Lecture 4:

Input 1: Conventional Input Models for Handling Input Events

05-431/631 Software Structures for User Interfaces (SSUI)

Fall, 2022

© 2022 - Brad Myers

Logistics

- HW 1 due next Tuesday

Input Handling

- Event handlers attached to UI elements
- Most window managers and toolkits have used this same model
  - True of JavaScript as well
  - Quite old and has problems
  - Quickly gets to be complex
  - Even more difficult today with multi-finger interactions
- Don’t face it much if just use built in widgets and just “click” actions
  - Most important for highly interactive systems
    - Games, graphical editors
  - Or to create custom behaviors and widgets

© 2022 - Brad Myers

Quotes

- "One of the most complex aspects of Xlib programming is designing the event loop, which must take into account all of the possible events that can occur in a window."
  -- Nye & O'Reilly X Toolkit Intrinsics Programming Manual, vol. 4, 1990, p. 241.

- "The dispatching and handling of events is rather complicated."
  -- Galaxy Reference Manual, v1.2, p. 20-5.

© 2022 - Brad Myers

Event Records

- Structures (records) composed of all information about events
- Created by window manager, sent to a queue for each window
- X/11 window manager from 1987 defines 33 different types of events

© 2022 - Brad Myers

X Event Types

1. buttonPress
2. keyPress
3. keyRelease
4. buttonRelease
5. motionNotify
6. enterNotify
7. leaveNotify
8. focusIn
9. focusOut
10. keymapNotify (change keymap)
11. Expose
12. graphicsExpose (source of copy not available)
13. noExpose (source of copy is available)
14. colormapNotify
15. propertyNotify (some property changed)
16. visibilityNotify (become covered)
17. resizeRequest
18. circulateNotify (stacking order)
19. configureNotify (resize or move)
20. destroyNotify (was destroyed)
21. gravityNotify (moved due to gravity)
22. mapNotify (became visible)
23. createNotify
24. reparentNotify (in diff. window)
25. unmapNotify (invisible)
26. circulateRequest
27. configureRequest
28. mapRequest
29. mappingNotify (keyboard mapping)
30. clientMessage
31. selectionClear (for cut and paste)
32. selectionNotify
33. selectionRequest

© 2022 - Brad Myers

Windows .Net Events

Human-Computer Interaction Institute

1. AutoSizeChanged
2. BackColorChanged
3. BackgroundImageChanged
4. BackgroundImageLayoutChanged
5. BindingContextChanged
6. CausesValidationChanged
7. ChangeUICues
8. Click
9. ClientSizeChanged
10. ContextMenuChanged
11. ContextMenuStripChanged
12. ControlAdded
13. ControlRemoved
14. CursorChanged
15. Disposed
16. DockChanged
17. DoubleClick
18. DragDrop
19. DragEnter
20. DragLeave
21. DragOver
22. EnabledChanged
23. Enter
24. FontChanged
25. ForeColorChanged
26. GiveFeedback
27. GotFocus
28. HandleCreated
29. HandleDestroyed
30. HelpRequested
31. ImeModeChanged
32. Invalidated
33. KeyDown
34. KeyPress
35. KeyUp
36. Layout
37. Leave
38. LocationChanged
39. LostFocus
40. MarginChanged
41. MouseCaptureChanged
42. MouseClick
43. MouseDoubleClick
44. MouseDown
45. MouseEnter
46. MouseHover
47. MouseLeave
48. MouseMove
49. MouseUp
50. MouseWheel
51. Move
52. PaddingChanged
53. Paint
54. ParentChanged
55. PreviewKeyDown
56. QueryAccessibilityHelp
57. QueryContinueDrag
58. RegionChanged
59. Resize
60. RightToLeftChanged
61. SizeChanged
62. StyleChanged
63. SystemColorsChanged
64. TabIndexChanged
65. TabStopChanged
66. TextChanged
67. Validated
68. Validating
69. VisibleChanged

