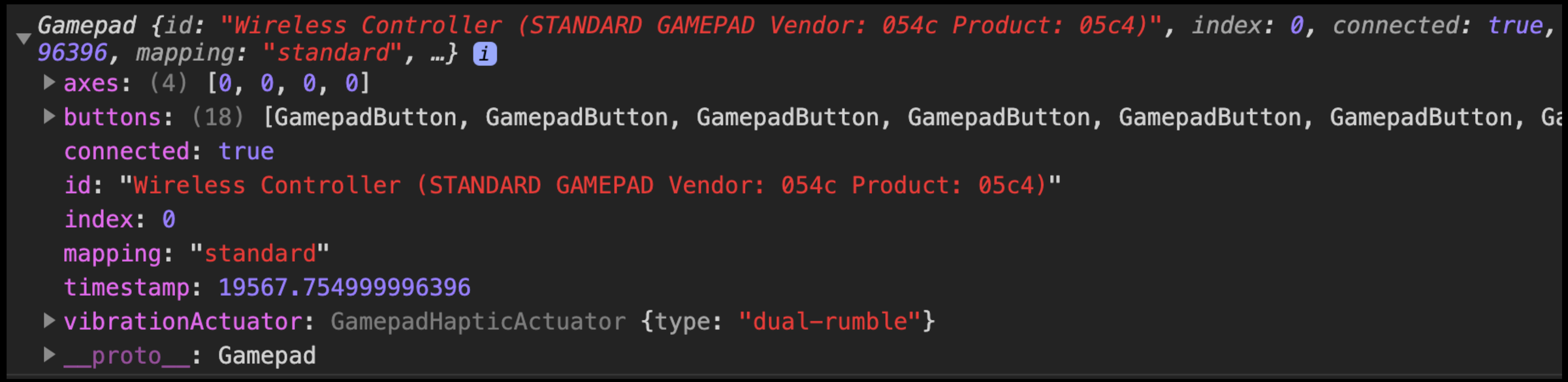
Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
4-20																
21-24		3.1-10	2-28	10-23			3.2-10.2									
25-115	12-114	10.1-16.4	29-115	24-100	6-10		10.3-16.4	4-20		12-12.1		2.1-4.4.4				2.5
116	115	16.5	116	101	11	115	16.5	21	all	73	15.5	115	115	13.1	13.18	3.1
117-119		16.6-TP	117-119				16.6-17									

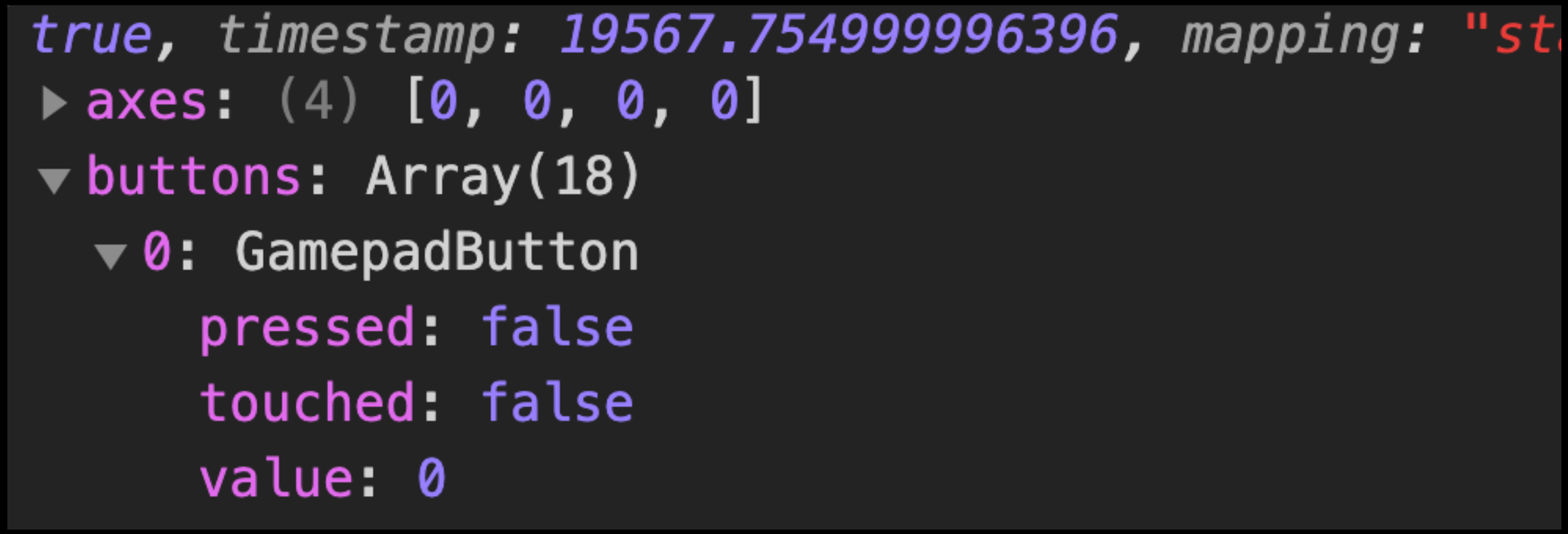
Image source: caniuse.com

```
window.navigator.getGamepads()
▶ (4) [Gamepad, null, null, null]
```

