Event Handling in jQuery

Understanding Event Driven Programming



Robert Lindley
Software Architect

@coderrob

www.coderrob.com

Overview



- Understanding browser event processing
- Introducing the events
- Introducing event handlers
- Understanding event propagation
- Understanding event delegation
- Creating custom events
- Referencing jQuery documentation

Understanding Browser Event Processing

Design Patterns

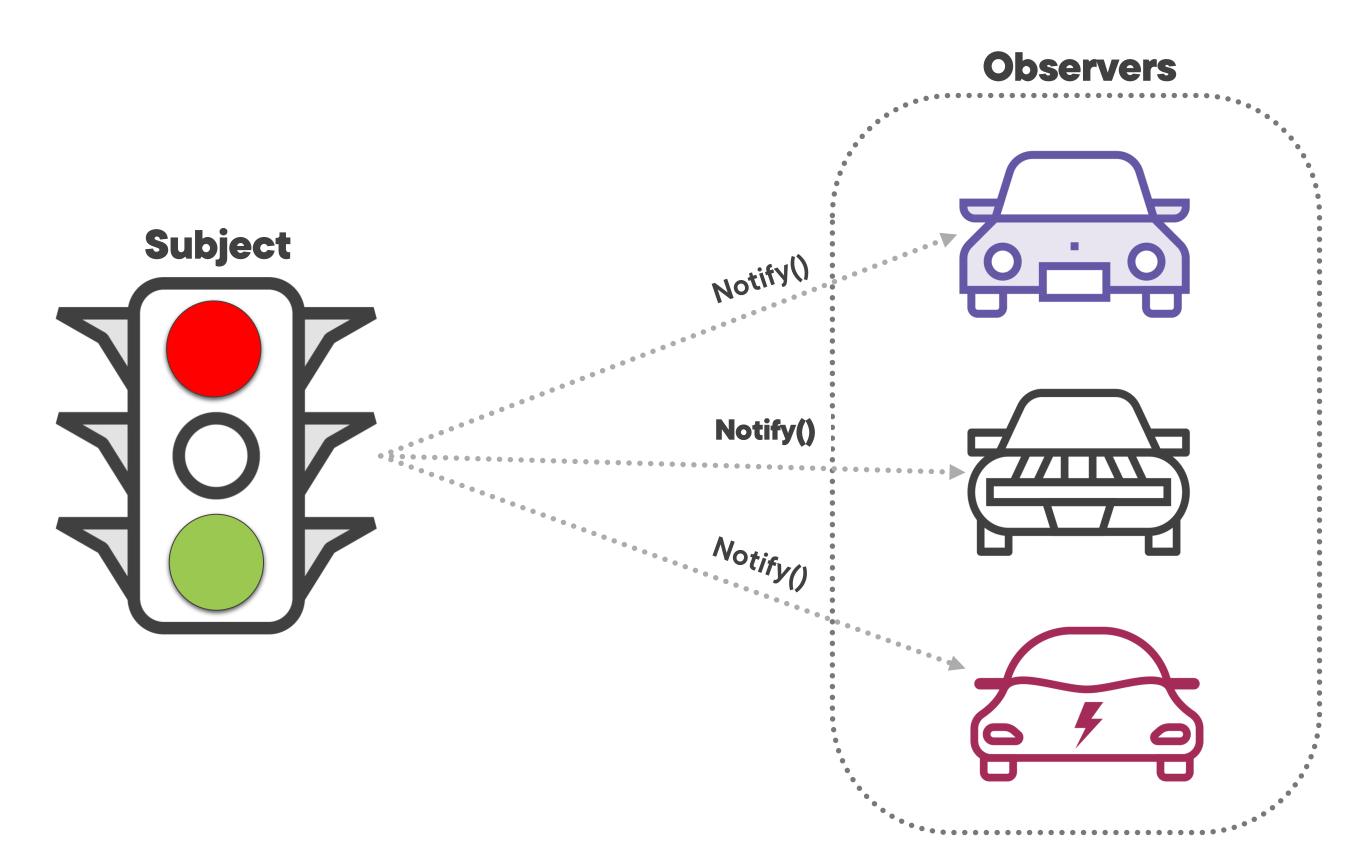
Observer

A subject maintains a list of subscribers, called observers, and notifies them directly of changes.

