IN4MATX 285:
Interactive Technology Studio

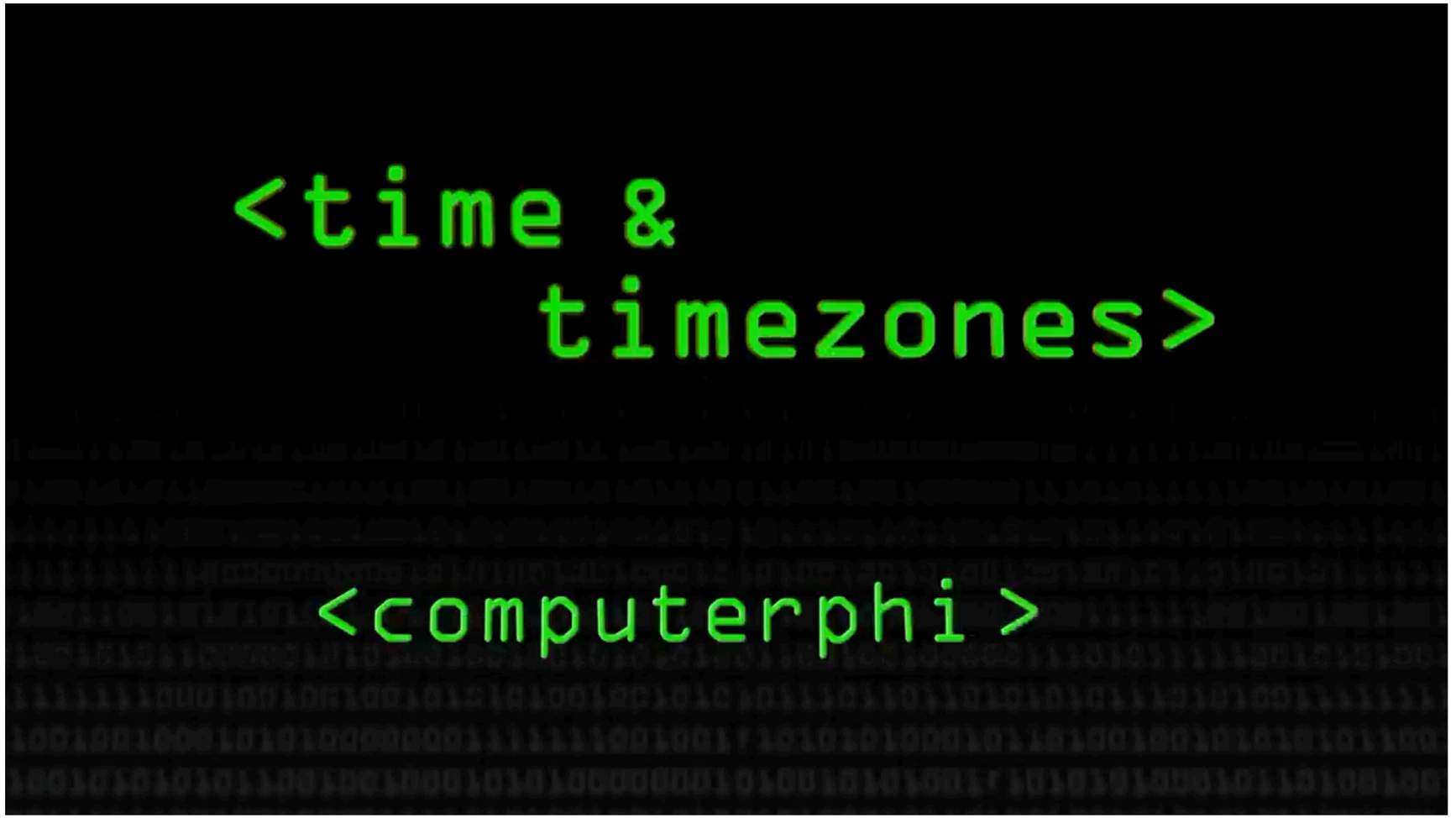
Practice: Libraries and Toolkits

Today’s goals

By the end of today, you should be able to…

- Articulate the role of libraries in software development, and the financial models that underlie them
- Describe the concepts of threshold and ceiling in interface tools and what tool designers should be striving to create
- Explain the relative threshold and ceilings of tools like Photoshop and Paint