© 2022 - Brad Myers

Source: http://msdn.microsoft.com/en-us/library/1dk48x94.aspx

JavaScript DOM events

Human-Computer Interaction Institute

1. AnimationEvent
2. AudioProcessingEvent
3. BeforeInputEvent
4. BeforeUnloadEvent
5. BlobEvent
6. ClipboardEvent
7. CloseEvent
8. CompositionEvent
9. CSSFontFaceLoadEvent
10. CustomEvent
11. DeviceLightEvent
12. **DeviceMotionEvent**
13. DeviceOrientationEvent
14. DeviceProximityEvent
15. DOMTransactionEvent
16. DragEvent
17. EditingBeforeInputEvent
18. ErrorEvent
19. FetchEvent
20. **FocusEvent**
21. GamepadEvent
22. HashChangeEvent
23. IDBVersionChangeEvent
24. **InputEvent**
25. **KeyboardEvent**
26. MediaStreamEvent
27. MessageEvent
28. **MouseEvent**
28. MutationEvent
29. OfflineAudioCompletionEvent
30. OverconstrainedError
31. PageTransitionEvent
32. PaymentRequestUpdateEvent
33. PointerEvent
34. PopStateEvent
35. ProgressEvent
36. RelatedEvent
37. RTCDataChannelEvent
38. RTCIdentityErrorEvent
39. RTCIdentityEvent
40. RTCPeerConnectionIceEvent
41. SensorEvent
42. StorageEvent
43. SVGEvent
44. SVGZoomEvent
45. TimeEvent
46. **TouchEvent**
47. TrackEvent
48. TransitionEvent
49. UIEvent
50. UserProximityEvent
51. WebGLContextEvent
52. **WheelEvent**

Source: https://developer.mozilla.org/en-US/docs/Web/API/Event

© 2022 - Brad Myers

Sub types of MouseEvent

onclick	The event occurs when the user clicks on an element
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu
ondblclick	The event occurs when the user double-clicks on an element
onmousedown	The event occurs when the user presses a mouse button over an element
onmouseenter	The event occurs when the pointer is moved onto an element
onmouseleave	The event occurs when the pointer is moved out of an element
onmousemove	The event occurs when the pointer is moving while it is over an element
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children
onmouseup	The event occurs when a user releases a mouse button over an element

© 2022 - Brad Myers

Source: https://www.w3schools.com/jsref/obj_mouseevent.asp

Mouse Event Properties

Human-Computer Interaction Institute

altKey	Returns whether the "ALT" key was pressed when the mouse event was triggered
button	Returns which mouse button was pressed when the mouse event was triggered
buttons	Returns which mouse buttons were pressed when the mouse event was triggered
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
ctrlKey	Returns whether the "CTRL" key was pressed when the mouse event was triggered
getModifierState()	Returns true if the specified key is activated
metaKey	Returns whether the "META" key was pressed when an event was triggered
movementX	Returns the horizontal coordinate of the mouse pointer relative to the position of the last mousemove event
movementY	Returns the vertical coordinate of the mouse pointer relative to the position of the last mousemove event
offsetX	Returns the horizontal coordinate of the mouse pointer relative to the position of the edge of the target element
offsetY	Returns the vertical coordinate of the mouse pointer relative to the position of the edge of the target element
pageX	Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
pageY	Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
region	
relatedTarget	Returns the element related to the element that triggered the mouse event
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered
shiftKey	Returns whether the "SHIFT" key was pressed when an event was triggered
which	Returns which mouse button was pressed when the mouse event was triggered

© 2022 - Brad Myers

Source: https://www.w3schools.com/jsref/obj_mouseevent.asp

Sub Types of Touch Event

ontouchcancel	The event occurs when the touch is interrupted
ontouchend	The event occurs when a finger is removed from a touch screen
ontouchmove	The event occurs when a finger is dragged across the screen
ontouchstart	The event occurs when a finger is placed on a touch screen

© 2022 - Brad Myers

Touch Event Properties

- Some touch devices also have keyboard keys

altKey	Returns whether the "ALT" key was pressed when the touch event was triggered
changedTouches	Returns a list of all the touch objects whose state changed between the previous touch and this touch
ctrlKey	Returns whether the "CTRL" key was pressed when the touch event was triggered
metaKey	Returns whether the "meta" key was pressed when the touch event was triggered
shiftKey	Returns whether the "SHIFT" key was pressed when the touch event was triggered
targetTouches	Returns a list of all the touch objects that are in contact with the surface and where the touchstart event occured on the same target element as the current target element
touches	Returns a list of all the touch objects that are currently in contact with the surface