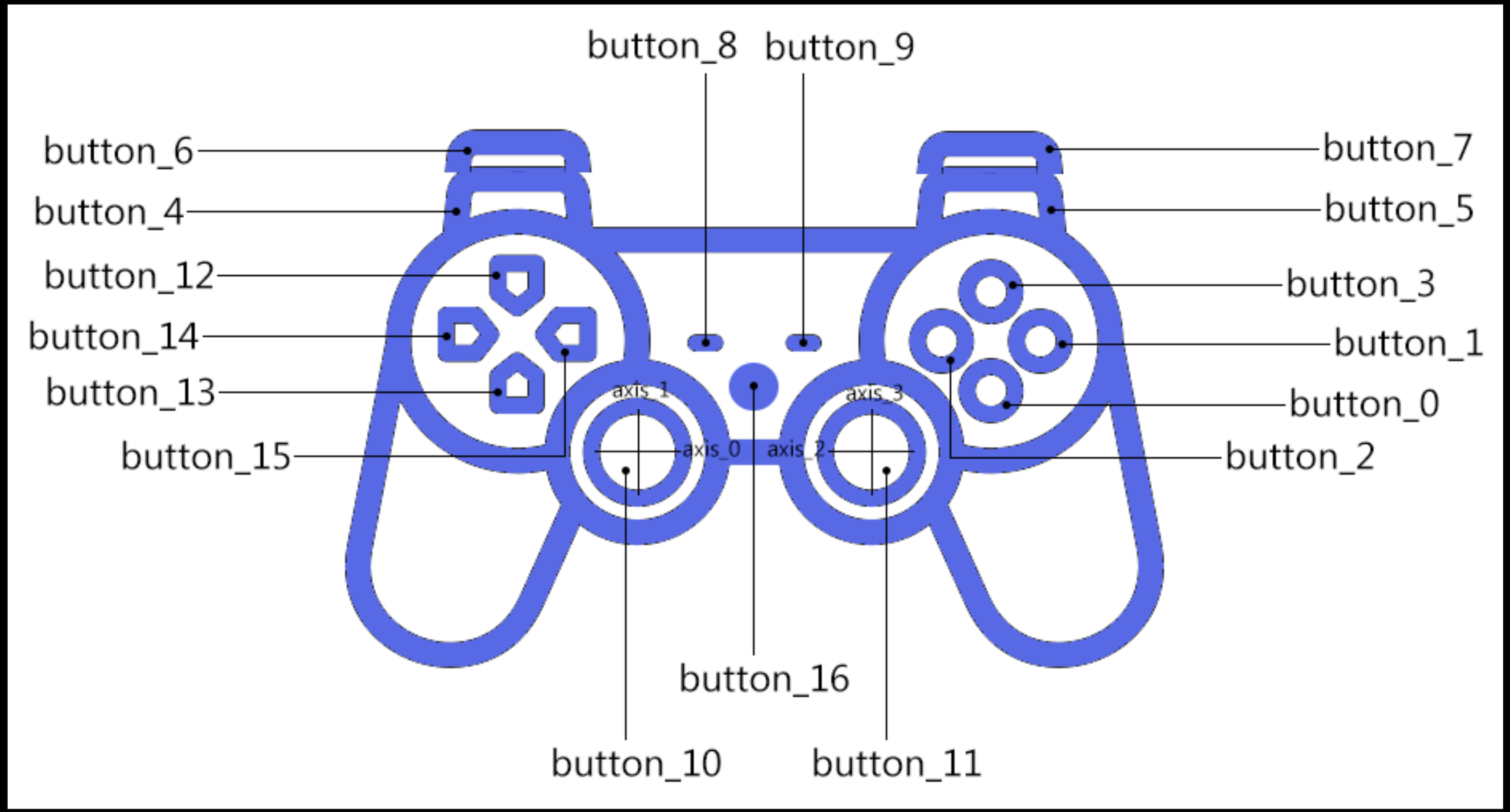
Gamepad Interface

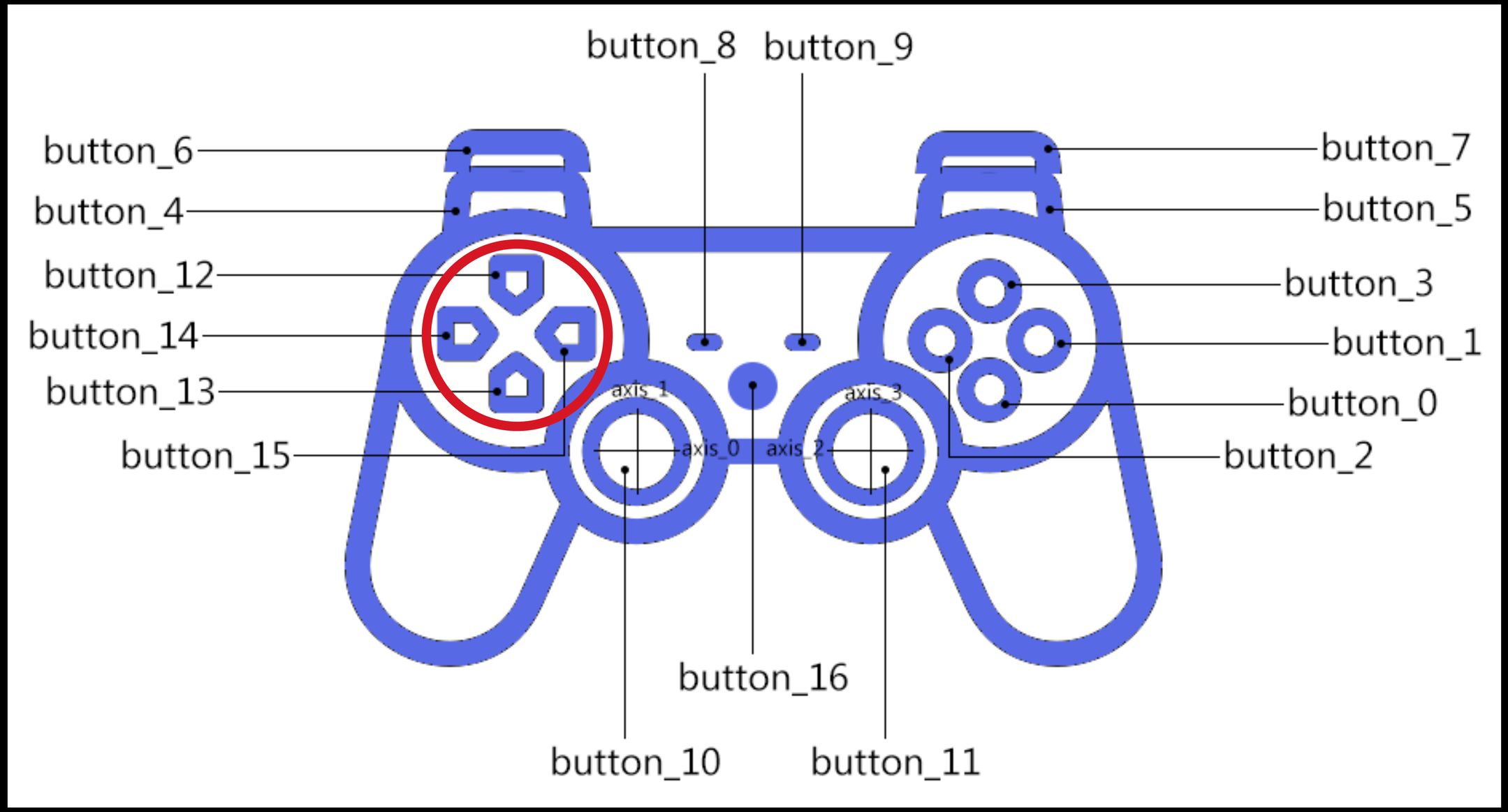
bit.ly/gamepad_interface

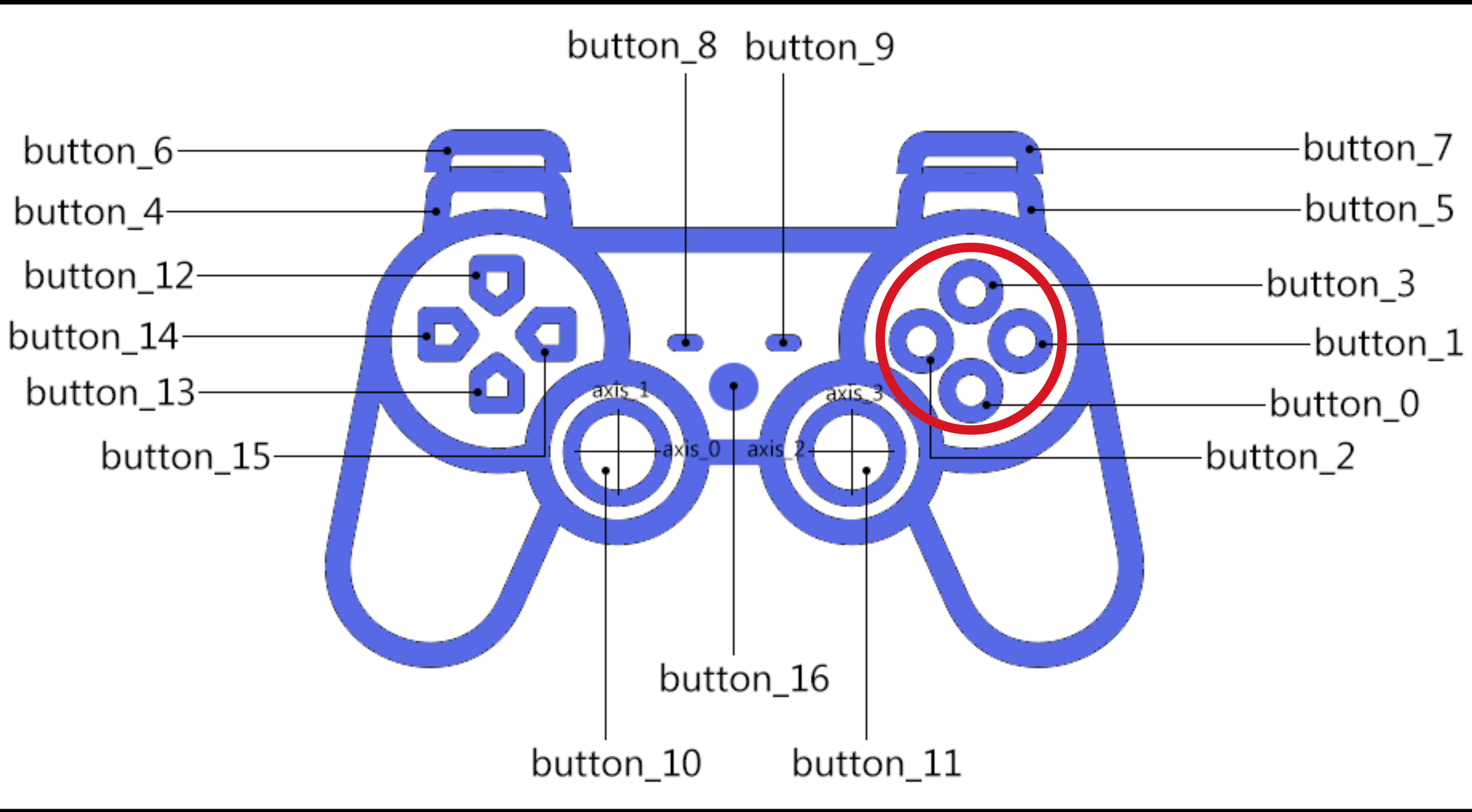

Gamepad {id: "Wireless Controller (STANDARD GAMEPAD Vendor: 054c Product: 05c4)", index: 0, connected: true,
96396, mapping: "standard", …}
axes: (4) [0, 0, 0, 0]
buttons: (18) [GamepadButton, GamepadButton, GamepadButton, GamepadButton, GamepadButton, GamepadButton, Ga
connected: true
id: "Wireless Controller (STANDARD GAMEPAD Vendor: 054c Product: 05c4)"
index: 0
mapping: "standard"
timestamp: 19567.754999996396
vibrationActuator: GamepadHapticActuator {type: "dual-rumble"}
__proto__: Gamepad

