IN4MATX 231:
User Interface Design & Evaluation

Class 19:
Interface development tools

Daniel Epstein

Today’s goals

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

- Describe the concepts of threshold and ceiling in software tools and what tool designers should be striving to create
- Describe why tools eventually limit design thinking
- Explain a few approaches to interface development which do not require programming
- Explain a few considerations for interface development when programming is used

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

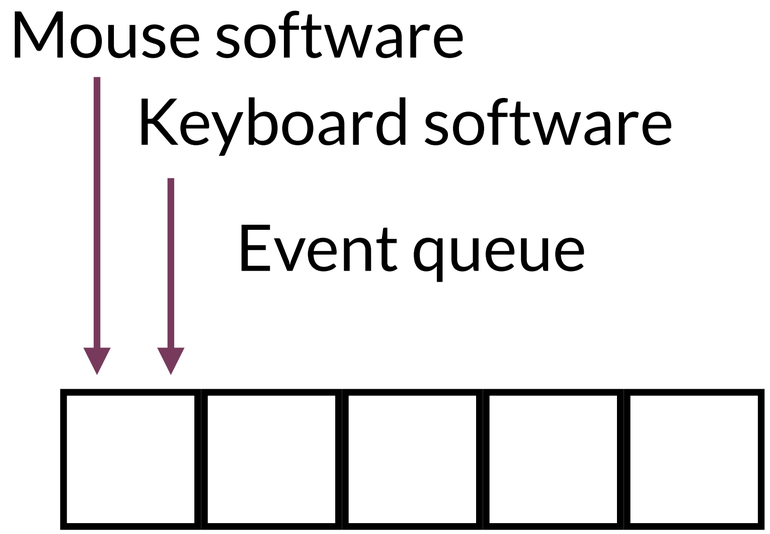
Sequential programs (command line)

```
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've made a game, you've built 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?

Understanding tools

What is a user interface tool?

- Software or libraries which help you build a user interface
  - Bootstrap is a user interface tool, designed to help make interfaces responsive
  - Angular, React, etc. are all user interface tools

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

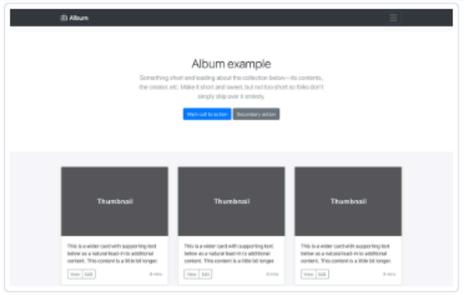
Tools influence what interfaces are created

- 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

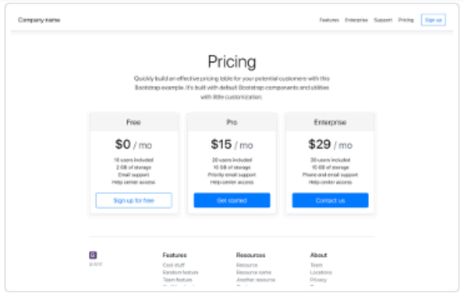
Path of least resistance

Tools can lead to similar-looking webpages



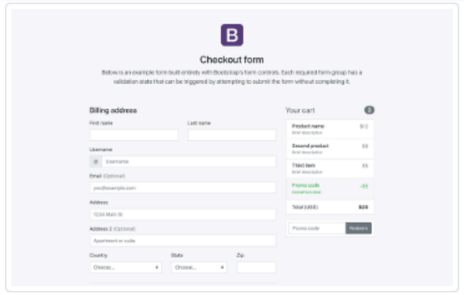
Album

Simple one-page template for photo galleries, portfolios, and more.



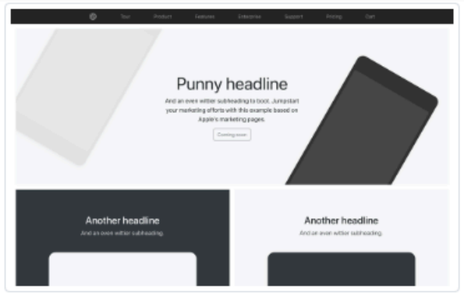
Pricing

Example pricing page built with Cards and featuring a custom header and footer.