© 2022 - Brad Myers

Inherited Properties for both (from Event)

Human-Computer Interaction Institute

bubbles	Returns whether or not a specific event is a bubbling event
cancelBubble	Sets or returns whether the event should propagate up the hierarchy or not
cancelable	Returns whether or not an event can have its default action prevented
composed	Returns whether the event is composed or not
createEvent()	Creates a new event
composedPath()	Returns the event's path
currentTarget	Returns the element whose event listeners triggered the event
defaultPrevented	Returns whether or not the preventDefault() method was called for the event
eventPhase	Returns which phase of the event flow is currently being evaluated
isTrusted	Returns whether or not an event is trusted
preventDefault()	Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur
stopImmediate-Propagation()	Prevents other listeners of the same event from being called
stopPropagation()	Prevents further propagation of an event during event flow
target	Returns the element that triggered the event
timeStamp	Returns the time (in milliseconds relative to the epoch) at which the event was created
type	Returns the name of the event

© 2022 - Brad Myers

Examples: Mouse

```
document.addEventListener(
  "click", // or "dblclick"
  (event) => {
        event.target  // what clicked on
        event.clientX // x location of the click
        event.clientY // y location of the click
      }
);
```

- Or might attach to a particular target object:

```
const workarea = document.querySelector("#workarea");
workarea.addEventListener("click",
      () => {console.log("click on workarea");}
);
```

© 2022 - Brad Myers

Examples: touch

```
document.addEventListener(
  "touchstart",
  (event) => {
    if (event.touches.length === 1) { // first finger
      …
    } else if (event.touches.length === 2) { //second finger
      …
    }
});
```

© 2022 - Brad Myers

What events are generated?

- When double click with the mouse?
  - See https://www.cs.cmu.edu/~bam/uicourse/05631fall2021/HW2/input-test/

What events are generated?

- When double click with the mouse?
  - Answer:
    - mousedown
    - mouseup
    - click -- `e.detail` has click count = 1
    - mousedown
    - mouseup
    - click -- `e.detail` has click count = 2
    - dbclick -- `e.detail` has click count = 1
- Can keep clicking and click count will go up, but no named events (no “tripleclick”)

© 2022 - Brad Myers

What about for touch events?

- Important: no touch “click” equivalent in JavaScript
  - But wants to have code that works across regular and touch devices
  - So touches *also* (sometimes) generate mouse events
  - Generates `touchstart, touchend, mouseover, mousemove, mousedown, mouseup, click, [dblclick]`
  - See https://www.cs.cmu.edu/~bam/uicourse/05631fall2021/HW2/input-test/index.html

© 2022 - Brad Myers

What does “click” mean?

What does “click” mean?

- In most systems, must be short time, and not move before release

What does “click” mean?

- In JavaScript, for *mouse*, “click” does *not* depend on movement *or* time
  - Can press mouse button and hold for arbitrary amount of time
  - Can press and move, as long as stay over same object
- But dblclick *does* depend on time
- But on iPhone, touch-ing only generates click if short <u>and</u> *don’t move*
  - Can feel the “long-press” timeout – phone vibrates

© 2022 - Brad Myers

Implications for the UI

- What does all this mean for what you can have click, double click, and triple click do?

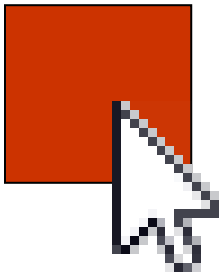
Implications for the UI

- What does all this mean for what you can have click, double click, and triple click do?
  - Click behavior *always* happens before double click behavior
    - Otherwise, would need a timeout to wait and see if the double click happens
  - Single click => select, double click => open
  - Selecting text with 1,2,3 clicks in Chrome, PowerPoint or Word

