

# JavaScript, Widgets, & Events

Prof. Lydia Chilton  
COMS 4170  
31 January 2024

# Warm-up: Section Preference

(Can you come to class Wednesdays in person?)

## Section Preferences

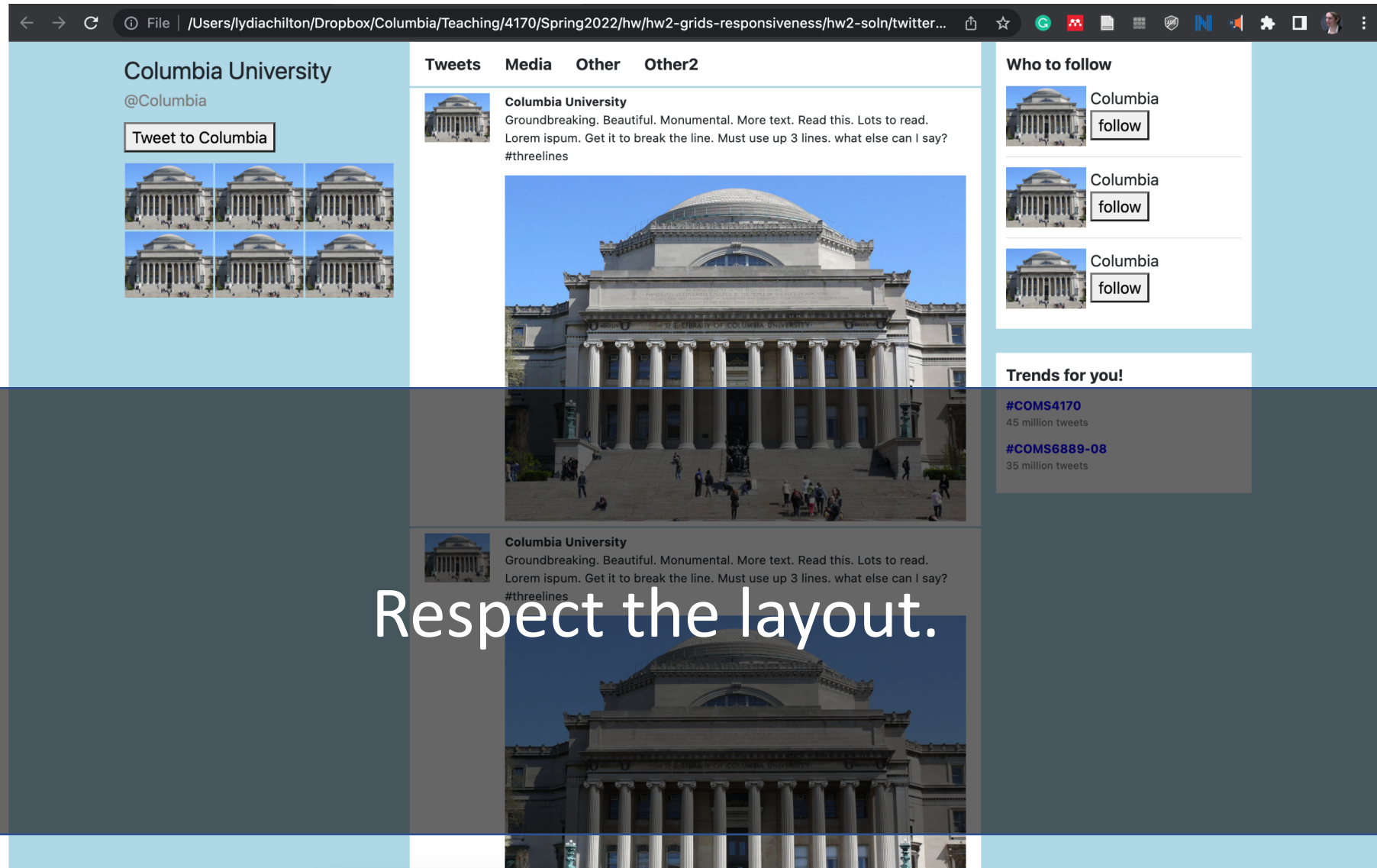
After spring break, can you attend class during class time in person on Wednesdays? \*

If you can only attend a part of class (like 1:10-1:45) that is good enough.

- Yes, I can attend class Wednesdays during class time in person
- I am free during class time Wednesdays, but can only meet on zoom
- I cannot make it during Wednesdays class time (neither in person nor zoom)
- I am a CVN student (your feedback sessions will be asynchronous)



# Homework 2 was hard!



The image shows a browser window displaying the Twitter profile of Columbia University (@Columbia). The browser's address bar shows the file path: /Users/lydiachilton/Dropbox/Columbia/Teaching/4170/Spring2022/hw/hw2-grids-responsiveness/hw2-soln/twitter... The profile header includes the name 'Columbia University' and handle '@Columbia'. Below the header is a 'Tweet to Columbia' button and a grid of six small images of the university's main building. The main content area is titled 'Tweets' and shows a tweet from 'Columbia University' with a large image of the building and text: 'Groundbreaking. Beautiful. Monumental. More text. Read this. Lots to read. Lorem ispum. Get it to break the line. Must use up 3 lines. what else can I say? #threelines'. To the right, there is a 'Who to follow' section with three entries for 'Columbia' and a 'Trends for you!' section with two trending hashtags: #COMS4170 (45 million tweets) and #COMS6889-08 (35 million tweets). A large, semi-transparent grey 'X' is overlaid on the bottom half of the page, with the text 'Respect the layout.' centered in white.

Columbia University  
@Columbia

Tweet to Columbia

Tweets Media Other Other2

Columbia University  
Groundbreaking. Beautiful. Monumental. More text. Read this. Lots to read. Lorem ispum. Get it to break the line. Must use up 3 lines. what else can I say? #threelines

Who to follow

Columbia follow

Columbia follow

Columbia follow

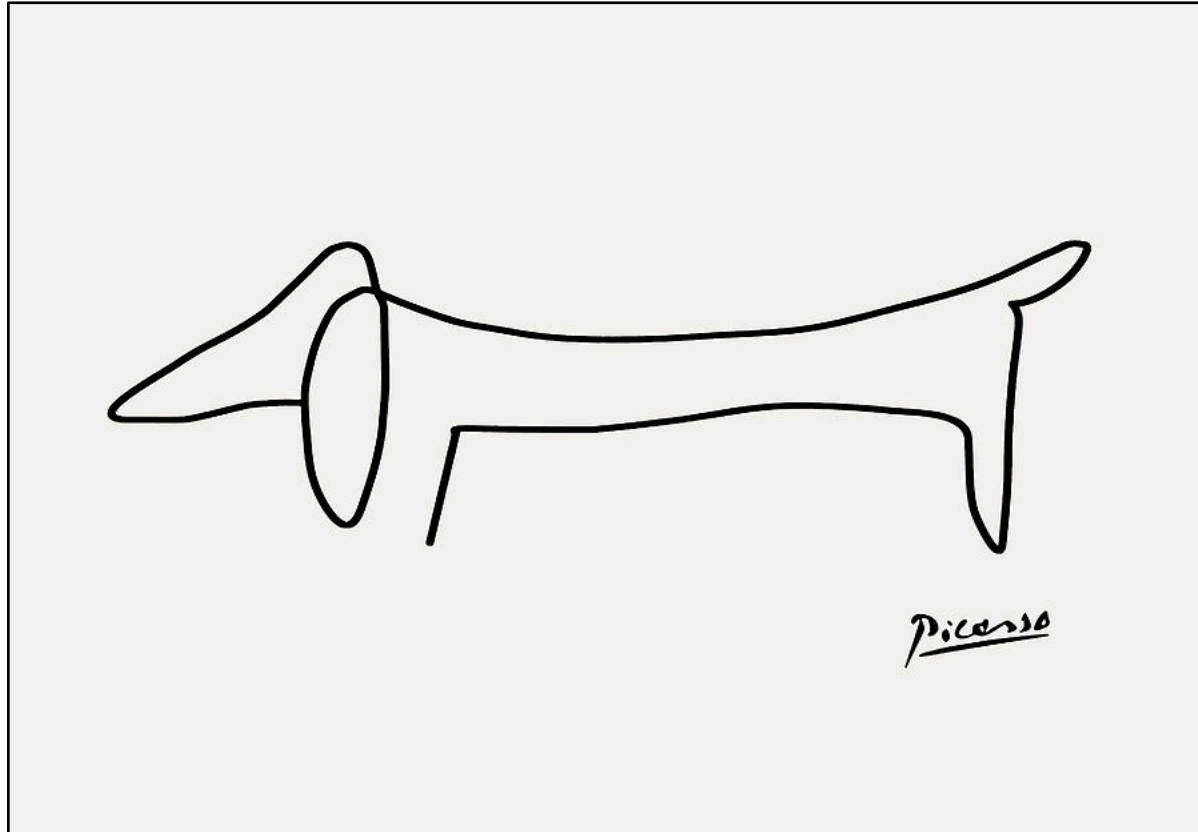
Trends for you!

#COMS4170  
45 million tweets

#COMS6889-08  
35 million tweets

Respect the layout.

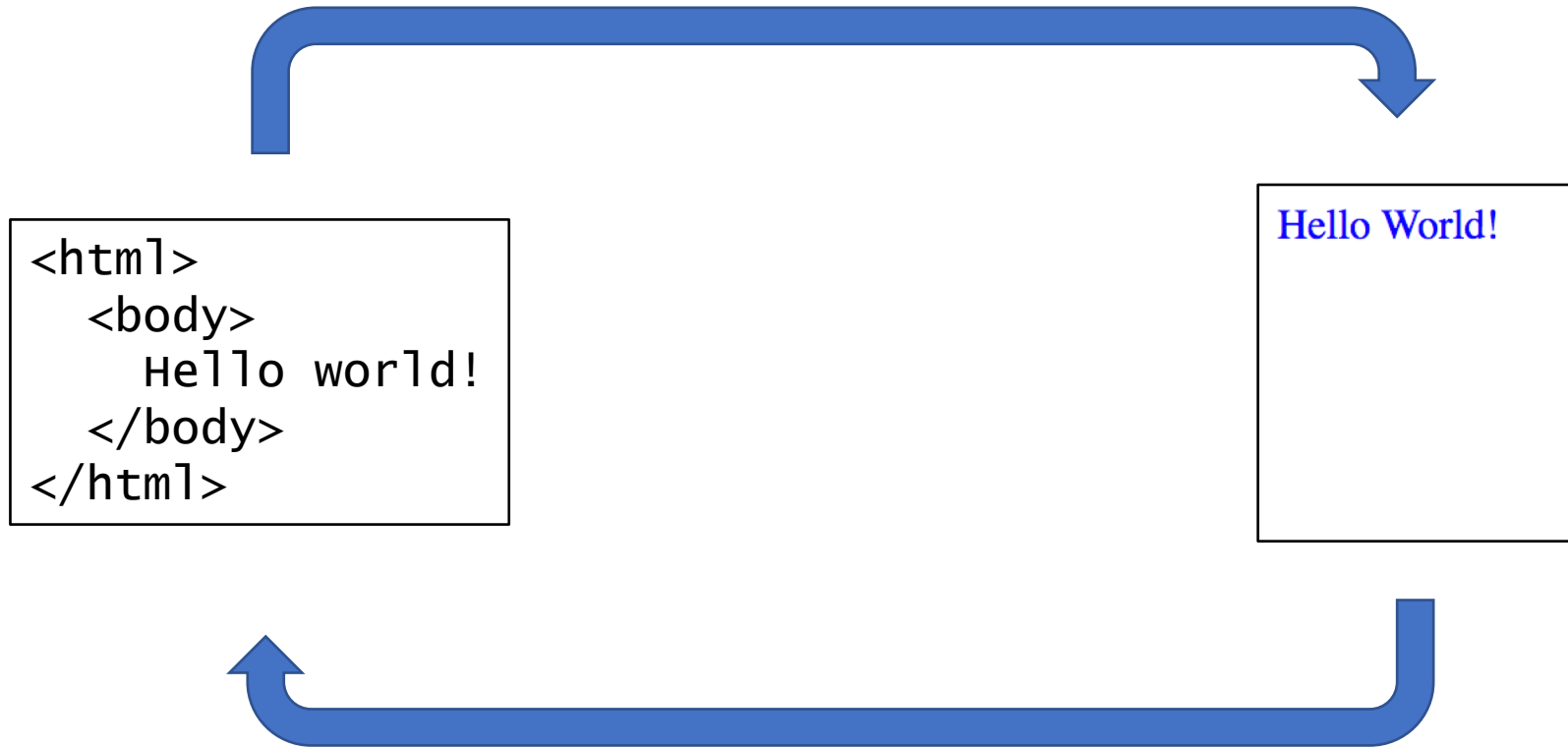
Simple is hard.



# Iterative Style of Programming

helps you build a mental model of your code.

What's the **smallest** unit of progress I can make?



Does it look ok?



# JavaScript, Widgets, & Events

Prof. Lydia Chilton  
COMS 4170  
2 February 2022

Raise your hand or type in zoom



# Users interact with the system to accomplish a goal.

The screenshot shows the Amazon Books homepage. At the top, there is a search bar with the word "Books" on the left and a magnifying glass icon on the right. To the right of the search bar is a link to "Shop Valentine's Day Deals". Below the search bar is a navigation bar with "Departments" and various links like "Browsing History", "Lydia's Amazon.com", "Today's Deals", "Gift Cards", "Registry", "Sell", and "Help". On the right side of the navigation bar, there are links for "EN", "Hello, Lydia", "Account & Lists", "Orders", "Prime", and a shopping cart icon with "1" item. Below the navigation bar is a horizontal menu with links for "Books", "Advanced Search", "New Releases", "Amazon Charts", "Best Sellers & More", "The New York Times® Best Sellers", "Children's Books", "Textbooks", "Textbook Rentals", "Sell Us Your Books", "Best Books of the Month", and "Kindle eBooks".