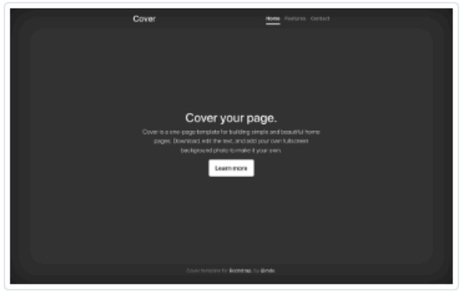
Checkout

Custom checkout form showing our form components and their validation features.



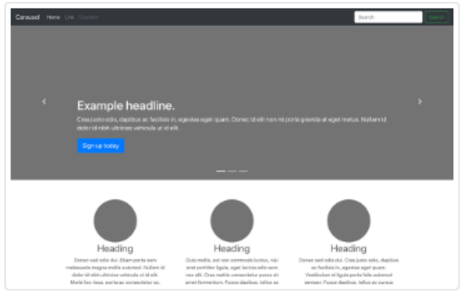
Product

Lean product-focused marketing page with extensive grid and image work.



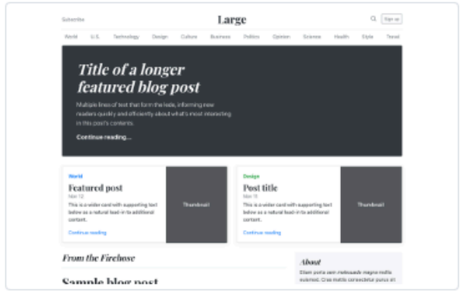
Cover

A one-page template for building simple and beautiful home pages.



Carousel

Customize the navbar and carousel, then add some new components.



Blog

Magazine like blog template with header, navigation, featured content.

Dashboard

Basic admin dashboard shell with fixed sidebar and navbar.

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 can stifle creativity

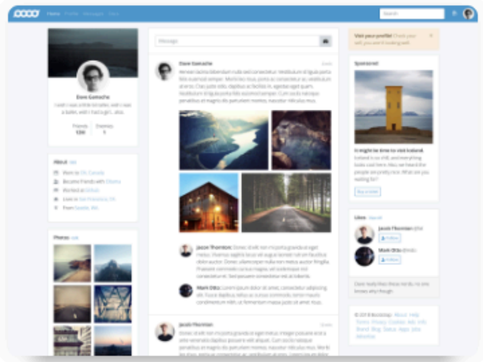
Themes built by or reviewed by Bootstrap's creators.

Why our themes?

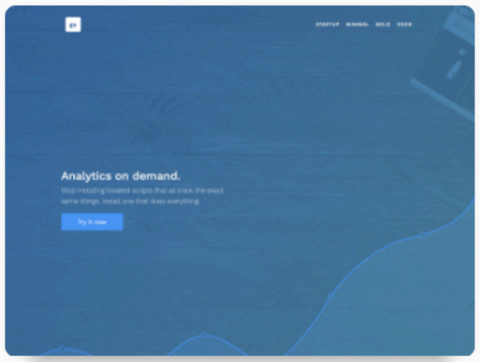
Built by Bootstrap Team

Component-based frameworks designed, built, and supported by the Bootstrap Team.