© 2022 - Brad Myers

Strategies for Multiple Behaviors

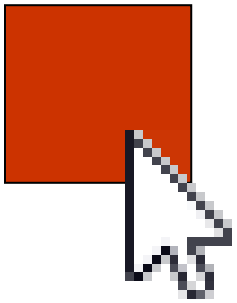
- Different things happen on mouse down depending on:
  - Mode
  - What press down on
  - What do next (release, move, …)
- E.g., press-move vs. click
- How control this? Two main strategies
  - 1) put event handlers on the objects only when relevant, and remove them when not, dynamically
    - E.g., put a click handler on each target when waiting for click, remove it when not.
  - 2) put event handlers globally, e.g., on `document`, and control behavior with global variable(s)



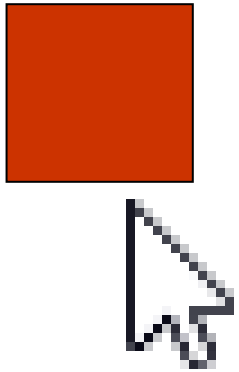
Select vs. Move

- Why doesn't this work:

```
target.addEventListener(
        "mousedown",
workarea (event) => { … } );
 target.addEventListener(
        "mousemove",
        (event) => { … } );
```



- Need to have move handler on the background, *not on* the target



Click vs. Move Differentiation

- If need to differentiate single click from double (or two fingers down), then do need a timeout
  `setTimeout(function, milliseconds, param1, param2, ...)`
  - Parameters sent to the function
  - Function might remove the double-click handler, and perform the single click function
- Alternatively, may be sufficient to perform action on mouseup (or touchstop) if no extra mousedown or mousemove (or touchdown, touchmove)

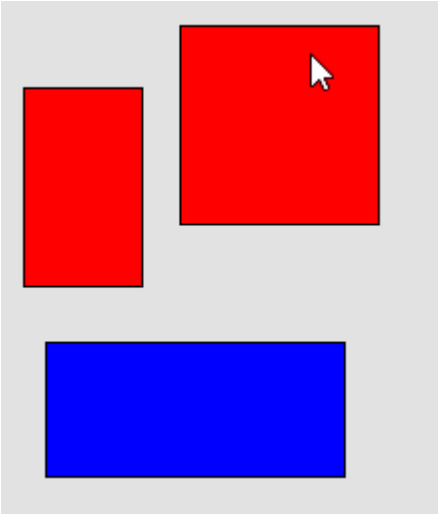
© 2022 - Brad Myers

Coordinating multiple behaviors

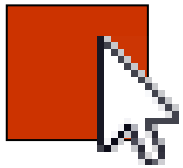
- For button like behavior, can have one onclick handler per button
- But if doing graphics in JavaScript, seems better to have fewer, more global event handlers, like on the document
  - Then use event.target to find where actually happened
- Control modes with one (or a few) global variables – set with values to indicate the mode:
  - Want enum (but not in JS), can use strings: “idle”, “pressdown”, moving”, “doubleclickmoving”, “aborting”, …

© 2022 - Brad Myers

Hints for Homework 2



- Will need a lot of global variables to keep track of the state and modes:
  - `selected_object`, `object_being_dragged_around`, `double_click_mode`, `object_being_resized`, `touch_count`, `touch_point_delta`, `orig_size_position`, etc.
- Will need a lot of different kinds of event handlers attached to different kinds of objects – targets and background



- Each handler will need lots of `if` statements, depending on the global variables