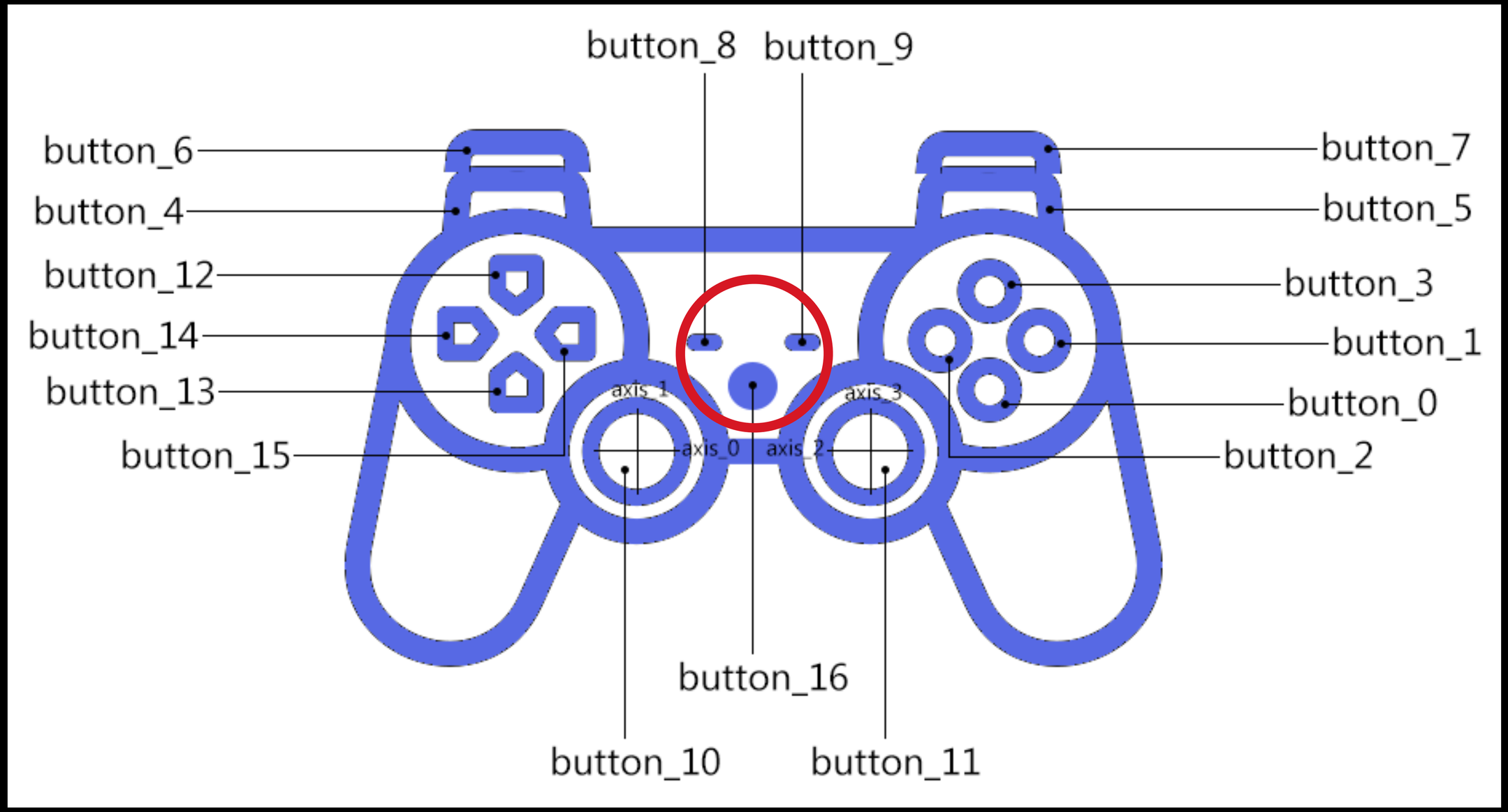

true, timestamp: 19567.754999996396, mapping: "st
axes: (4) [0, 0, 0, 0]
buttons: Array(18)
0: GamepadButton
pressed: false
touched: false
value: 0

