

Please fill in the rows middle-out.



Make a friend - one on each side of you 😊

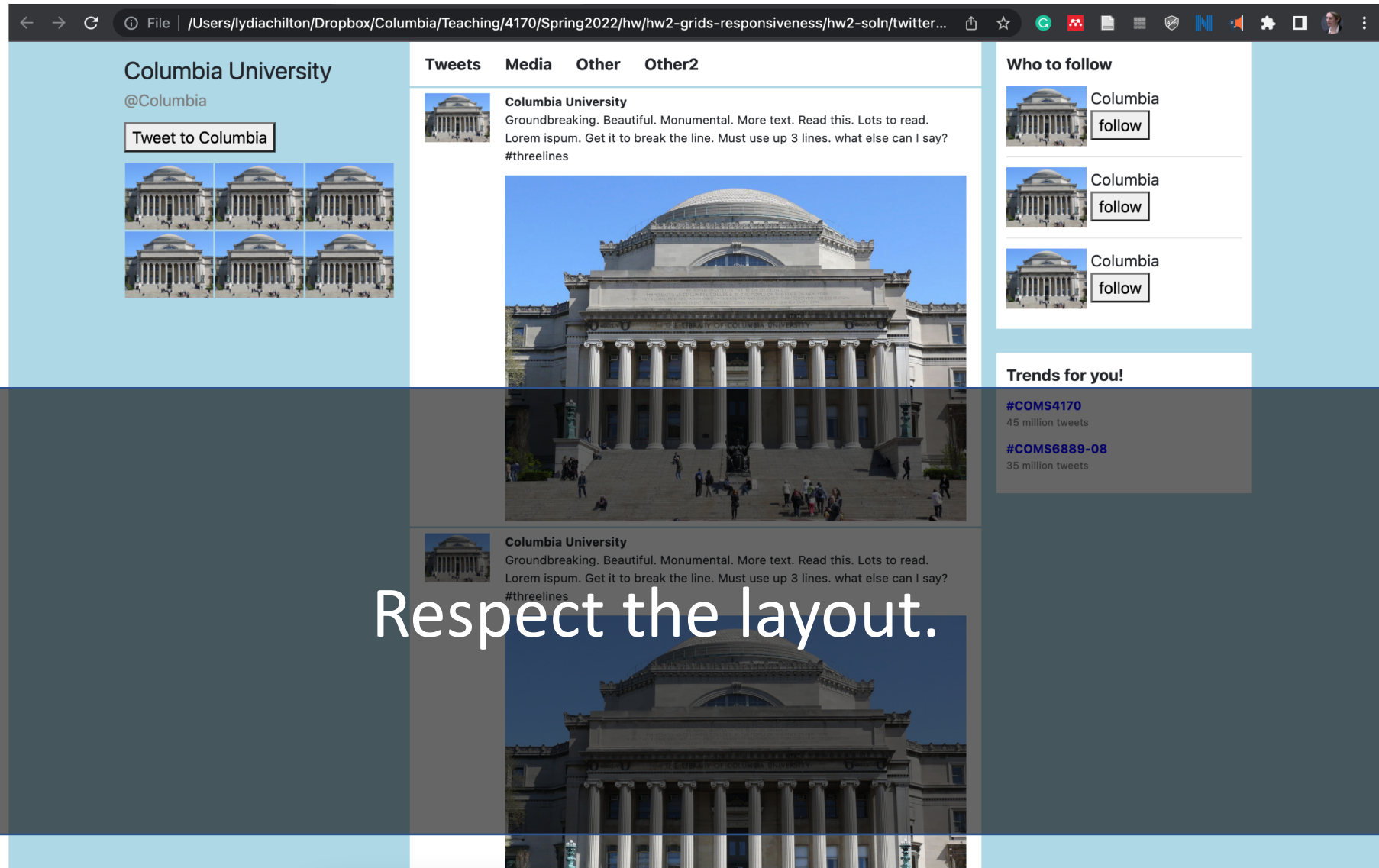
JavaScript, Widgets, & Events

Prof. Lydia Chilton
COMS 4170
2 February 2022

Raise your hand or type in zoom



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 divided into three columns: 'Tweets', 'Media', and 'Other'. The 'Tweets' column 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'. The 'Media' column is empty. The 'Other' column contains 'Who to follow' and 'Trends for you!' sections. The 'Who to follow' section lists three 'Columbia' accounts with 'follow' buttons. The 'Trends for you!' section lists two hashtags: '#COMS4170' (45 million tweets) and '#COMS6889-08' (35 million tweets). A large grey rectangular overlay covers 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

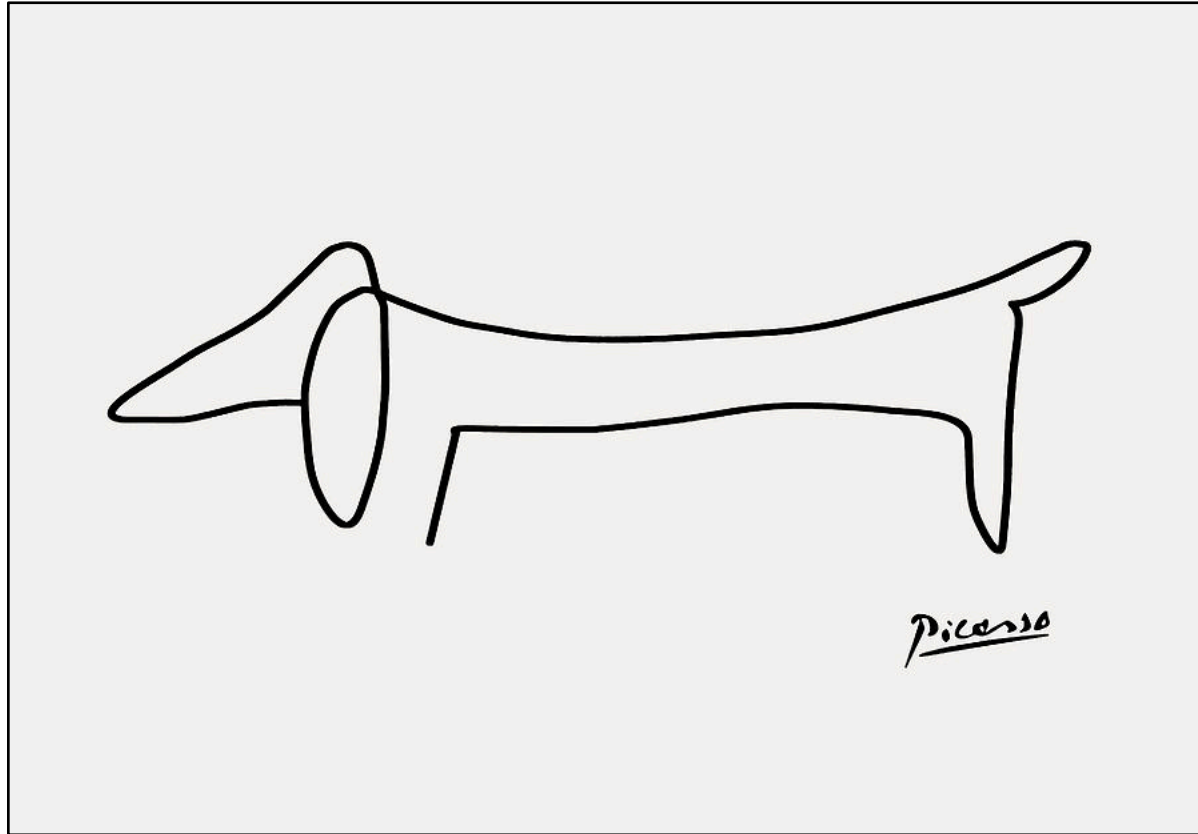
Trends for you!

#COMS4170
45 million tweets

#COMS6889-08
35 million tweets

Respect the layout.

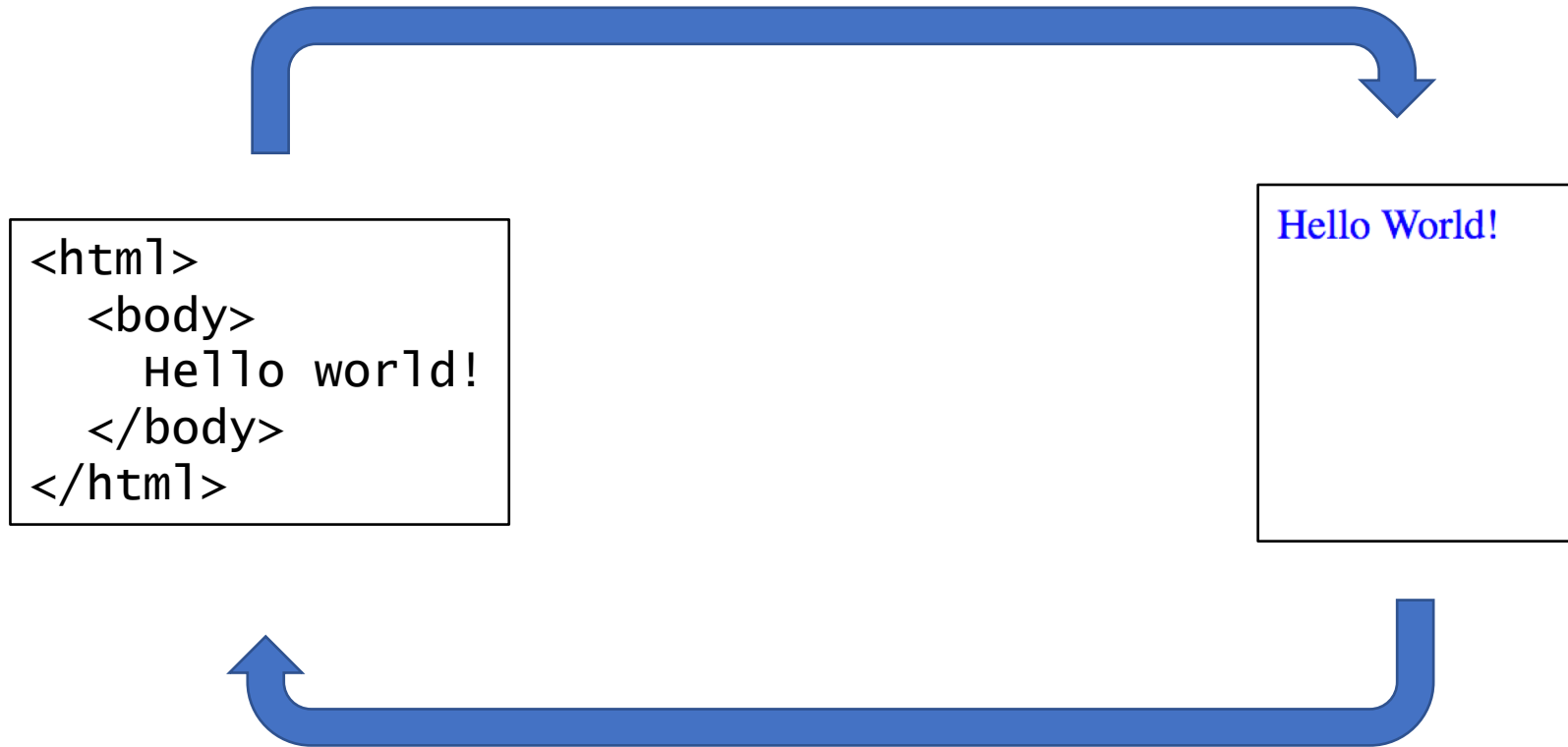
Simple is hard.



