src/**desktop.rs**

```
if let Some(virt_keycode) = input.virtual_keycode {
    let button = match virt_keycode {
        VirtualKeyCode::A => Button::A,
        VirtualKeyCode::B => Button::B,
        VirtualKeyCode::Z => Button::Select,
        VirtualKeyCode::X => Button::Start,
        VirtualKeyCode::Left => Button::Left,
        VirtualKeyCode::Right => Button::Right,
        VirtualKeyCode::Down => Button::Down,
        VirtualKeyCode::Up => Button::Up,
        _ => {
            *control_flow = glutin::event_loop::ControlFlow::Poll;
            return;
        }
    };
    match input.state {
        ElementState::Pressed => gb.keydown(button),
        ElementState::Released => gb.keyup(button),
    }
}
```

src/**web.rs**

```
let current_key_code: Rc<RefCell<i32>> = Rc::new(RefCell::new(0));
{
    let key_code = current_key_code.clone();
    let closure =
        Closure::<dyn FnMut(_)>::new(move |event: KeyboardEvent| {
            *key_code.borrow_mut() = event.key_code() as i32;
        });
    add_event_listener(“keydown”, closure.as_ref().unchecked_ref());
    closure.forget();

    let key_code = current_key_code.clone();
    let closure =
        Closure::<dyn FnMut(_)>::new(move |event: KeyboardEvent| {
            *key_code.borrow_mut() = (event.key_code() as i32) * -1;
        });
    add_event_listener(“keyup", closure.as_ref().unchecked_ref());
    closure.forget();
}
```