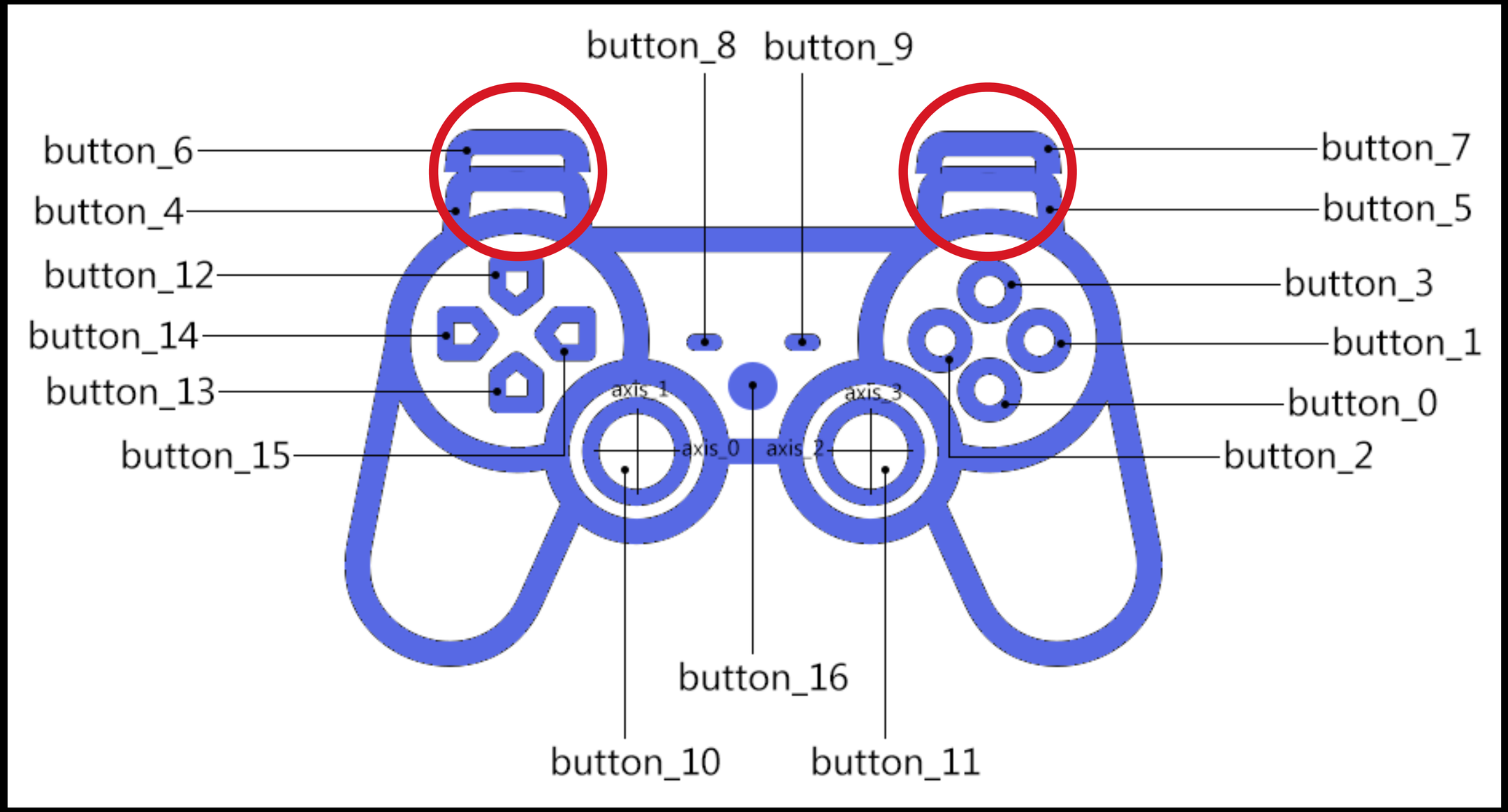
Standard Gamepad Layout

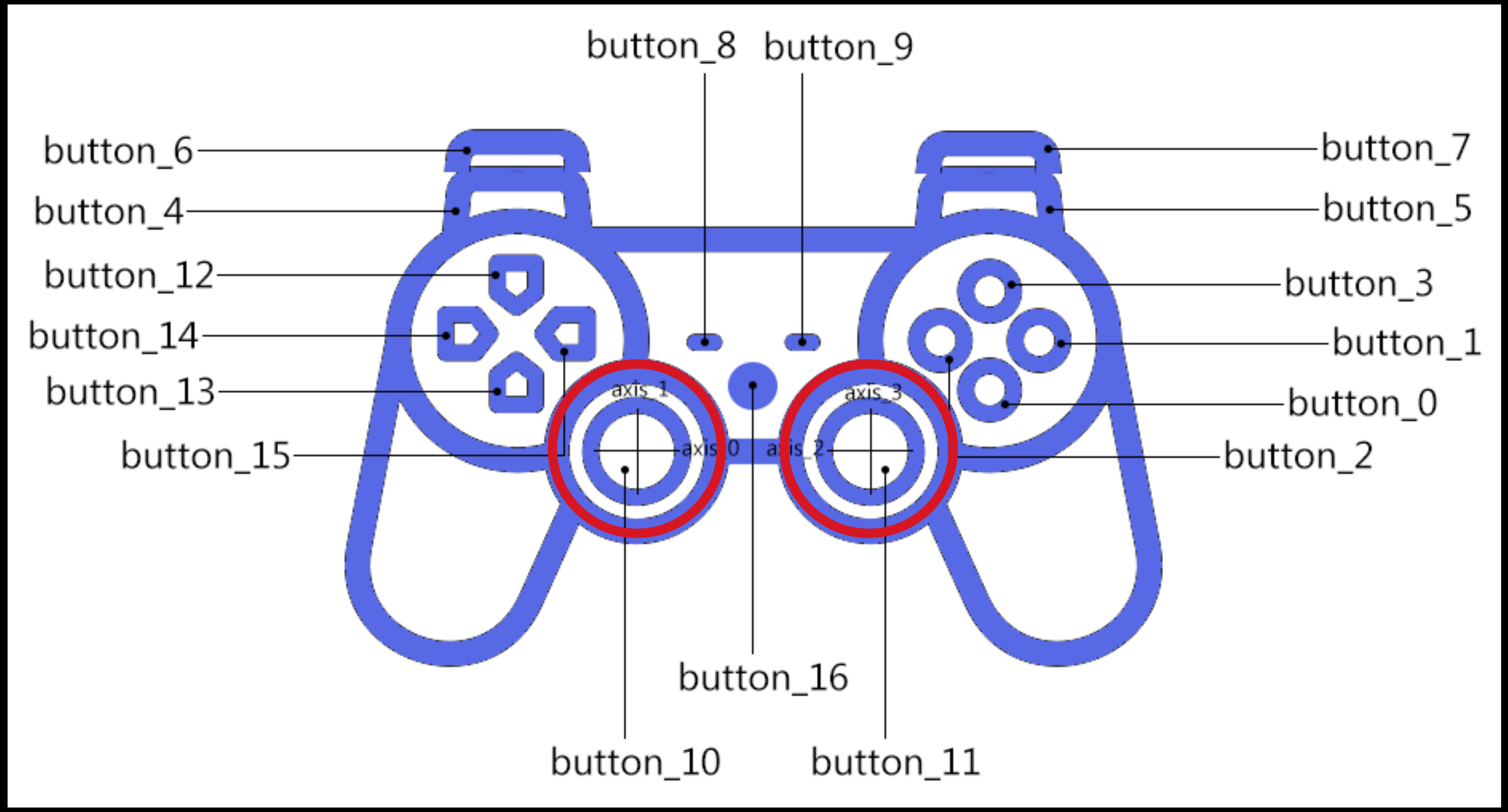

button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11

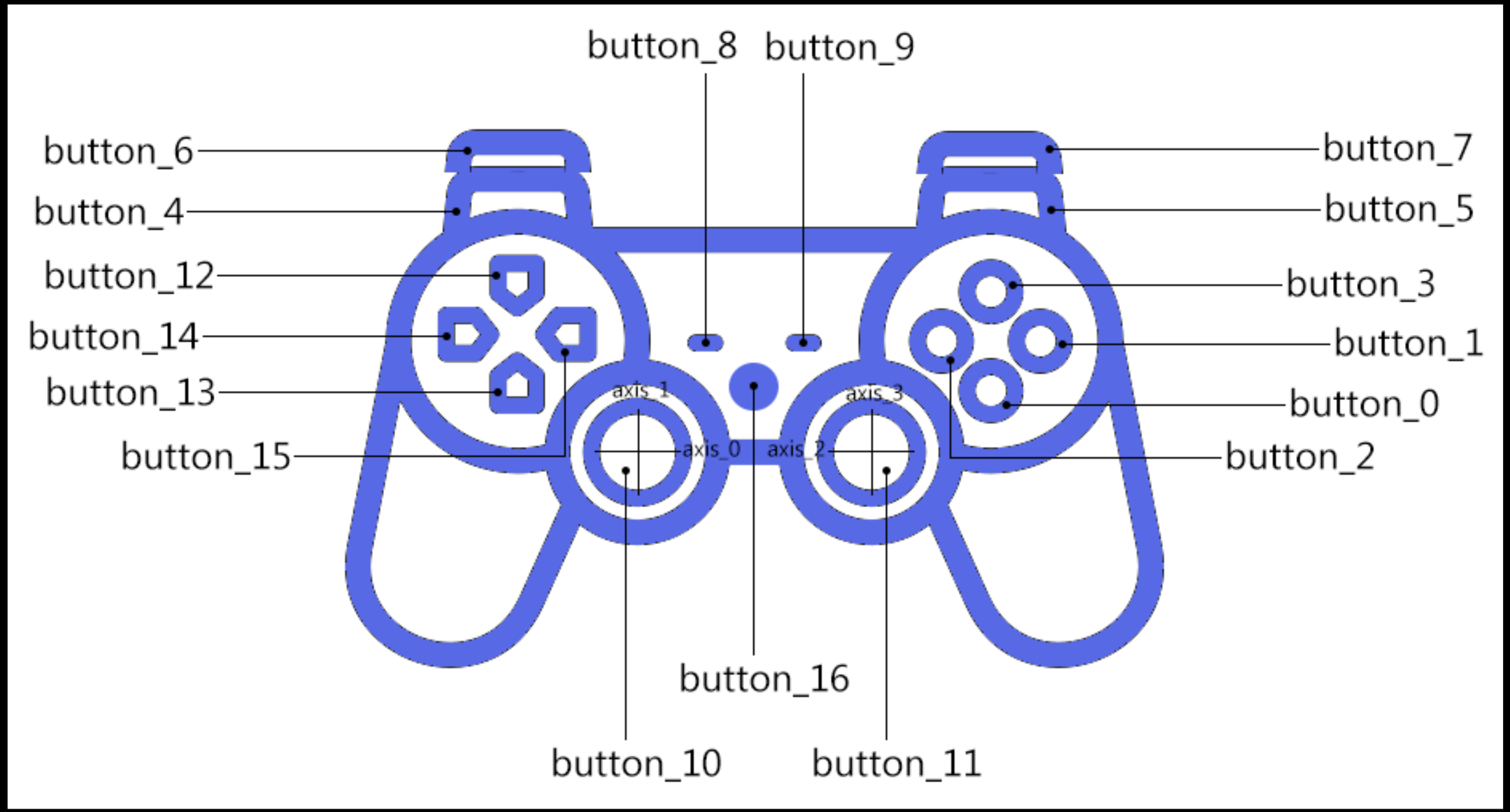

button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11


button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11


button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11


button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11


button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11


button_8
button_9
button_6
button_7
button_4
button_5
button_12
button_3
button_14
button_1
button_13
button_0
button_15
button_2
axis_1
axis_3
axis_0
axis_2
button_16
button_10
button_11

Events?

```
window.addEventListener("gamepadconnected", function(e) {
    // Info of connected gamepad
    console.log(e.gamepad);
});
```

bit.ly/gamepad_connected_event

```
window.addEventListener("gamepaddisconnected", function(e) {
    // Info of disconnected gamepad
    console.log(e.gamepad);
});
```

bit.ly/gamepad_disconnected_event

```
window.addEventListener("gamepadbuttonpressed", function(e) {
    // Cannot detect button press
});
```

```
setInterval(() => console.log(window.navigator.getGamepads()[0].buttons[0]), 10);
```

```
▶ GamepadButton {pressed: true, touched: true, value: 1}
▶ GamepadButton {pressed: true, touched: true, value: 1}
▶ GamepadButton {pressed: true, touched: true, value: 1}
▶ GamepadButton {pressed: true, touched: true, value: 1}
▶ GamepadButton {pressed: false, touched: false, value: 0}
▶ GamepadButton {pressed: false, touched: false, value: 0}
▶ GamepadButton {pressed: false, touched: false, value: 0}
▶ GamepadButton {pressed: false, touched: false, value: 0}
▶ GamepadButton {pressed: false, touched: false, value: 0}
```