The main content area is divided into two columns. The left column is titled "Popular in Books" and lists categories like "Award Winners", "Bargain Books", "Best Books of the Month", "Best Books of 2017", "Books in Spanish", "Children's Books", "Deals in Books", and "Top 20 Lists in Books". Below this is a "More in Books" section with links for "100 Books to Read in a Lifetime", "Amazon Book Review Blog", "Amazon Books on Facebook", "Amazon Books on Twitter", and "Amazon First Reads".

The right column displays five book covers with their respective titles and authors. Each book listing includes the title, author(s), format (Paperback), a star rating, and the number of reviews. The first book is "The Craft of Research, Third Edition" by Wayne C. Booth, Gregory G. Colomb, and Joseph M. Williams, with a 4.5-star rating and 384 reviews. The second is "A Manual for Writers of Research Papers, Theses, and Dissertations, 7th Edition" by Kate L. Turabian and Wayne C. Booth, with a 4.5-star rating and 753 reviews. The third is "Academic Writing for Graduate Students, 3rd Edition" by John M. Swales and Christine B. Feak, with a 4.5-star rating and 128 reviews, priced at \$25.65 with Prime. The fourth is "The Craft of Research, 2nd edition" by Wayne C. Booth and Joseph M. Williams, with a 4.5-star rating and 384 reviews. The fifth is "A Manual for Writers of Research Papers, Theses, and Dissertations, Ninth Edition" by Kate L. Turabian and Wayne C. Booth, with a 4.5-star rating and 384 reviews, priced at \$18.00 with Prime.

## To buy a book.

# The designer must create the subgoals and interactions to help them accomplish it.

**Goal:** Buy a book

**Subgoal:**

Find it

Add to cart

Enter payment info

Place order

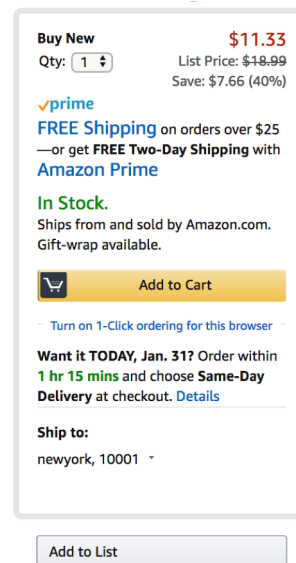
**Interaction:**

Type, click

click

Type, click, point

Click



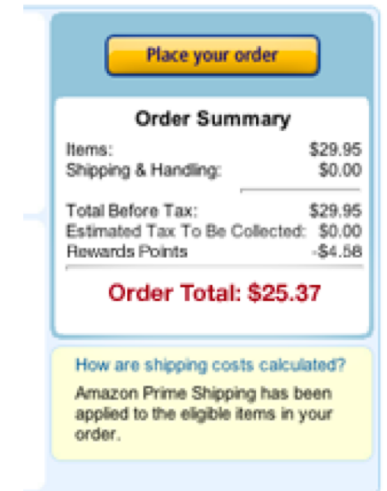
Name (as it appears on your card)

Card number (no dashes or spaces)

Expiration date

Security code (3 on back, Amex: 4 on front)

Low-level interactions take time and effort.  
Minimize them because you do them a lot.

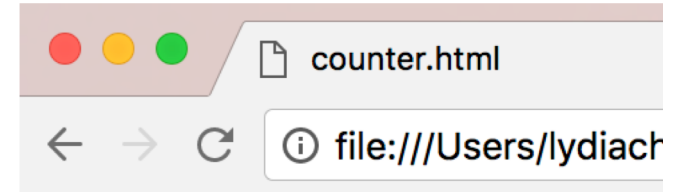
Move  
Click  
Move Click  
TypeTypeTypeType  
Move Click  
TypeTypeTypeType  
Move Tunnel Click  
Move Tunnel Click  
TypeTypeTypeType



# Creating Interactions on the web has two parts:

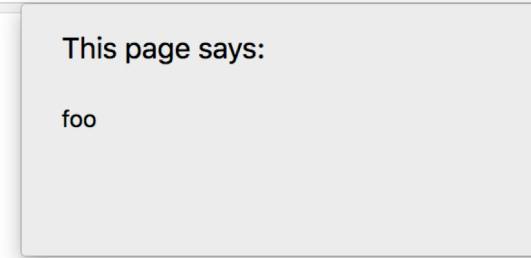
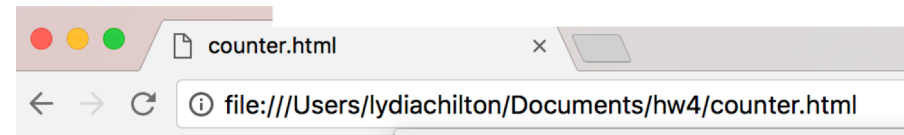
## 1. Program the interface and style in HTML & CSS

```
30  
31 <body>  
32  
33   <button id="counter" class="btn btn-primary">Counter (0)</button>  
34  
35 </body>  
36
```



## 2. Program interactions is JavaScript

```
25  
26 $(document).ready(function(){  
27   $("#counter").click(function(){  
28     alert("foo")  
29   })  
30 })  
31
```

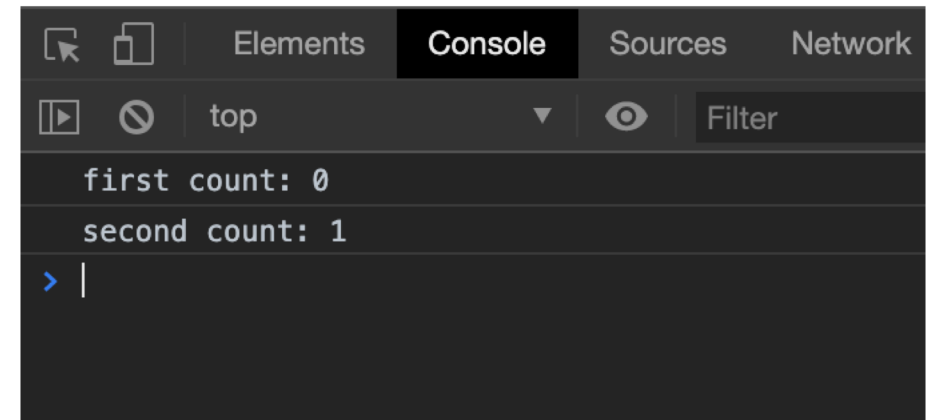
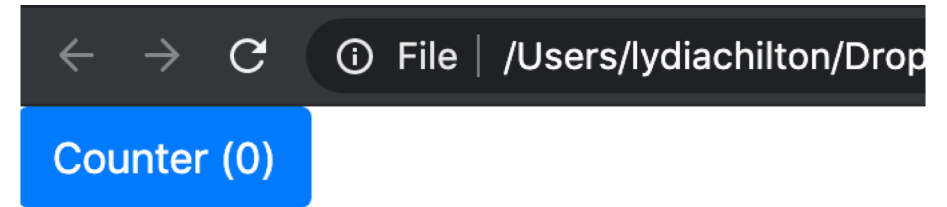




# Web Page Execution

# Browsers execute an HTML file from top to bottom. What will this execute?

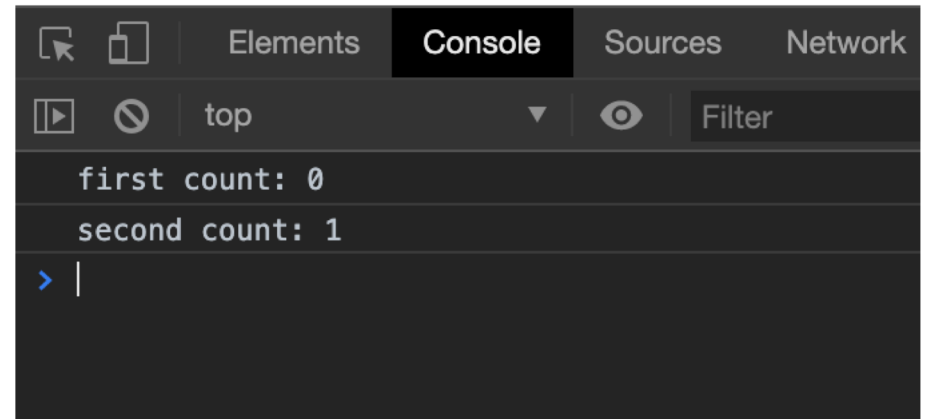
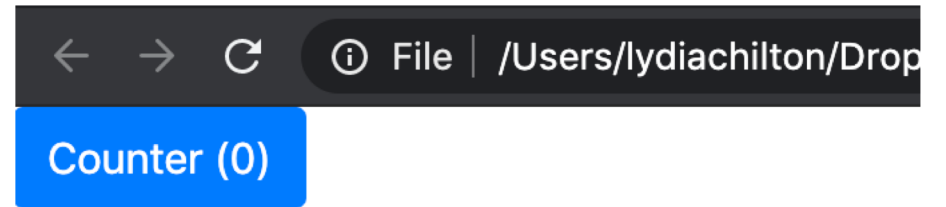
```
1 <html>
2 <head>
3   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boot
4     <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossor
5
6   <script>
7
8     var count = 0
9
10    function incrementCount(c) {
11      return c + 1;
12    }
13
14    console.log("first count: "+count)
15    count = incrementCount(count)
16
17    console.log("second count: "+count)
18
19  </script>
20
21 </head>
22
23
24
25
26 <body>
27   <button id="counter" class="btn btn-primary">Counter (0)</button>
28 </body>
29
30 </html>
31
```



However, JavaScript functions will get “hoisted.”  
Meaning, you can use them anywhere in scope.

```
1 <html>
2 <head>
3   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boot
4     <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossor
5
6   <script>
7
8     var count = 0
9
10    console.log("first count: "+count)
11    count = incrementCount(count)
12
13    console.log("second count: "+count)
14
15    function incrementCount(c) {
16      return c + 1;
17    }
18
19  </script>
20
21 </head>
22
23
24
25
26 <body>
27   <button id="counter" class="btn btn-primary">Counter (0)</button>
28 </body>
29
30 </html>
31
```

Still works!



# There is another (worse) way to declare functions that will not be hoisted.

```
<script>

  var count = 0

  count = incrementCount(count)

  // this is a function expression, it will execute whenever it is called
  function incrementCount1(c) {
    return c + 1;
  }

  // This is a variable
  // it will only be available only after it is executed by the browser
  // this will create an error in this case
  var incrementCount2 = function(c) {
    return c + 1;
  }

</script>
```

Do it this way.

# Three ways of declaring variables: let, const, var

freeCodeCamp (🔥)

Learn to code – free 3,000-hour curriculum

APRIL 2, 2020 / #JAVASCRIPT

## Var, Let, and Const – What's the Difference?



Sarah Chima Atuonwu



# Let is the preferred way to declare variables.

```
1
2 let greeting = "hello"
3
4 console.log("0: "+greeting)
5
6 ▾ if(true){
7   greeting = "hi"
8   console.log("1: "+greeting)
9 }
10
11 console.log("2: "+greeting)
12
13
```

```
>_ Console (beta) 3 0 0 0 0 Clear console Minimize
"Running fiddle"
"0: hello"
"1: hi"
"2: hi"
>_
```

let is block scoped, and can be re-assigned.

# let will not be defined outside of scope.

```
1
2 ▾ if(true){
3   let hello = "hello"
4   console.log("1: "+hello)
5 }
6
7 console.log("2: "+hello) // this is undefined
```

>\_ Console (beta) ⓘ 1 ⓘ 0 ⚠ 0 ❌ 1 Clear console Minimize

☁ "Running fiddle"

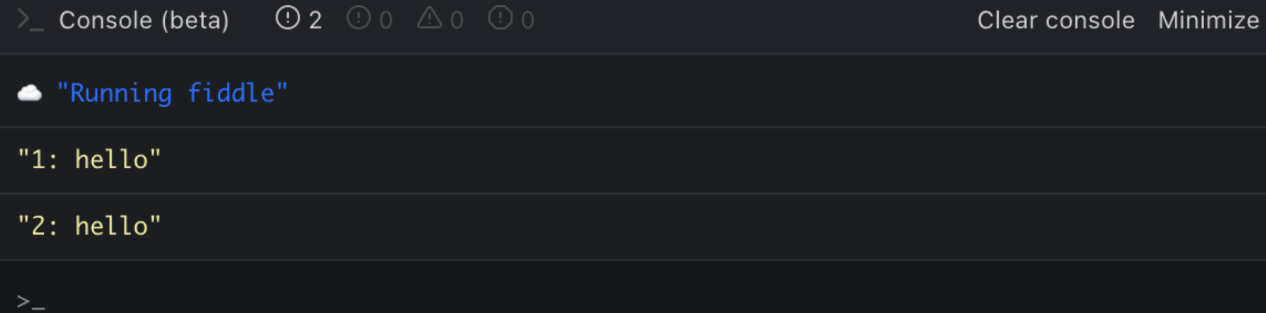
"1: hello"

"<a class='gotoLine' href='#47:19'>47:19</a> Uncaught ReferenceError: hello is not def

>\_

# Only use **var** when you really want a global variable

```
1
2 ▾ if(true){
3   var hello ="hello"
4   console.log("1: "+hello)
5 }
6
7 console.log("2: "+hello)
```



>\_ Console (beta) ⓘ 2 ⓘ 0 ⚠ 0 ⓘ 0 Clear console Minimize

☁ "Running fiddle"

"1: hello"

"2: hello"

>\_

**var** is globally scoped (or function scoped)  
**var** variables can be updated and re-declared  
"foo = 1" is the same as "var foo = 1"



# const is good for constant variables

```
1  const debug_mode = true
2
3  ▼ if(debug_mode){
4    let hello = "hello"
5    console.log("1: "+hello)
6  }
7
8
```

```
>_ Console (beta) 1 0 0 0 Clear console Minimize
"Running fiddle"
"1: hello"
>_
```

**const** is block scoped, and cannot be re-assigned.

# let and const are good ways to declare variables.

(var is globally scoped, and can get you into trouble)

```
1  const debug_mode = true
2
3  if(debug_mode){
4    let hello = "hello"
5    console.log("1: "+hello)
6  }
7
8  console.log("2: "+hello) // not declared
```

> Console (beta) 2 0 0 1 Clear console Minimize

"hello"

☁ "Running fiddle"

"1: hello"

"<a class='gotoLine' href='#48:19'>48:19</a> Uncaught ReferenceError: hello is not de

**let** is block scoped, and can be re-assigned.

**const** is block scoped and cannot be re-assigned.

Adding events

# When you click this button, what will it do?

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boots

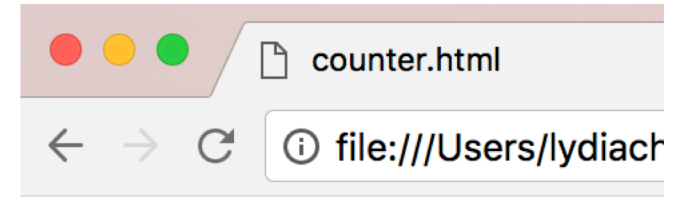
  <script>

    var count = 0

    function incrementCount() {
      return c + 1;
    }
  </script>

</head>

<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>
</html>
```



Counter (0)

Nothing

# To add click handlers nicely, we're first going to include JQuery (a JS extension)

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css" >
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorigin="anonymous"></script>

  <script>
    var count = 0

    function incrementCount() {
      return c + 1;
    }
  </script>

</head>

<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>

</html>
```



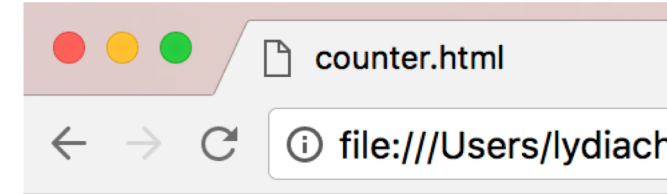
Syntax is similar to but different from...