Dashboard $49.00
Admin & Dashboard ★★★★★



Application $49.00
Application ★★★★★



Marketing $49.00
Landing & Corporate ★★★★★

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

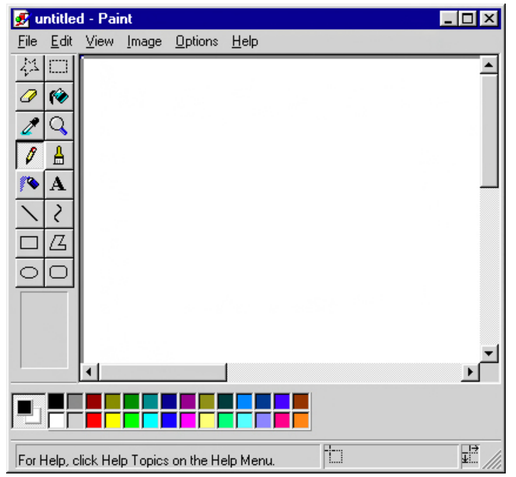
Changing needs make tools obsolete

- 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

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



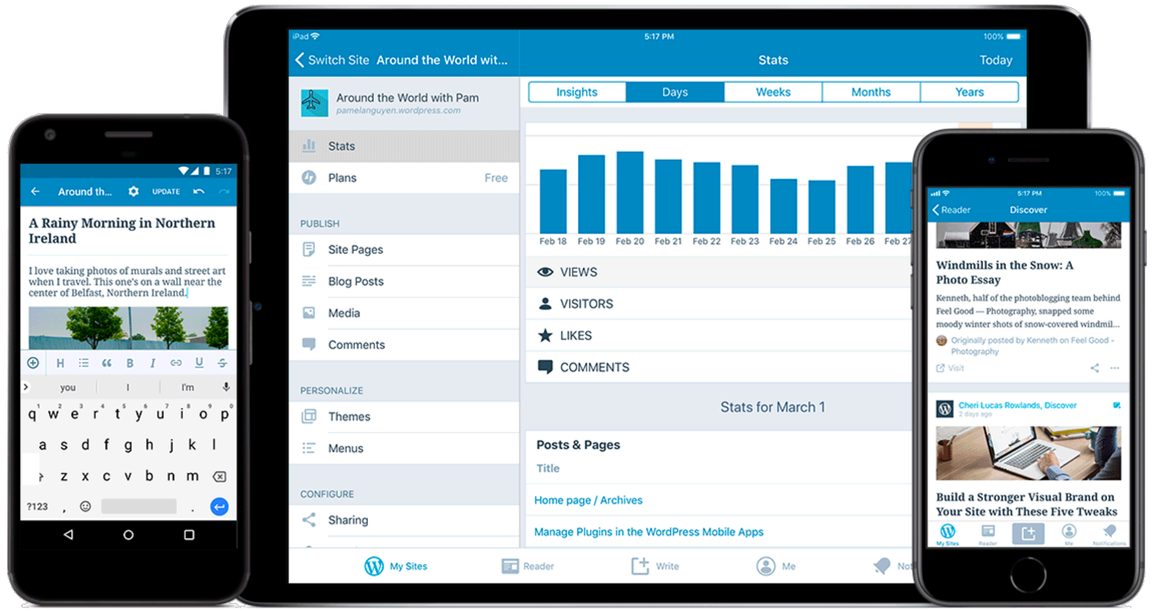
Switching gears: tools

Interface development tools

- You all are a diverse crowd, it’s inherently hard to create material useful for all of you