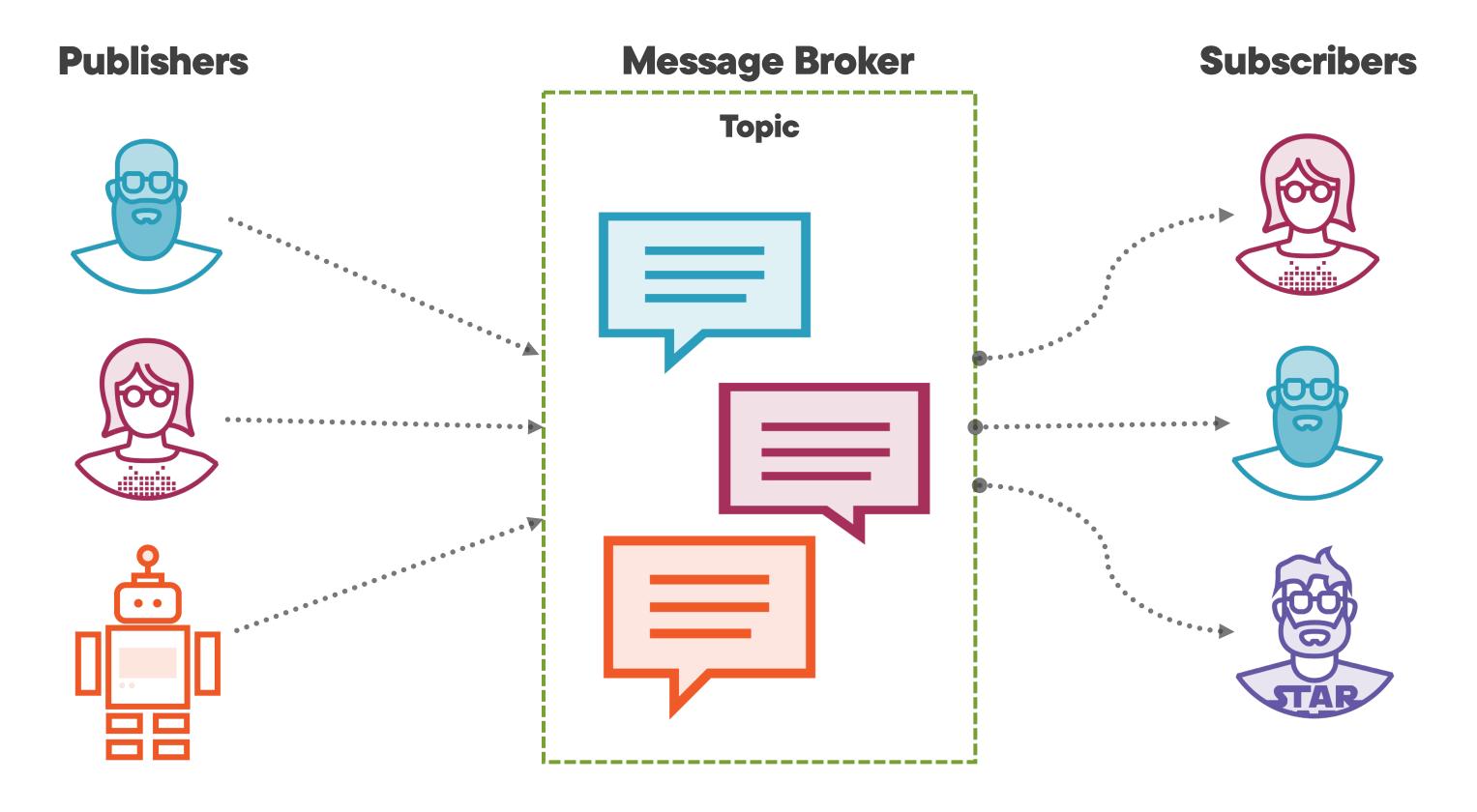
Publisher-Subscriber

A message broker maintains a list of subscribers to a topic. Publishers post messages to the broker.

