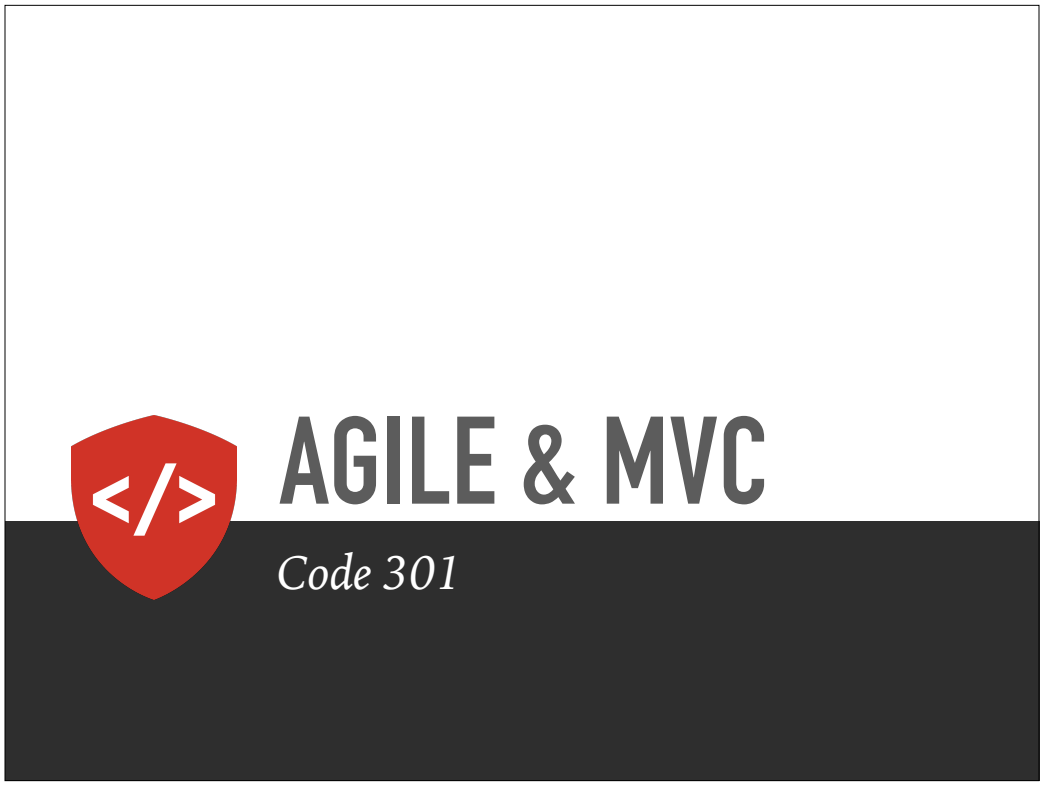
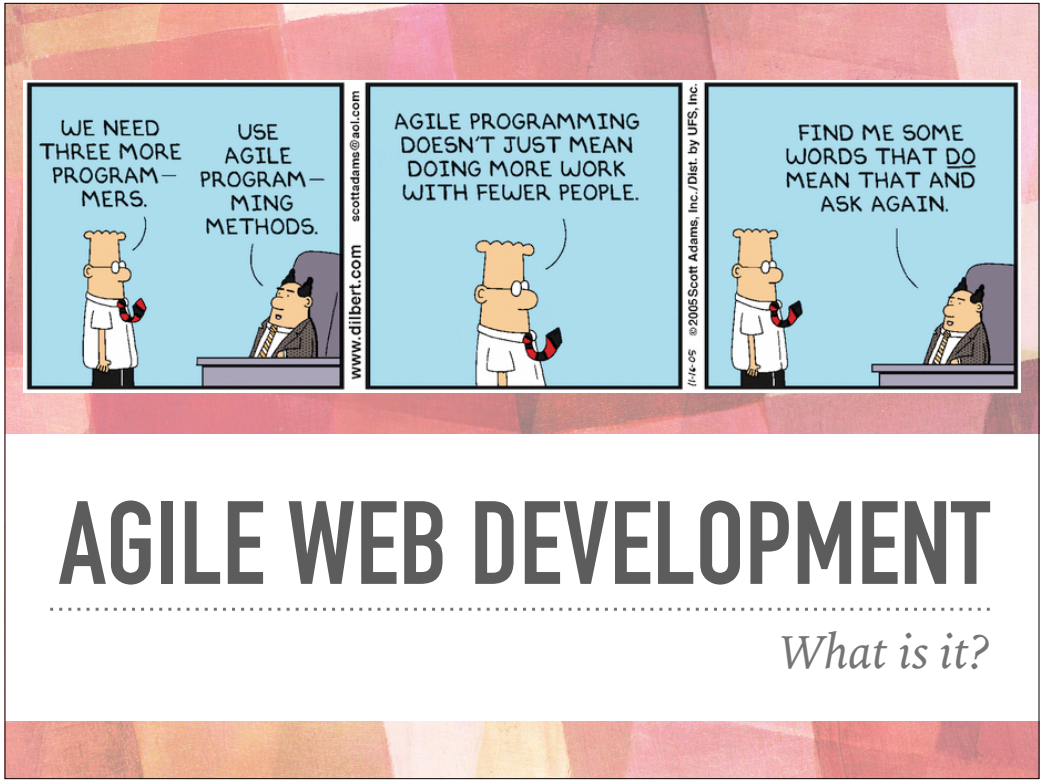