Timezones



https://www.youtube.com/watch?v=-5wpm-gesOY

Timezones

https://www.youtube.com/watch?v=-5wpm-gesOY

Software library

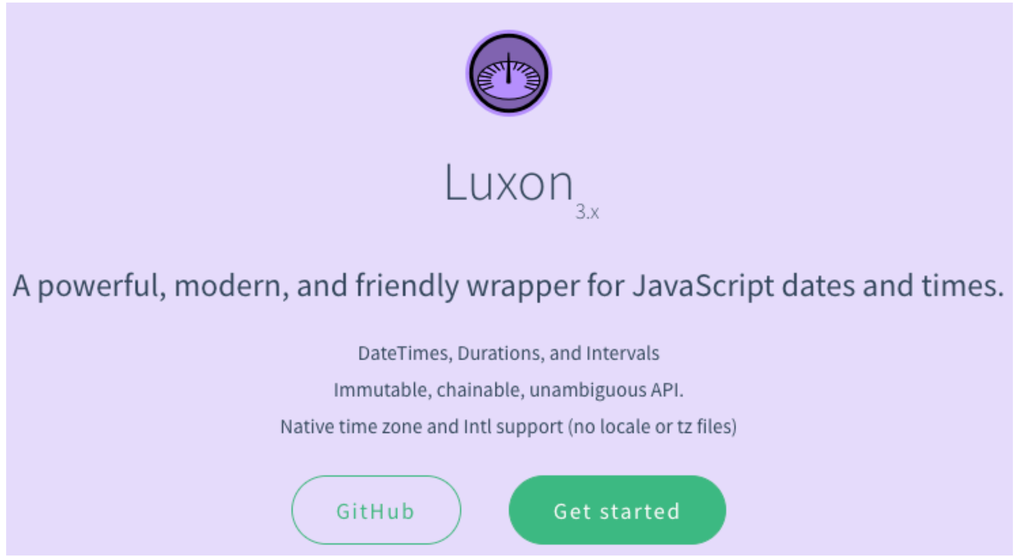
- A piece of software that solves a common coding problem that people have
- Programmers can then use, copy, extend, etc. that library for their own purposes
- Programmers only need to understand *how to use* the library, and don't need to understand how it works

One example: Luxon

- Solves time zones for us, hooray!

```
DateTime.now().setZone('America/New_York').minus({weeks:1}).endOf('day').toISO();
```

- I have no idea how it works, but I can look up how to use it



https://moment.github.io/luxon/#/

Libraries seem great, but who writes them? Why?

Why do we have good libraries?

- One part philosophical, one part economical
- Philosophical: It’s the right thing to do, and it helps us have good software
  - Programmers have all benefitted from libraries, and want to pay it forward
- Economical: Companies collectively benefit from good libraries
  - The public will build something better than you can make in-house
  - Many programmers want to contribute to open source, so allowing them to will help you retain them

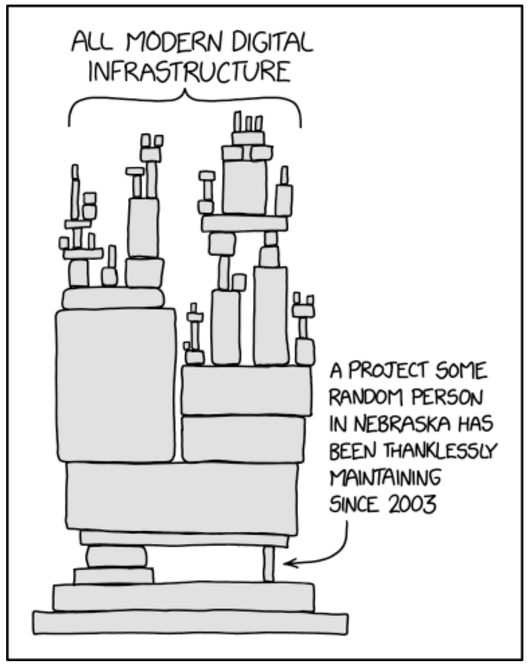
Libraries and licenses

- Software libraries are typically released with a *license*, which indicates how it can be used
- Some licenses don’t allow for commercial use
  - Others require attribution in some form
- If you plan to use a software library in a commercial product, you need to check the license

