# Event-driven Prototype

## Aims

1. **Demonstrate working knowledge of HTML and CSS, such that the user can interact and gain a good user experience (UX).**
2. **Advance your work into a good UI through desk research and remote field research.**
3. **Cover 3 different events within your system.**

## Background

You have had training sessions on the 3 web languages, HTML CSS and Javascript. You must now get comfortable producing something using your current knowledge, to see what you can remember, and what areas you find difficult.

Javascript was made to improve interactivity on the web. For a programmer, the interactions are all called ‘events’. This is an abstraction; rather than calling it what it is for a person (an interaction), we call it what it is for the ‘system’ or ‘program’ (an event). This abstraction is also important because when you remove the person from the story, you can see that the system is prepared for events that are not directly from people, such as a sensor device that might send data to the system.

### What is an event?

User actions such as mouse clicks and key presses are the popular examples. You must understand the logical breakdown of these: When a user clicks a mouse, there are many more real-world events within that and they are actually all common sense:

1. Mouse button touch
2. Mouse button down
3. Mouse button hold
   1. Longer mouse button hold
4. Mouse button release
   1. A fast release is a click
      1. This allows for a double click, or even multiple clicks
   2. A slow release is a hold

Do not underestimate how much applying good common sense can help you as a programmer. The above breakdown of a mouse button, though is common sense, is actually what you need in understanding the ‘mouse events’.

Even beyond the mouse button itself, come the UX questions:

1. Where can you click?
2. When?
3. How long for?
4. Can that same button be pressed with your finger for a touch screen, is it big enough?

Event-driven programming is the dominant paradigm used in graphical user interfaces (GUI). Javascript is focused on responding to user input and each input is an event.

### How do you track events?

You have already been told about listeners.

### How do you react to events?

You have already been told about handlers, which contain ‘callbacks’.

# Task

Imagine you are building a tool like powerpoint and there are two ‘views’. The public view for everyone to see whilst the presentation takes place, and the speaker view which is private for the speaker and may contain hints and notes.

## Part 1

You must make a web app that works from two different screens i.e. two different URLs.

Simply code the UI design of both. Use any open source libraries you wish.

## Part 2

Make it possible for the presentation view to take mouse clicks and button presses to control what part of the presentation you are on. It can be slides or paragraphs or even one page that is scrolled. You have full flexibility on the semantics of your code to produce the UX.

## Part 3

Now make it possible to control the presentation view from your speaker view using keyboard shortcuts.