Welcome to class!

AGENDA

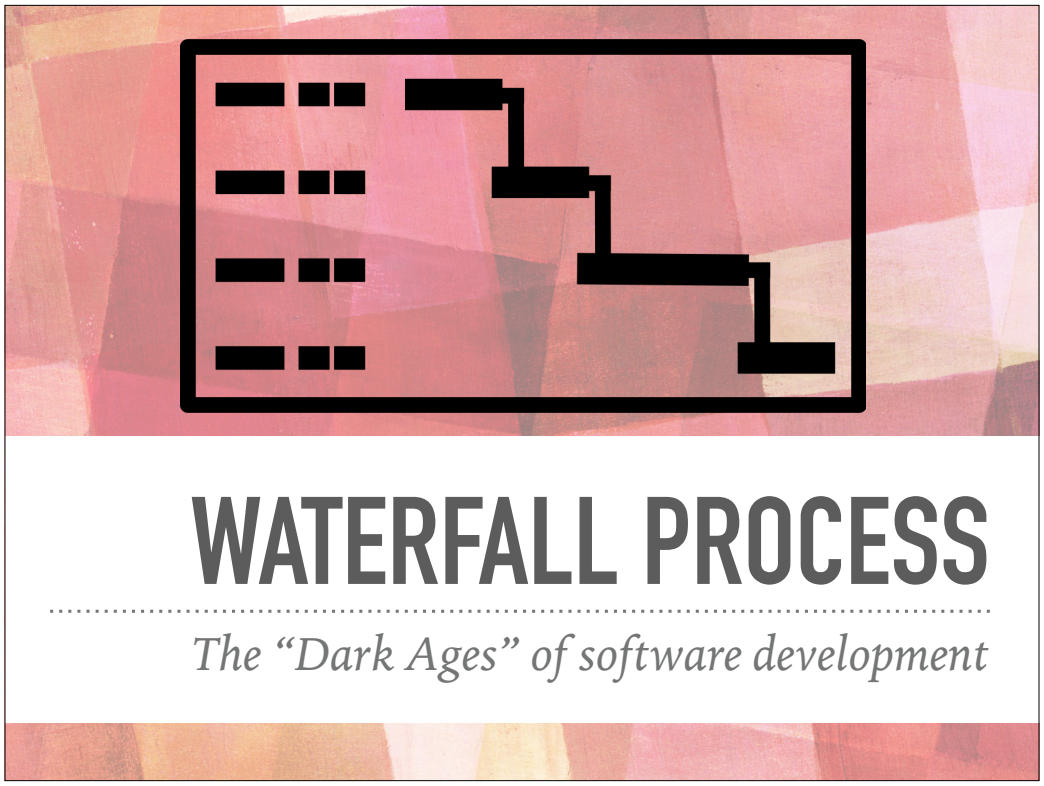
- Agile Web Development
- The Model-View-Controller Pattern
- The DOM
- jQuery

AGILE WEB DEVELOPMENT



What is Agile Software Development?