https://en.wikipedia.org/wiki/Software_license

Open source

- Code that's publicly available and can be modified
- Often related to the *open source model* of development, which is decentralized and collaborative
- Open source libraries can have varied licenses



https://en.wikipedia.org/wiki/Open_source

https://xkcd.com/2347/

Privately maintained libraries

- A lot of the most popular web libraries are maintained by private companies
- Meta maintains React, Google maintains Angular

How do you learn how to use libraries?

- Documentation
  - As important as the library!
  - Usually some form of “quick start”, and longer list of supported functions
  - Lots of examples of how to use it
- Look at the code of the library itself
  - Extremely tedious, but occasionally needed

Comparing DateTimes

DateTime implements `#valueOf` to return the epoch timestamp, so you can compare DateTimes with `<`, `>`, `<=`, and `>=`. T

```
d1 < d2 // is d1 before d2?
```

However, be aware that `===` compares object identity, which is not a useful concept in a library with immutable types. Use `#eq` timestamps, you can use:

```
d1.toMillis() === d2.toMillis() // are d1 and d2 the same instant in time?
+d1 === +d2 // same test, using object coercion
```

You may also use `#hasSame` to make more subtle comparisons:

```
d1.hasSame(d2, 'year');   // both DateTimes have the same calendar year
d1.hasSame(d2, 'day');    // both DateTimes have the same calendar day (which implies they also have th
```

Switching to a specific kind of library: toolkits

Thought exercise: How does a computer process user input?

Sequential programs (command line)

- Program takes control, prompts for input
- Person waits on the program
- Program says when it is ready for more input, which the person then provides

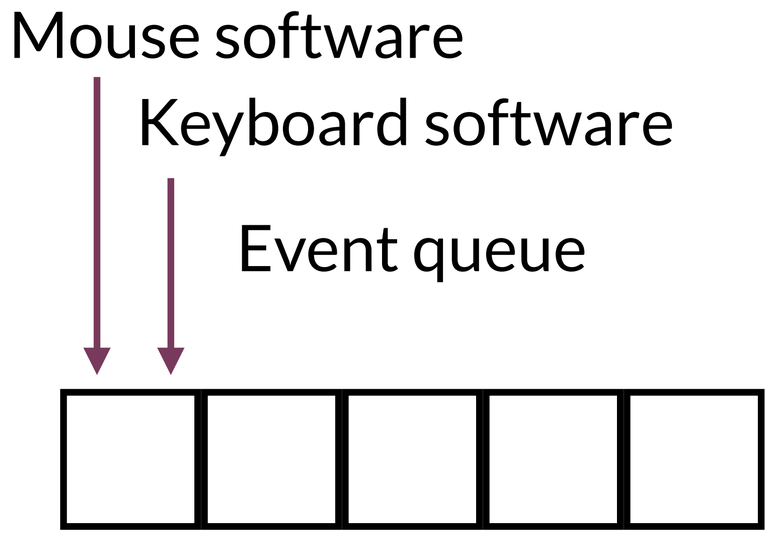
Sequential programs (command line)

```
while true {
   print “Prompt for Input”
   input = read_line_of_text()
   output = do_work()
   print output
}
```

- Person is literally modeled as a file

Event-driven programming

- A program waits for a person to provide input
- All communication is done via events
  - Mouse down, item drag, key up
- All events go in a queue
  - Ensures events are handled in order
  - Hides specifics from applications