Including Bootstrap

# If we add an event, what will it do?

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" >
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorigin="anonymous">
  <script>
    $("#counter").click(function(){
      alert("foo")
    })
  </script>
</head>

<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>
</html>
```



Counter (0)

Nothing

# If we add an event after the document is loaded, will it finally work??

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boo
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorig

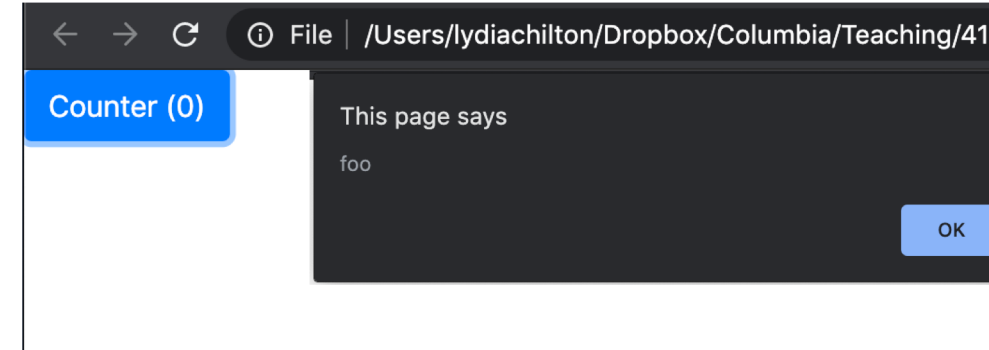
  <script>

    $(document).ready(function(){
      $("#counter").click(function(){
        alert("foo")
      })
    })
  </script>

</head>

<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>

</html>
```



YES!!!!

# We added an event. Yay!

## How do we increment the counter?

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boot
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorig

  <script>

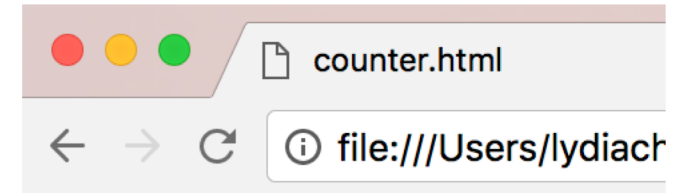
    $(document).ready(function(){
      $("#counter").click(function(){
        alert("foo")
      })
    })

  </script>

</head>

<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>

</html>
```

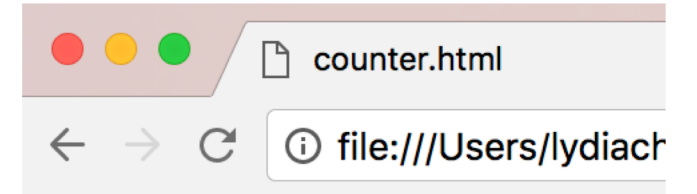


Counter (0)

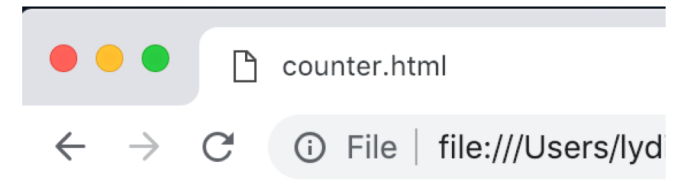


# How do we increment the count?

```
1 <html>
2 <head>
3   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" >
4   <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorigin="anonymous">
5
6   <script>
7
8     var count = 0
9
10    function incrementCount(c) {
11      return c + 1;
12    }
13
14
15    $(document).ready(function(){
16      $("#counter").click(function(){
17        count = incrementCount(count)
18        $("#counter").html("Counter (" + count + ")")
19      })
20    })
21  </script>
22
23
24 </head>
25
26
27
28
29 <body>
30   <button id="counter" class="btn btn-primary">Counter (0)</button>
31 </body>
32
33 </html>
34
```



Counter (0)



Counter (1)

# Incrementing the count differently.

```
1 <html>
2 <head>
3   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
4   <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorigin="anonymous"></script>
5
6   <script>
7
8     var count = 0
9
10    function incrementCount(c) {
11      return c + 1;
12    }
13
14
15    $(document).ready(function(){
16      $("#counter").click(function(){
17        count = incrementCount(count)
18        // $("#counter").html("Counter (" + count + ")")
19        $("#count").html(count)
20      })
21    })
22
23  </script>
24
25
26 </head>
27
28
29
30 <body>
31   <button id="counter" class="btn btn-primary">Counter (<span id="count">0</span></button>
32 </body>
33
34 </html>
```

# Jquery vs. Pure JavaScript

jQuery is a JavaScript library that make JavaScript easier (and standard across browsers)

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

## jQuery

```
$("#counter").click(function(){  
    $("#counter").html("Counter (0)");  
});
```

What's the JS equivalent to `$("#counter")` ?

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

## JQuery

```
$("#counter").click(function(){  
    $("#counter").html("Counter (0)");  
});
```

# document.getElementById("counter")

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

## jQuery

```
$("#counter").click(function(){  
    $("#counter").html("Counter (0)");  
});
```

# What's the JavaScript equivalent of \$(element).click(...)

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

## jQuery

```
$("#counter").click(function(){  
    $("#counter").html("Counter (0)");  
});
```

We used `$("#counter")` again...  
Is that normal?

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

## jQuery

```
$("#counter").click(function(){  
    $("#counter").html("Counter (0)");  
});
```



# Can use **this** within scope

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    this.innerHTML = "Counter (0)";  
});
```

## jQuery

```
$("#counter").click(function(){  
    $(this).html("Counter (0)");  
});
```

# What's the JavaScript equivalent of setting html?

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    this.innerHTML = "Counter (0)";  
});
```

## JQuery

```
$("#counter").click(function(){  
    $(this).html("Counter (0)");  
});
```

# Will this work?

JavaScript

JQuery

```
document.getElementById("counter").click(function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

No.

Don't mix pure JavaScript with JQuery in the same line.

For your own sanity. Only use JQuery

Don't do this (even though it will work)

```
<button onclick="myFunction()">Click me</button>
```

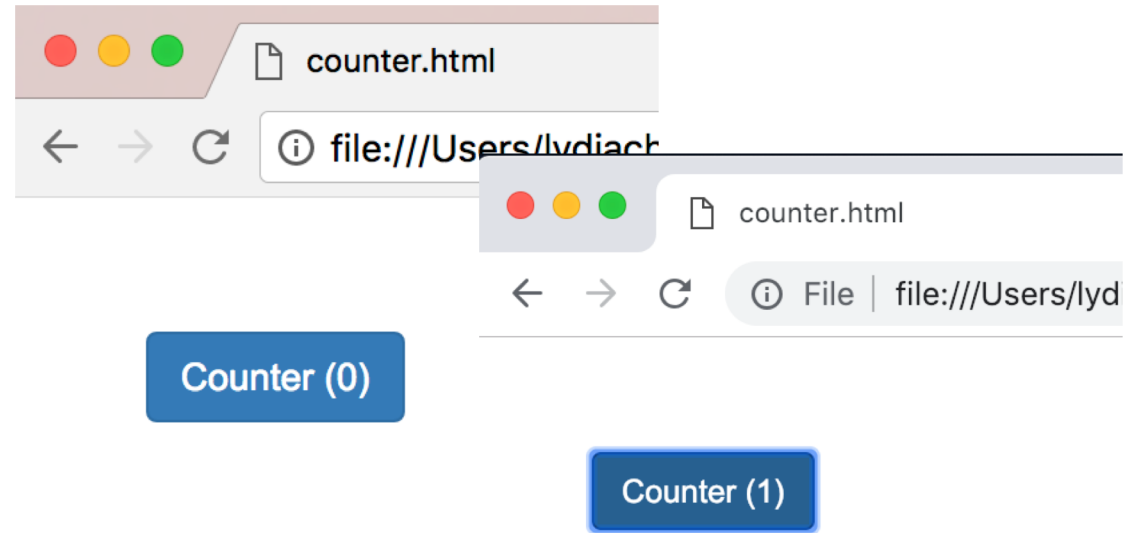
```
<button onclick="incrementCount(1)">Counter (1)</button>
```

# Good style of attaching events in JQuery

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" >
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" >
  </script>
  <script>
    var count = 0;

    function incrementCount(c) {
      return c + 1;
    }

    $(document).ready(function(){
      $("#counter").click(function(){
        count = incrementCount(count);
        $("#counter").html("Counter (" + count + ")");
      });
    });
  </script>
</head>
<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>
</html>
```



1. Uses JQuery (not pure JavaScript)
2. Attaches click handler as in the `<script>`  
`$(element).click(...)`  
(doesn't attach in HTML)
2. Uses `$(document).ready(...)`



**I Am Developer**  
@iamdeveloper



It's only jQuery if it's from the jQueré  
region of France. Otherwise it's just  
sparkling javascript

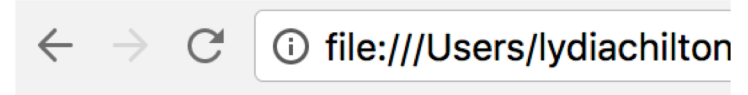
5:20 AM · 7/23/19 · [Twitter Web App](#)

# Creating Widgets Dynamically

# Statically created widget: created on page load.

## HTML

```
61 <body>
62
63     <button id="counter" class="btn btn-primary"></button>
64
65 </body>
66
```



Static Button (0)

## JavaScript

```
60
61 $(document).ready(function(){
62     $("#counter").click(function(){
63         // increment the counter
64     })
65 })
66
```



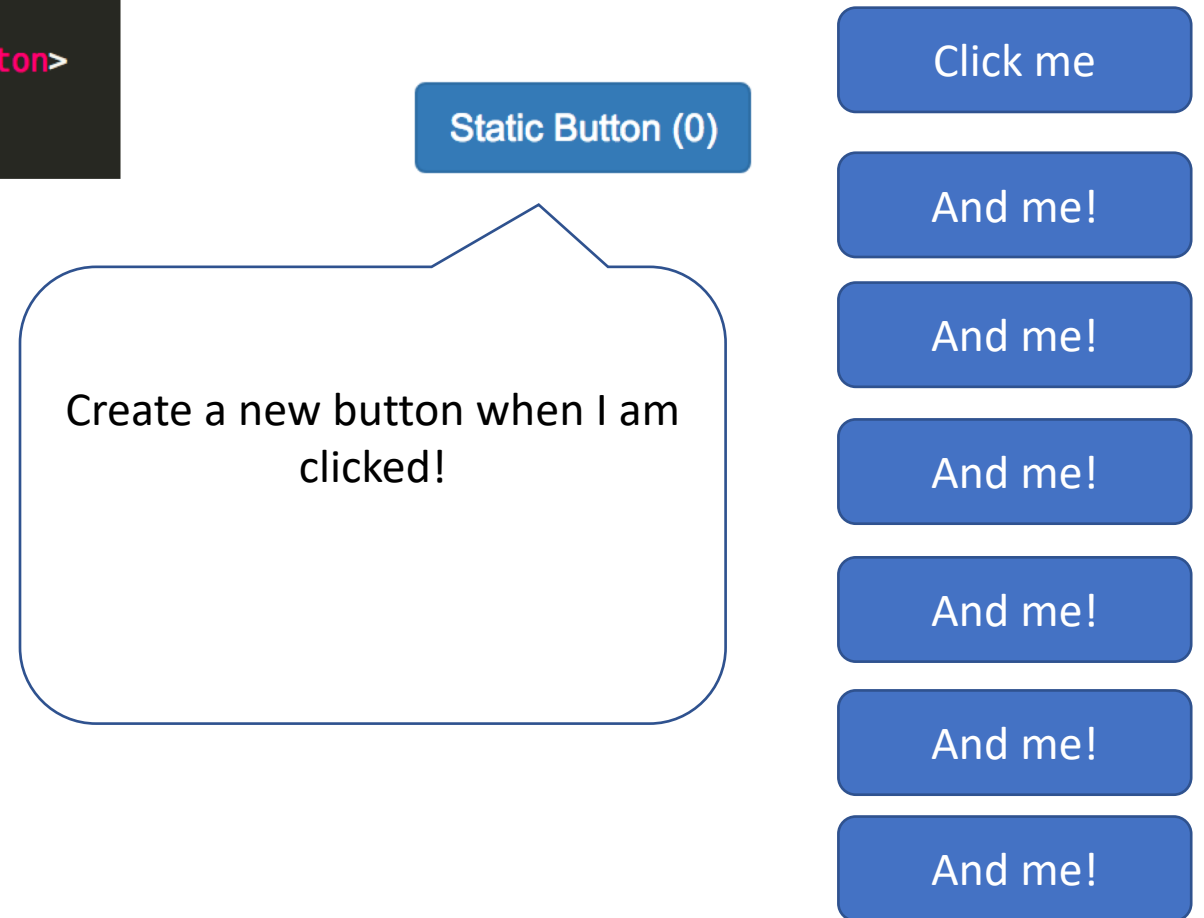
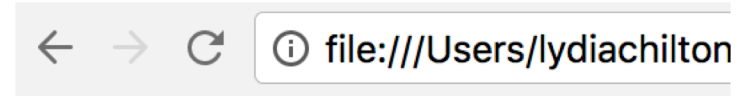
# Dynamically created widget: created on demand based on user interaction.

## HTML