- I’ll first talk a little about implementation tools without programming, then some with programming

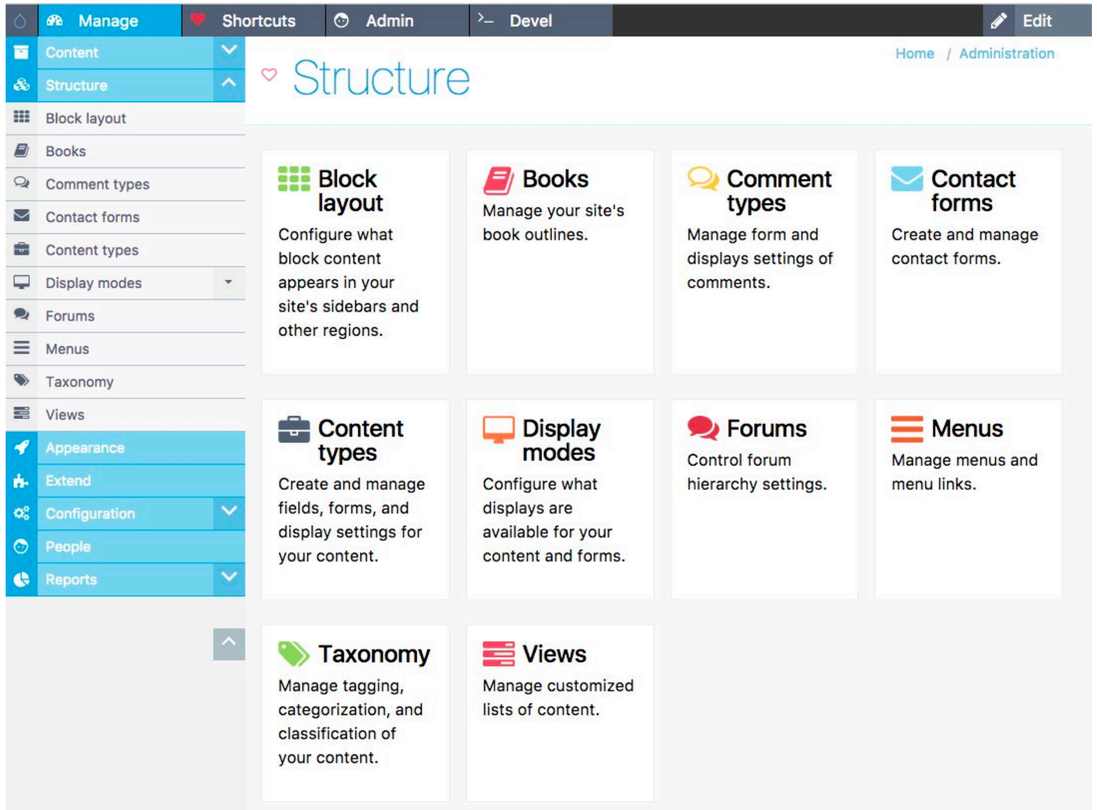
Avoiding programming

- There are dozens of drag-and-drop tools for creating websites and mobile apps
  - WIX
  - WordPress
  - BuildFire
  - Thunkable



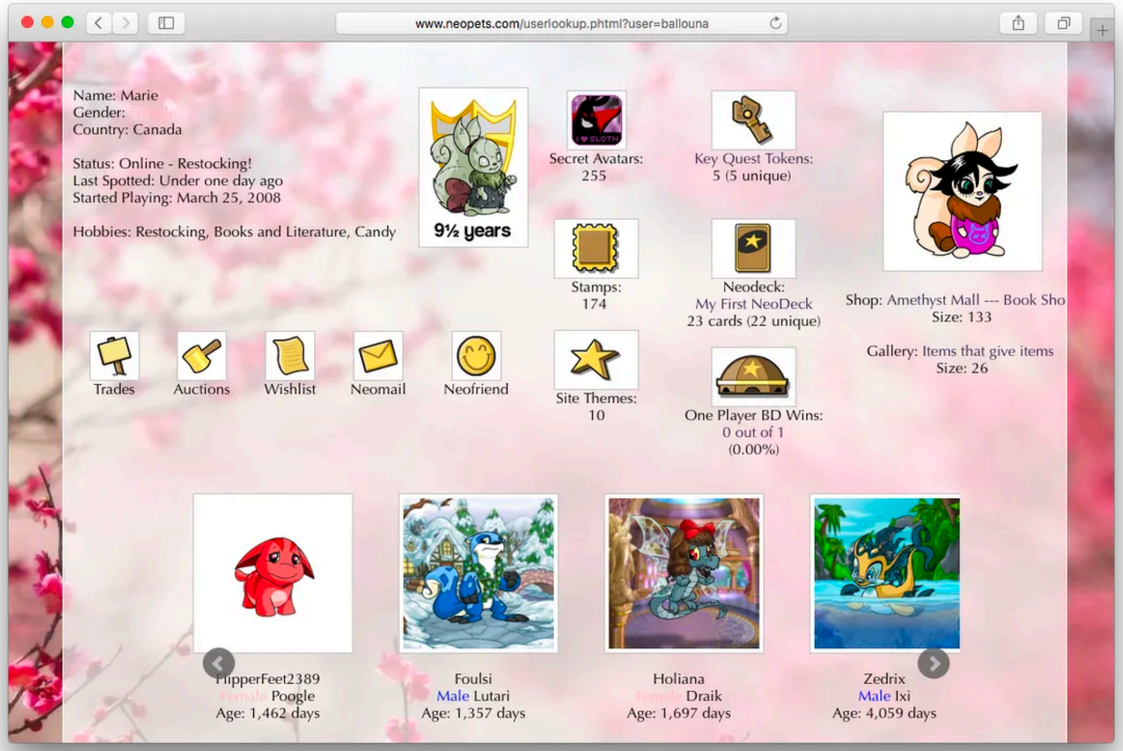
Avoiding programming

- Most drag-and-drop tools use some sort of content management system (CMS)
  - For indexing data, specifying layout, producing input forms, etc.