Basic interactive software loop

- All interactive software has this loop somewhere

```
do {
  e = read_event();
  dispatch_event(e);
  if (damage_exists())
     update_display();
} while (e.type != WM_QUIT);
```

Input
Processing
Output

Basic interactive software loop

- Maybe if you're programming a game, you build this loop
- But imagine you had to write this loop every time you wanted to write a webpage, desktop app, or mobile app
- Instead, we rely on tools to handle common operations

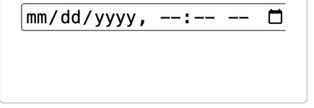
```
do {
  e = read_event();
  dispatch_event(e);
  if (damage_exists())
     update_display();
} while (e.type != WM_QUIT);
```

Example: a button

- What's behind a button?
  - Set X and Y boundaries
  - Check if mouse down is within those boundaries
  - Check if mouse up is *also* within those boundaries
  - If so, then fire an event
- What if you had to program this sequence every time you wanted to add a button to your website?

Interface standards

- We've developed lots of standard types of inputs, which lets us skip to what makes our interface and product interesting/unique

checkbox	A check box allowing single values to be selected/deselected.	Play 
color	A control for specifying a color; opening a color picker when active in supporting browsers.	Play 
date	A control for entering a date (year, month, and day, with no time). Opens a date picker or numeric wheels for year, month, day when active in supporting browsers.	Play 
datetime-local	A control for entering a date and time, with no time zone. Opens a date picker or numeric wheels for date- and time-components when active in supporting browsers.	Play 

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

Understanding tools

What is a user interface tool?

- Software or libraries which help you build a user interface
  - Design systems are user interface tools, designed to help make interfaces consistent

Understanding tools

We use tools because they…

- Identify common or important practices
- Package those practices in a framework
- Make it easy to follow those practices
- Make it easier to focus on the application we’re building

Understanding tools

Tools enable…

- Faster and more iterative design
- Better implementation than without the tool
- Consistency across applications using the same tool

Understanding tools

Why is designing tools difficult?

- Need to understand the core practices and problems
- Those are often evolving with technology and design
- The tasks people are trying to solve change quickly, so tools struggle to keep up

Brad Myers, Scott E. Hudson, and Randy Pausch. TOCHI, 2000. Past, present, and future of user interface software tools. https://doi.org/10.1145/344949.344959

Understanding tools

Key terms

- Threshold: How hard to get started
- Ceiling: How much can be achieved
- Path of least resistance: Tools influence what interfaces are created
- Moving targets: Changing needs make tools obsolete

Brad Myers, Scott E. Hudson, and Randy Pausch. TOCHI, 2000. Past, present, and future of user interface software tools.
https://doi.org/10.1145/344949.344959

Threshold

How hard to get started

- Some tools are harder to pick up
- Depends on what a person knows already
  - A new programming language adds to the threshold
  - If a tool borrows concepts from another popular tool, it will be easier for many people to pick up

Brad Myers, Scott E. Hudson, and Randy Pausch. TOCHI, 2000. Past, present, and future of user interface software tools. https://doi.org/10.1145/344949.344959

Ceiling

How much can be achieved

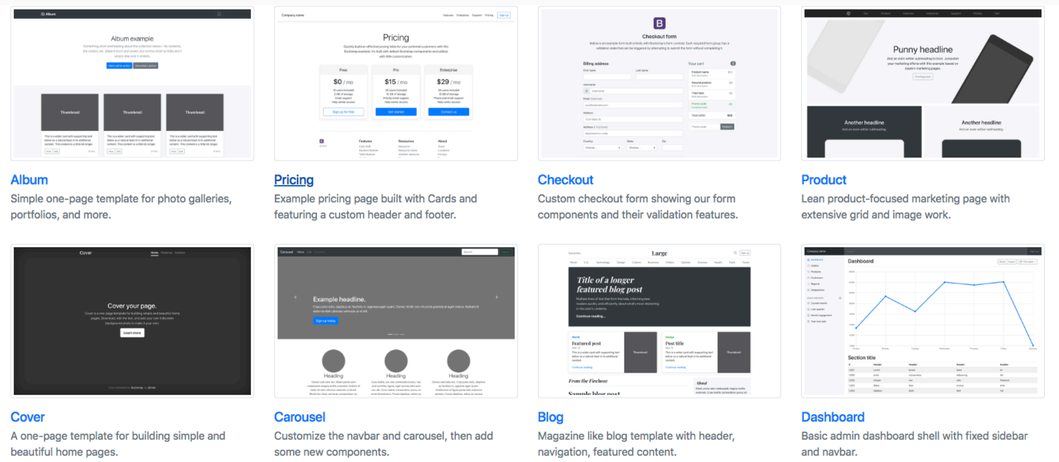
- Tools restrict what's possible
  - Your program could do much more if it had direct access to the bits on your computing device