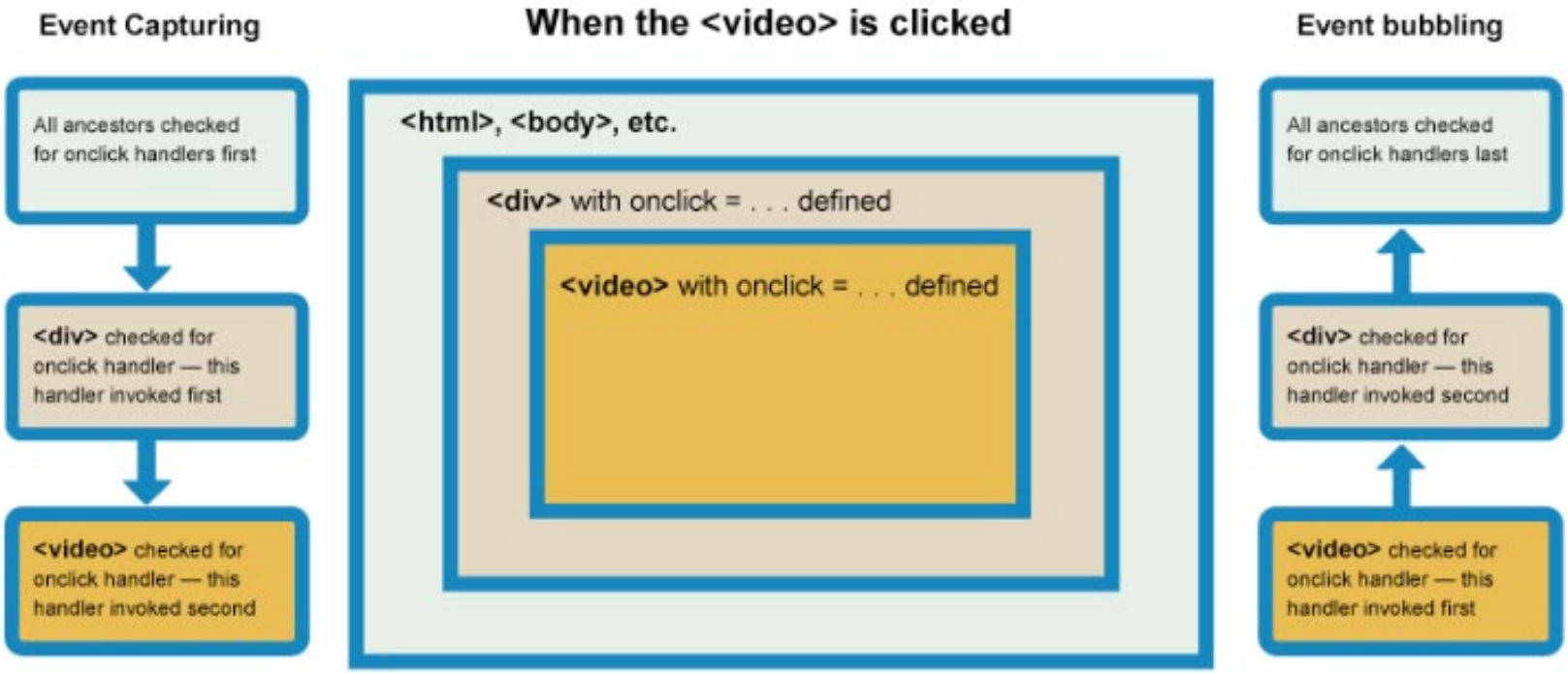
Propagation

- Events sent to the lowest level DOM element containing the pointer
- If event not consumed by a handler, then sent to the container element, etc.
- Handlers say whether they handled (consumed) the event or not
  - JavaScript: `event.stopPropagation();`
    - See 3 and more clicks in `input-test`
  - Android: `onXXXX()` handlers return Booleans

© 2022 - Brad Myers

JavaScript Propagation

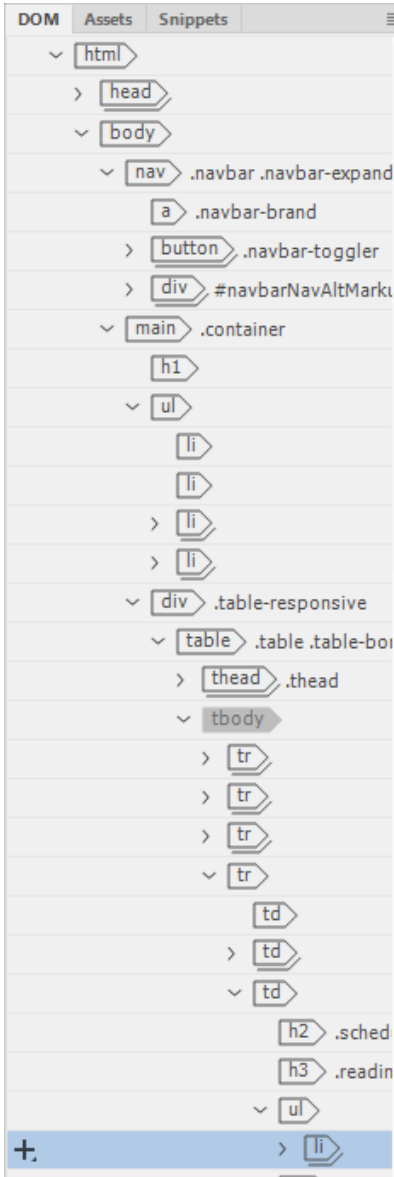
- JavaScript for the web goes *both* down and then up the container tree, due to combining the way that IE and Mozilla did it, and handlers can pick which one they want.
  - "Capturing" vs. "bubbling"
  - Default = bubbling, but runs all, unless call `stopPropagation()`