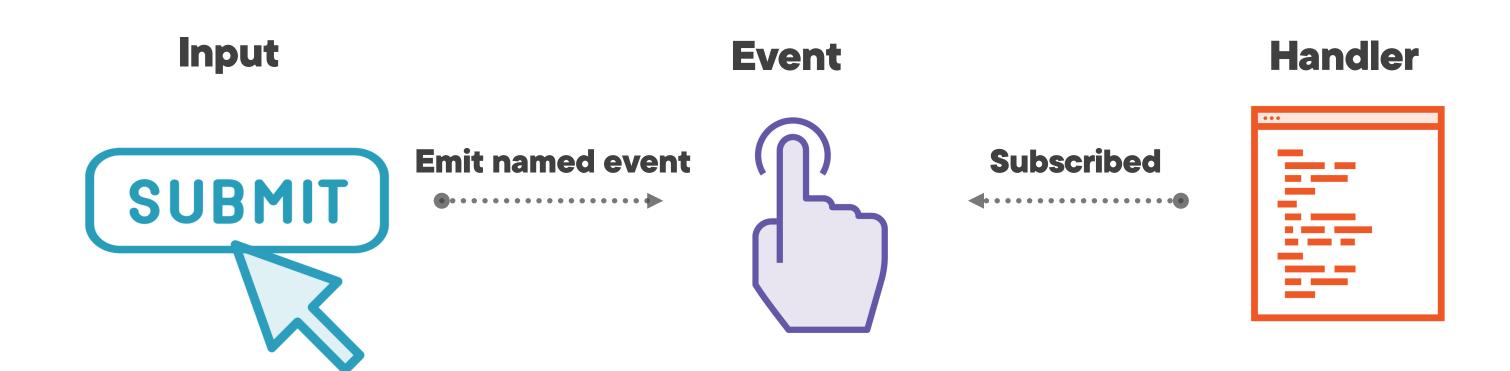
Observer Pattern



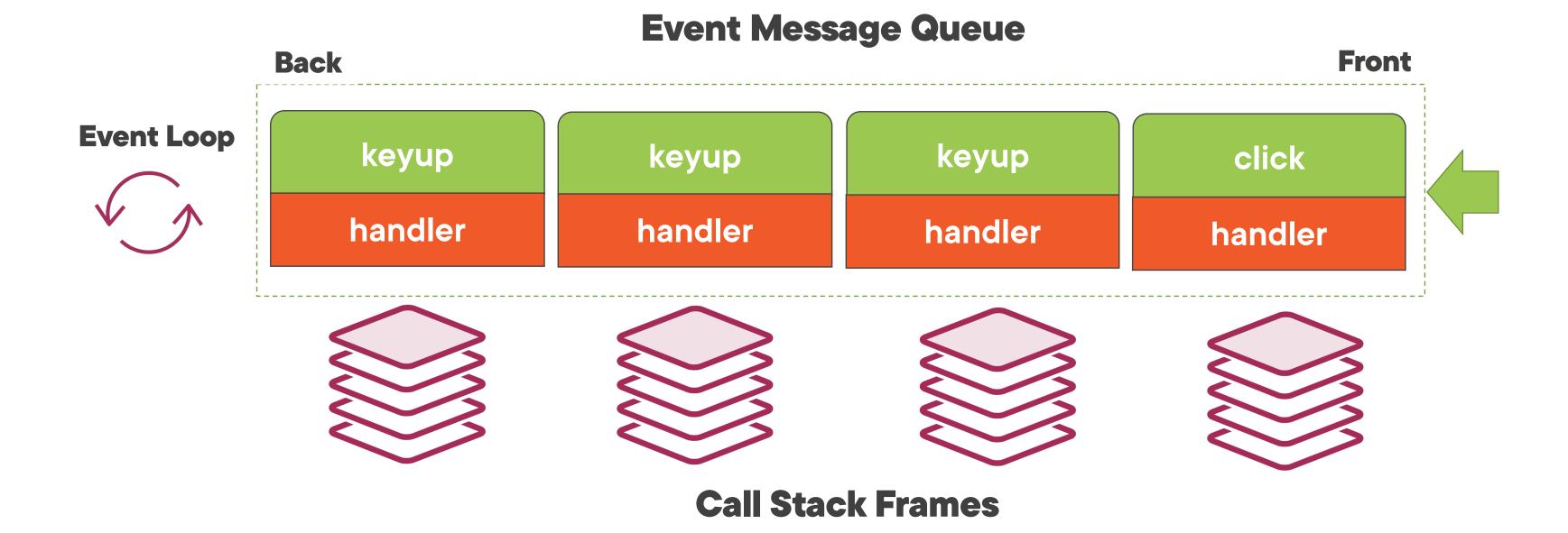
Publisher-Subscriber Pattern



Browser Events



Event Processing



Introducing Events

Common Types of Events



Mouse Events



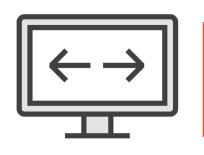
Keyboard Events



Form Events



Document Events



Window Events

Event Object Properties

event.bubbles event.cancelable event.composed event.defaultPrevente dvent.eventPhase event.target event.timeStamp event.type