Brad Myers, Scott E. Hudson, and Randy Pausch. TOCHI, 2000. Past, present, and future of user interface software tools. https://doi.org/10.1145/344949.344959

Path of least resistance

Tools influence what interfaces are created

- Sapir-Whorf Hypothesis
  - Roughly, some thoughts in one language cannot be expressed or understood in another language
- Our tools frame how we think about interaction and design



Brad Myers, Scott E. Hudson, and Randy Pausch. TOCHI, 2000. Past, present, and future of user interface software tools. https://doi.org/10.1145/344949.344959

Moving targets

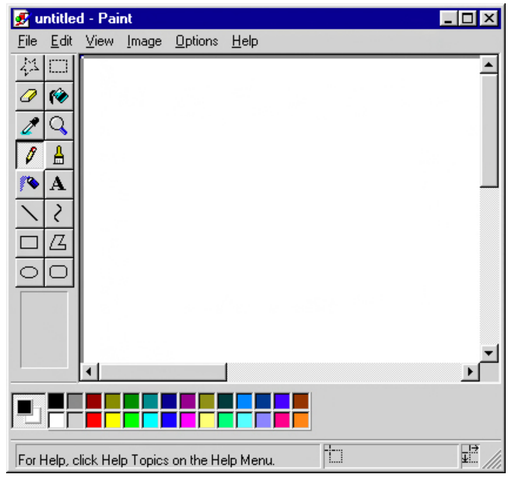
Changing needs make tools obsolete

- Codification eventually constrains design
  - Our understanding of how people interact with technology improves
  - New technology comes along to change the needs of tools
  - Example: Virtual reality has wildly different interactions and tool needs

Brad Myers, Scott E. Hudson, and Randy Pausch. TOCHI, 2000. Past, present, and future of user interface software tools. https://doi.org/10.1145/344949.344959

Between Photoshop and Paint…

Which will have a higher *threshold*? A higher *ceiling*?