Iterative Style of Programming

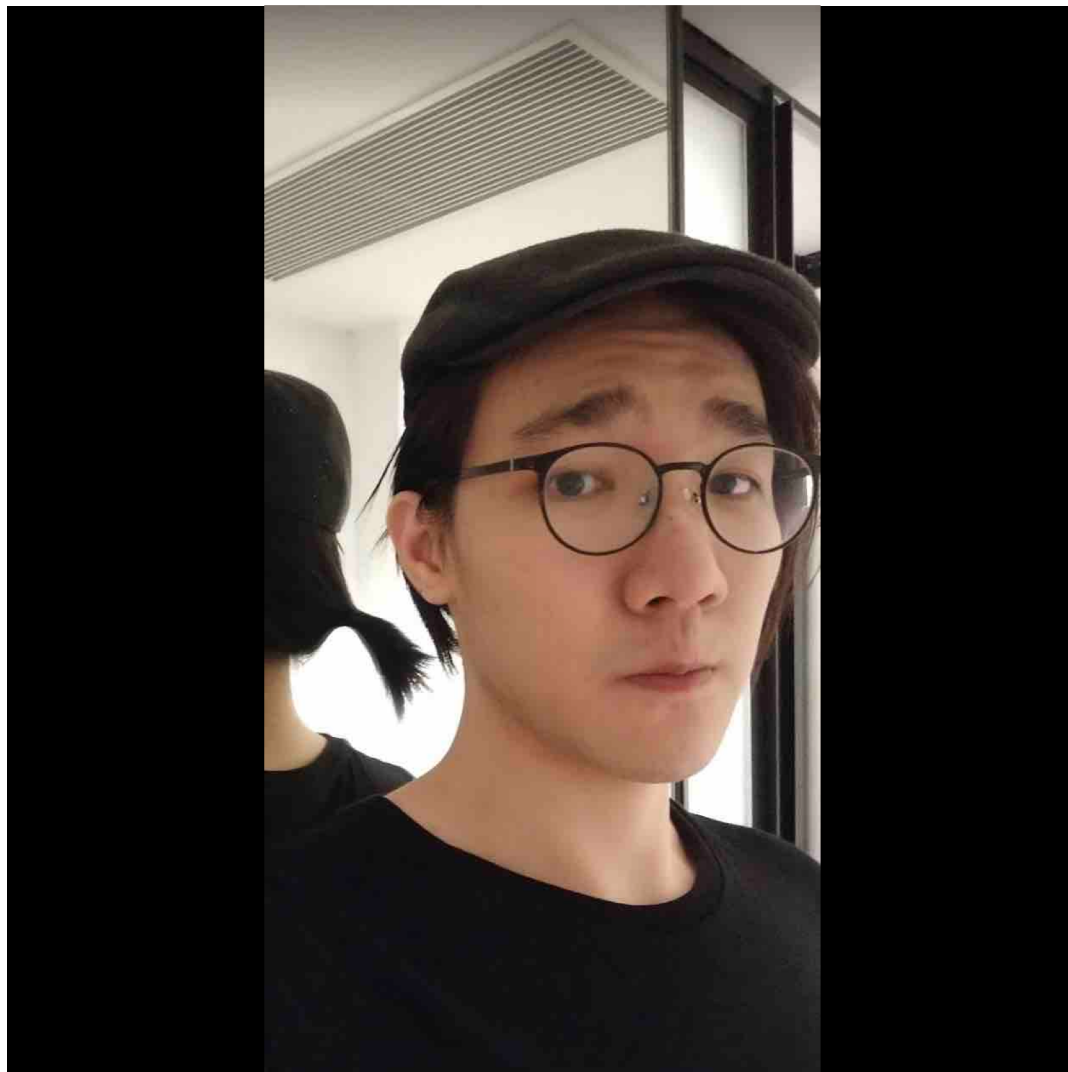
helps you build a mental model of your code.

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



Does it look ok?

Andreas



- GS Senior in CS
- Took Prof. Smith's UI Design last fall
- Interested in competitive programming other than web development
- Quite into *JoJo's Bizarre Adventure* recently
- Nice to meet you all!

Hong



- SEAS Junior studying CS + Econ
- Took UI Design Spring 2021 but enjoyed it so much I came back a year later to TA!
- From Bronx, New York 🏠
- Exploring the intersection of CS, journalism, social good, and politics
- Recent interest in exploring use cases in virtual reality!
- Love hiking, running, and anything naturey
- Current show I'm binging: *Kingdom*
- Super excited to meet you all :D

Sofia



- Senior in GS studying CS on the applications track
- Interested in UI, designing for accessibility, and the web3/crypto space.
- Took UI fall 2021 with Professor Smith
- In my spare time I organize Zoom classes through an organization I founded called weareausome, whose mission is to to mitigate the taboos of autism spectrum disorder :) We offer classes taught by skilled people on the autism spectrum open to both neurodivergent and neurotypical young adults.
- I worked as a UX designer at a sports-metaverse tech startup
- I love to draw and I love art museums. My inspirations are Hilma af Klint, Georgia O'Keefe, and Escher
- Why GS? I was a ballerina in another life!

Wesley



- MSCS student in the Software Systems track
- Took Advanced Web Design Studio with Lydia in Fall 2021
- Full Stack Software Engineer for 2 years before coming back to complete a masters
- Might find me at the gym, playing table tennis, or cooking in my spare time
- Currently following along with airings for season 2 of Demon Slayer
- Looking forward to meeting you all!

Jeff



- MSCS student
- Took Prof Smith's UI Design class last semester
- Interested in mobile app development and ubiquitous computing.
- Currently watching Formula 1 and Attack on Titan!
- Grew up in Taiwan!
- Very excited to meet you all :)

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

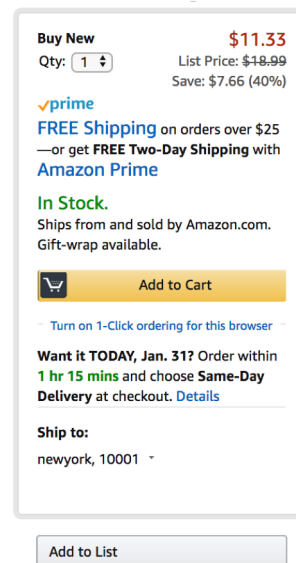
Interaction:

Type, click

click

Type, click, point

Click

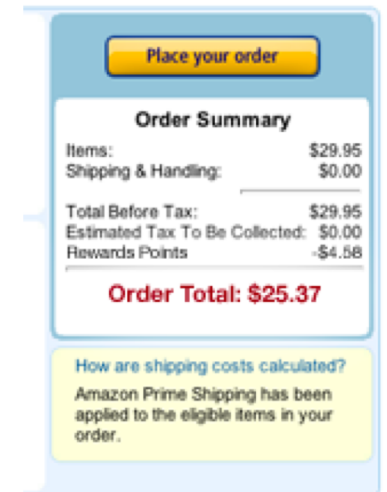


Name (as it appears on your card)

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.

Secure Payment Info

MasterCard VISA AMEX DISCOVER PayPal

Name (as it appears on your card)

Card number (no dashes or spaces)

Expiration date

01 - January 2013

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

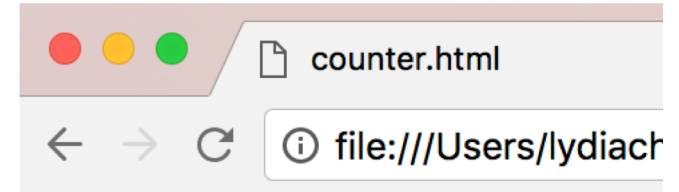
123 1234

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

Creating Interactions on the web has two parts:

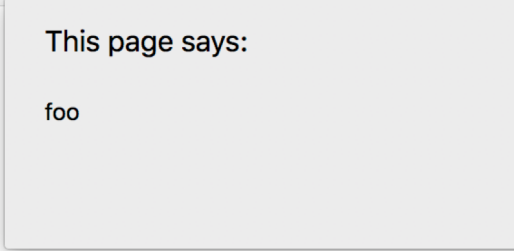
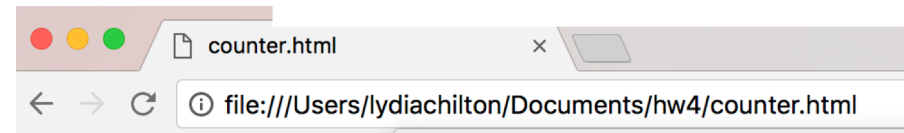
1. Program the interface and style in HTML & CSS

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



2. Program interactions is JavaScript

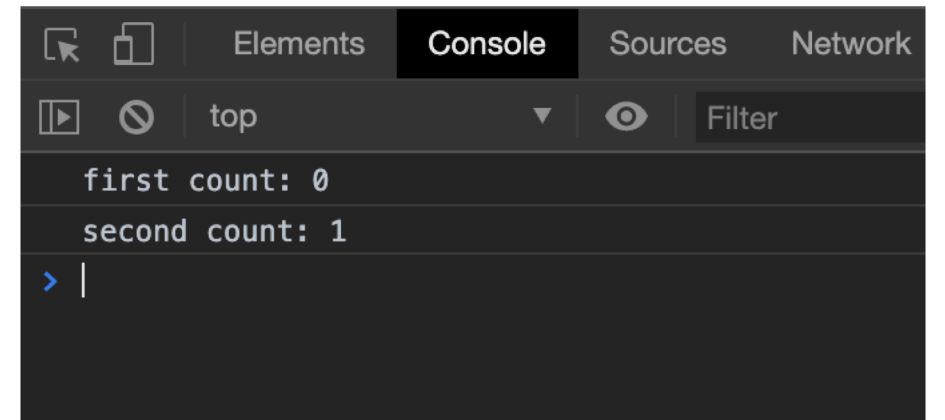
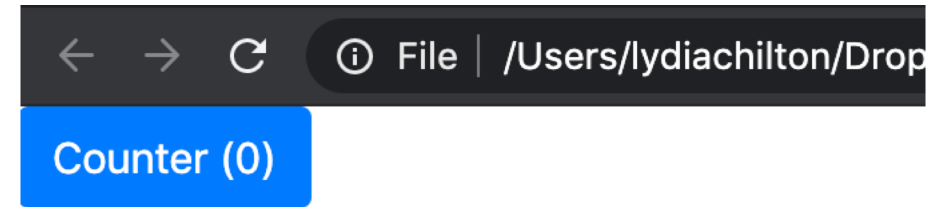
```
25  
26 $(document).ready(function(){  
27   $("#counter").click(function(){  
28     alert("foo")  
29   })  
30 })  
31
```



Web Page Execution

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

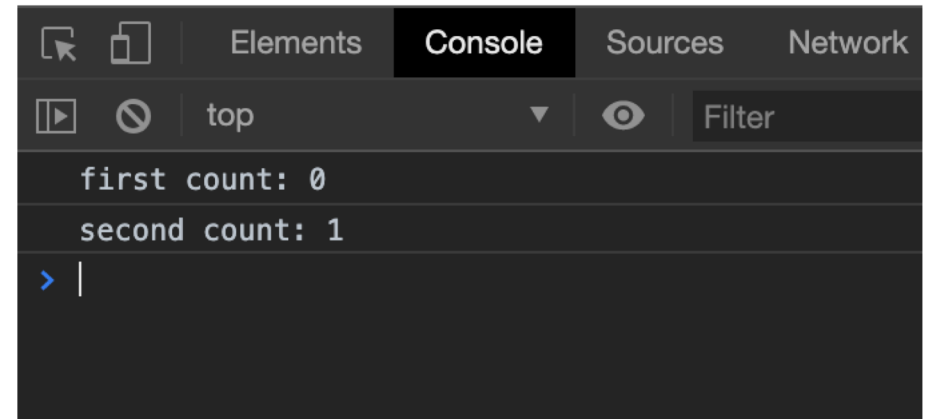
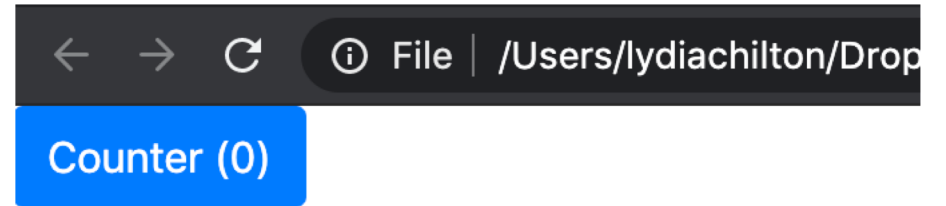
```
1 <html>
2 <head>
3   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/boot
4     <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossor
5
6   <script>
7
8     var count = 0
9
10    function incrementCount(c) {
11      return c + 1;
12    }
13
14    console.log("first count: "+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: var, let, const

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"

"47:19 Uncaught ReferenceError: hello is not def

>_

Const is good for constant variables

```
JavaScript + No-Library (pure JS) ▼  
1  
2  const debug_mode = true  
3  
4  ▼ if(debug_mode){  
5    var hello = "hello"  
6    console.log("1: "+hello)  
7  }  
8  
9
```

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

☁ "Running fiddle"

"1: hello"

>_

Const is block scoped, and cannot be re-assigned.

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"

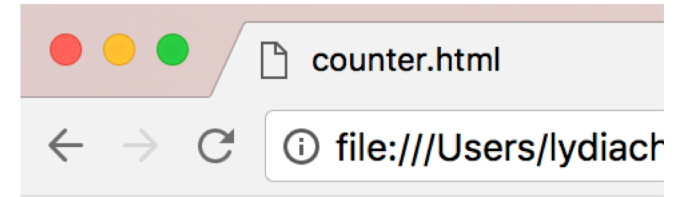
>_

Var are globally scoped (or function scoped)
Var variables can be updated and re-declared

Adding events

When you click this button, what will it do?