  - Are especially useful when content needs to change regularly, like a news website



Avoiding programming

- These CMS sometimes allow for writing custom HTML or CSS for layout or styling
  - The first HTML I ever wrote was to customize my Neopets profile, inside of their CMS
- But, they often restrict what's possible
  - e.g., they have a low ceiling



Avoiding programming

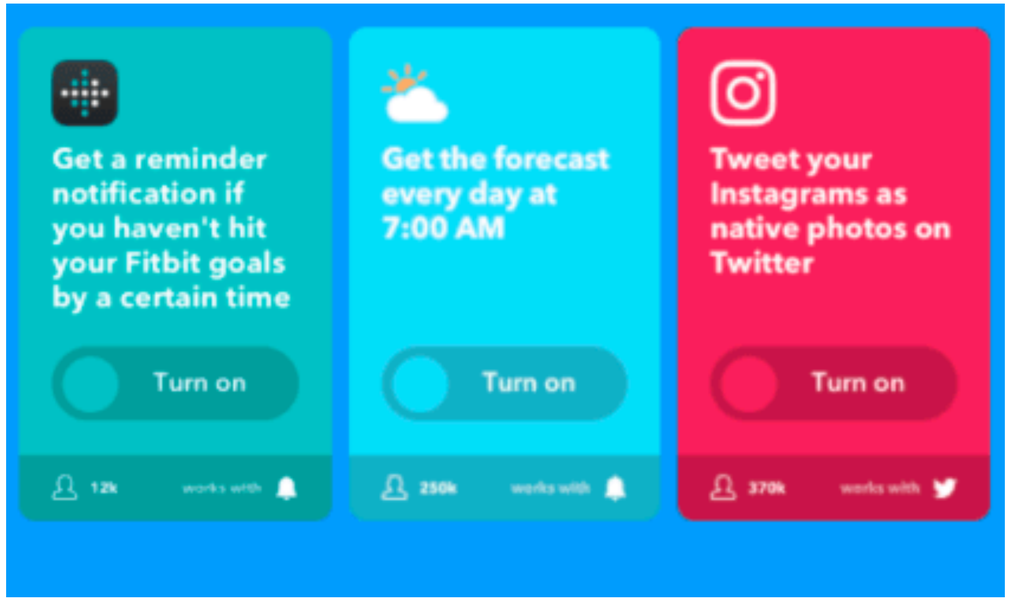
- You can do quite a lot with drag-and-drop tools
  - These tools are great for first prototypes, research tools, personal websites, etc.

Avoiding programming

- These tools are useful when:
  - We want to change content frequently
  - We don’t need databases, sensors, or other features which require coding
  - We don’t want to worry about things which increase development time like device compatibility, varied resolutions, etc.