```
61 <body>
62
63   <button id="counter" class="btn btn-primary"></button>
64
65 </body>
66
```

## JavaScript

```
61 $(document).ready(function(){
62   $("#counter").click(function(){
63     // increment the counter
64   })
65 })
66
```



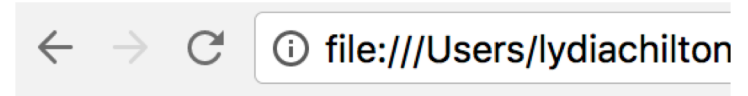
# Where in the code should we add the dynamic behavior?

## HTML

```
61 <body>
62
63   <button id="counter" class="btn btn-primary"></button>
64
65 </body>
66
```

## JavaScript

```
60
61 $(document).ready(function(){
62   $("#counter").click(function(){
63     // increment the counter
64   })
65 })
66
```



Static Button (0)

Create a new button when I am clicked!

Click me

And me!

And me!

And me!

And me!

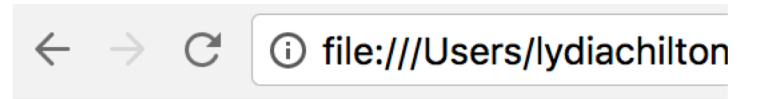
And me!

And me!

# How did we create the button in JavaScript?

## HTML

```
61 <body>
62
63   <button id="counter" class="btn btn-primary"></button>
64
65 </body>
66
```



Static Button (0)

## JavaScript

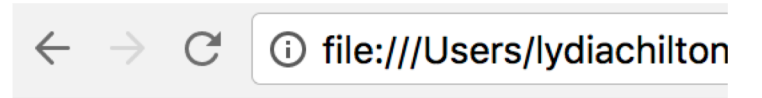
```
42 function createButton(){
43   var new_button = $("<button>")
44   $(new_button).text("dynamic button "+Date.now())
45 }
```

```
61 $(document).ready(function(){
62   $("#counter").click(function(){
63     // increment the counter
64     createButton()
65   })
66 })
67
```

First, add a div to HTML to contain the new buttons.

## HTML

```
61 <body>
62     <button id="counter" class="btn btn-primary"></button>
63     <hr><hr>
64     <div id="updates"></div>
65 </body>
```



## JavaScript

```
42 function createButton(){
43     var new_button = $("<button>")
44     $(new_button).text("dynamic button "+Date.now())
45 }
```

```
61 $(document).ready(function(){
62     $("#counter").click(function(){
63         // increment the counter
64         createButton()
65     })
66 })
```

# Add widget to UI dynamically

## HTML

```
61 <body>
62     <button id="counter" class="btn btn-primary"></button>
63     <br><br>
64     <div id="updates"></div>
65 </body>
```

Static Button (2)

dynamic button 1519059719092    dynamic button 1519059720090

## JavaScript

```
44 function createButton(){
45     var new_button = $("<button>")
46     $(new_button).text("dynamic button "+Date.now())
47     $("#updates").append(new_button)
48 }
```

```
61 $(document).ready(function(){
62     $("#counter").click(function(){
63         // increment the counter
64         createButton()
65     })
66 })
```

# How do we create a line break *dynamically*?

## HTML

```
61 <body>
62     <button id="counter" class="btn btn-primary"></button>
63     <br><br>
64     <div id="updates"></div>
65 </body>
```

Static Button (2)

dynamic button 1519059891686

dynamic button 1519059892439

## JavaScript

```
44 function createButton(){
45     var new_button = $("<button>")
46     $(new_button).text("dynamic button "+Date.now())
47     $("#updates").append(new_button)
48     $("#updates").append("<br>")
49 }
```

```
61 $(document).ready(function(){
62     $("#counter").click(function(){
63         // increment the counter
64         createButton()
65     })
66 })
67
```

# How do we create a bootstrap button dynamically?

## HTML

```
61 <body>
62     <button id="counter" class="btn btn-primary"></button>
63     <br><br>
64     <div id="updates"></div>
65 </body>
```

Static Button (2)

dynamic button 1519060044460

dynamic button 1519060044905

## JavaScript

```
44 function createButton(){
45     var new_button = $("<button class='btn btn-default'>")
46     $(new_button).text("dynamic button "+Date.now())
47     $("#updates").append(new_button)
48     $("#updates").append("<br>")
49 }
```

```
61 $(document).ready(function(){
62     $("#counter").click(function(){
63         // increment the counter
64         createButton()
65     })
66 })
```

# Where do we create a click event *dynamically*?

## HTML

```
61 <body>
62     <button id="counter" class="btn btn-primary"></button>
63     <br><br>
64     <div id="updates"></div>
65 </body>
```

Static Button (2)

dynamic button 1519060044460

dynamic button 1519060044905

## JavaScript

```
44 function createButton(){
45     var new_button = $("<button class='btn btn-default'>")
46     $(new_button).text("dynamic button "+Date.now())
47     $("#updates").append(new_button)
48     $("#updates").append("<br>")
49 }
```

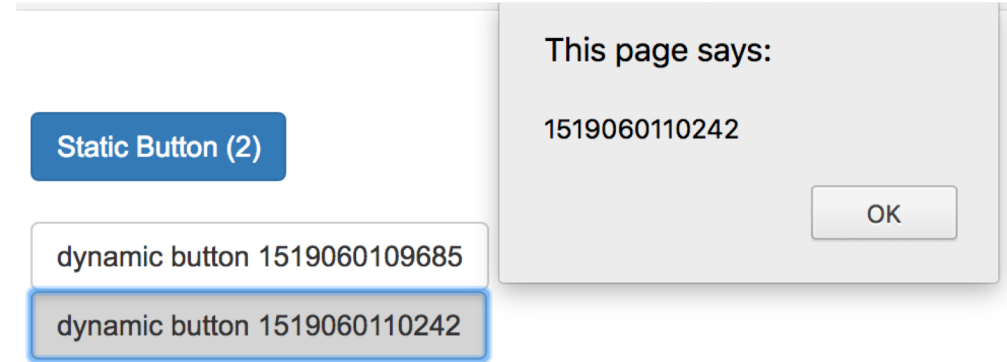
```
61 $(document).ready(function(){
62     $("#counter").click(function(){
63         // increment the counter
64         createButton()
65     })
66 })
67
```



# How do we create a click event *dynamically*?

## HTML

```
61 <body>
62     <button id="counter" class="btn btn-primary"></button>
63     <br><br>
64     <div id="updates"></div>
65 </body>
66
```



## JavaScript

```
44 function createButton(){
45
46     var new_button = $("<button class='btn btn-default'>")
47     $(new_button).text("dynamic button "+Date.now())
48     $("#updates").append(new_button)
49     $("#updates").append("<br>")
50
51     var d = Date.now()
52     $(new_button).click(function(){ alert(d) })
53 }
```

# You can create elements **statically** in HTML Or **dynamically** in JavaScript (jQuery)

## **Static:** HTML, JavaScript onReady

```
61 <body>
62   <button id="counter" class="btn btn-primary"></button>
63   <br><br>
64   <div id="updates"></div>
65 </body>
```

```
61 $(document).ready(function(){
62   $("#counter").click(function(){
63     // increment the counter
64     createButton()
65   })
66 })
```

## **Dynamic:** All JavaScript

```
44 function createButton(){
45
46   var new_button = $("<button class='btn btn-default'>")
47   $(new_button).text("dynamic button "+Date.now())
48   $("#updates").append(new_button)
49   $("#updates").append("<br>")
50
51   var d = Date.now()
52   $(new_button).click(function(){ alert(d) })
53 }
```

Static Button (2)

dynamic button 1519060109685

dynamic button 1519060110242

This page says:

1519060110242

OK

# Widgets and Events

Basic elements for users to interact with your UI

Buttons are one type of **widget**  
the main event they can respond to is **clicks**.

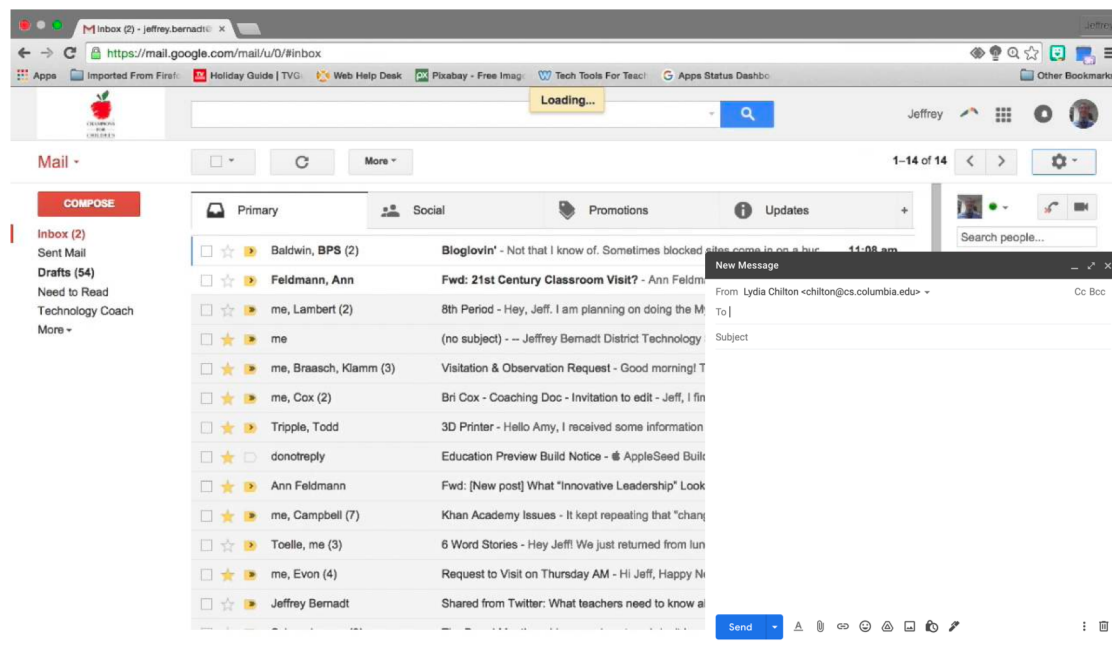
---



COMPOSE

```
$("#compose").click(function(){  
    //compose new email  
});
```

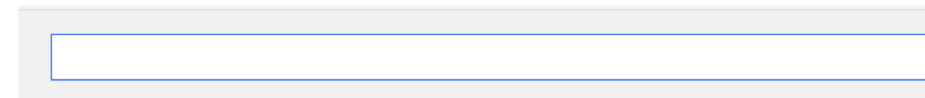
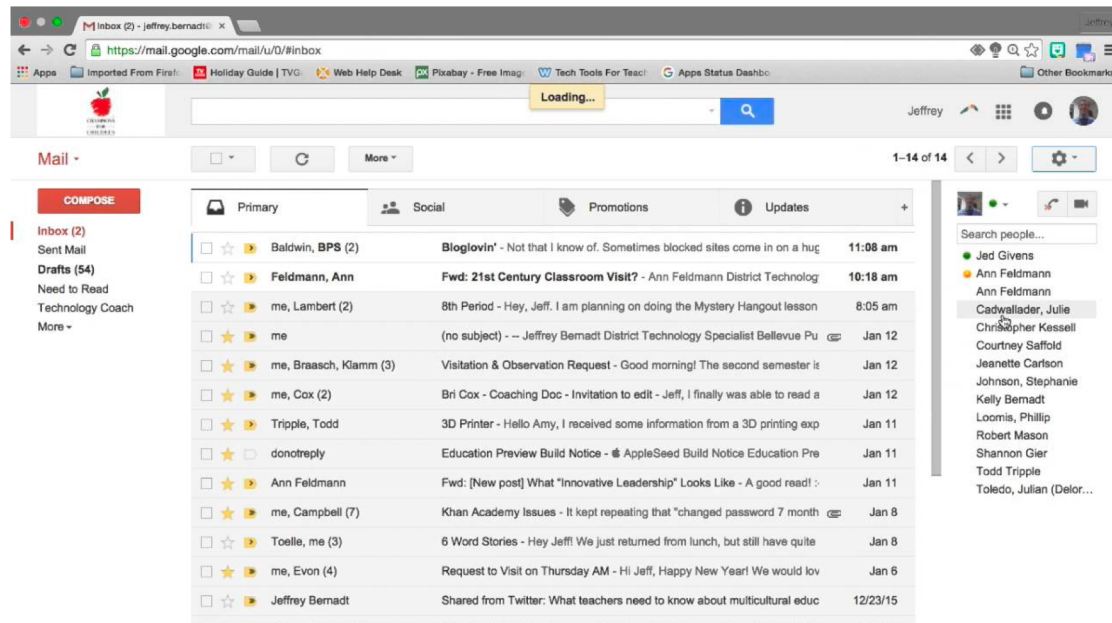
# Every time a button is clicked, a click event fires.