```
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
  <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>
```



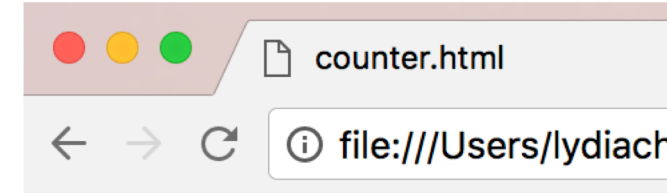
Syntax is similar to but different from...

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/boot
  <script src="https://code.jquery.com/jquery-3.3.1.min.js" crossorig

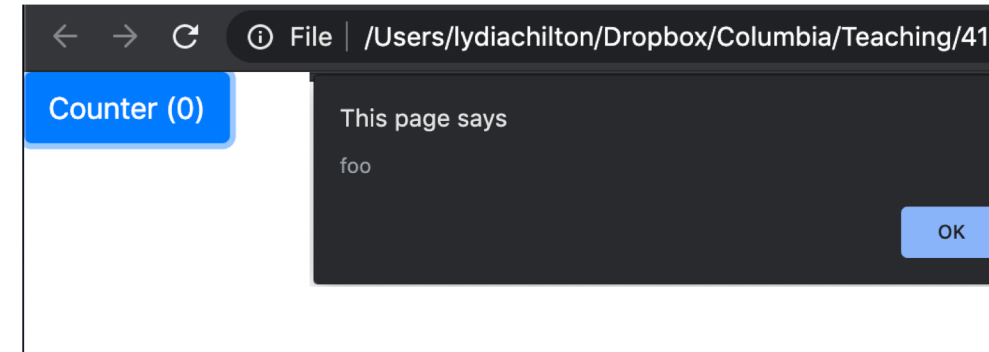
  <script>

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

</head>

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

</html>
```



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>

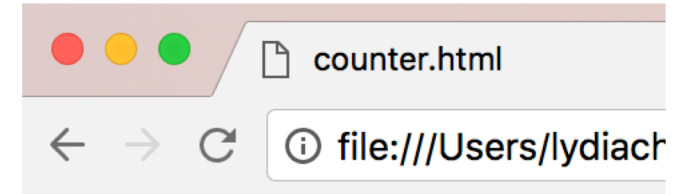
    $(document).ready(function(){
      $("#counter").click(function(){
        alert("foo")
      })
    })

  </script>

</head>

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

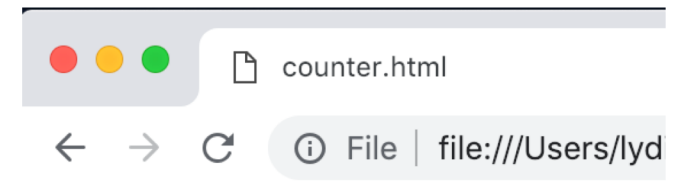
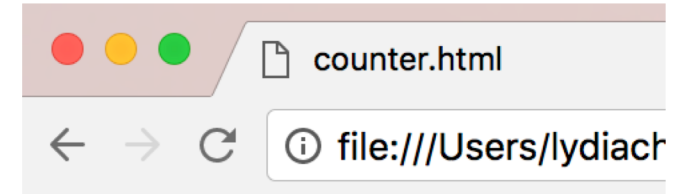
</html>
```



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
```



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>
```

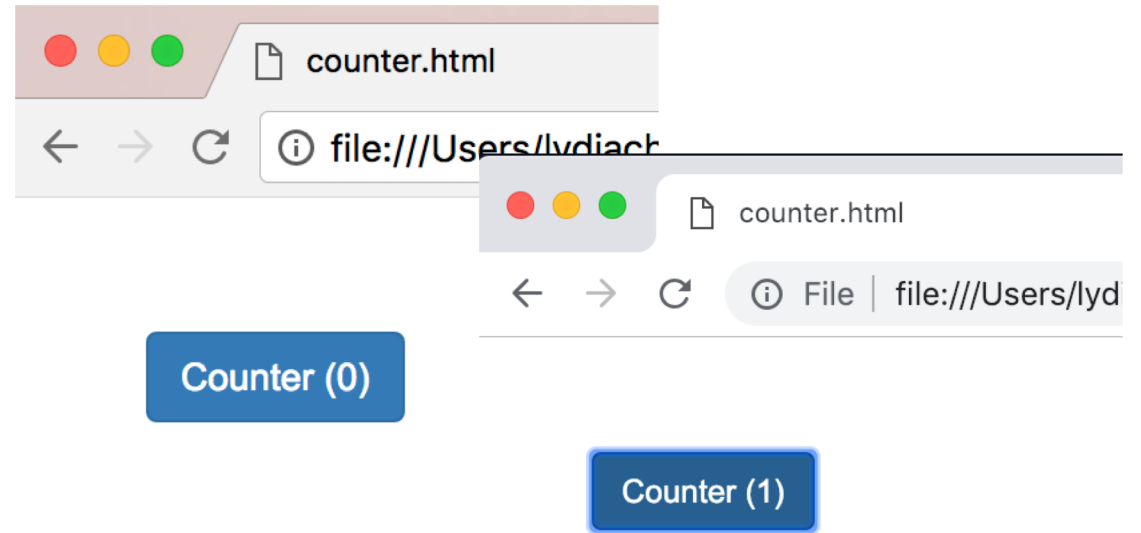
```
<button onclick="incrementCount(1)">Counter (1)</button>
```

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(...)`

Creating Widgets Dynamically

Statically created widget: created on page load.

HTML

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



Static Button (0)

JavaScript

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

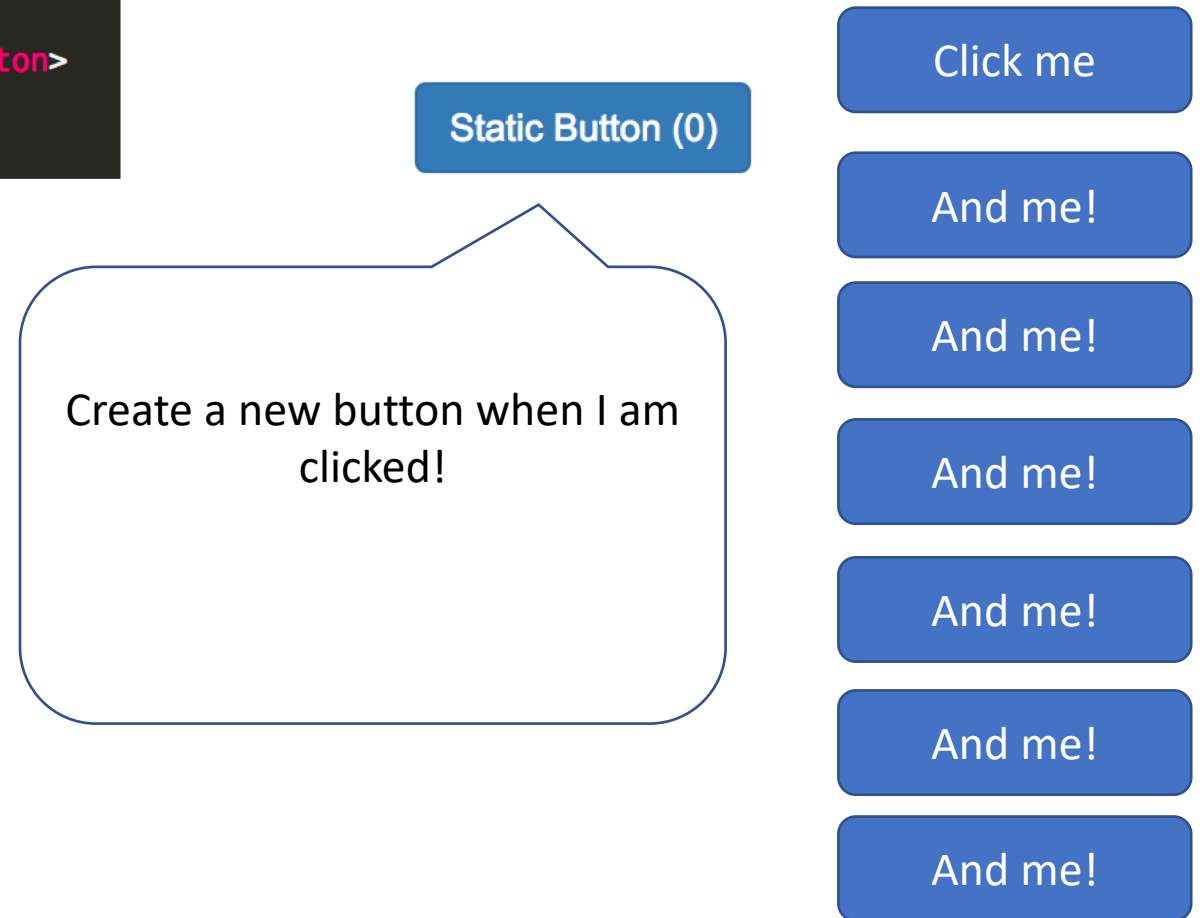
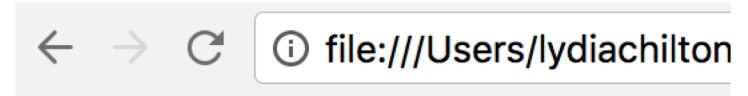
Dynamically created widget: created on demand based on user interaction.

HTML

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

JavaScript

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



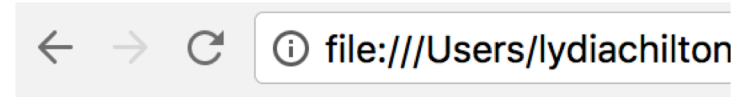
Where in the code should we add the dynamic behavior?

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!

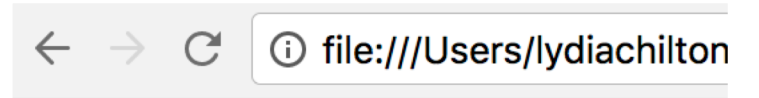
And me!

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

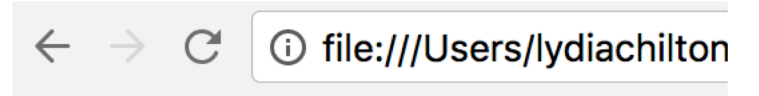
```
42 function createButton(){
43   var new_button = $("<button>")
44   $(new_button).text("dynamic button "+Date.now())
45 }
46
```

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


First, add a div to HTML where we can append stuff.

HTML

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



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 })
```

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 })
```

Where 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 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>
66
```

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
```

Where 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 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 = $("
```

```
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

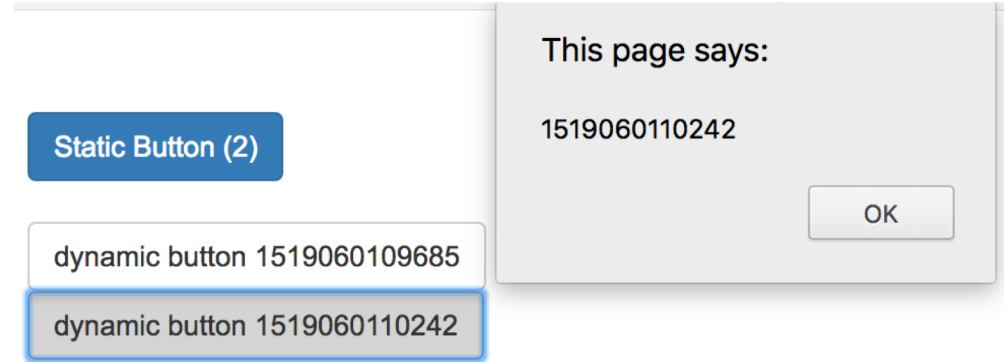
```
44 function createButton(){
45     var new_button = $("<button class='btn btn-default'>")
46     $(new_button).text("dynamic button "+Date.now())
47     $("#updates").append(new_button)
48     $("#updates").append("<br>")
49 }
```

```
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 }
```

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


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>
66
```

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

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

You don't need to know this, but

Problem: Var is global when you don't want it to be.