https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events#Event_bubbling_and_capture

JavaScript Propagation

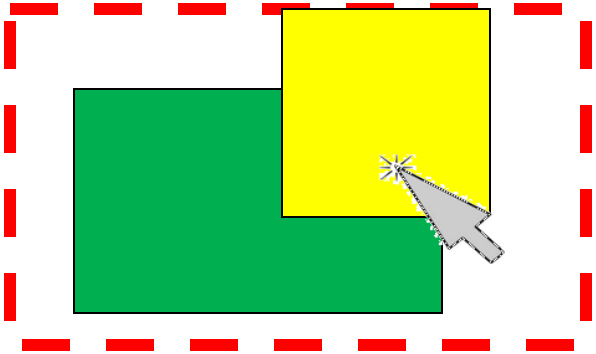
- Third parameter to `addEventListener` is whether down (capturing) or up (bubbling)
- Default = `false` = up ("bubble");
- `true` – handled on the way down
- Can have multiple handlers, both up and down!
  - See special divs in `input-test`: https://www.cs.cmu.edu/~bam/uicourse/05631fall2021/HW2/input-test/
- Capturing useful in rare situations
  - Often with `stopPropagation()`
  - E.g., only scroll container; touches ignore internal objects, etc.



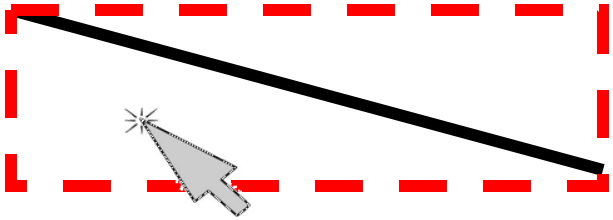
© 2022 - Brad Myers

Issue: Covering

- Which objects get an event when overlapping
  - "Z" order vs. containment
  - What about when top object doesn't want event?
  - Can't necessarily use `obj.contains(eventX, eventY)`



- Input mechanism must know about graphical objects
  - Handled automatically by DOM



- Bounding box vs. on object
- Complexities:
  http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/event_delivery_responder_chain/event_delivery_responder_chain.html#//apple_ref/doc/uid/TP40009541-CH4-SW2

© 2022 - Brad Myers

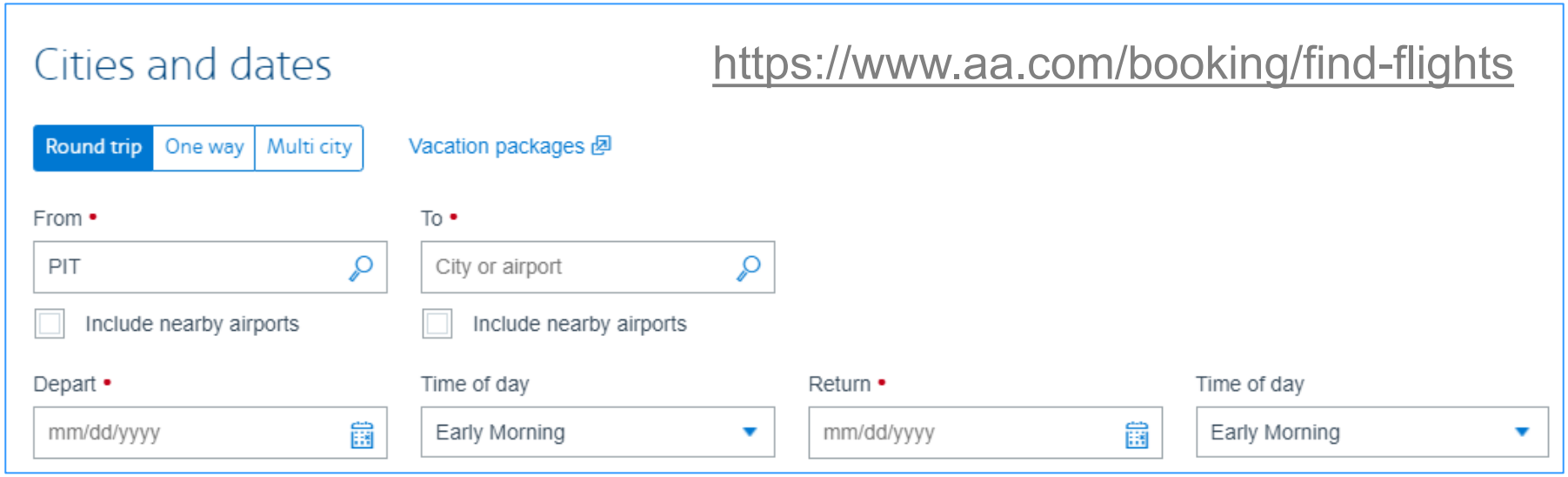
Text Events - Which Window?