```
$("#compose").click(function(){  
    //compose new email  
});
```

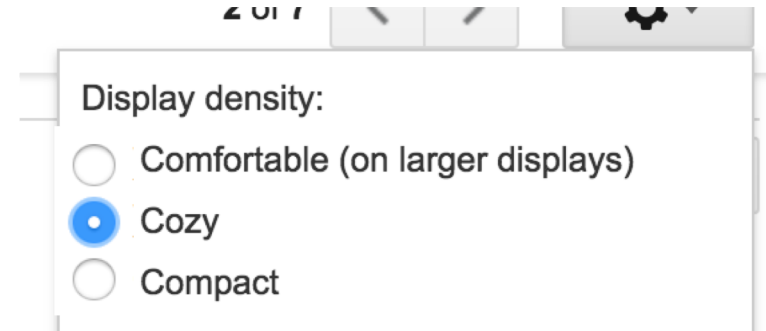
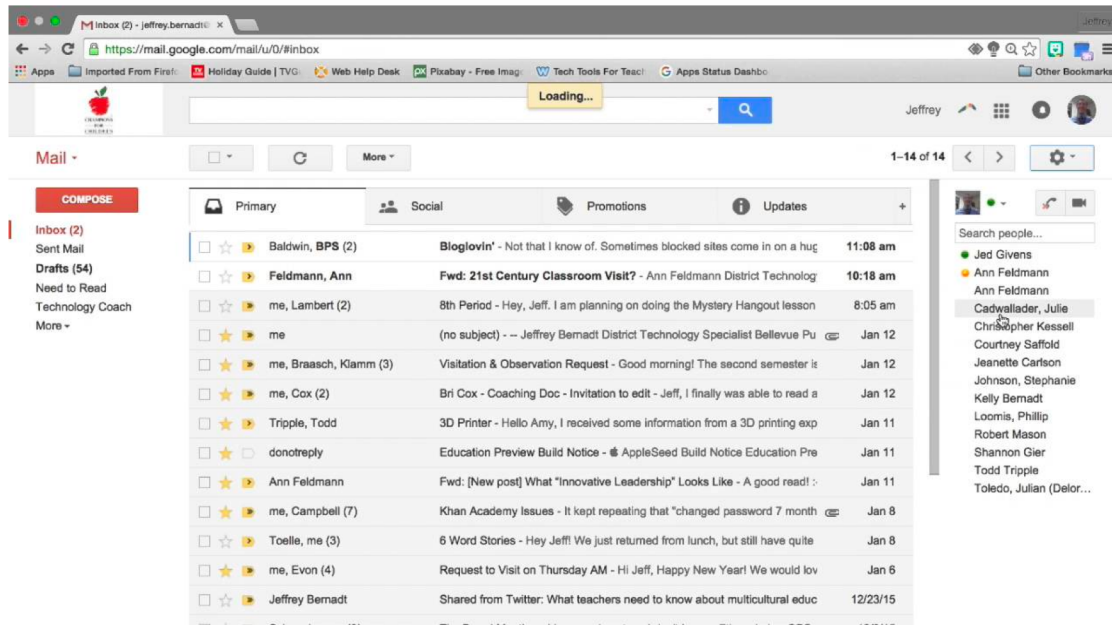
This code listens for the click event fire and does something is called the “click handler” (more generally: “event handler”)

# Text Input interaction: What event fires?



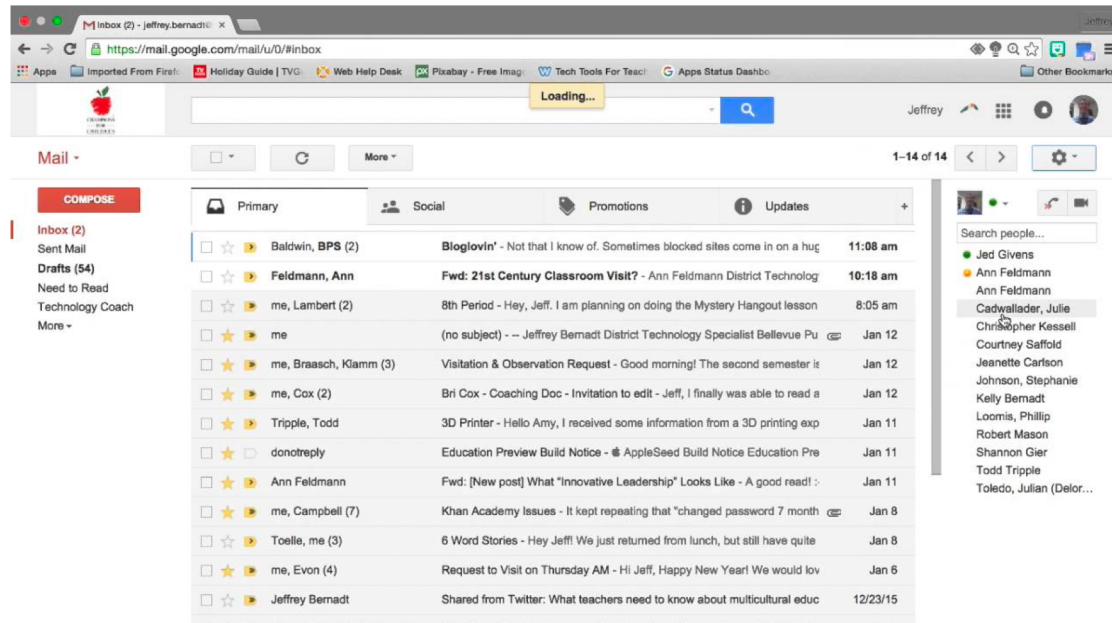
“Keypress” event

# Radio Input interaction: What event fires?



“Change” event

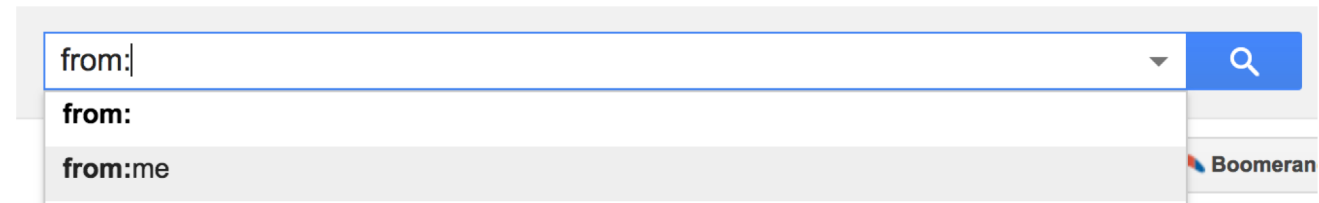
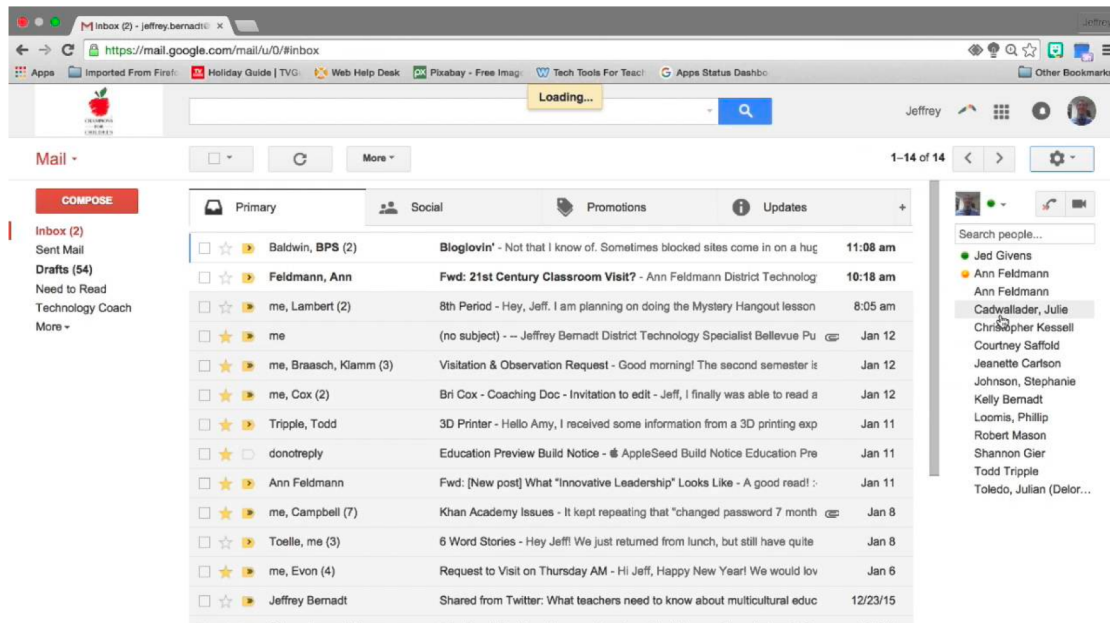
# Select Element interaction: What event fires?



“Change” event

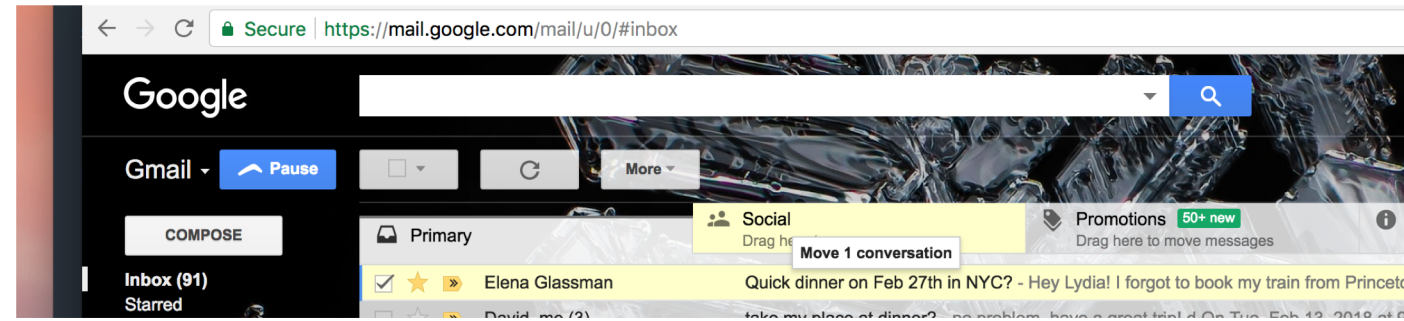
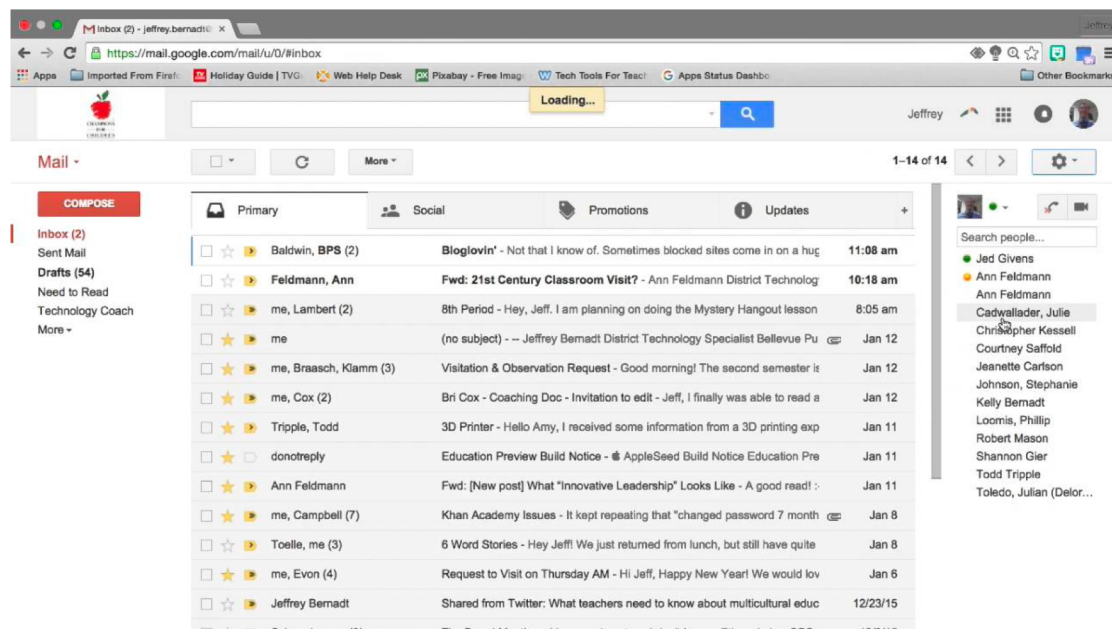


# Dropdown interaction: What event fires?



“Select” event

# Drag and Drop interaction: What events fire?

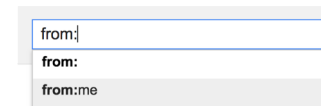
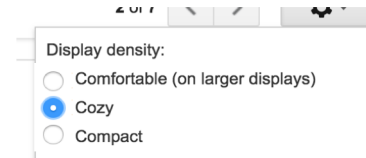
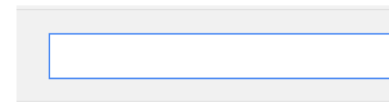
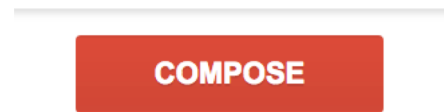


- “Drag” event
- “Drop” event

# Widgets are standardized low-level interaction interfaces that trigger events

When you create a widget...

The **appearance** is standardized,



The **types of events** it responds to are standardized

“Click”  
“hover”

“Keypress”

“Change”

“Select”  
“Search”

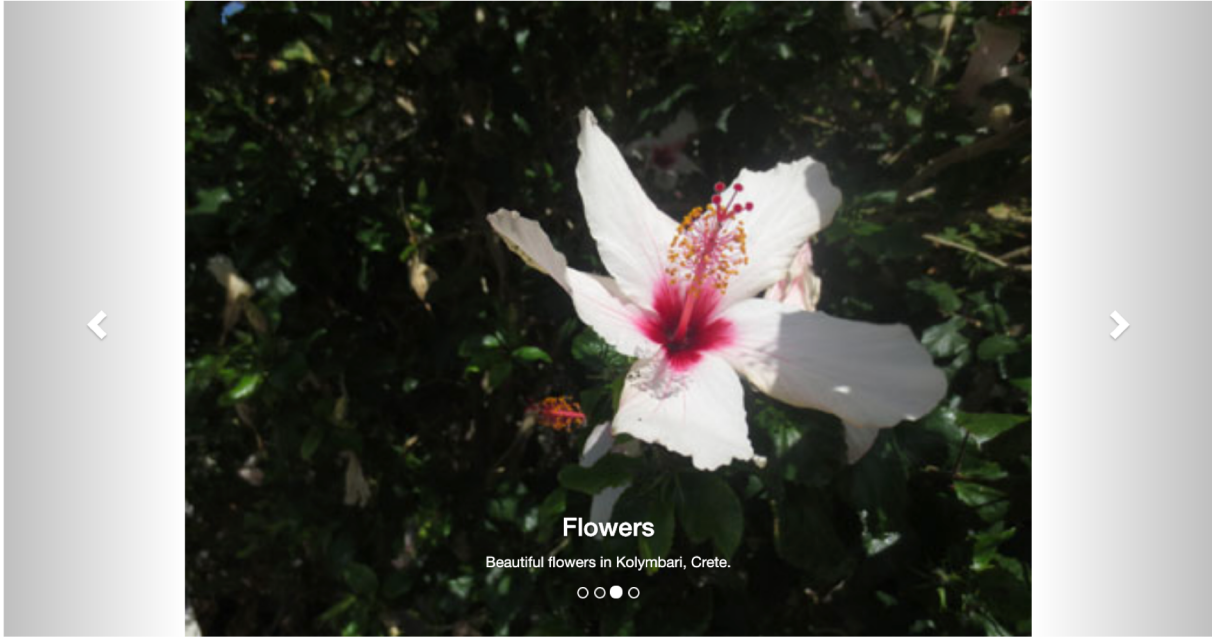
But the actions taken after an event is fired, are not standardized

# Widgets can also be big

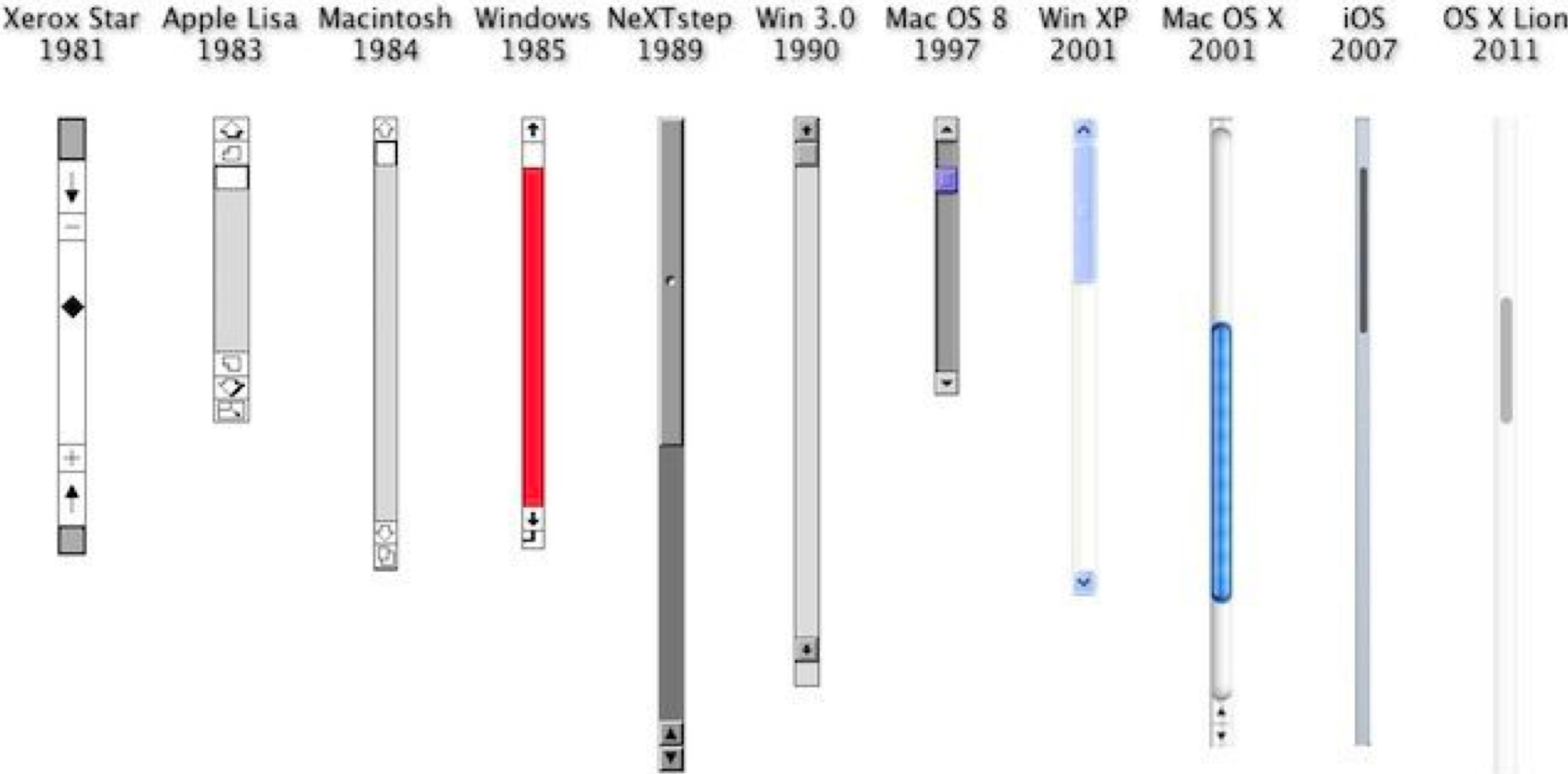
```
Run » Result Size: 1573 x 723
<div id="myCarousel" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
    <li data-target="#myCarousel" data-slide-to="1"></li>
    <li data-target="#myCarousel" data-slide-to="2"></li>
    <li data-target="#myCarousel" data-slide-to="3"></li>
  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner" role="listbox">