The screenshot displays a web browser's developer console with three main sections: HTML, CSS, and JavaScript + No-Library (pure JS). The HTML section shows three button elements with values 0, 1, and 2. The CSS section is empty. The JavaScript section contains a script that uses `var` to declare a `buttons` array and attach click event listeners to each button. The event listener function logs the value of `i` to the console. Below the code, a visual representation of the `buttons` array is shown as a vertical list of three boxes containing the values 0, 1, and 2. The console output shows three log entries, each displaying the string `"My value: 3"`, which is a result of the global scope issue where the `var` declaration is hoisted and `i` is always 3 by the time the event listeners are executed.

```
HTML ▼  
1 <button>0</button>  
2 <br />  
3 <button>1</button>  
4 <br />  
5 <button>2</button>  
  
CSS ▼  
1  
  
JavaScript + No-Library (pure JS) ▼  
1 var buttons = document.getElementsByTagName("button");  
2 // let's create 3 functions  
3 for (var i = 0; i < buttons.length; i++) {  
4 // as event listeners  
5 buttons[i].addEventListener("click", function() {  
6 // each should log its value.  
7 console.log("My value: " + i);  
8 });  
9 }  
  
0  
1  
2  
  
>_ Console (beta) ⓘ 12 ⓘ 0 ⚠ 0 ⓘ 0 Clear console Minimize  
"My value: 3"  
"My value: 3"  
"My value: 3"  
>_
```

Solution: Use let

The image shows a web development IDE with four main panels:

- HTML:** Contains three button elements with values 0, 1, and 2, separated by line breaks.

```
1 <button>0</button>
2 <br />
3 <button>1</button>
4 <br />
5 <button>2</button>
```
- CSS:** Currently empty.
- JavaScript + No-Library (pure JS):** Contains a script that uses `let` in a `for` loop to attach click event listeners to the buttons. Each listener logs the button's value to the console.

```
1 var buttons = document.getElementsByTagName("button");
2 // let's create 3 functions
3 for (let i = 0; i < buttons.length; i++) {
4   // as event listeners
5   buttons[i].addEventListener("click", function() {
6     // each should log its value.
7     console.log("My value: " + i);
8   });
9 }
```
- Preview/Console:** Shows three buttons with values 0, 1, and 2. Below them is a console window with the following log output:

```
>_ Console (beta) 15 0 0 0 0
"My value: 0"
"My value: 1"
"My value: 2"
>_
```

Widgets and Events

Basic elements for users to interact with your UI

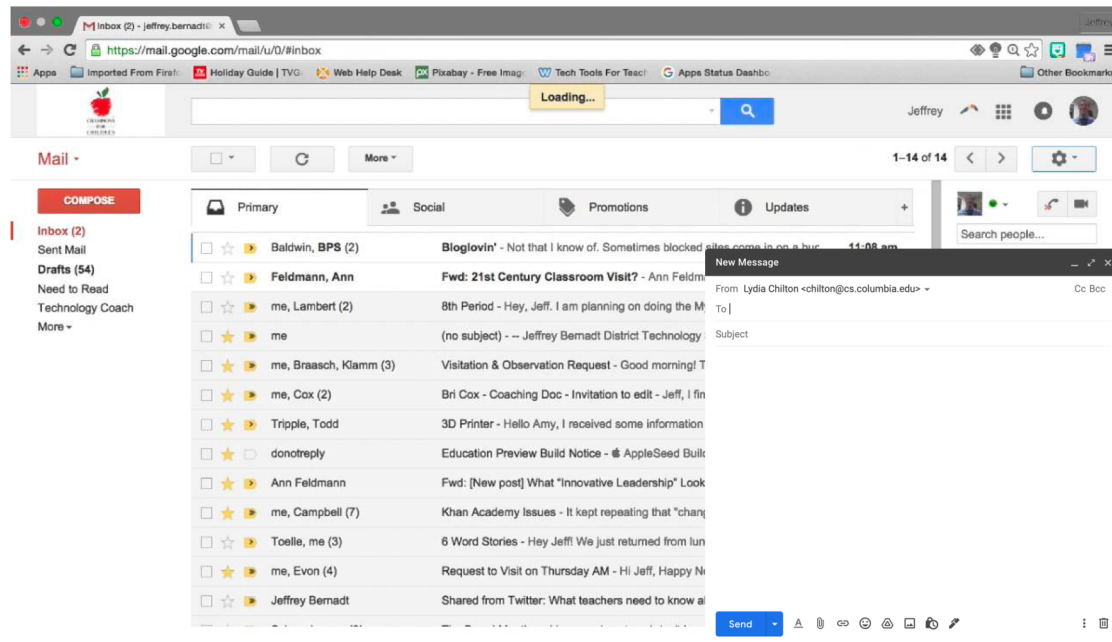
Buttons are one type of **widget**
the main event they can respond to is **clicks**.



COMPOSE

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

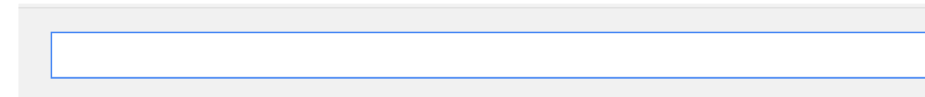
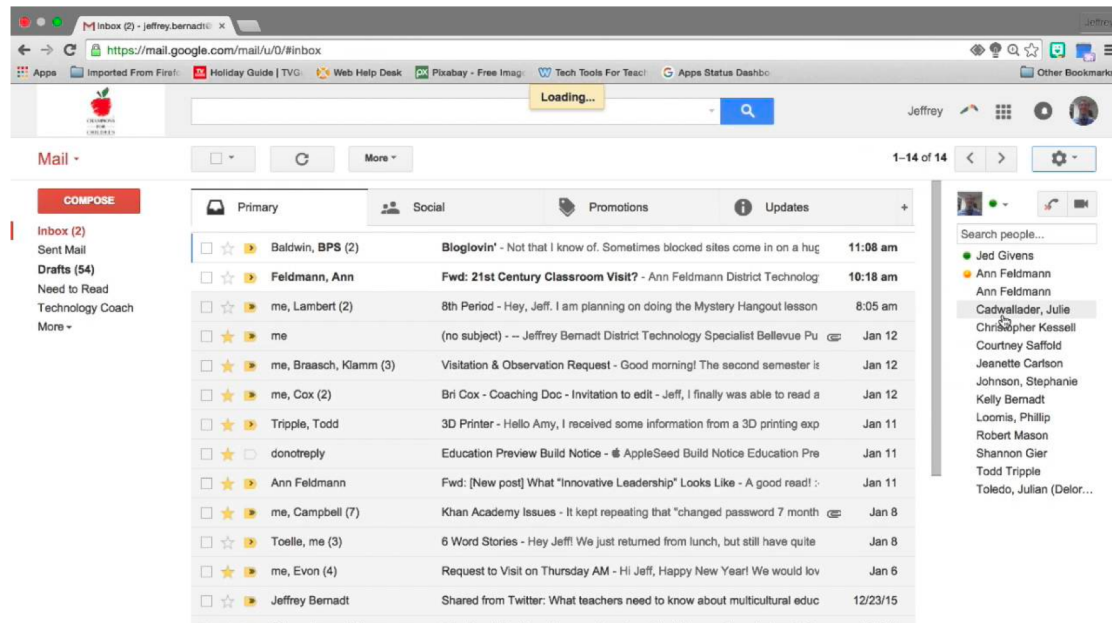
Every time a button is clicked, a click event fires.



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

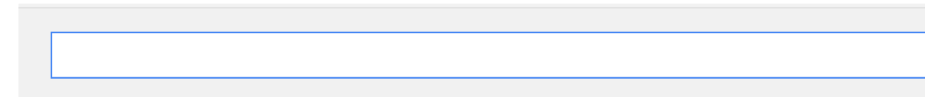
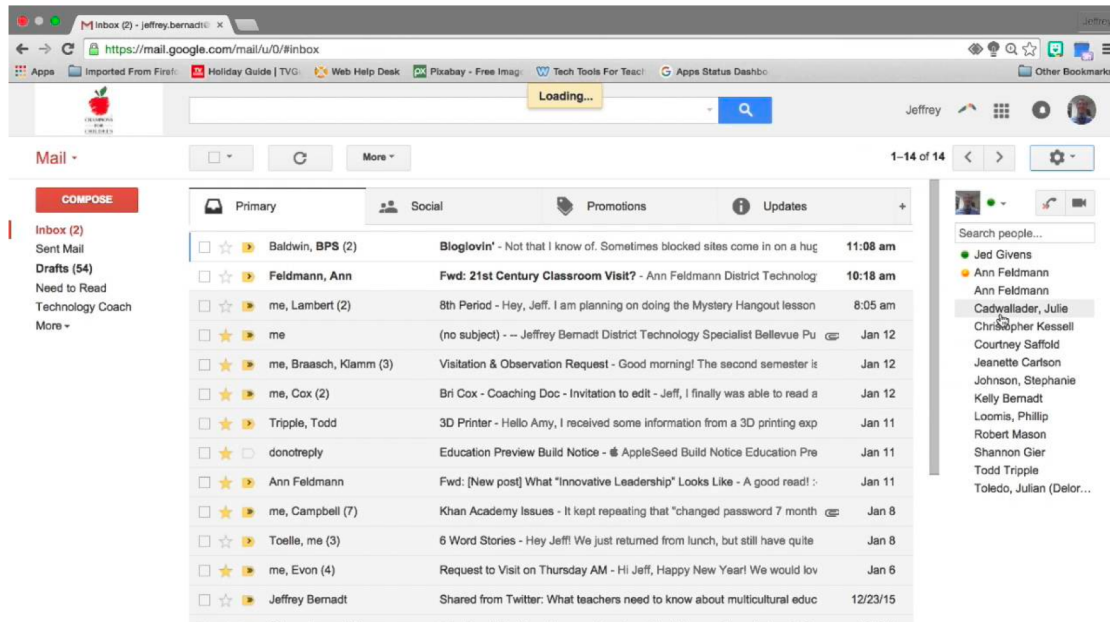
This code listens for the click event fire and does something is called the “click handler” (more generally: “event handler”)

Text Input interaction: What event fires?



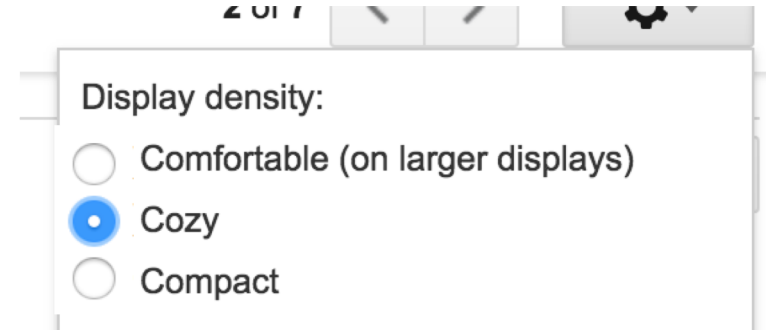
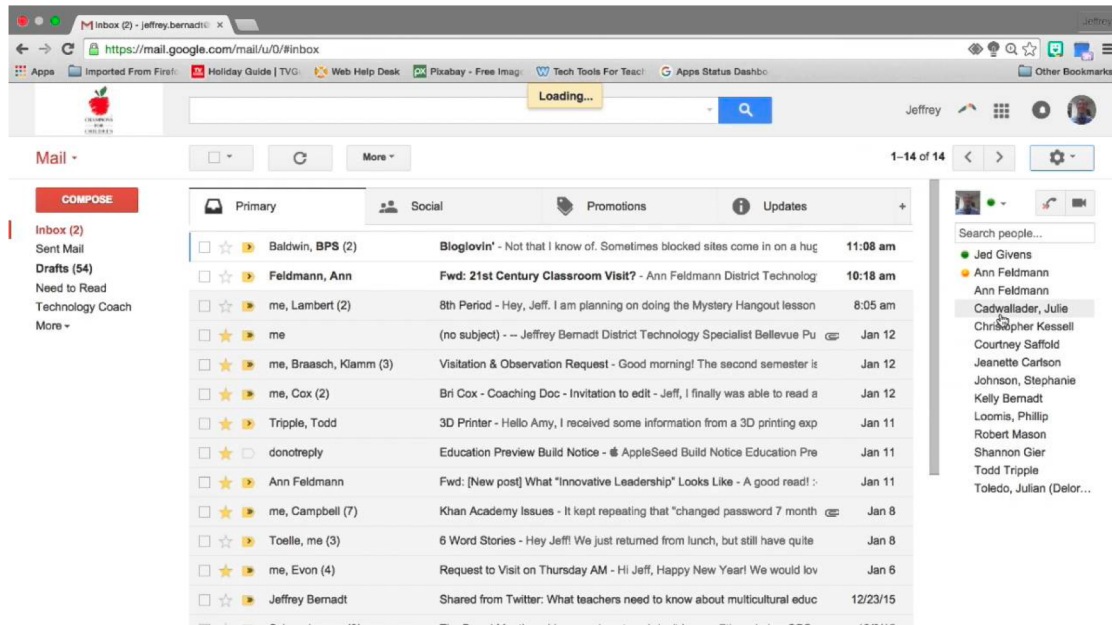
“Keypress” event

Button interaction: What event fires?