Text Events - Which Window?

- Called "focus"
  - Old name: "active" window
  - My old name: "Listener" window
- Click to Type
- Move to Type
- Affects what kinds of interactions are possible
  - Mac single menubar not possible with move-to-type
- Note difference with "mouse" events focus vs. scroll events focus!
  - Windows and Mac
- Also which text input widget has the focus
  - So also relevant to smartphones

© 2022 - Brad Myers

Multiple Text Fields



ref

- Click or tab to select which field
- Tab order includes all kinds of input
  - Especially important for accessibility
- Default = DOM order
- Control with `tabindex` property in html

```
<form>
  Field 1 (first tab selection):
  <input type="text" name="field1" tabindex=1 /><br/>
  Field 2 (third tab selection):
  <input type="text" name="field2" tabindex=3 /><br/>
  Field 3 (second tab selection):
  <input type="text" name="field3" tabindex=2 /><br/>
</form>
```

© 2022 - Brad Myers

JavaScript: Assign Focus manually

- Need to do the following

```
// enable getting the keyboard focus
// https://stackoverflow.com/questions/18928116/javascript-keydown-event-listener-is-not-working
workspace.setAttribute("tabindex", -1);
// give it the keyboard focus to start
// https://stackoverflow.com/questions/6754275/set-keyboard-focus-to-a-div/6809236
workspace.focus();
// now can listen for keyboard events
workspace.addEventListener("keydown", (event) => {
  console.log("key=" + event.code);
  if (event.code == "Escape") {
    console.log("abort");
    … do abort stuff
  }
});
```

© 2022 - Brad Myers

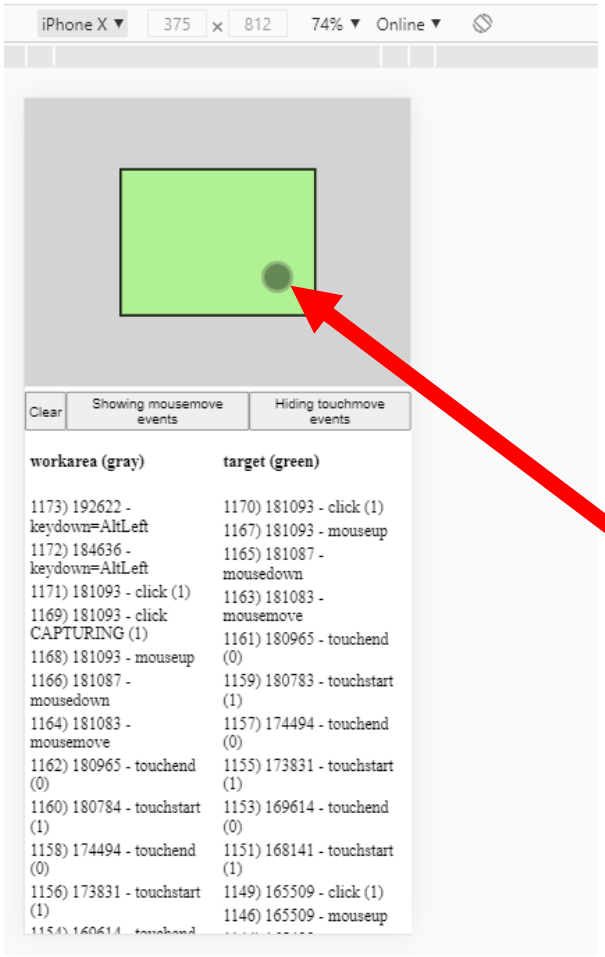
Gestural "Events"