  - Our team doesn’t have much programming knowledge

Avoiding programming

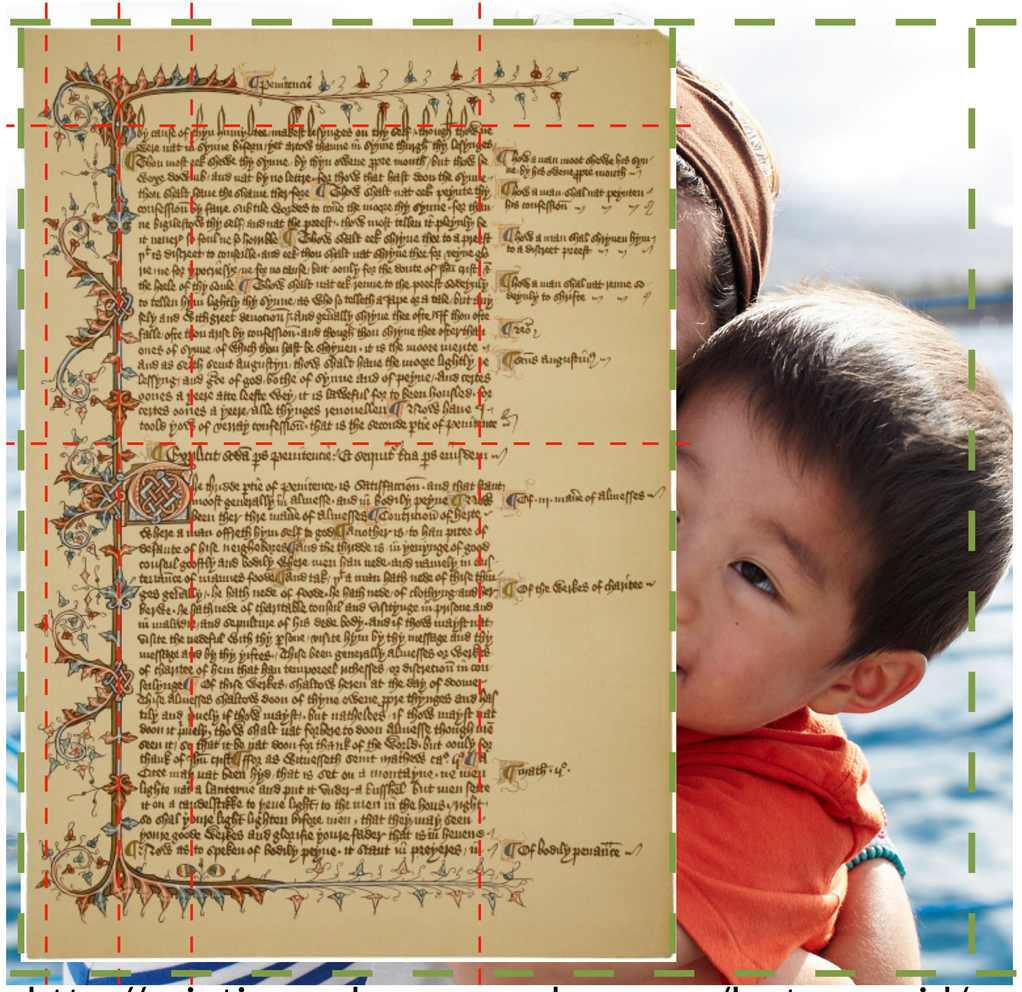
- Tools can also replace challenging parts of study design and maintenance
  - Cloud spreadsheet instead of a database (Google Docs)
  - Trigger-action programming (IFTTT)
  - Advanced survey tools (SurveyGizmo)
  - Mail merge extensions (Boomerang)



Interface development with programming

Grid-based layouts

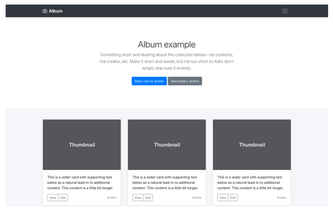
- Established tool for content arrangement
- Gridded content is familiar and easy to follow
- In general, it's good to target fewer lines
  - But breaking that rule is important for creativity and attention-grabbing

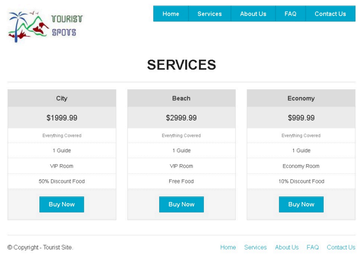


http://printingcode.runemadsen.com/lecture-grid/

Grid-based frameworks

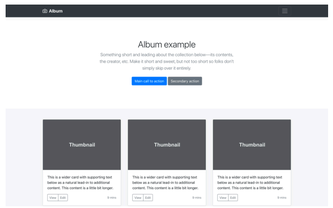
- There are many frameworks which provide grids by default
  - Arrange content into rows/columns, handle spacing, etc.