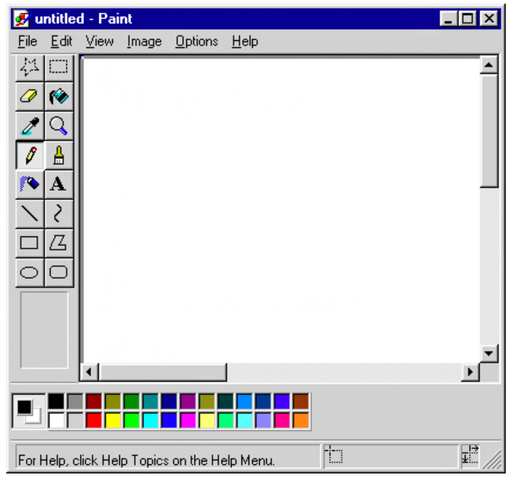


Microsoft Paint:
Low threshold,
but low ceiling

Adobe Photoshop:
High threshold,
but high ceiling

Threshold and ceiling

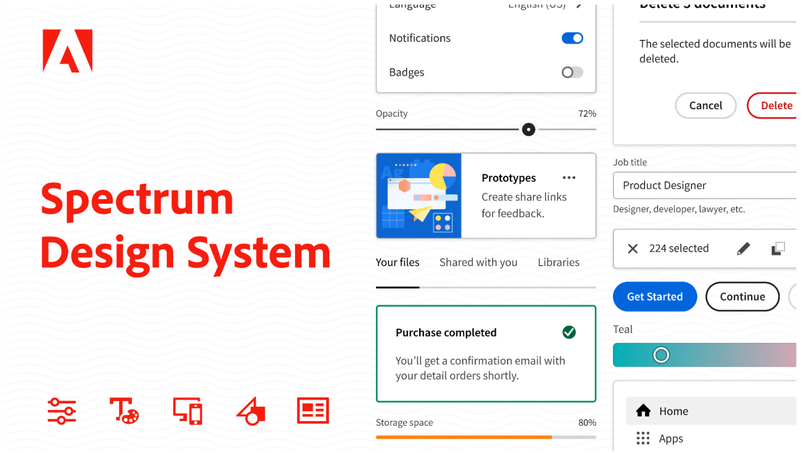
- It’s all relative; no absolute measure
- Tools should be *low threshold*
  - Easy to pick up
- But tools should also be *high ceiling*
  - Can do a lot
- The best tools are both
  - Photoshop introduces tutorials, etc. to lower the threshold

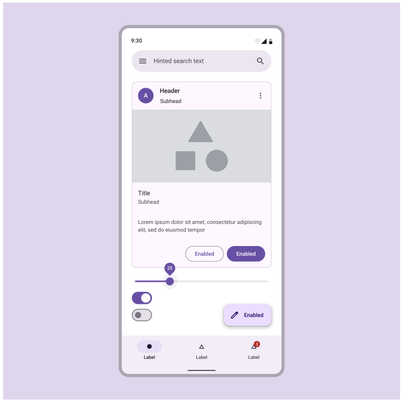


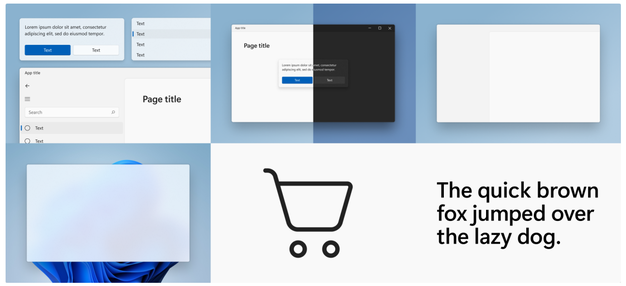
Other interface tools

Design systems

- Lower threshold than making a good design from scratch
- But, lower ceiling
  - Only support the flexibility that the system developers envision
- Definitely constrain what you make