    <div class="item active">
      
      <div class="carousel-caption">
        <h3>Chania</h3>
        <p>The atmosphere in Chania has a touch of Florence and Venice.</p>
      </div>
    </div>
  </div>
</div>
```

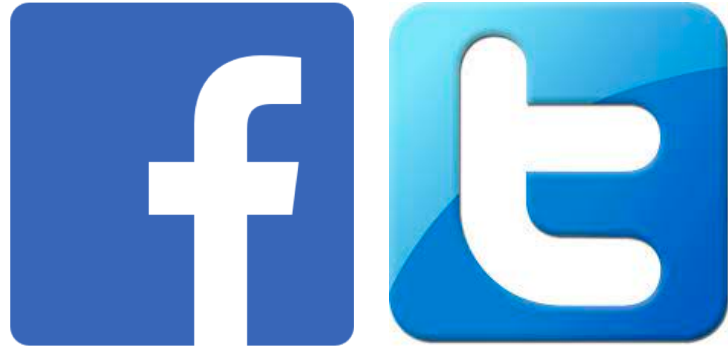


Because you did not program them yourself, widgets may appear and act differently on different devices

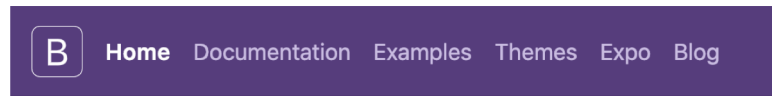


# Pros and Cons of Standardization

# Things that have become standardized



Because people people copy successful designs



## Bootstrap

Build responsive, mobile-first projects on the world's most popular front-end component lib

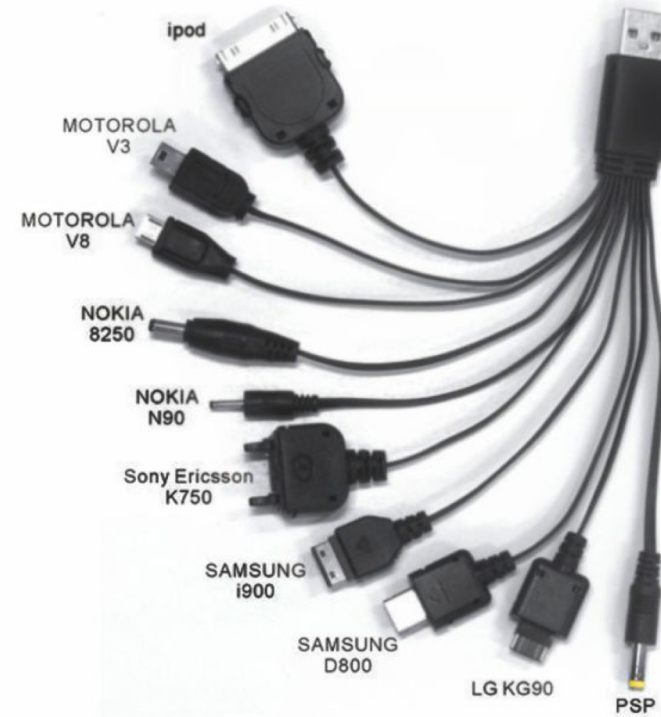
Because people create good, reusable solutions



Because one version domains the market



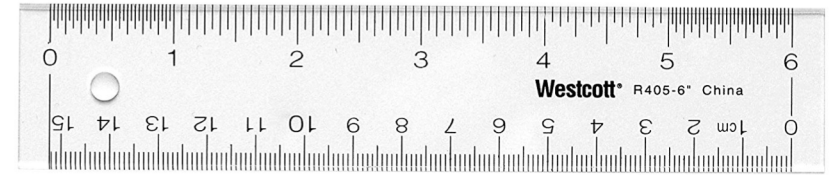
# Things that have not become standardized



“colour”, “honour”,  
“cheque”, “connexion”



# Old things that got standardized



# What's **good** about standardization?

Standardized



Non-Standardized





# What's **bad** about standardization?

Standardized



Non-Standardized

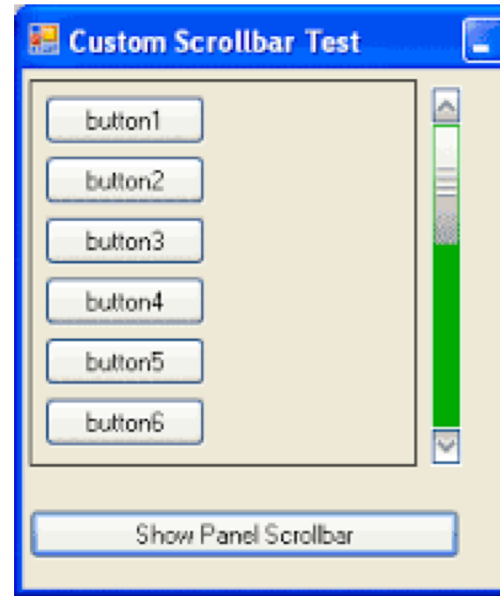