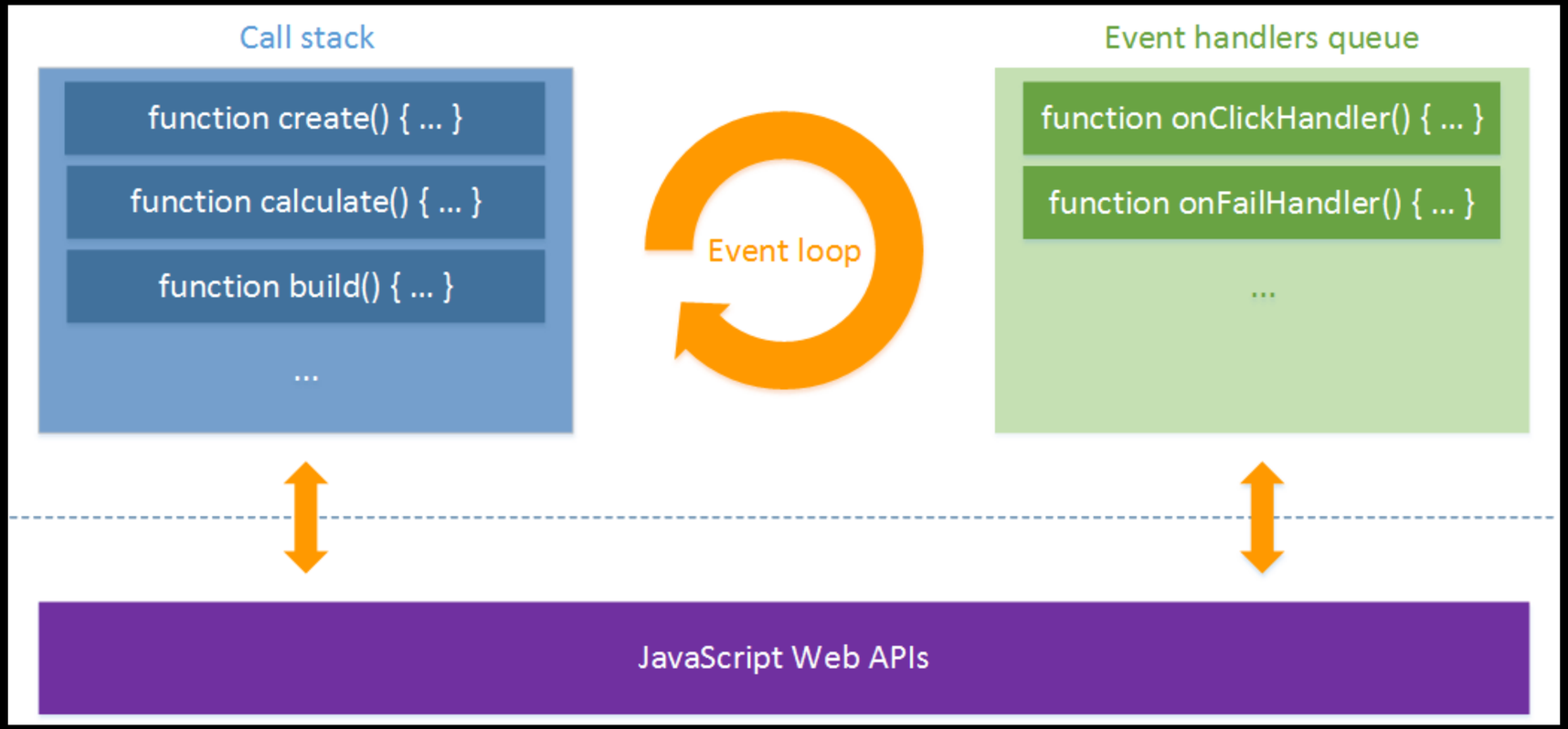
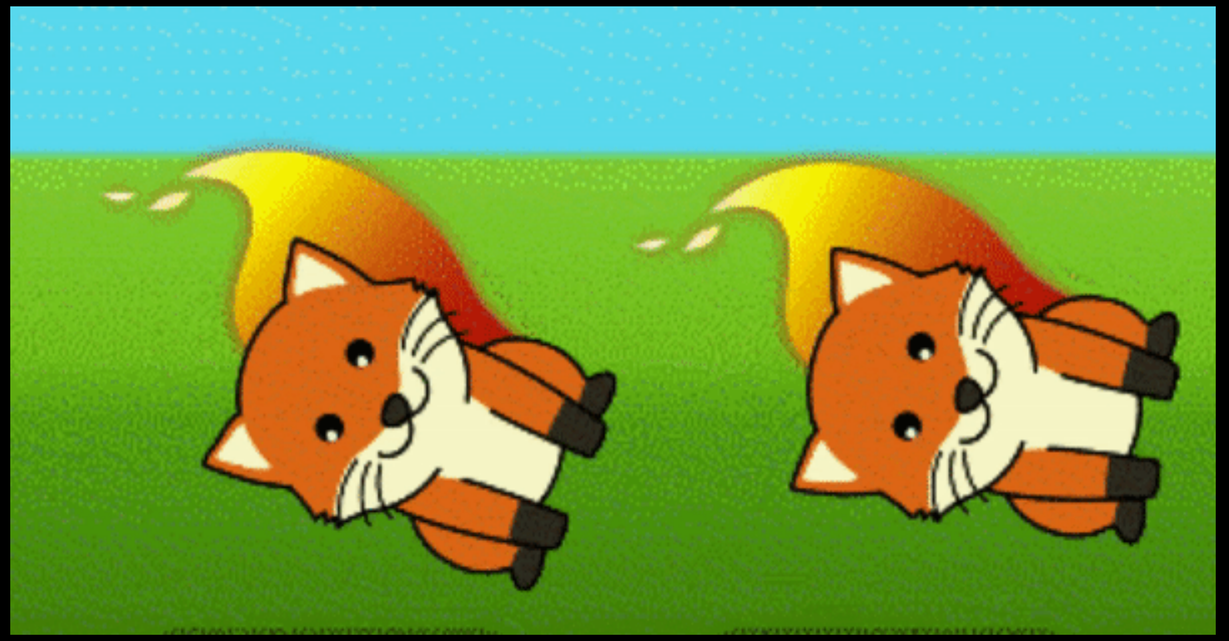


Image source: red-gate.com/



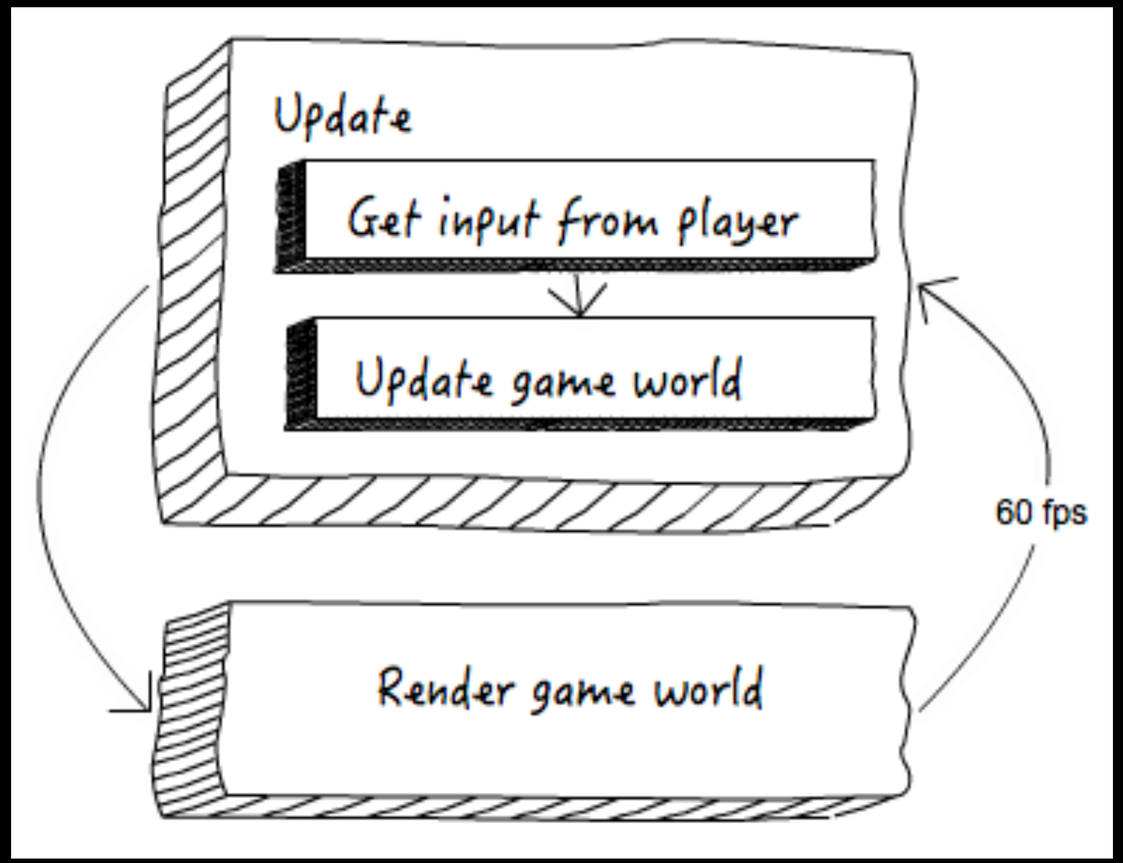
bit.ly/javascript_layout_thrashing

Image source: hacks.mozilla.org

requestAnimationFrame( )

```
const loop = () => {
    console.log(navigator.getGamepads()[0].buttons[0]);
    requestAnimationFrame(loop);
}

loop();
```