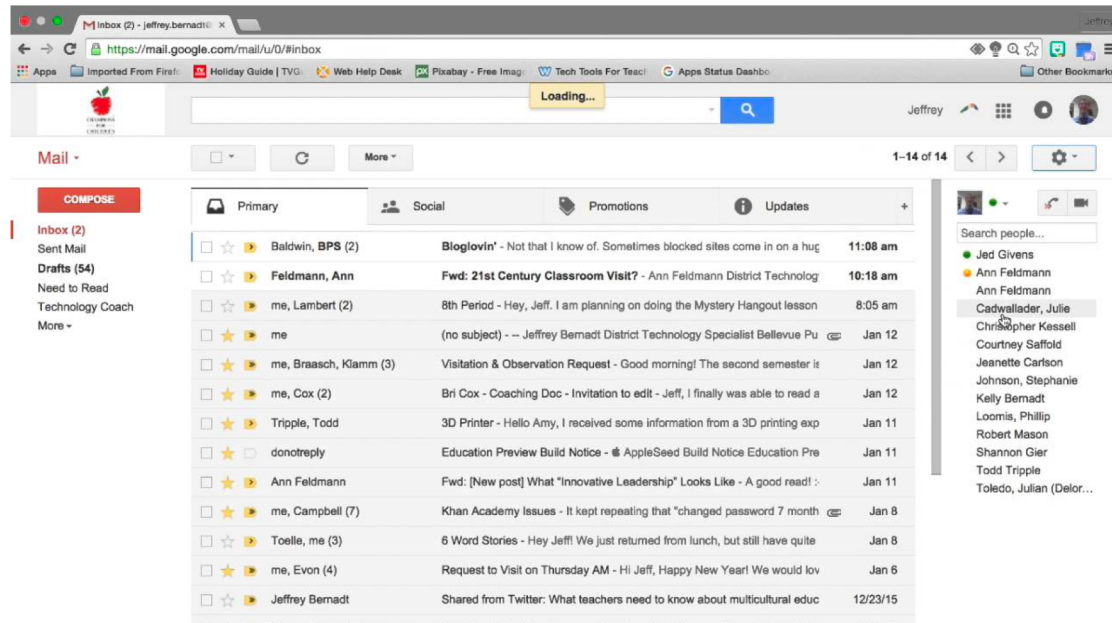
“Click” event

Radio Input interaction: What event fires?



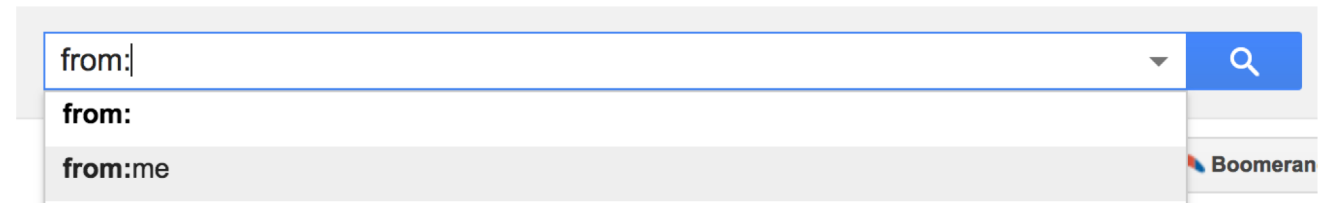
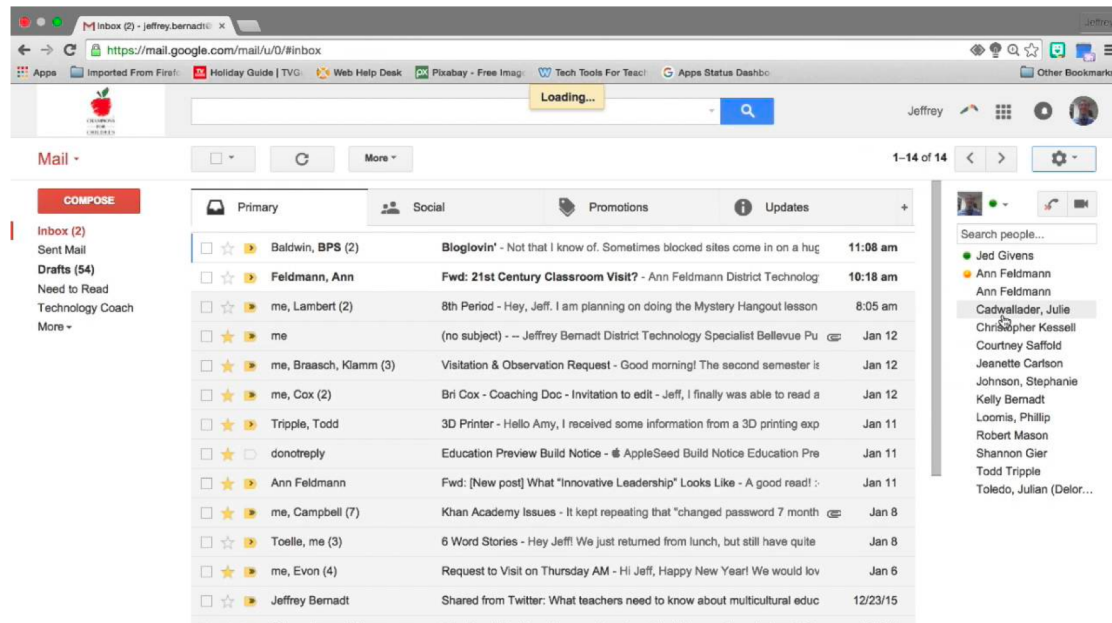
“Change” event

Select Element interaction: What event fires?



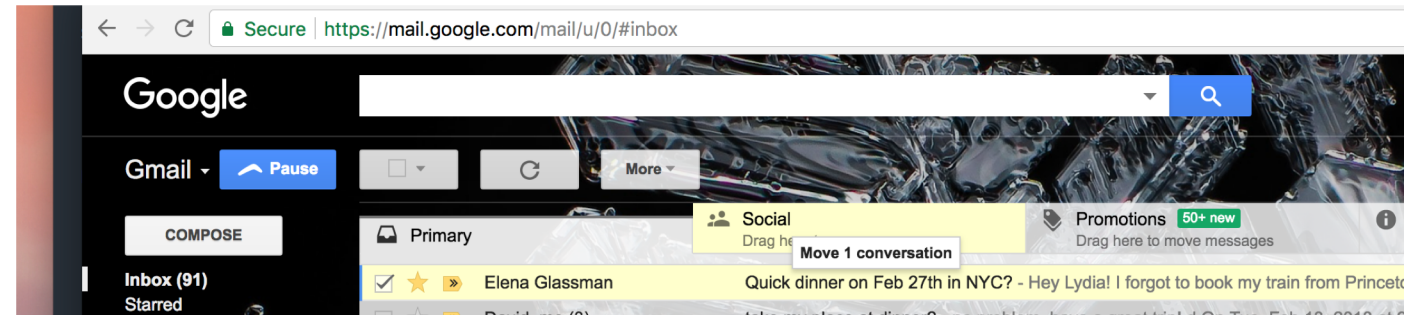
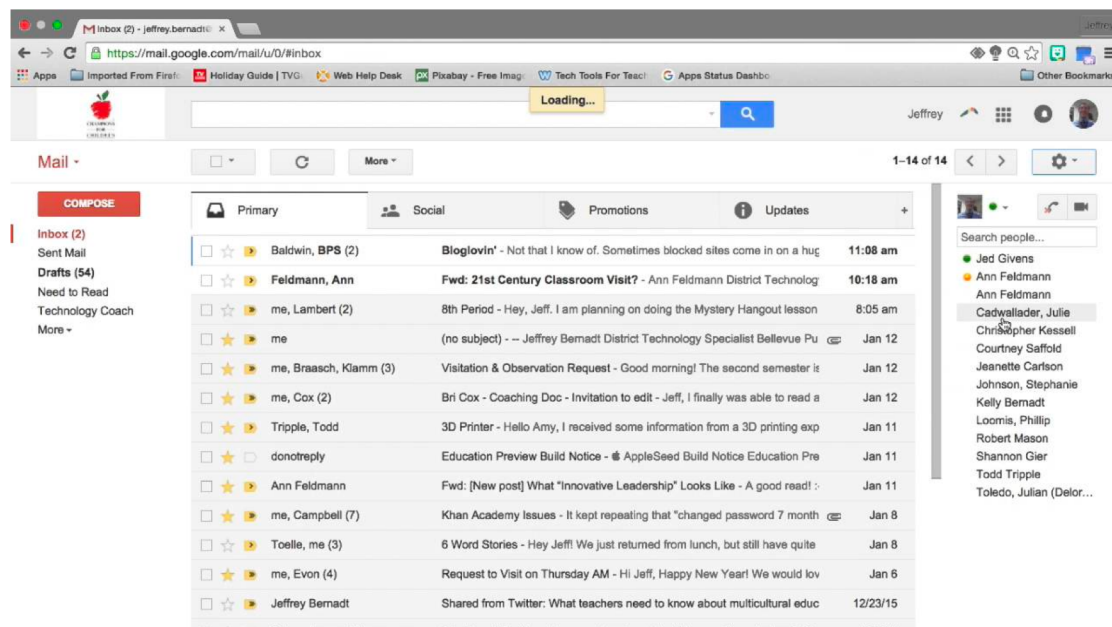
“Change” event

Dropdown interaction: What event fires?



“Select” event

Drag and Drop interaction: What events fire?