# Widgets allow customization

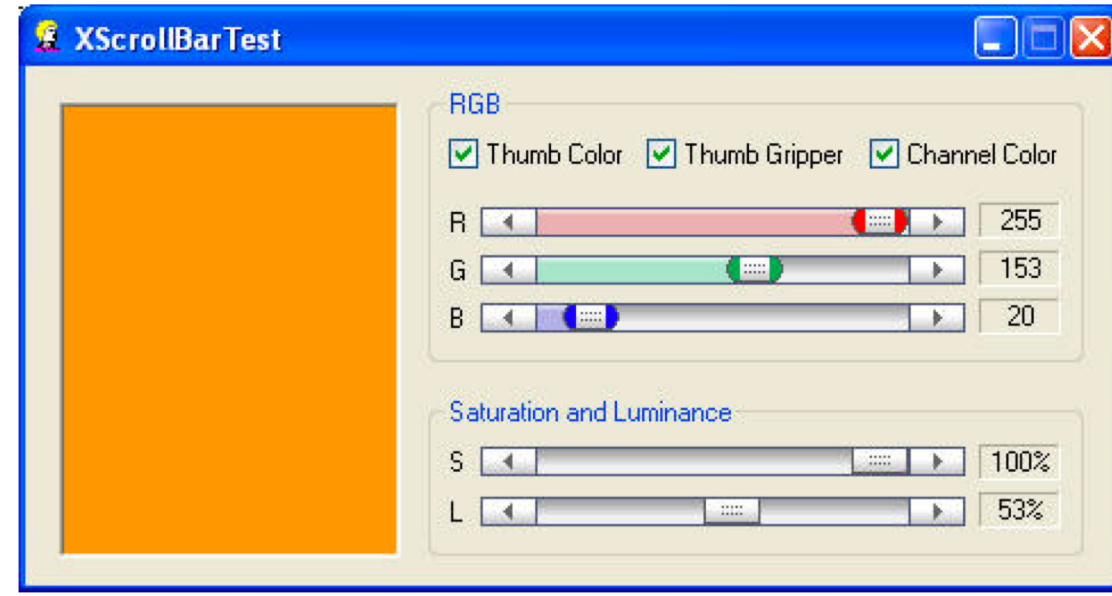
Customizable scroll bars



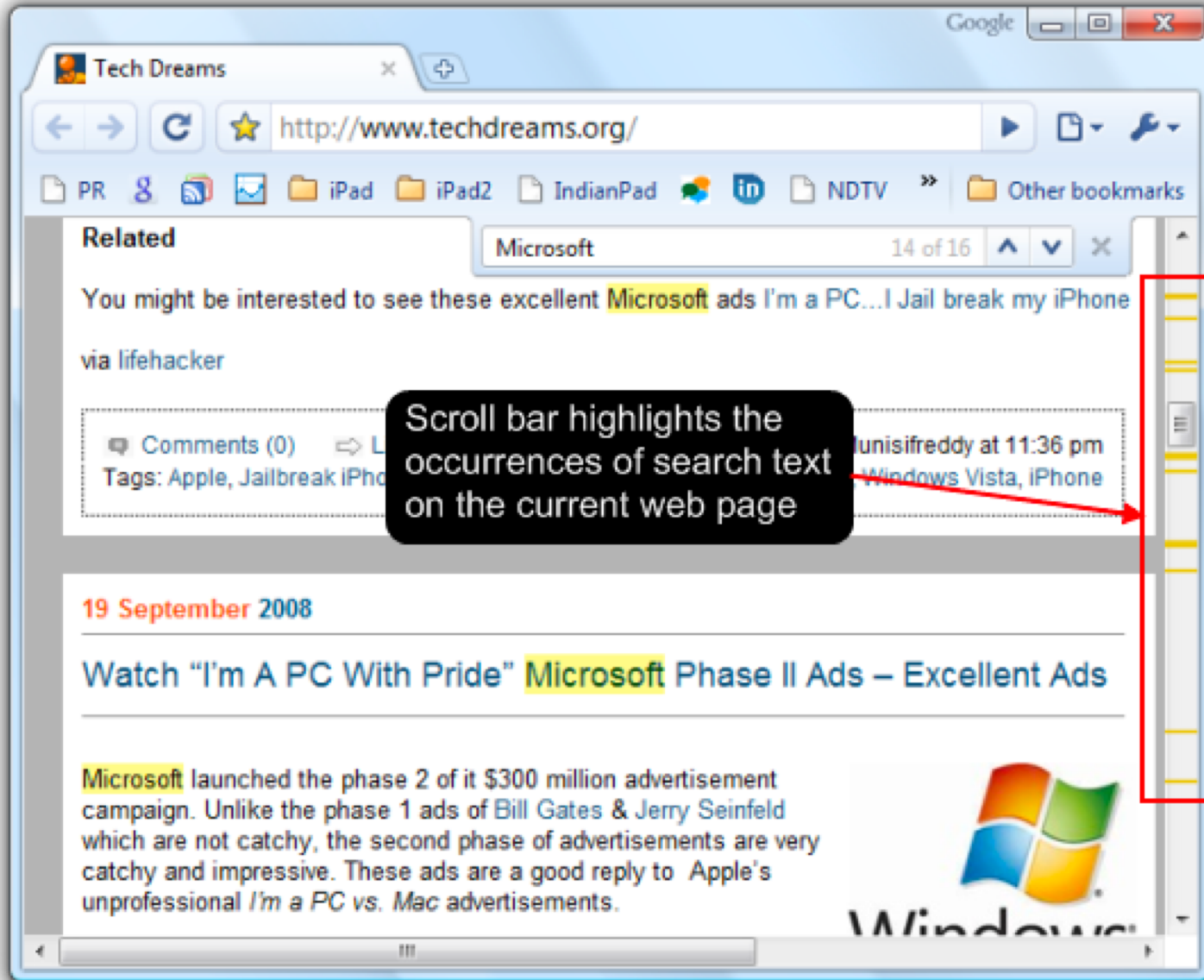
**Bad** use of customization.



**Good** use of customization.

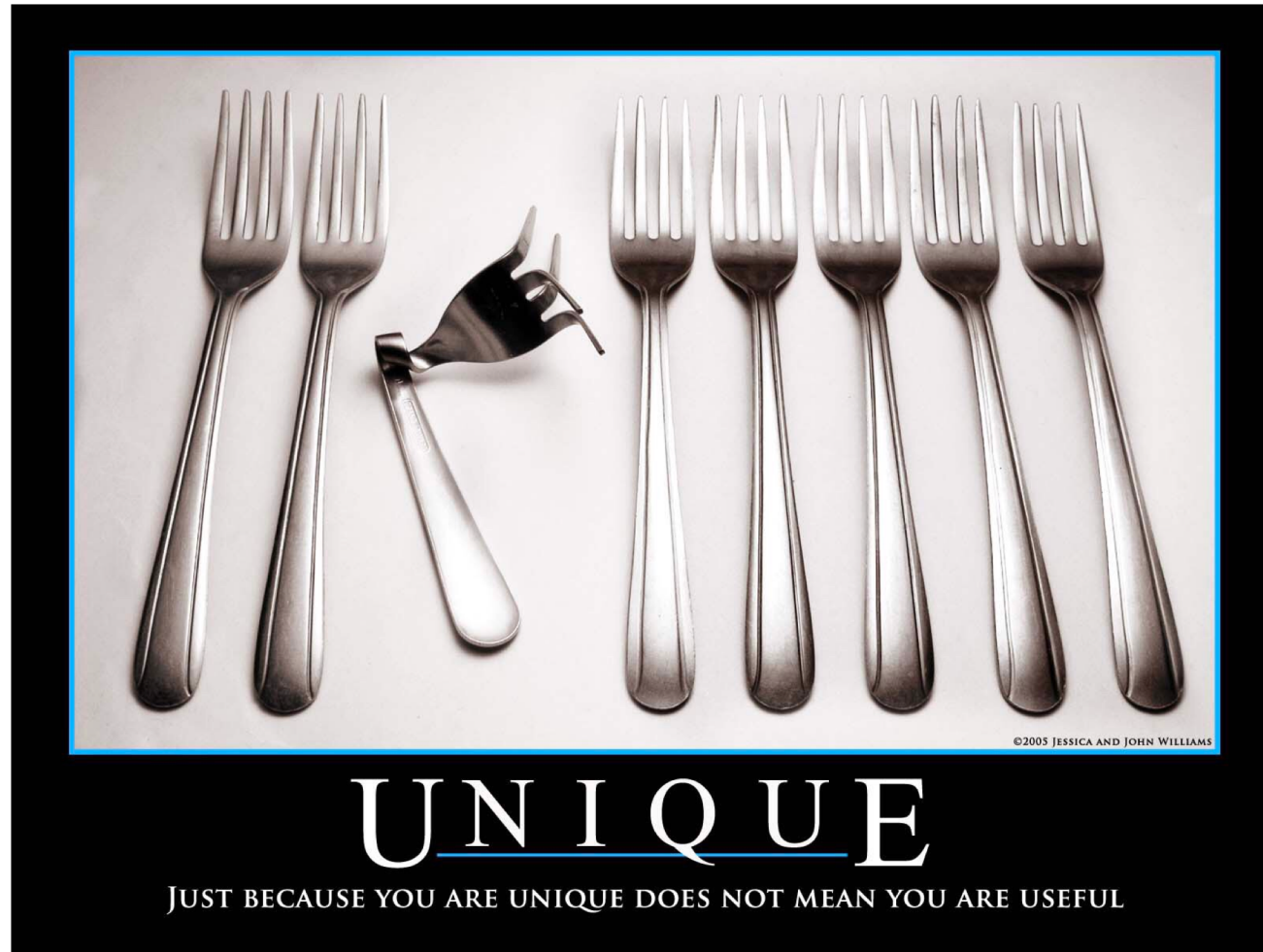


# Widgets allow customization



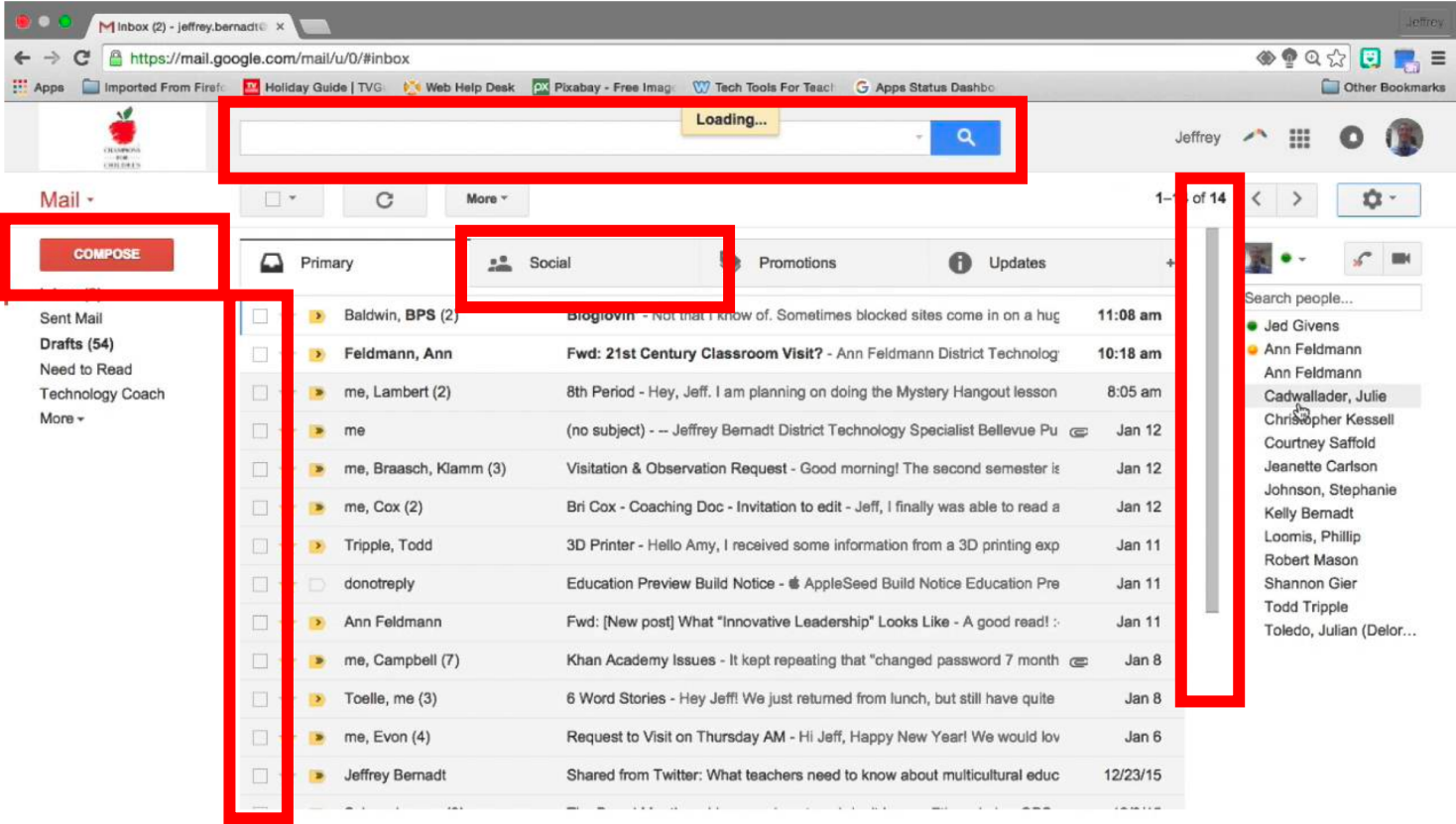


Use your powers of customization wisely.



Summary

# We interact with webpages through **widgets**: Elements with standardized appearance and events

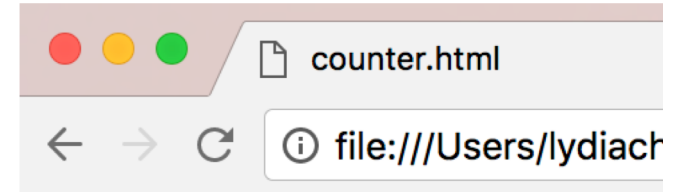




# Creating Interactions on the web has two parts:

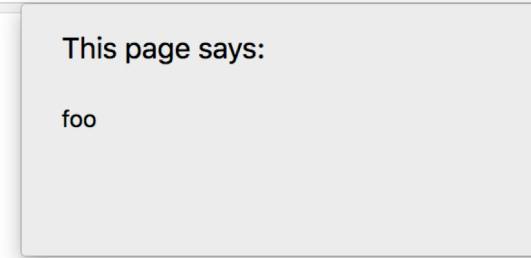
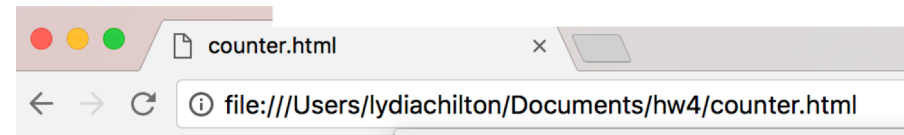
## 1. Program the interface and style in HTML & CSS

```
30  
31 <body>  
32  
33   <button id="counter" class="btn btn-primary">Counter (0)</button>  
34  
35 </body>  
36
```



## 2. Program interactions is JavaScript

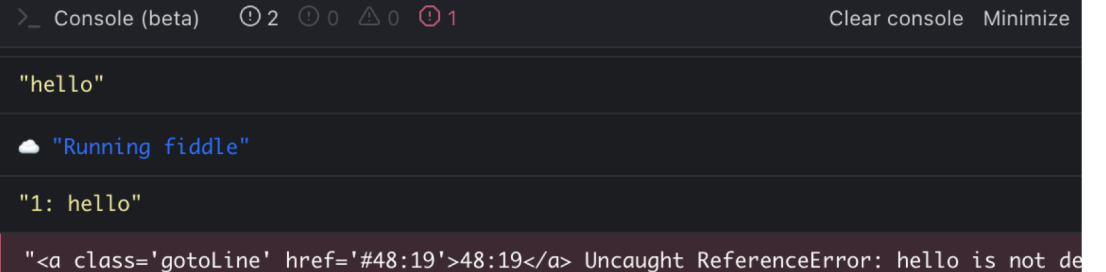
```
25  
26 $(document).ready(function(){  
27   $("#counter").click(function(){  
28     alert("foo")  
29   })  
30 })  
31
```



# In JavaScript, **let** and **const** are good ways to declare variables.

(var is globally scoped, and can get you into trouble)