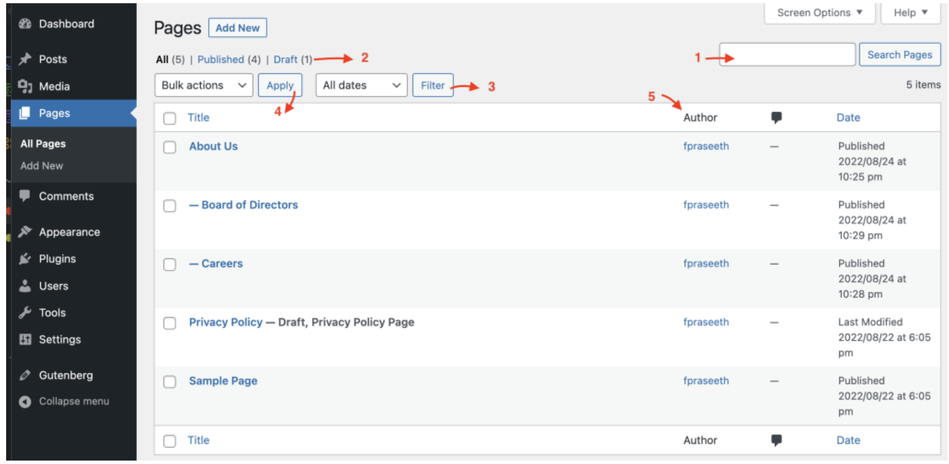


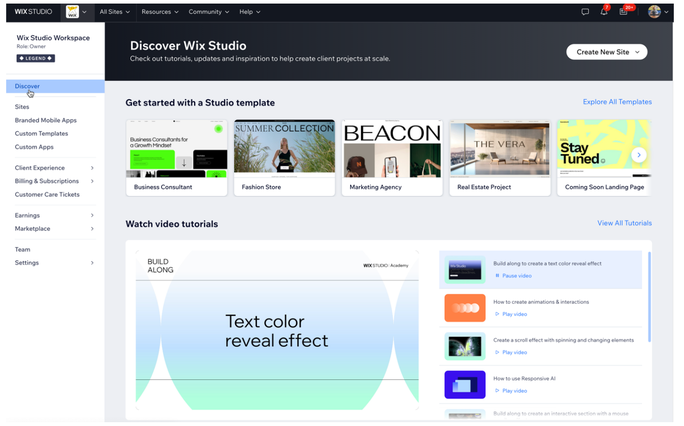




Website builders

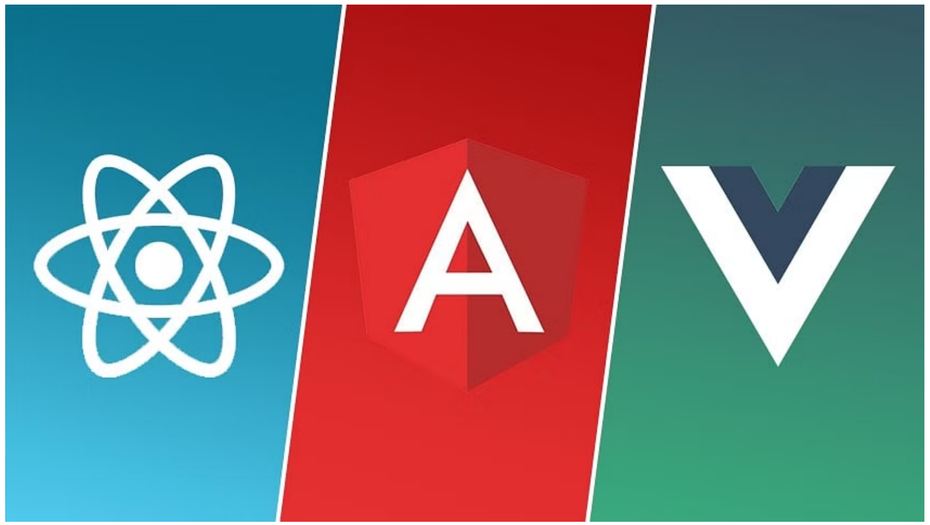
- Ever notice how all WordPress sites feel the same?
- Lower threshold, lower ceiling
  - Templates lower the threshold further, but also limit your options more





Web frameworks

- React, Vue, Angular
  - All support making more complex webpages
- High threshold, high ceiling
  - Harder to learn than plain HTML/CSS/ JavaScript
  - But you can make more complicated webpages



Web frameworks: why?

Web frameworks

A “small” client interface

- 3 pages
- Limited interactivity between pages; each page was fairly self-contained
- Interface was static, not personalized to an individual user

My Widget Store

Toy Boat

$15.99

4.6

How many? 10 What color?

Shipping

Address

6093 Donald Bren Hall | Irvine | CA | 92617

Shipping Speed

Standard Shipping (5 days, $3.99)

Place Order

Order in!

Thanks for shopping at My Widget Store. Your business is greatly appreciated!