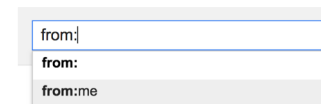
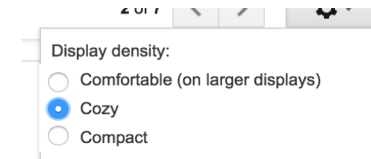
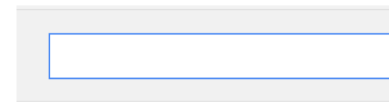
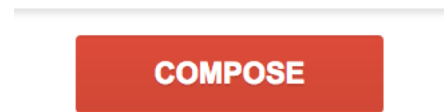


- “Drag” event
- “Drop” event

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”

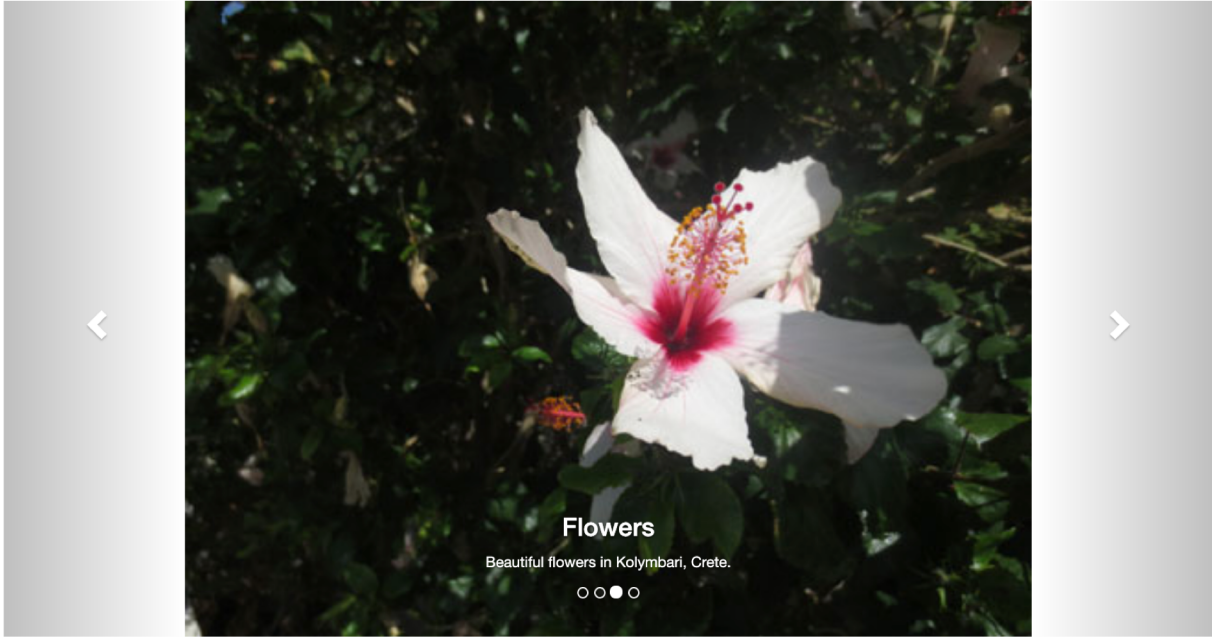
But the actions taken after an event is fired, are not standardized