```
1  const debug_mode = true
2
3  if(debug_mode){
4    let hello = "hello"
5    console.log("1: "+hello)
6  }
7
8  console.log("2: "+hello) // not declared
```



> Console (beta) 2 0 0 1 Clear console Minimize

"hello"

"Running fiddle"

"1: hello"

"<a class='gotoLine' href='#48:19'>48:19</a> Uncaught ReferenceError: hello is not de

Let is block scoped, and can be re-assigned.

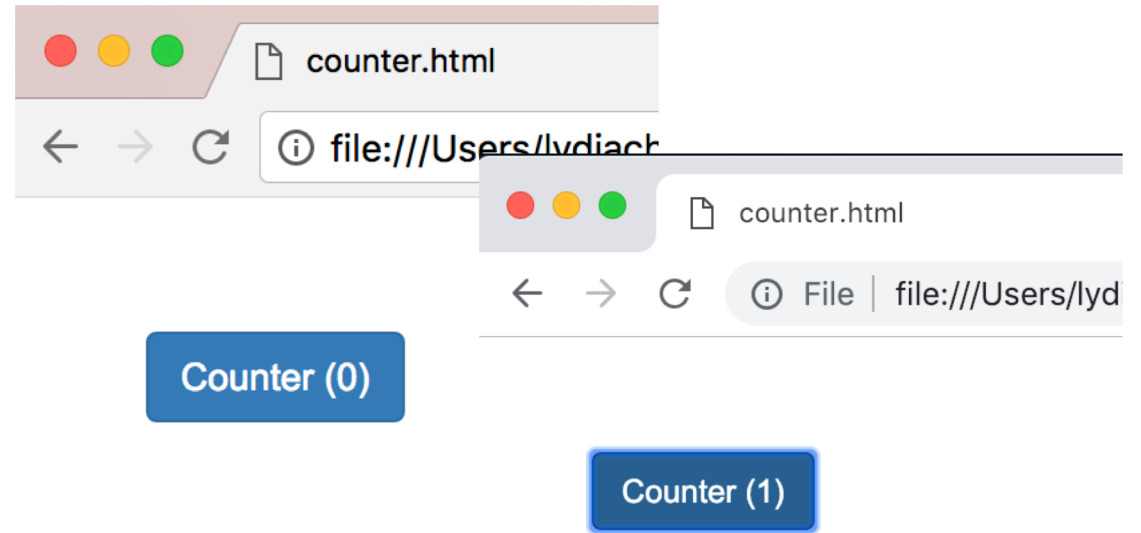
Const is block scoped and cannot be re-assigned.

# Good style of attaching events in JQuery

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" >
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" >
  </script>
  <script>
    var count = 0

    function incrementCount(c) {
      return c + 1;
    }

    $(document).ready(function(){
      $("#counter").click(function(){
        count = incrementCount(count)
        $("#counter").html("Counter (" + count + ")")
      })
    })
  </script>
</head>
<body>
  <button id="counter" class="btn btn-primary">Counter (0)</button>
</body>
</html>
```



1. Uses JQuery (not pure JavaScript)
2. Attaches click handler as in the `<script>`  
`$(element).click(...)`  
(doesn't attach in HTML)
2. Uses `$(document).ready(...)`

jQuery is a JavaScript Library that make JavaScript easier (and standard across browsers)

## JavaScript

```
document.getElementById("counter").addEventListener("click", function(){  
    document.getElementById("counter").innerHTML = "Counter (0)";  
});
```

## jQuery

```
$("#counter").click(function(){  
    $("#counter").html("Counter (0)");  
});
```

# You can create elements **statically** in HTML Or **dynamically** in JavaScript (jQuery)

## **Static:** HTML, JavaScript onReady

```
61 <body>
62   <button id="counter" class="btn btn-primary"></button>
63   <br><br>
64   <div id="updates"></div>
65 </body>
```

```
61 $(document).ready(function(){
62   $("#counter").click(function(){
63     // increment the counter
64     createButton()
65   })
66 })
```

## **Dynamic:** All JavaScript

```
44 function createButton(){
45
46   var new_button = $("<button class='btn btn-default'>")
47   $(new_button).text("dynamic button "+Date.now())
48   $("#updates").append(new_button)
49   $("#updates").append("<br>")
50
51   var d = Date.now()
52   $(new_button).click(function(){ alert(d) })
53 }
```

Static Button (2)

- dynamic button 1519060109685
- dynamic button 1519060110242

This page says:  
1519060110242  
OK

# Widgets are standardized low-level interaction interfaces that trigger events

When you create a widget...

```
61 <body>
62
63   <button id="counter" class="btn btn-primary"></button>
64
65 </body>
66
```

The **appearance** is standardized,



The **types of events** it responds to are standardized

```
50
51   $("#counter").click(function(){
52     [REDACTED]
53   })
54
```

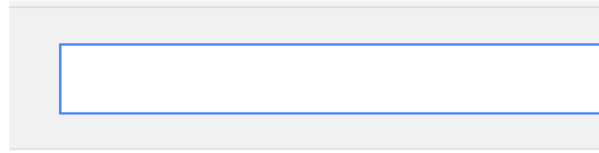
But the actions taken after an event is fired, are not standardized

```
50
51   $("#counter").click(function(){
52     count = count + 1
53     setCount(count)
54   })
```

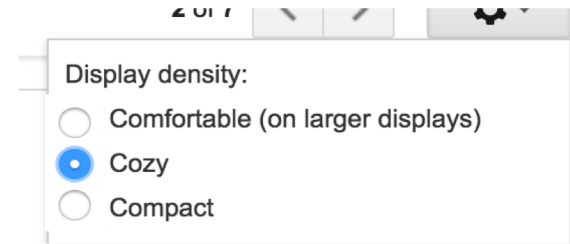
# There are many types of widgets and events



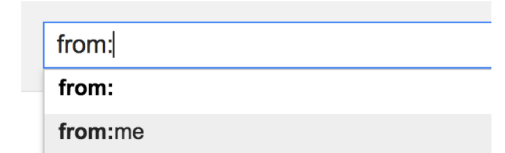
Click



Keypress



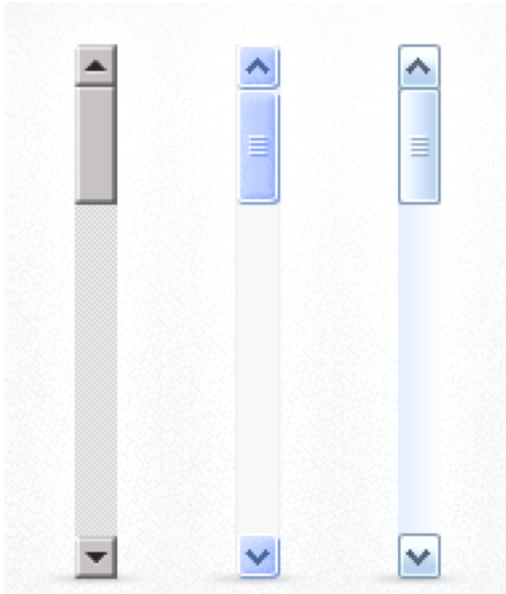
Change



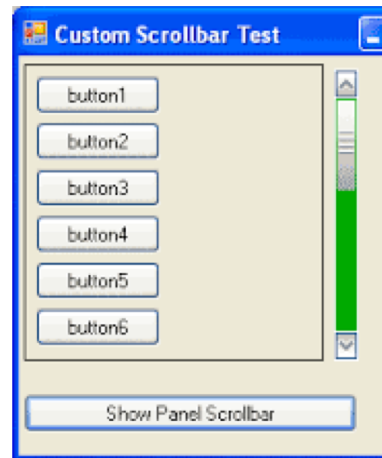
Select

# Widgets allow customization. Use it wisely.

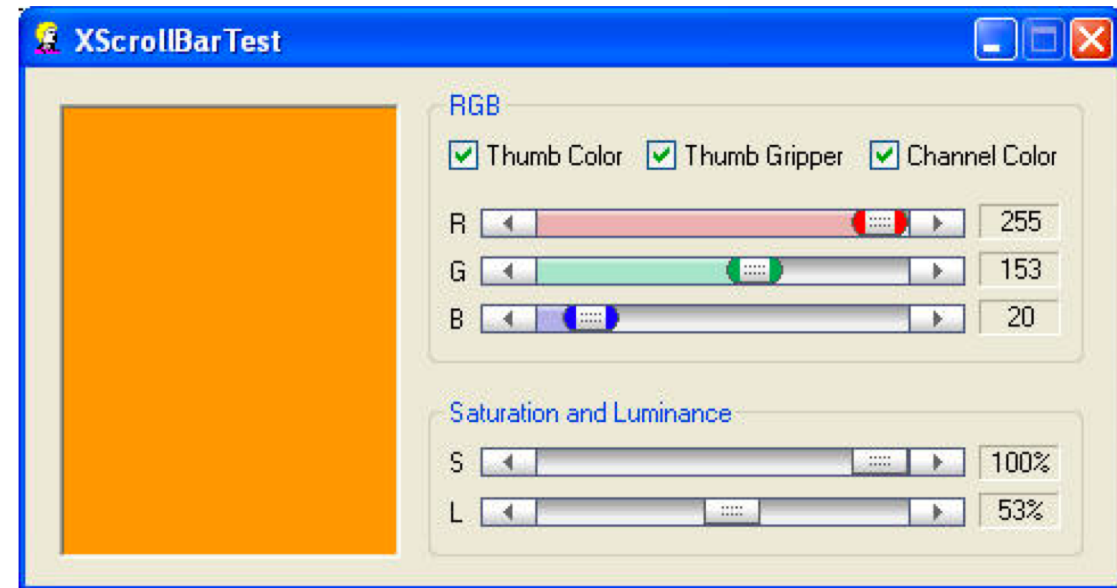
Customizable scroll bars



**Bad** use of customization.



**Good** use of customization.



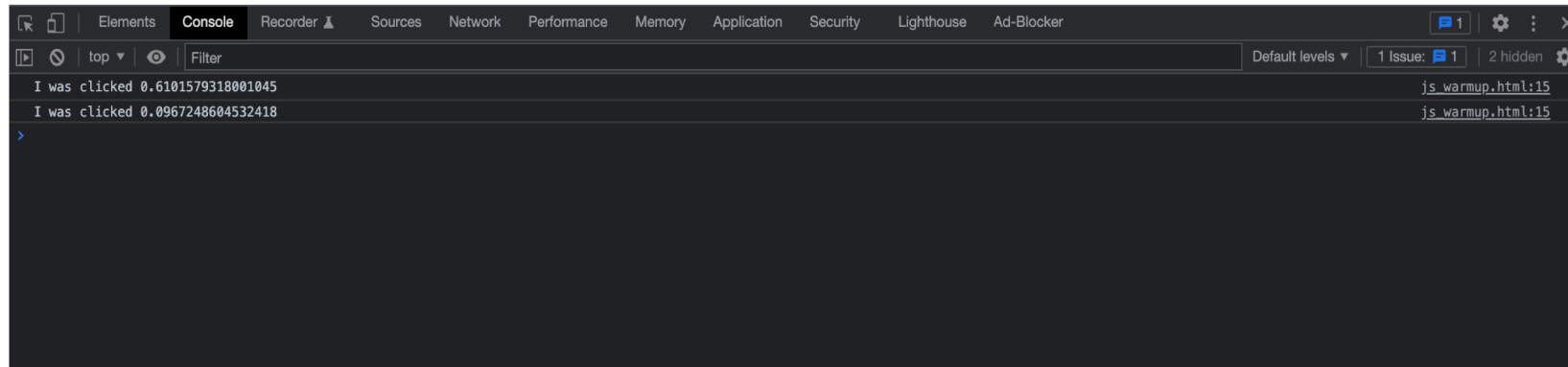
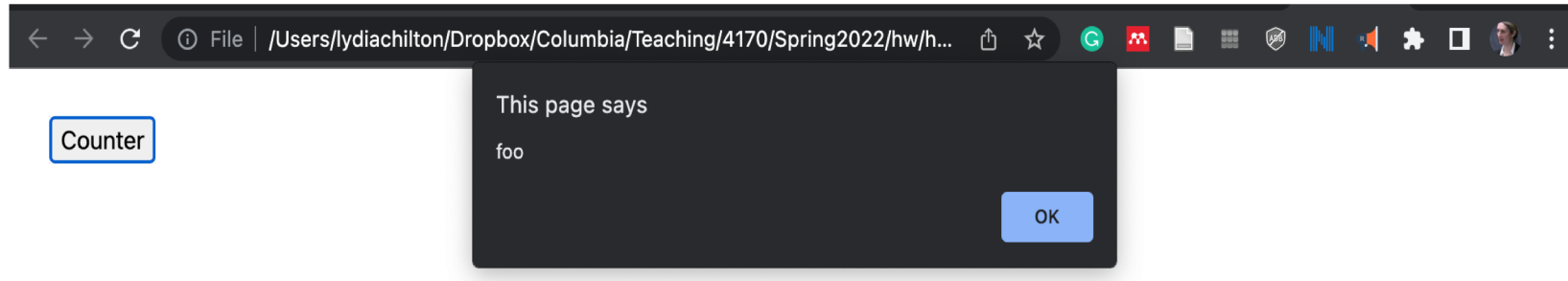


# Homework 3: User Models and JavaScript

Warm up: due Friday 2/4 @ 11:59pm on Courseworks

Main: due Tuesday 2/8 @ 11:59pm on Courseworks.

## Warm-up:



# Homework 3: User Models and JavaScript

Warm up: due Friday 2/2 @ 11:59pm on Courseworks

Main: due Tuesday 2/6 @ 11:59pm on Courseworks.

## Warm up

The screenshot shows a web form titled "Section Preferences". The form contains a question: "After spring break, can you attend class during class time in person on Wednesdays?" with a red asterisk indicating it is required. Below the question is a sub-question: "If you can only attend a part of class (like 1:10-1:45) that is good enough." There are four radio button options: "Yes, I can attend class Wednesdays durir", "I am free during class time Wednesdays,", "I cannot make it during Wednesdays clas", and "I am a CVN student (your feedback sessi". A blue "Counter" button is visible. A dark notification box says "This page says foo" with an "OK" button. The browser's developer console shows two log entries: "I was clicked 0.6181579318001045" and "I was clicked 0.0967248604532418".

## Main

Write a tweet

Call me Ishmael. Some years ago- never

-12 Post Tweet

POSTS

chilton Third post

chilton Second post

chilton First post