The Game Loop



bit.ly/anatomy_of_a_videogame

Image Source: mozdevs.github.io

```
const loop = {
    id: null,
    start: function () {
        // Get all the gamepads
        let gamepads = window.navigator.getGamepads();

        // Loop through the gamepads
        gamepads.forEach(gamepad => {
            if (gamepad) {
                // Listen to button press events
                listenToButtonEvents(gamepad);
                // Listen to axis movement events
                listenToAxisMovements(gamepad);
            }
        });
        // Call the start function on each frame
        this.id = window.requestAnimationFrame(this.start.bind(this));
    },
    stop: function (id) {
        return window.cancelAnimationFrame(id);
    }
};
```

```
const loop = {
    id: null,
    start: function () {
        // Get all the gamepads
        let gamepads = window.navigator.getGamepads();

        // Loop through the gamepads
        gamepads.forEach(gamepad => {
            if (gamepad) {
                // Listen to button press events
                listenToButtonEvents(gamepad);
                // Listen to axis movement events
                listenToAxisMovements(gamepad);
            }
        });
        // Call the start function on each frame
        this.id = window.requestAnimationFrame(this.start.bind(this));
    },
    stop: function (id) {
        return window.cancelAnimationFrame(id);
    }
};
```

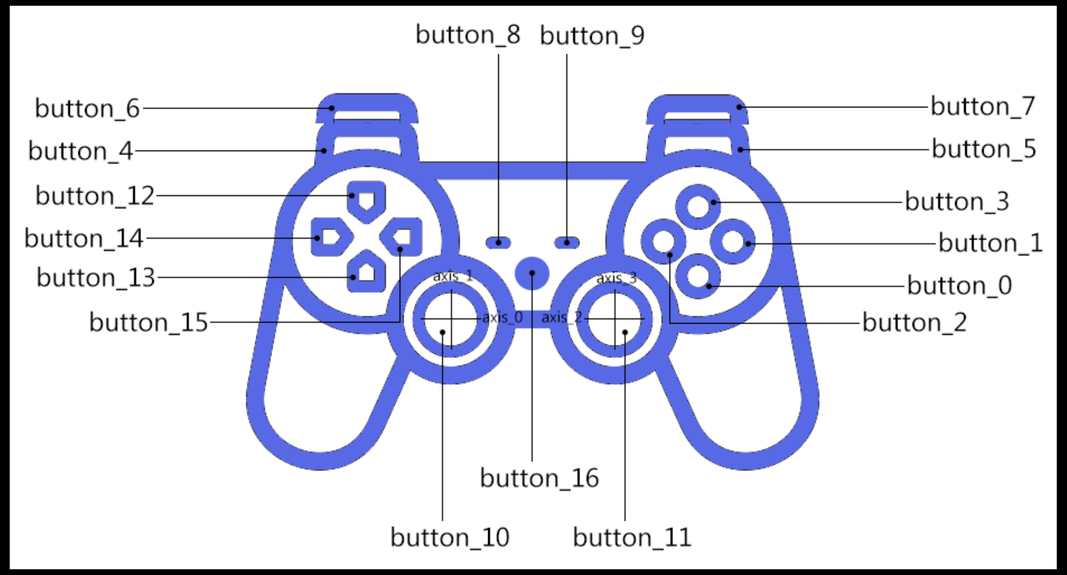
```
const loop = {
    id: null,
    start: function () {
        // Get all the gamepads
        let gamepads = window.navigator.getGamepads();

        // Loop through the gamepads
        gamepads.forEach(gamepad => {
            if (gamepad) {
                // Listen to button press events
                listenToButtonEvents(gamepad);
                // Listen to axis movement events
                listenToAxisMovements(gamepad);
            }
        });
        // Call the start function on each frame
        this.id = window.requestAnimationFrame(this.start.bind(this));
    },
    stop: function (id) {
        return window.cancelAnimationFrame(id);
    }
};
```

```
const loop = {
    id: null,
    start: function () {
        // Get all the gamepads
        let gamepads = window.navigator.getGamepads();

        // Loop through the gamepads
        gamepads.forEach(gamepad => {
            if (gamepad) {
                // Listen to button press events
                listenToButtonEvents(gamepad);
                // Listen to axis movement events
                listenToAxisMovements(gamepad);
            }
        });
        // Call the start function on each frame
        this.id = window.requestAnimationFrame(this.start.bind(this));
    },
    stop: function (id) {
        return window.cancelAnimationFrame(id);
    }
};
```

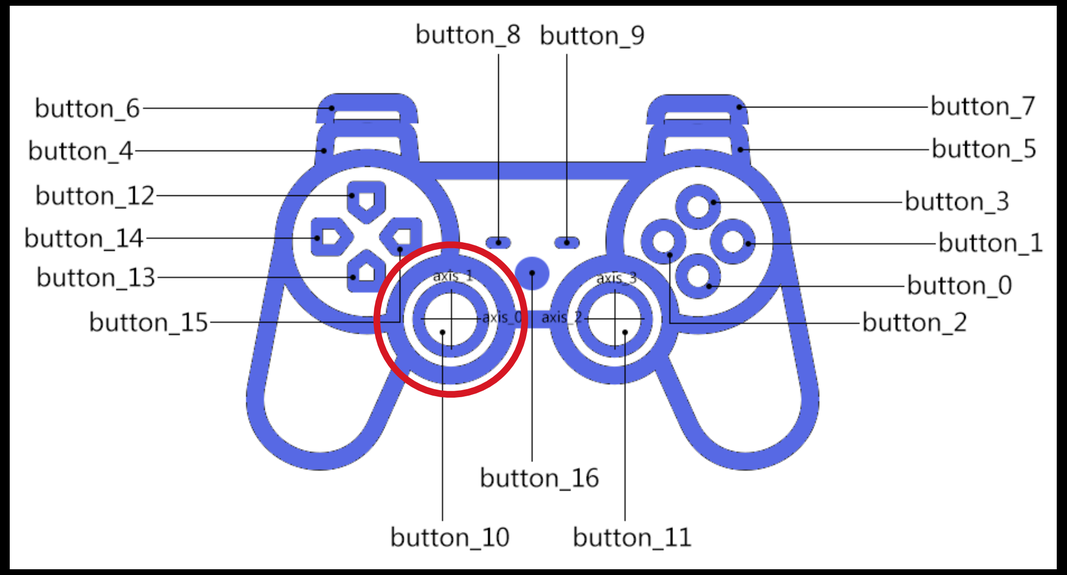
```
const loop = {
    id: null,
    start: function () {
        // Get all the gamepads
        let gamepads = window.navigator.getGamepads();

        // Loop through the gamepads
        gamepads.forEach(gamepad => {
            if (gamepad) {
                // Listen to button press events
                listenToButtonEvents(gamepad);
                // Listen to axis movement events
                listenToAxisMovements(gamepad);
            }
        });
        // Call the start function on each frame
        this.id = window.requestAnimationFrame(this.start.bind(this));
    },
    stop: function (id) {
        return window.cancelAnimationFrame(id);
    }
};
```

```
const listenToButtonEvents = gamepad => {
    // Loop through the gamepad buttons
    gamepad.buttons.forEach(button => {
        // Check if button is pressed and dispatch event
        if (button.pressed) {
            window.dispatchEvent(buttonPressEvent(button));
        } else {
            // Handle button release
        }
    });
};
```