Widgets can also be big

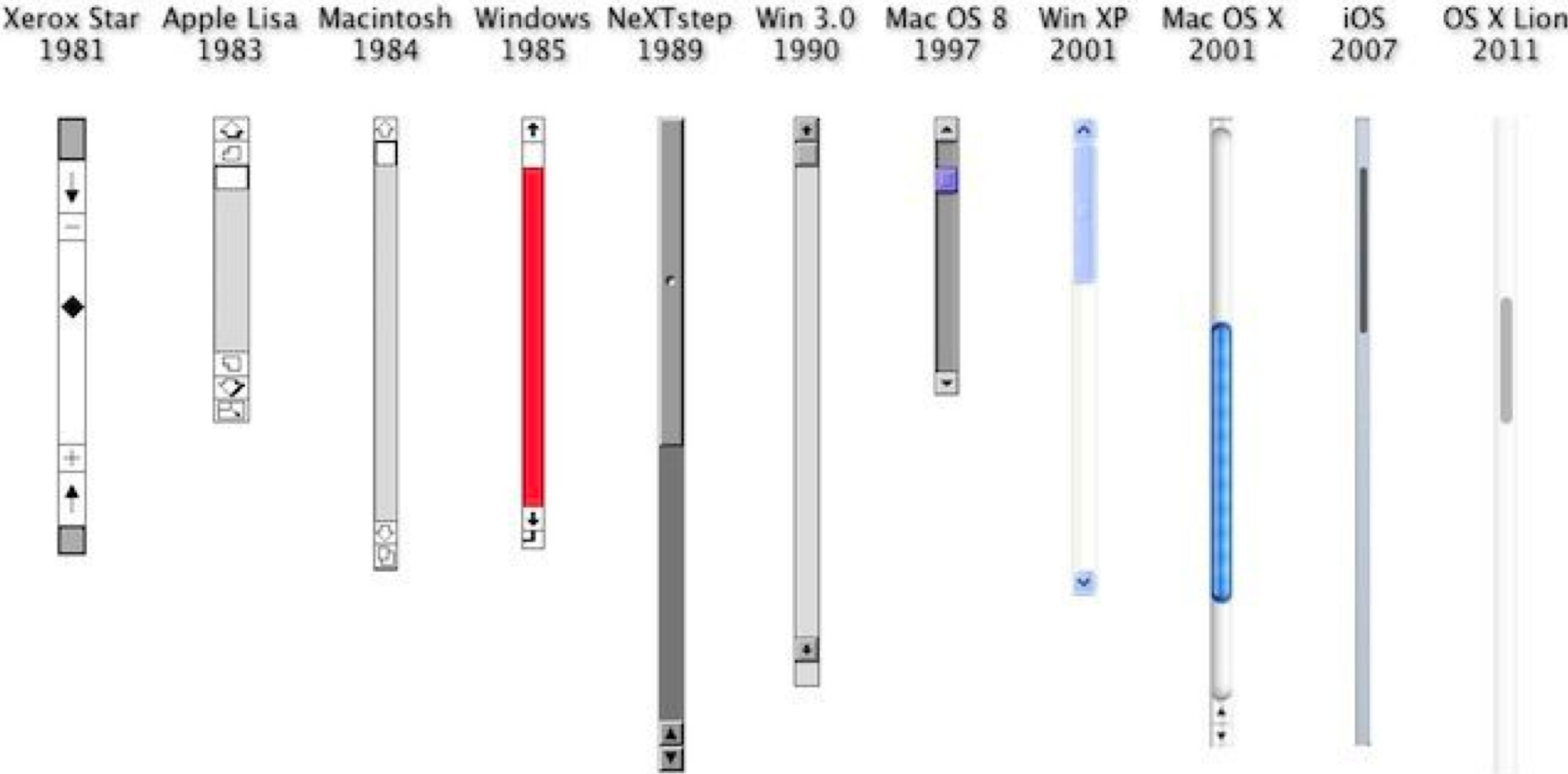
```
Run » Result Size: 1573 x 723
<div id="myCarousel" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
    <li data-target="#myCarousel" data-slide-to="1"></li>
    <li data-target="#myCarousel"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

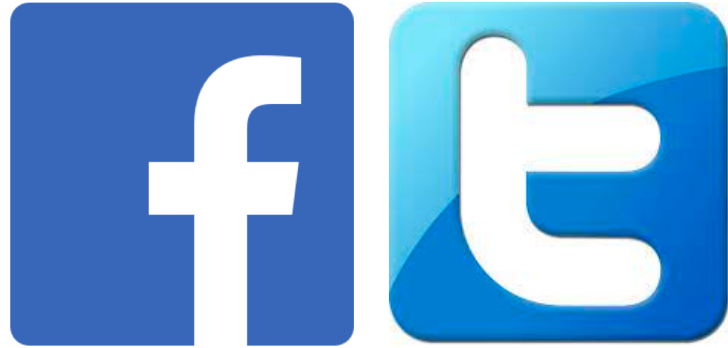


Because you did not program them yourself, widgets may appear and act differently on different devices

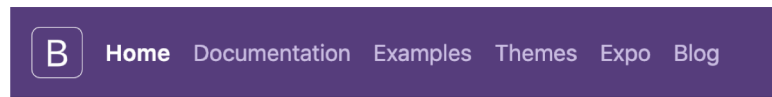


Pros and Cons of Standardization

Things that have become standardized



Because people people copy successful designs



Bootstrap

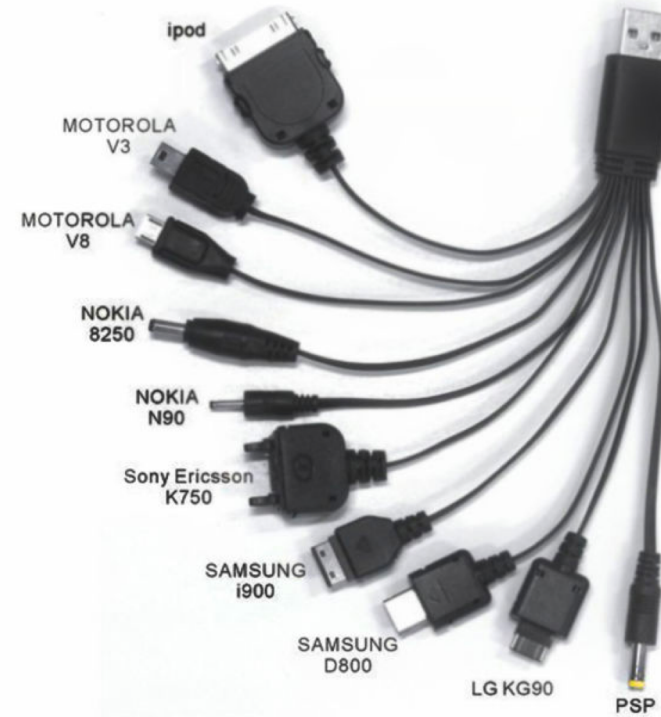
Build responsive, mobile-first projects on the world's most popular front-end component lib

Because people create good, reusable solutions



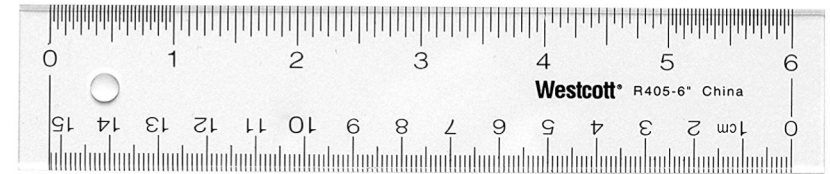
Because one version domains the market

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

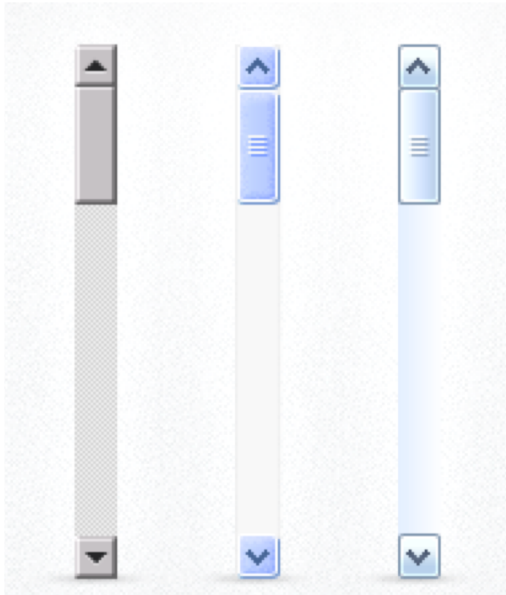


Non-Standardized

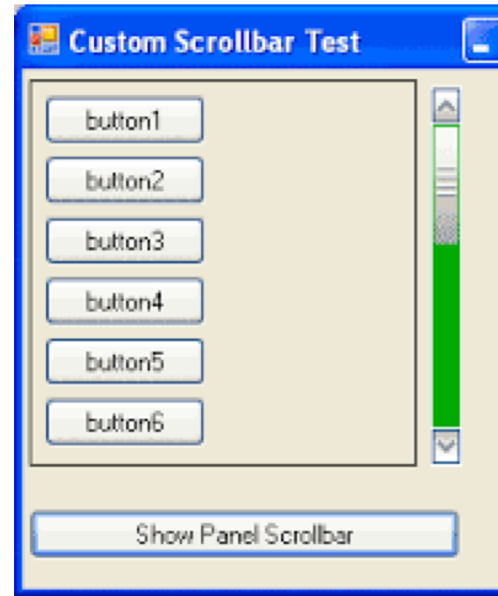


Widgets allow customization

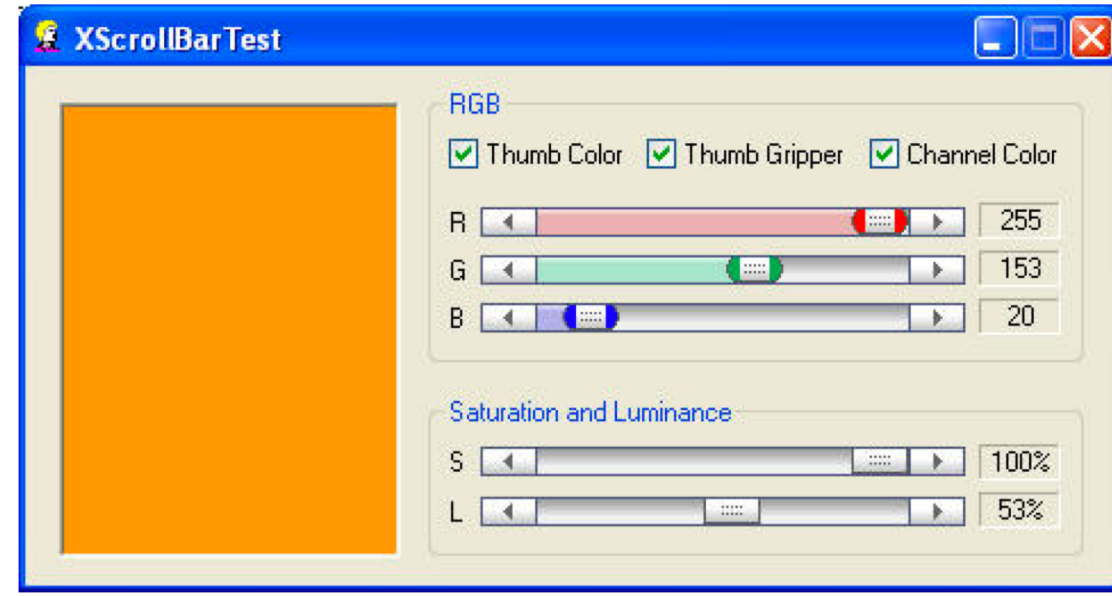
Customizable scroll bars



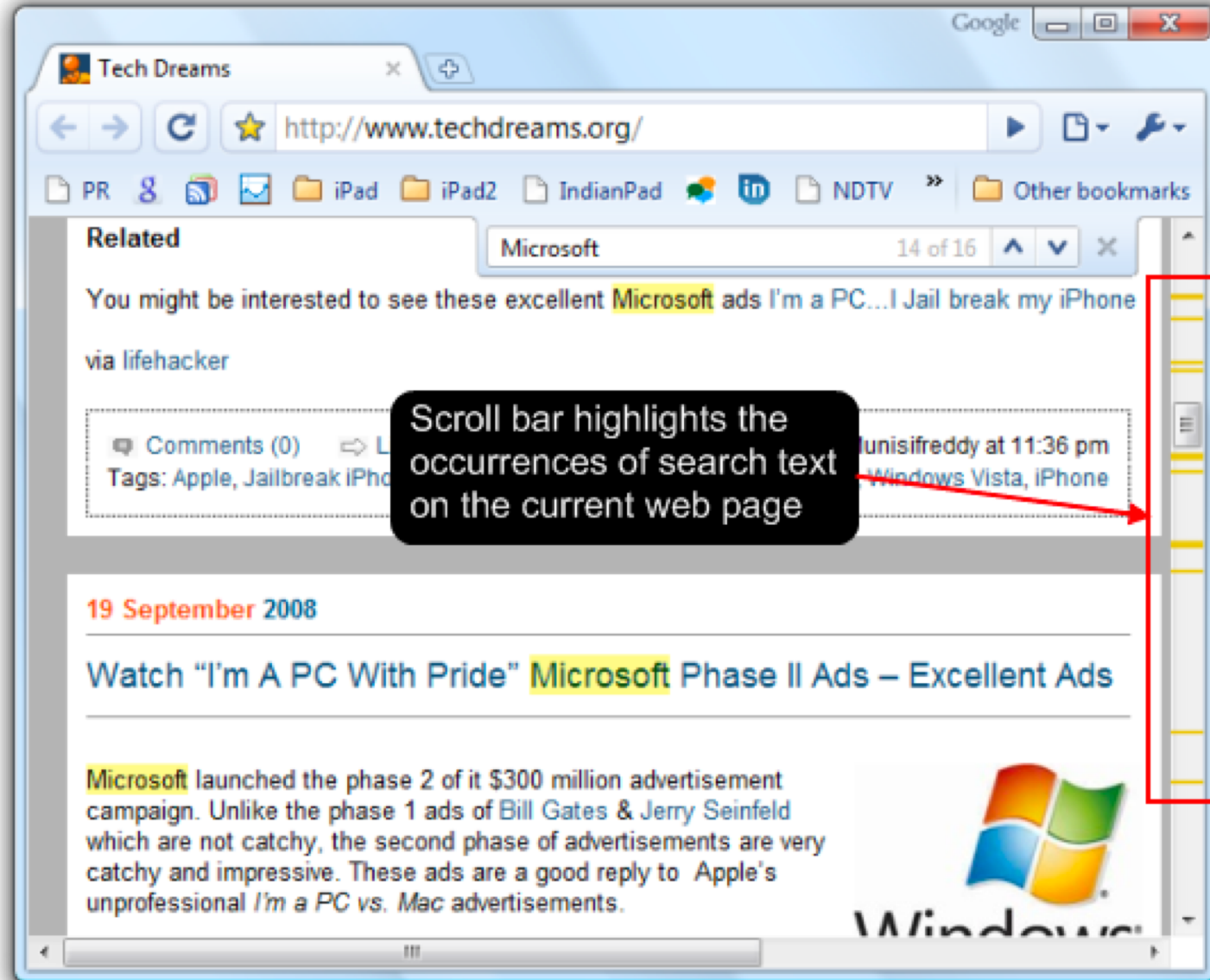
Bad use of customization.



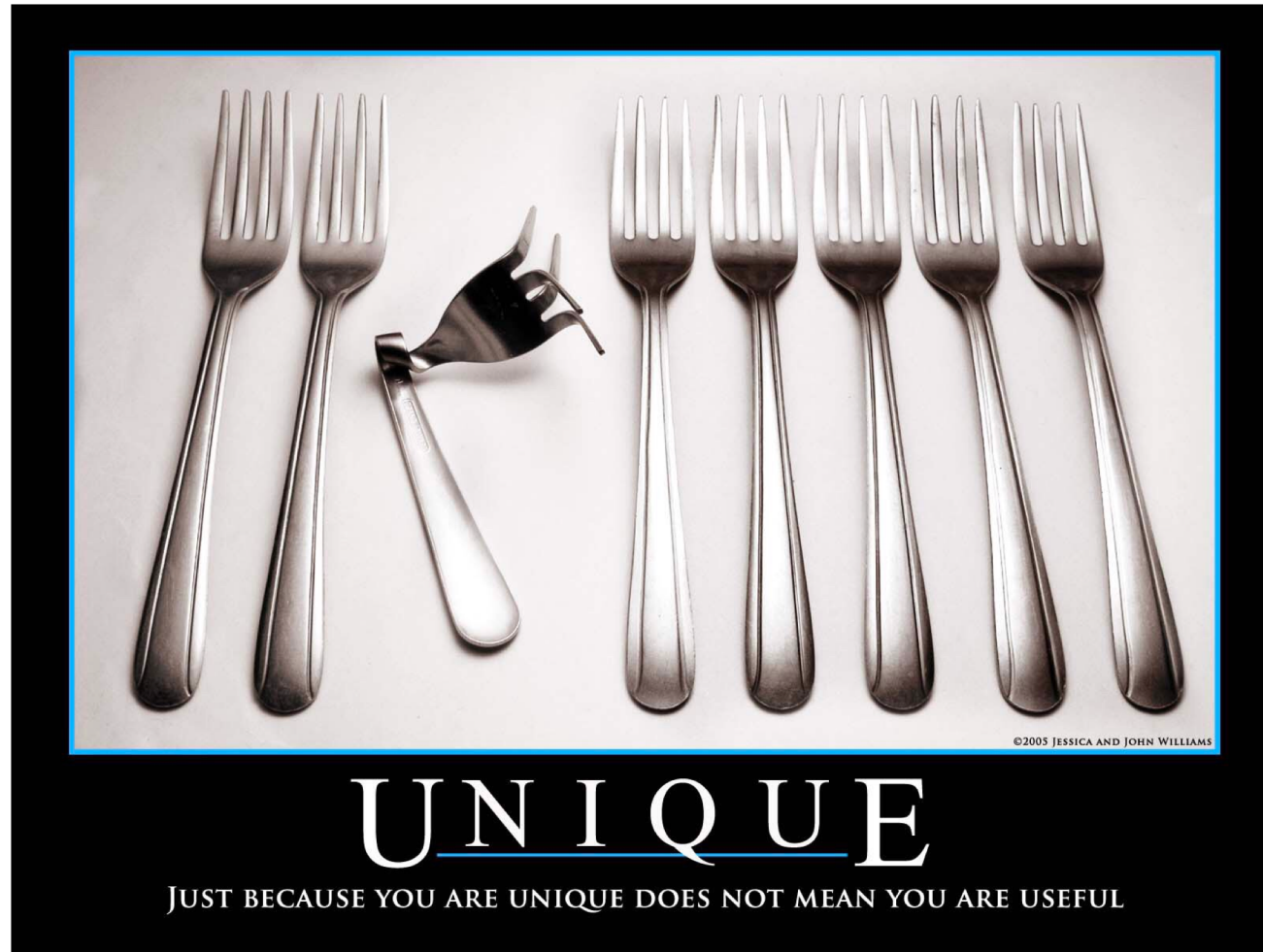
Good use of customization.



Widgets allow customization

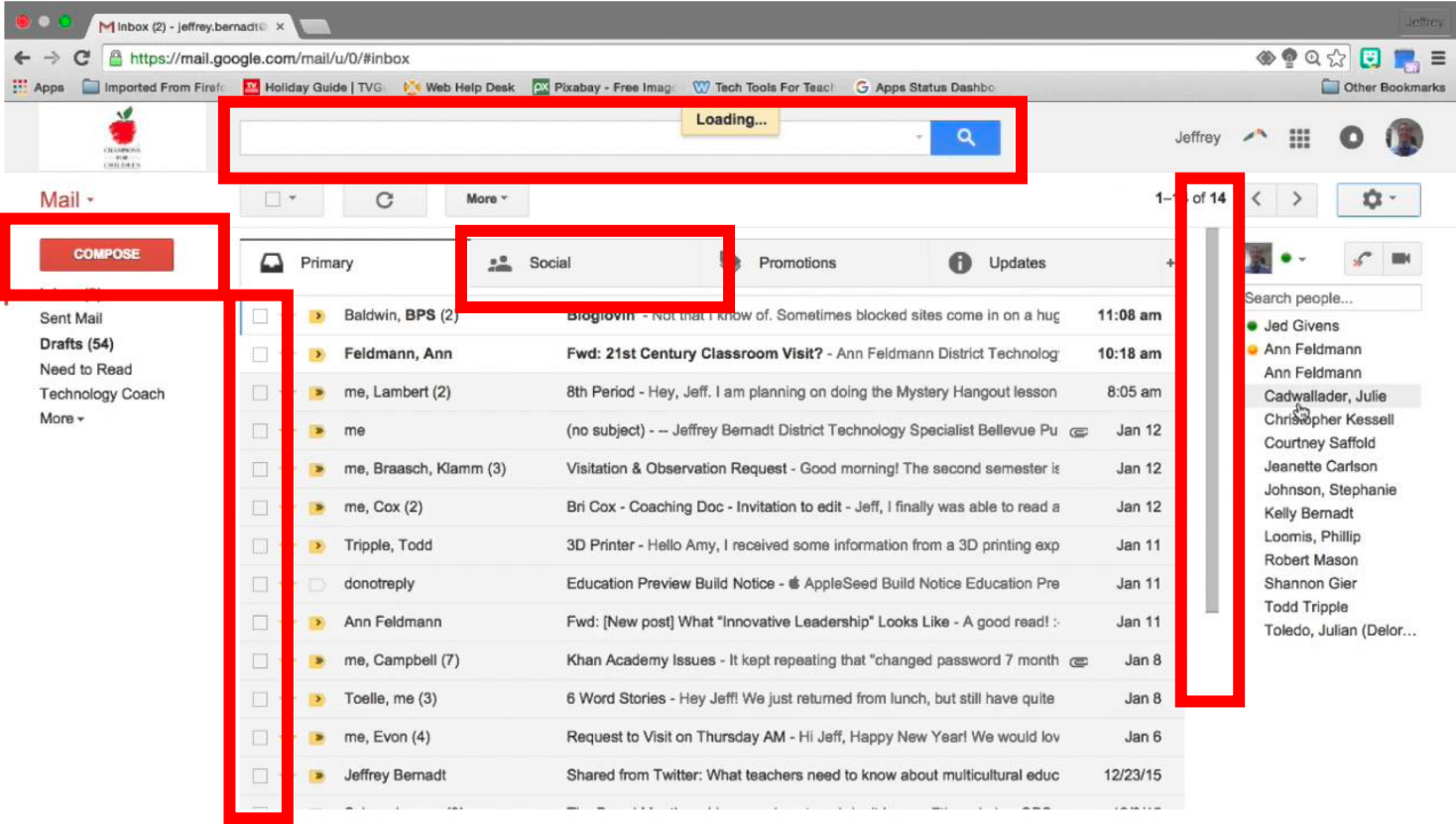


Use your powers of customization wisely.



Summary

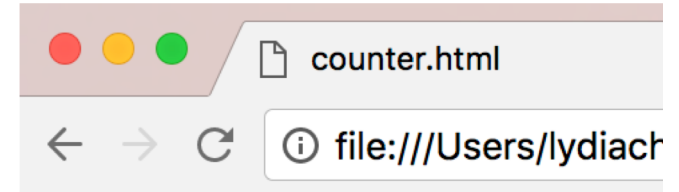
We interact with webpages through **widgets**: Elements with standardized appearance and events



Creating Interactions on the web has two parts:

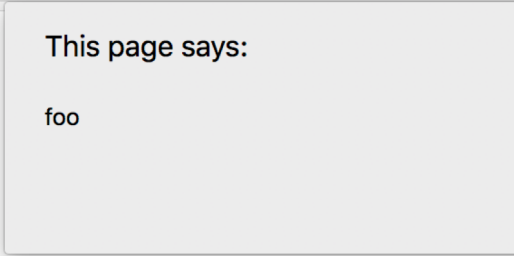
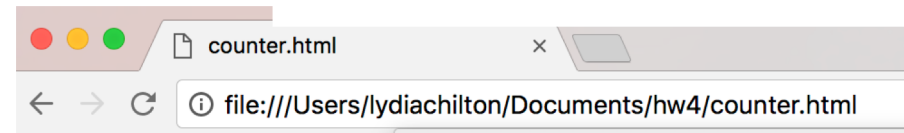
1. Program the interface and style in HTML & CSS