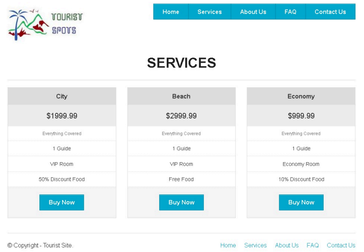




Grid-based frameworks

- Bootstrap (https://getbootstrap.com/)
  - Most popular, most extensions
- Foundation (https://foundation.zurb.com/)
  - Includes icons, drag&drop editor
- Pure.css (https://purecss.io/)
  - Small file size, 3.8KB



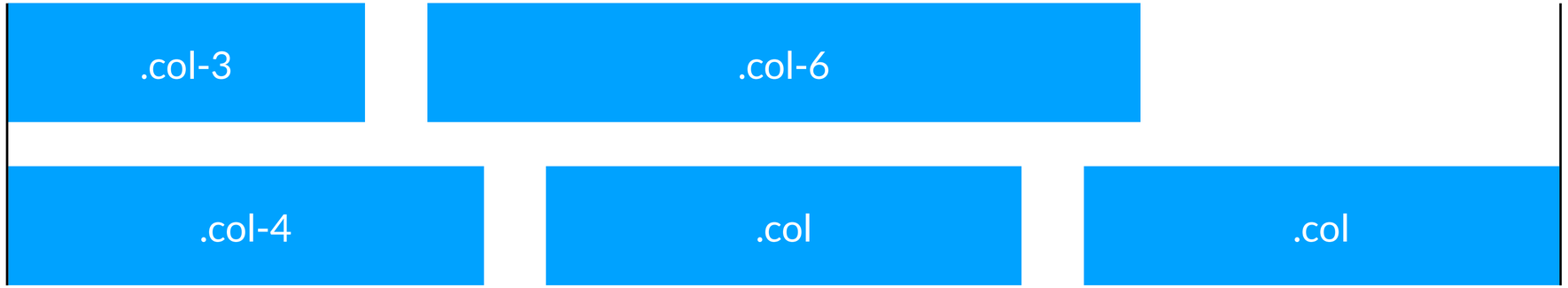


Grid-based frameworks

Example: Bootstrap

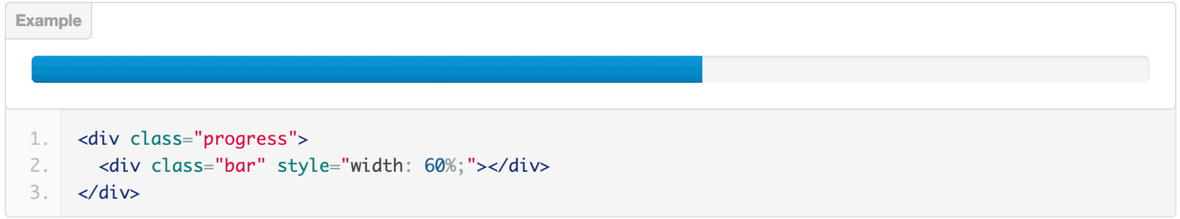
- Within the same row, content will wrap once it goes over 12 columns
    - Size parameter is optional; will divide space proportionally

```
<main class="container">
  <div class="row">
    <div class="col-3">A</div>
    <div class="col-6">B</div>
    <div class="col-4">C</div>
    <div class="col">D</div>
    <div class="col">E</div>
  </div>
</main>
```



Grid-based frameworks

- These frameworks also often provide common interface elements for common design patterns like breadcrumbs, progress bars
  - Reduces development burden
  - But stifles creativity



Native apps

- An app designed to work on a specific piece of hardware
- Usually built with tools created by the hardware or platform manufacturer
  - Android Studio for Android, in Java
  - Xcode for iOS, in Swift or Objective-C

Hybrid apps

- “Use a common code base to deploy native-like apps on a wide range of platforms”
- Two primary approaches:
  - WebView app (essentially, the app is a website)
  - Compiled hybrid app (essentially, the app is converted to a native app)
- I like Ionic, a WebView framework that provides some UI components and access to native resources

ionicframework.com

Hybrid apps vs. native apps

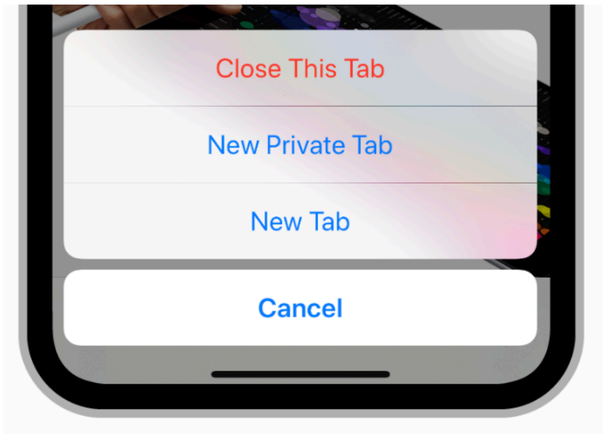
- Hybrid apps are great when time or money is a concern and you need to deploy on multiple platforms
- Native apps are great when performance and consistency with the platform are major concerns

Hybrid apps vs. native apps

- Hybrid apps
  - News sites
  - Informational apps
  - Product showcase
  - Seasonal/one-off
  - Research apps?