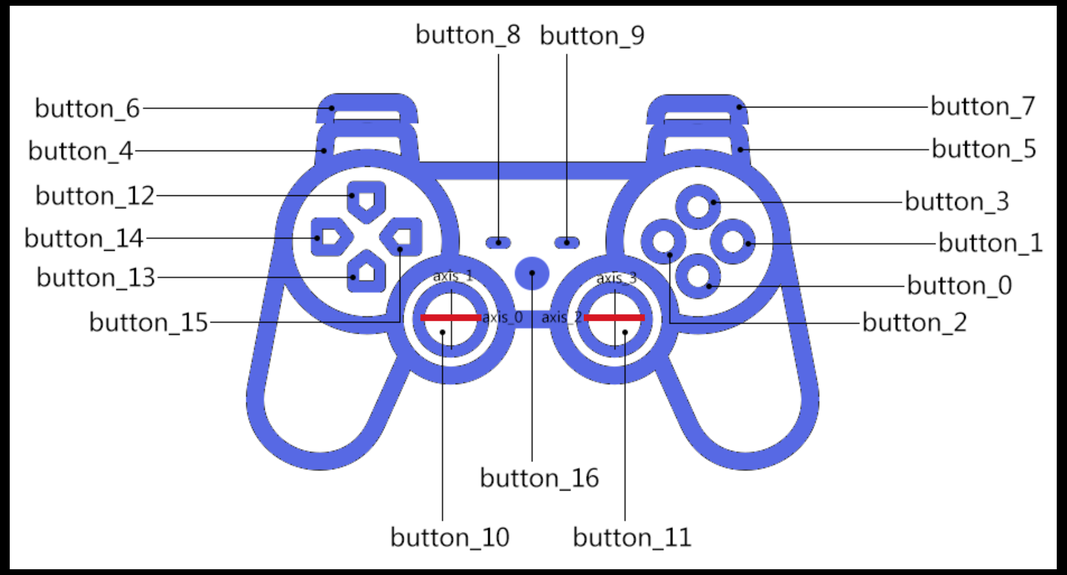
```
const buttonPressEvent = eventData => {
    return new CustomEvent("gamepadbuttonpressed", {
        detail: eventData
    });
};
```



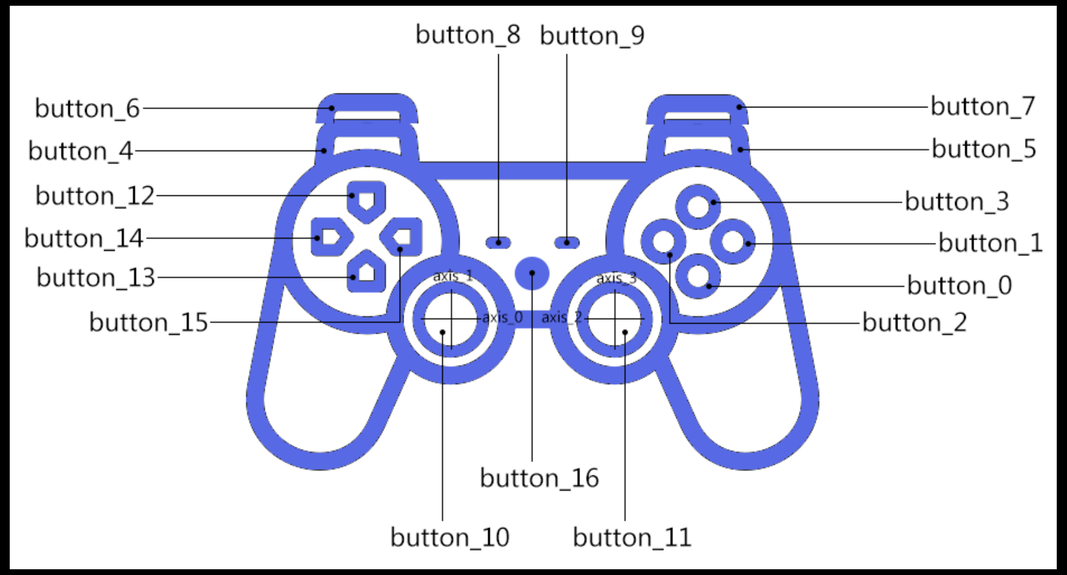

```
const listenToAxisMovements = gamepad => {
    const { axes } = gamepad;
    const totalAxisIndexes = axes.length;
    const totalSticks = totalAxisIndexes / 2;
    // Loop through all axes
    axes.forEach((axis, index) => {
        let stickMoved = null;
        let directionOfMovement = null;
        // Find which stick has been moved
        if (index < totalSticks) {
            stickMoved = 'left_stick';
        } else {
            stickMoved = 'right_stick';
        }
        // Find the direction of movement
        if (index === 0 || index === 2) {
            directionOfMovement = axis < 0 ? 'left' : 'right';
        }
        if (index === 1 || index === 3) {
            directionOfMovement = axis < 0 ? 'top' : 'bottom';
        }
        const eventData = { stickMoved, directionOfMovement };
        // Dispatch axis movement event
        return window.dispatchEvent(axisMovementEvent(eventData));
    });
};
```




```
const listenToAxisMovements = gamepad => {
    const { axes } = gamepad;
    const totalAxisIndexes = axes.length;
    const totalSticks = totalAxisIndexes / 2;
    // Loop through all axes
    axes.forEach((axis, index) => {
        let stickMoved = null;
        let directionOfMovement = null;
        // Find which stick has been moved
        if (index < totalSticks) {
            stickMoved = 'left_stick';
        } else {
            stickMoved = 'right_stick';
        }
        // Find the direction of movement
        if (index === 0 || index === 2) {
            directionOfMovement = axis < 0 ? 'left' : 'right';
        }
        if (index === 1 || index === 3) {
            directionOfMovement = axis < 0 ? 'top' : 'bottom';
        }
        const eventData = { stickMoved, directionOfMovement };
        // Dispatch axis movement event
        return window.dispatchEvent(axisMovementEvent(eventData));
    });
};
```




```
const listenToAxisMovements = gamepad => {
    const { axes } = gamepad;
    const totalAxisIndexes = axes.length;
    const totalSticks = totalAxisIndexes / 2;
    // Loop through all axes
    axes.forEach((axis, index) => {
        let stickMoved = null;
        let directionOfMovement = null;
        // Find which stick has been moved
        if (index < totalSticks) {
            stickMoved = 'left_stick';
        } else {
            stickMoved = 'right_stick';
        }
        // Find the direction of movement
        if (index === 0 || index === 2) {
            directionOfMovement = axis < 0 ? 'left' : 'right';
        }
        if (index === 1 || index === 3) {
            directionOfMovement = axis < 0 ? 'top' : 'bottom';
        }
        const eventData = { stickMoved, directionOfMovement };
        // Dispatch axis movement event
        return window.dispatchEvent(axisMovementEvent(eventData));
    });
};
```




```
const listenToAxisMovements = gamepad => {
    const { axes } = gamepad;
    const totalAxisIndexes = axes.length;
    const totalSticks = totalAxisIndexes / 2;
    // Loop through all axes
    axes.forEach((axis, index) => {
        let stickMoved = null;
        let directionOfMovement = null;
        // Find which stick has been moved
        if (index < totalSticks) {
            stickMoved = 'left_stick';
        } else {
            stickMoved = 'right_stick';
        }
        // Find the direction of movement
        if (index === 0 || index === 2) {
            directionOfMovement = axis < 0 ? 'left' : 'right';
        }
        if (index === 1 || index === 3) {
            directionOfMovement = axis < 0 ? 'top' : 'bottom';
        }
        const eventData = { stickMoved, directionOfMovement };
        // Dispatch axis movement event
        return window.dispatchEvent(axisMovementEvent(eventData));
    });
};
```

```
const axisMovementEvent = eventData => {
    return new CustomEvent("gamepadaxismoved", {
        detail: eventData
    });
};
```