```
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 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 Clear console Minimize
"Running fiddle"
"0: hello"
"1: hi"
"2: hi"
>_
```

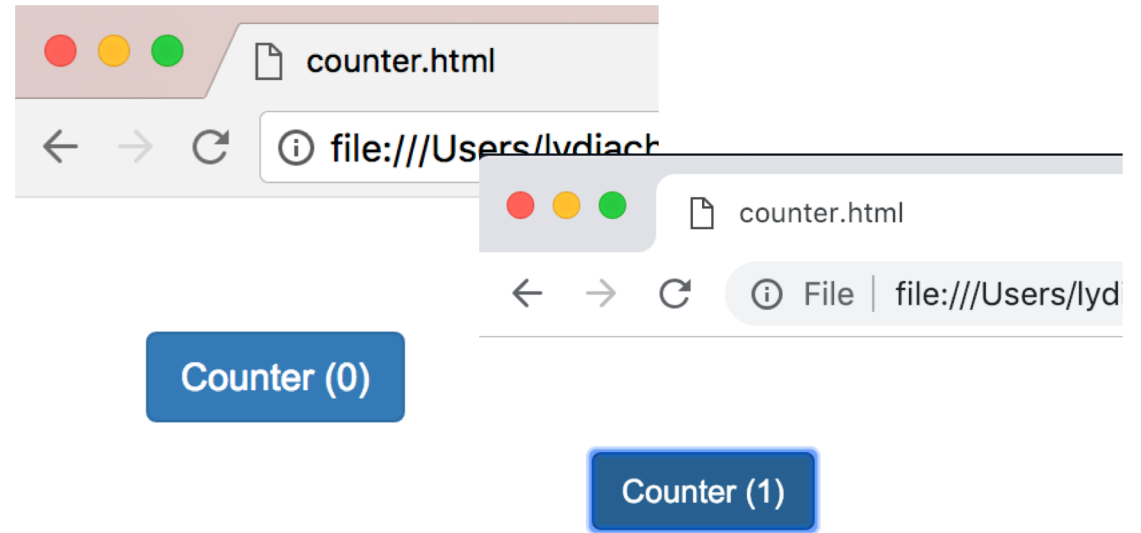
Let is block scoped, and can be re-assigned.

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
```

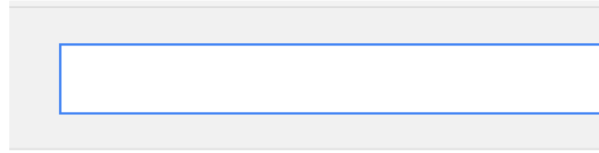
But the actions taken after an event is fired, are not standardized

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

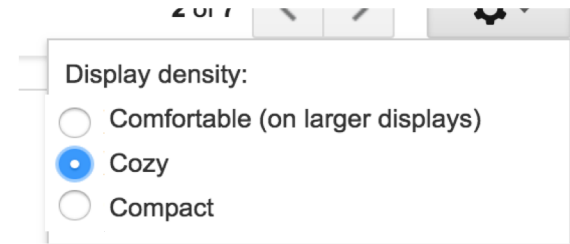
There are many types of widgets and events



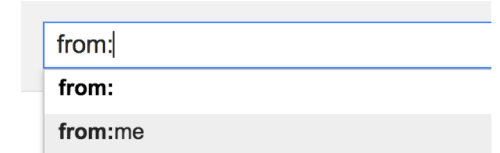
Click



Keypress



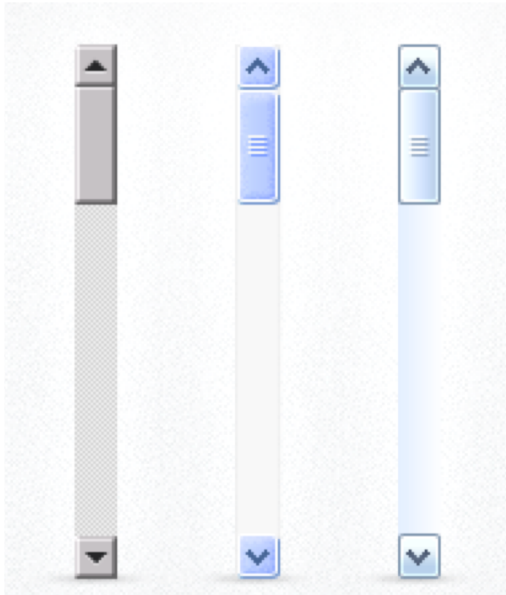
Change



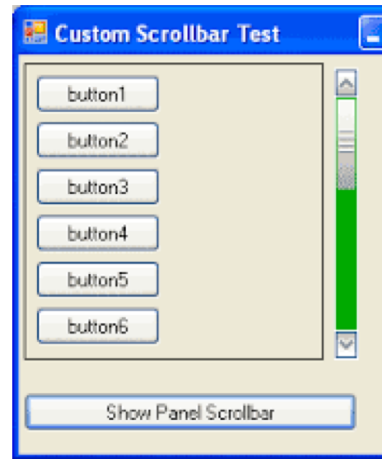
Select

Widgets allow customization. Use it wisely.

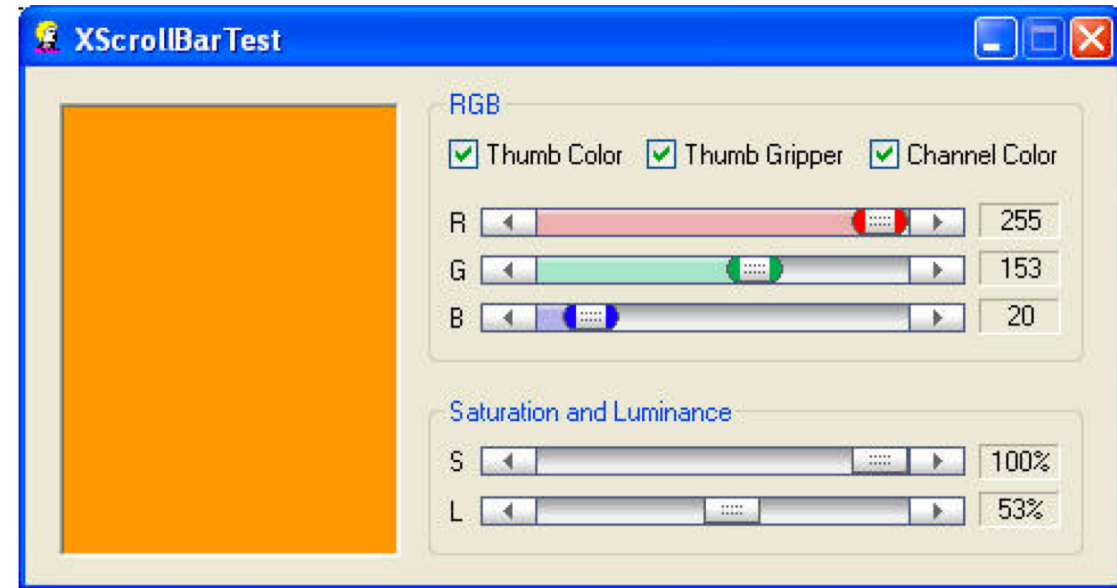
Customizable scroll bars



Bad use of customization.



Good use of customization.

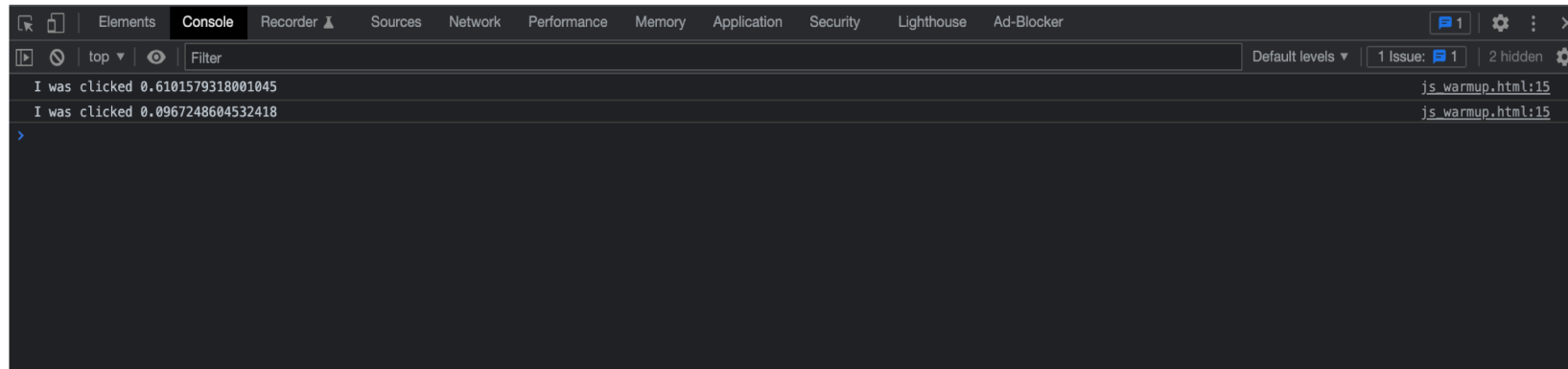
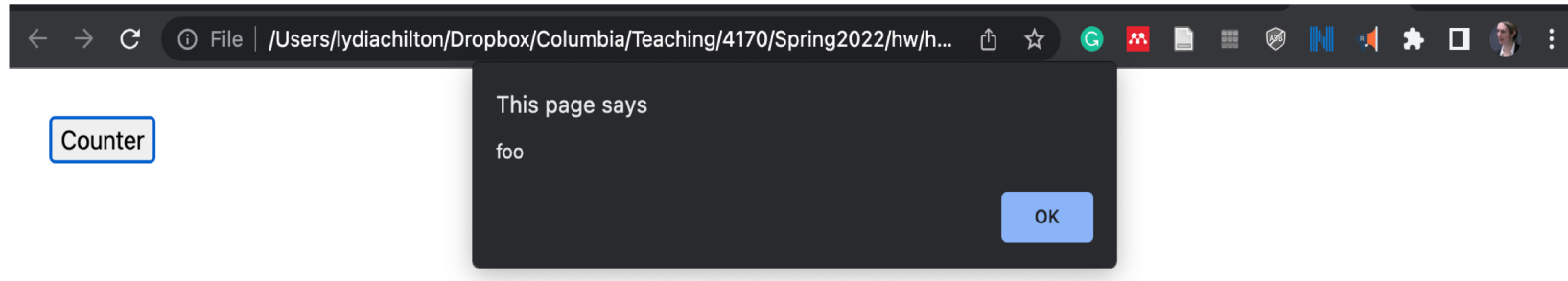


Homework 3: User Models and JavaScript

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 2/4 @ 11:59pm on Courseworks

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

Write a tweet

Call me Ishmael. Some years ago- never

-12 Post Tweet

POSTS

chilton Third post
chilton Second post
chilton First post