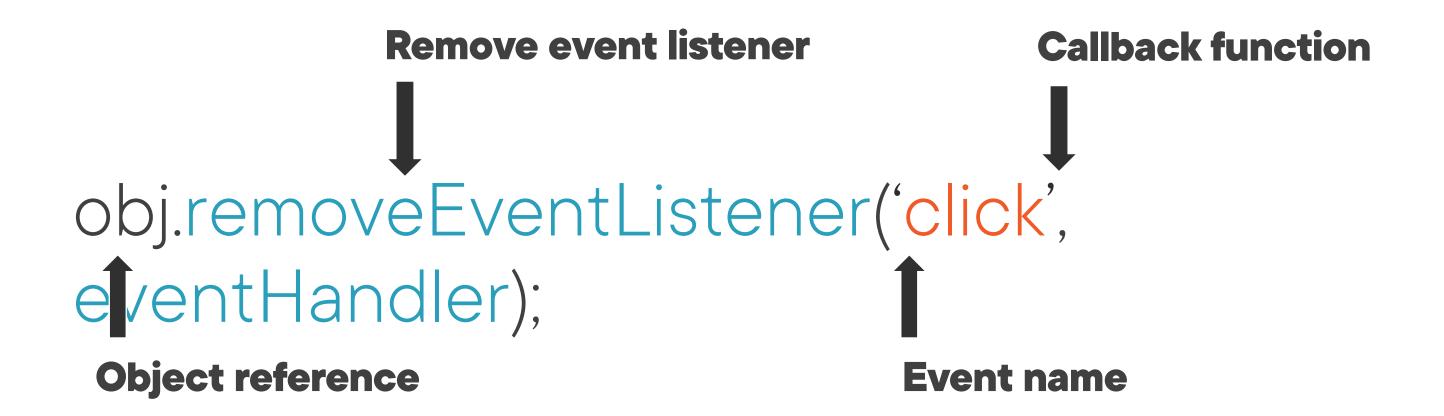
Event Object Functions

Introducing Event Handlers

Add Event Handler



Remove Event Handler



Callback Function

```
Function keyword

I

function eventHandler(event) {

Function name
```

```
// Index.html
<input id="rick" type="button"</pre>
value="Roll Me">
// Script.js
const btn =
document.getElementById('rick')
function roll(event) {
  console.log('never gonna give you up');
btn.addEventListener('click', roll)
```

■ Refer to the ID attribute's value

■ Get the element reference

■ Define the "roll" function

■ Register "roll" callback function to be invoked when the button's "click" event is dispatched.

Event Handler Best Practice

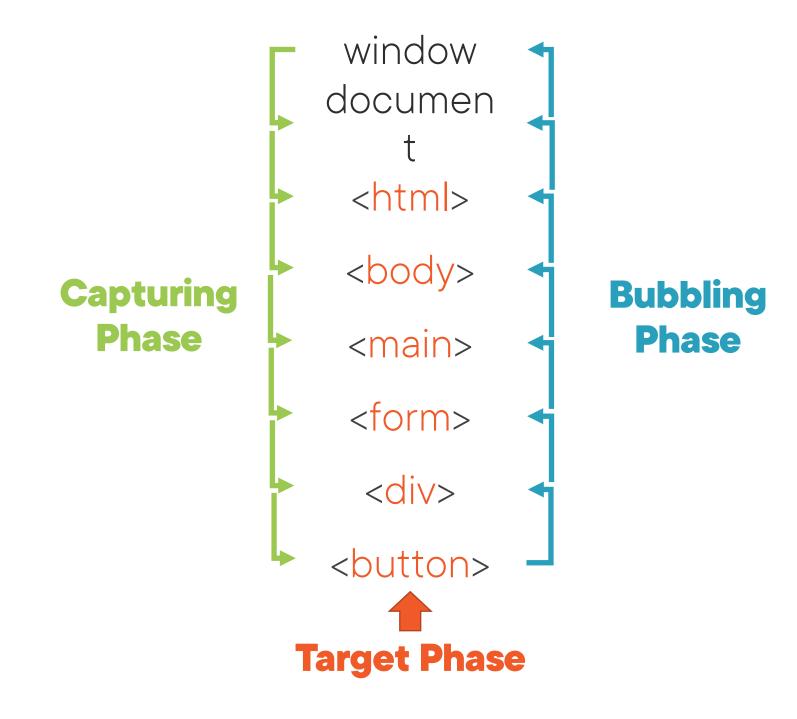
```
<input onclick='eventHandler()'>
obj.onclick = eventHandler;
obj.addEventListener('click', eventHandler);
```

Understanding Event Propagation

Event Propagation



```
<html>
  <body>
    <main>
      <form>
        <div>
          <button>Submit</button>
        </div>
      </form>
    </main>
  </body>
</html>
```