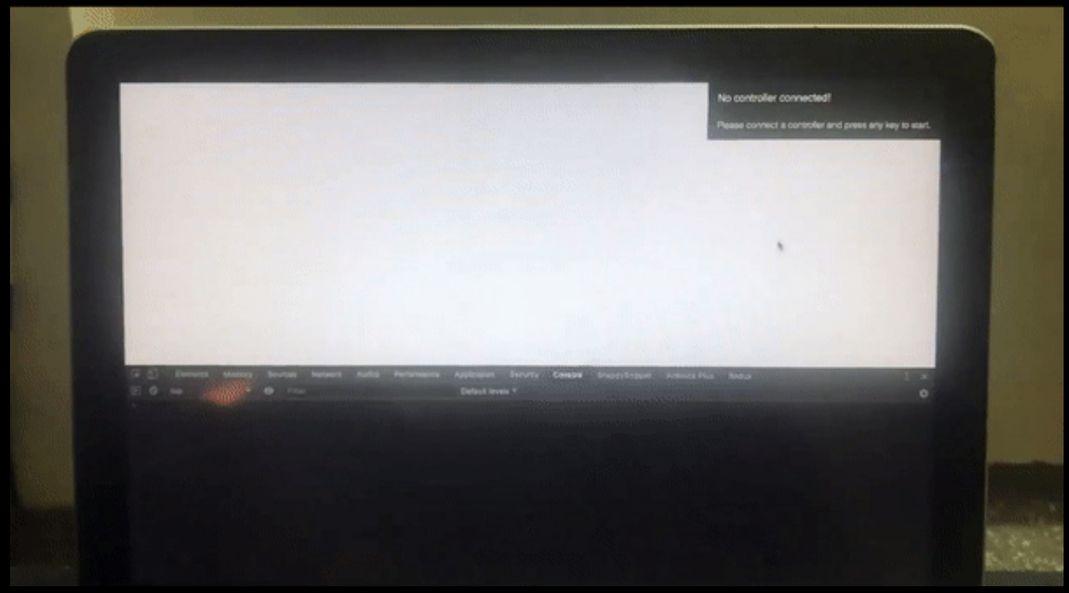
```
window.addEventListener("gamepadbuttonpressed", function(e) {
    // Info of pressed button
    console.log(e.detail);
});
```

```
window.addEventListener("gamepadaxismoved", function(e) {
    // Info of moved axis
    console.log(e.detail);
});
```

bit.ly/joypad_js

Connect/Disconnect

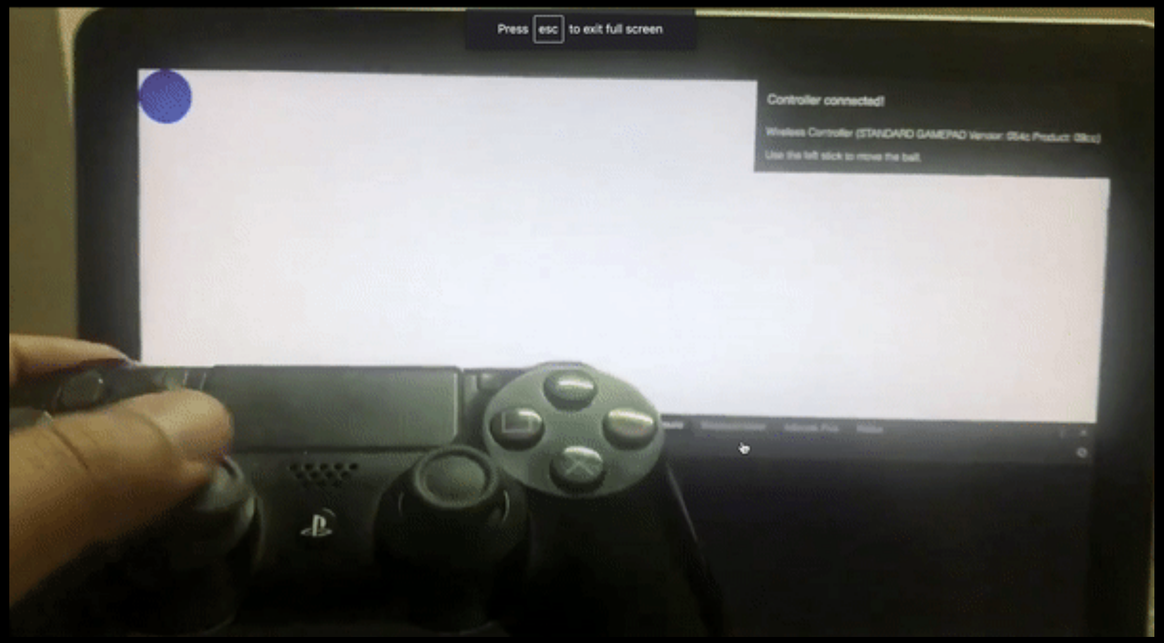
```
joypad.on('connect', e => {
    const { id } = e.gamepad;
    console.log(`${id} connected!`);
});

joypad.on('disconnect', e => {
    const { id } = e.gamepad;
    console.log(`${id} disconnected!`);
});
```



Button Press/Axis Movement

```
joypad.on('button_press', e => {
    const { buttonName } = e.detail;
    console.log(`${buttonName} was pressed!`);
});

joypad.on('axis_move', e => {
    const { stickMoved } = e.detail;
    console.log(`${stickMoved} was moved!`);
});
```

Button Press/Axis Movement



Vibration Effect

```
joypad.on('connect', e => {
    const { gamepad } = e;
    const options = {
        startDelay: 500,
        duration: 2000,
        weakMagnitude: 1,
        strongMagnitude: 1
    };

    joypad.vibrate(gamepad, options);
});
```

Settings

```
joypad.set({
    vibration: {
        startDelay: 500,
        duration: 3000,
        weakMagnitude: 1,
        strongMagnitude: 1
    }
});
```

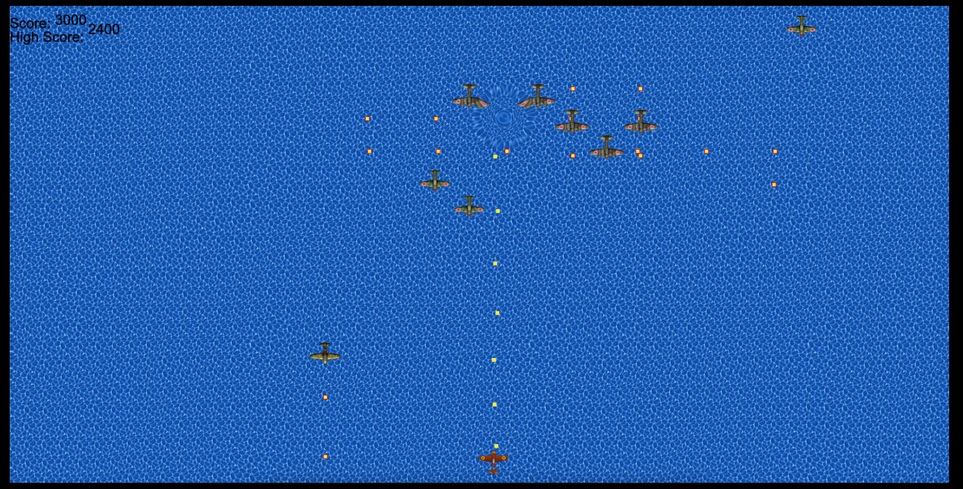
```
joypad.set({
    axisMovementThreshold: 0.3
});
```

```
function setCustomButtonMapping() {
    if (browserIs('Firefox')) {
        return {
            button_0: 1,
            button_7: 11,
            button_8: 12
        }
    } else {
        return null;
    }
};

joypad.set({
    customButtonMapping: setCustomButtonMapping()
});
```

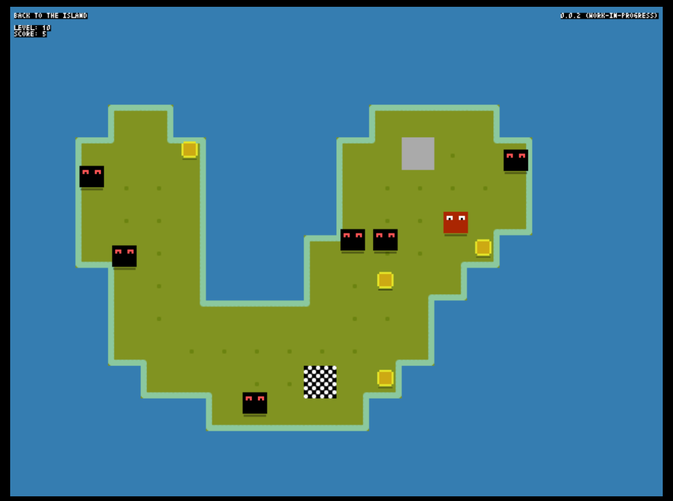
Browsers Support

 Edge	 Firefox	 Chrome	 Safari	 Opera
12+	29+	25+	10.1+	24+


0 / 5


Score: 3000
High Score: 2400


Welcome to the RPGJS demo!
A




Theodore
21:58
League Of Legends
CS:GO
BATTLEGROUNDS

bit.ly/joypad_js

Further Reading

arunmichaeldsouza.com/blog/using-joypad.js-for-a-better-gaming-experience-on-the-web

arunmichaeldsouza.com/blog/web-platform's-hidden-gems-(series)

arunmichaeldsouza.com/blog/web-platform's-hidden-gems---gamepad-api

Thank you

Arun Michael Dsouza

arunmichaeldsouza.com

@amdsouza92