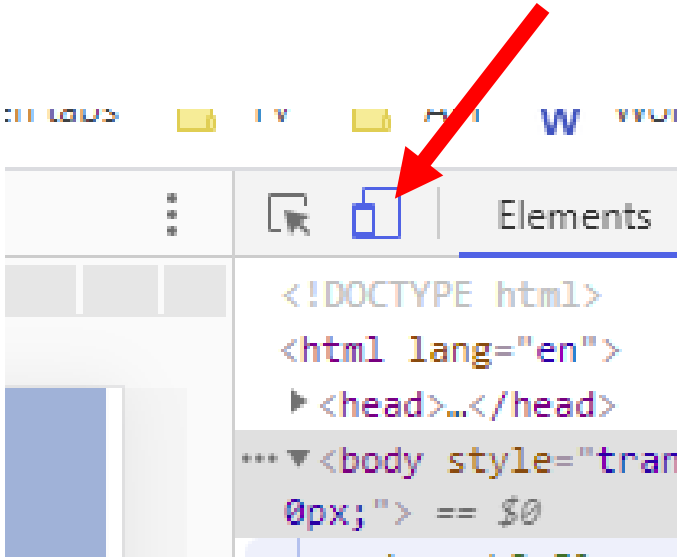
- Many libraries add gestural events
  - swipe, multi-touch pinch and rotate recognizers
  - Long press on Android or iPhone
- Come in as if they were regular events
- So lower-level code doesn't need to distinguish
- Android has separate "gesture" classes
- iOS: Gesture Recognizers

© 2022 - Brad Myers

Note: Debugging Touch on Chrome

- Can go into mouse/tablet mode
- Pick device size
- Mouse actions pretend to be touch
- Just 1 finger





Debugging events on phones

- See instructions on homework 2 for debugging Chrome on Android and iOS
  - Note: *not* Safari browser on iOS
- Can use console() output to debug and see what's happening

- *Thanks Clara!*

© 2022 - Brad Myers

Event Handling in Other Systems

- Other systems need additional kinds of events
- E.g., window manipulations – iconify, delete, etc.
- Keyboard keys down and up
- For when refresh needed
  - Not needed in JavaScript

Other Architectures

- Old (e.g., Windows SDK): giant `switch` statement per window
  - Branch for each event
  - But not dependent on mode, which object, etc.
- Global event handlers for each *type* of event
  - No matter *where* that event happens in the window
  - E.g., per `Activity` in Android
- Specific event handlers per object
  - Java Swing: `button1.addActionListener(this);`
  - Android: `View` event listeners (since widgets are views)
  - JavaScript: `obj1.addEventListener ( )`
- Lots of issues with multiple threading

© 2022 - Brad Myers

Translation Tables

- (Not available in JavaScript – only for native window systems)
- So particular mouse key or keyboard key not hard-wired into application.
  - Allows user customization and easier changes
- Supported in Motif by the *resources* mechanism
  - e.g. Shift<Btn1Down>: doit()
    can be put in .Xdefaults, and then application deals with doit, and user can change bindings.
- Keyboard translation is 2 step process in X:
  - Hardware "keycodes" numbers mapped to "keysyms"
  - "Keysyms" translated to events
- For double-clicking, Motif does translation, but not Xlib
  - For non-widgets, have to do it yourself
  - *Always* also get the single click events
  - Java – no built-in double click support
    - Does have click vs. drag
- Browser / OS level for JavaScript
  - E.g., can swap left/right mouse buttons for left-handed people

© 2022 - Brad Myers