- Native apps
  - Games
  - Content-heavy apps
  - Uses a lot of device resources
  - Needs specific OS capabilities

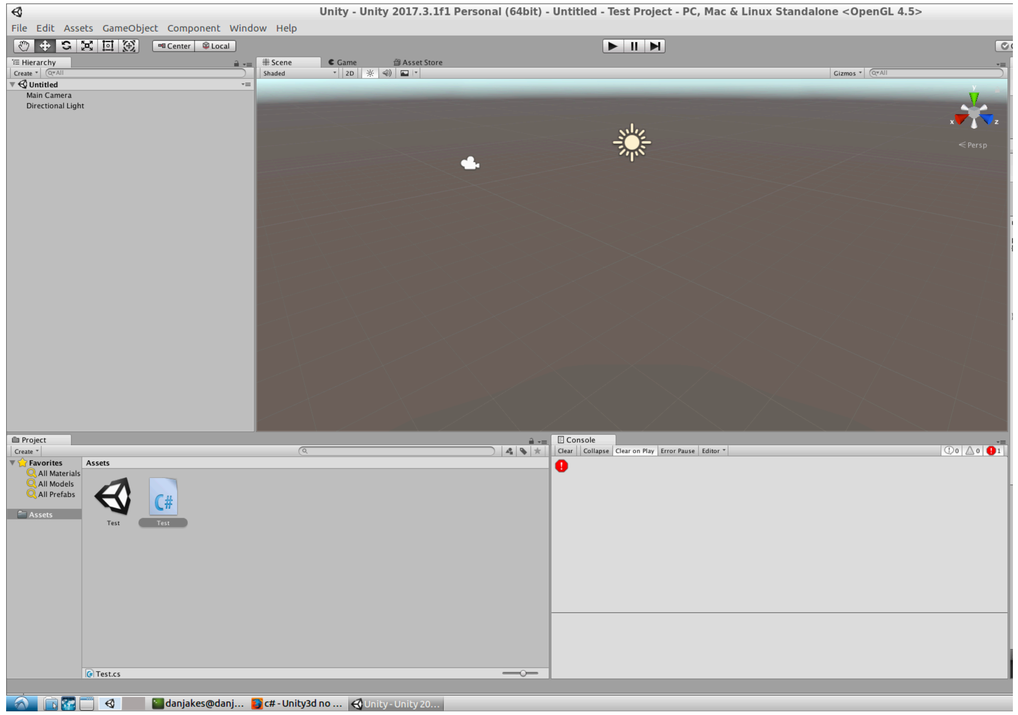
Native and Hybrid apps

- Provide interface elements like pickers and menus
  - These also reduce development burden
  - They’re also nice for platform consistency and familiarity
  - But, they also stifle creativity!



Summary

- To avoid stifling creativity, a tool needs should only render pixels on a screen
  - Game engines like Unity do this
- But supporting creativity is in directly in tension with development burden
  - I wouldn't want to program my personal website a pixel at a time, nevermind a more complex interface



Today’s goals

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

- Describe the concepts of threshold and ceiling in software tools and what tool designers should be striving to create
- Describe why tools eventually limit design thinking

IN4MATX 231:
User Interface Design & Evaluation

Class 19:
Interface development tools

Daniel Epstein