Summary

Product	Color	Unit Price	Quantity	Subtotal
Toy Boat	#fffd9d	$15.99	10	$159.90
Paper Plane	#372bd4	$8.52	3	$25.56
Stuffed Animal	#309179	$9.13	6	$54.78
Train Car	#000000	$25.00	0	$0.00
Shipping (standard)				$3.99
Total			19	$244.23

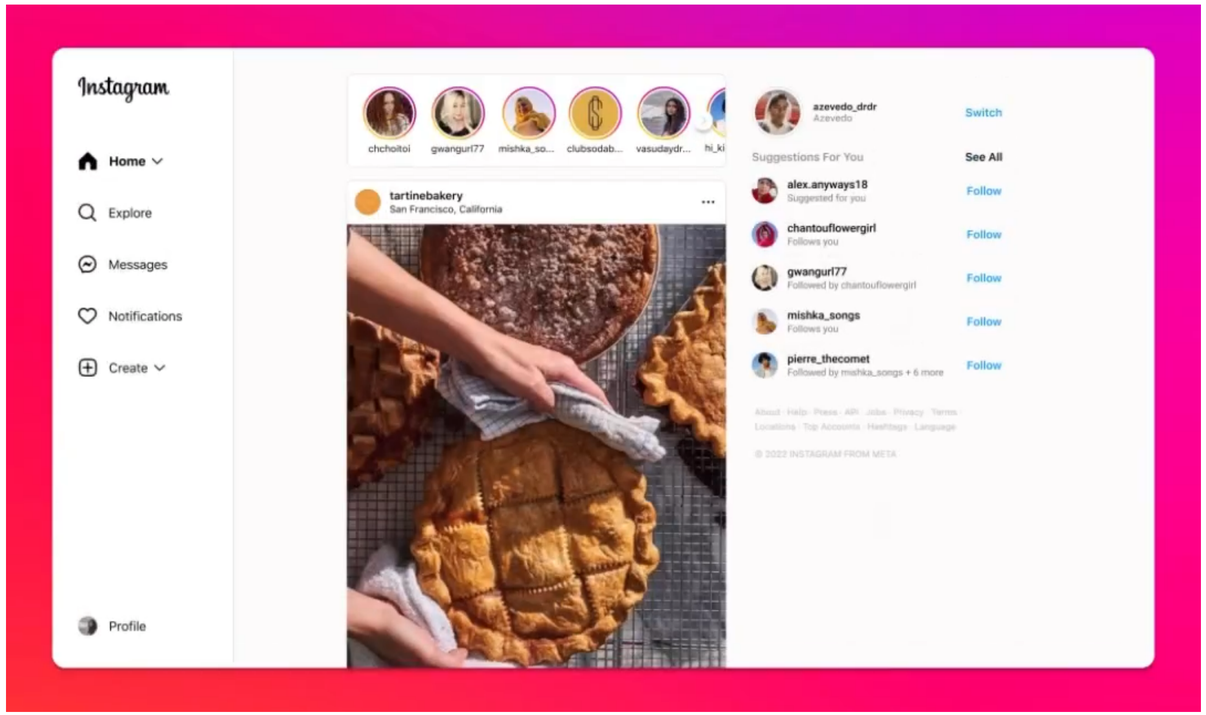
Shipping to: 6093 Donald Bren Hall Irvine, CA 92617.

Your order will arrive on: 3/5/2025.

Web frameworks

A “large” client interface

- Hundreds of pages and ways to navigate between pages
- Repeated UI components (posts, heart button)
- Different content, links, etc. displayed for each person



Web frameworks

- Frameworks make it possible to develop large interfaces
  - High ceiling: make it possible to create interfaces you wouldn't have made otherwise
  - Or at minimum, make it much easier
- Beyond a certain complexity, basically every site leverages a web framework
  - But, higher threshold. Need to understand the basics of HTML/CSS/JS, and the specifics of how that framework functions
  - We'll introduce frameworks this quarter, but won't require using them

Today’s goals

By the end of today, you should be able to…

- Articulate the role of libraries in software development, and the financial models that underlie them
- Describe the concepts of threshold and ceiling in interface tools and what tool designers should be striving to create
- Explain the relative threshold and ceilings of tools like Photoshop and Paint

IN4MATX 285:
Interactive Technology Studio

Practice: Libraries and Toolkits