Add Bubble Phase Event Handler

obj.addEventListener('click', eventHandler);

Add Capture Phase Event Handler

obj.addEventListener('click', eventHandler, true);

Short for { capture: true }

Remove Bubble Phase Event Handler

obj.removeEventListener('click', eventHandler);

Remove Capture Phase Event Handler

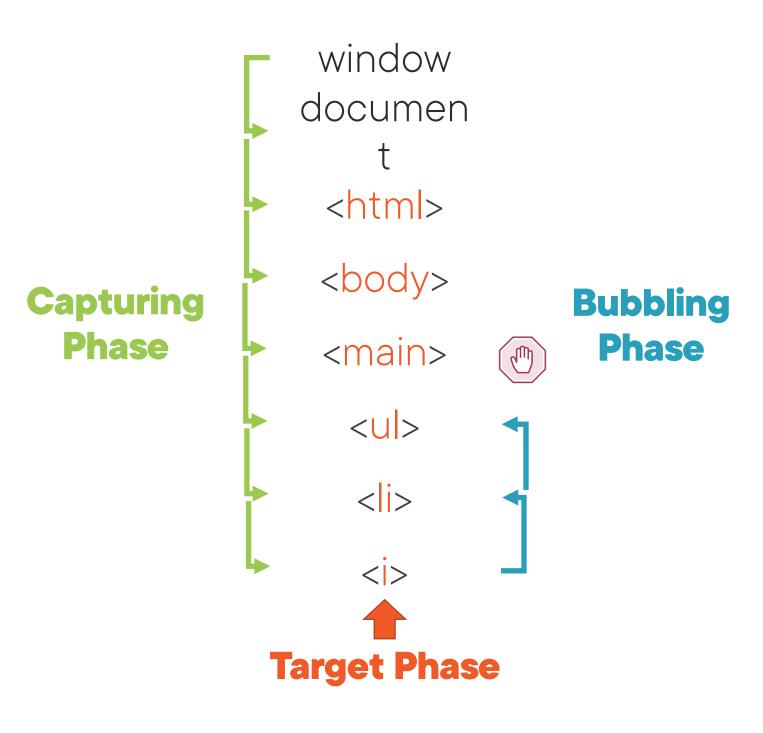
obj.removeEventListener('click', eventHandler, true);

false by default

Understanding Event Delegation

Event Delegation





Prevent Event Propagation

```
event.stopPropagation ();
```

Prevents any further propagation of the event in either the capturing or bubbling phases.

```
const list = document.querySelector('ul');
list.addEventListener('click', event => {
  event.stopPropagation();
  const reaction =
event.target?.dataset?.reaction;
  if (!reaction) return;
  console.log(`${event.type}:${reaction}`);
});
```

- **◄** Get element reference
- Add new click event listener
- Call the event's 'stopPropagation()'
- **◄** Get the attribute value for 'data-reaction'

- Check to ensure the reaction is not undefined
- Log the event type and class name to the console

Creating and Publishing Custom Events

Creating Custom Events

```
const event = new Event('ping')

Event object

Const event = new Event('ping')

Event name
```

Publish Events

```
Publish event

obj.dispatchEvent(event);

Object reference Event object
```

Handling Custom Events

```
Add event handler

obj.addEventListener('ping',
etentHandler);

Object reference

Callback function

the continuous properties of th
```

Passing Custom Event Data

```
const event = new CustomEvent('ping', {
    detail: { pong: true }
})
Custom event data
```

```
// Index.html
<input id="clickMe" type="button"</pre>
value="Ping">
// Script.js
const btn =
document.getElementById('clickMe');
btn.addEventListener('ping', (event) => {
    console.log(event.detail);
});
const event = new CustomEvent('ping', {
  detail: 'pong'
});
btn.dispatchEvent('ping', event);
```

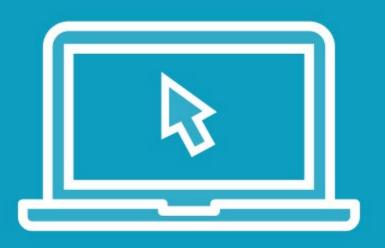
■ Refer to the ID attribute's value

◄ Get the element reference

■ Add event listener to the custom event

◄ Publish the event

Demo



- Referencing jQuery.com documentation
 - Accessing API documentation
 - Accessing the learning center

Summary



- Understand browser event processing
- Introduced to the Event object model
- Introduced to event handlers
- Understand event propagation
- Understand event delegation
- Learned to create custom events
- How-to reference jQuery documentation