A methodology for creating great software.



Before Agile, we had “Waterfall”…

Let me describe to you the Dark Ages of software development:

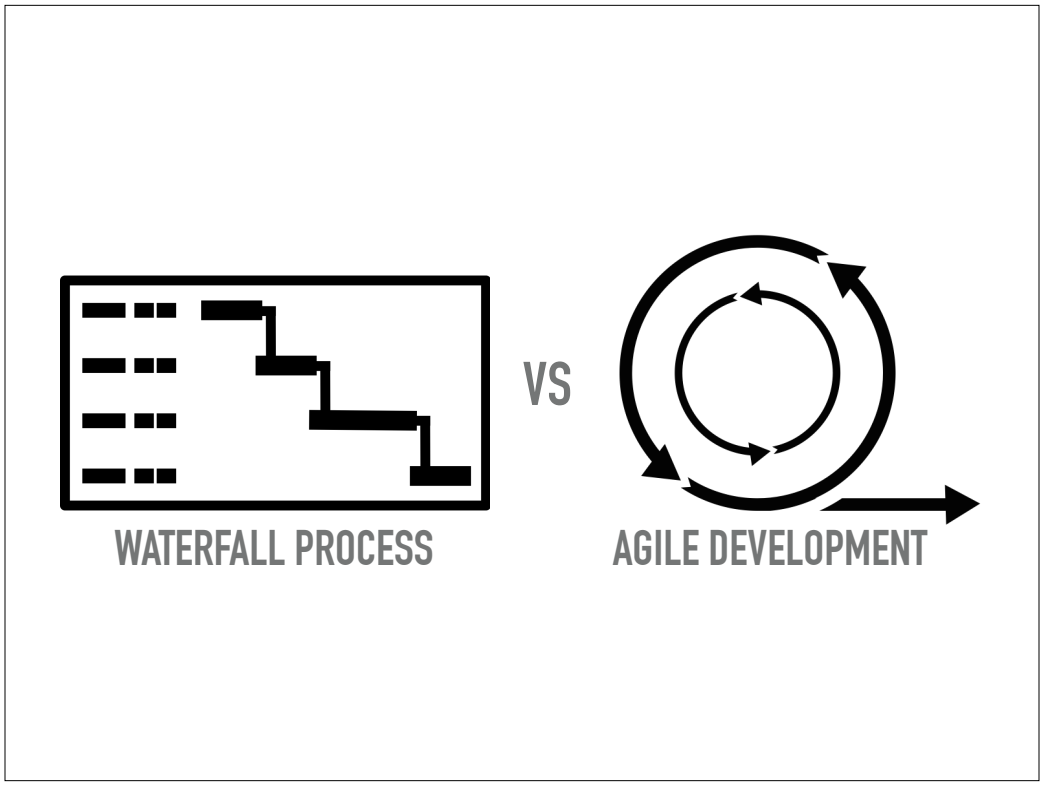
- Business requirements were written into documents 100s of pages long
- Designers worked from the doc to produce static images of the final project
- Developers worked from these “comps” to produce working software
- QA teams took over the source code and worked out all the bugs
- The final project was compiled, burned onto a floppy or CD, wrapped in plastic, put on trucks…
- And eventually placed on shelf stores, where someone might decide to buy it… or the one next to it.

A LIGHT IN THE DARKNESS

- While corporations plodded along with Waterfall…
- New “lightweight” methodologies were explored:
  - Extreme Programming
  - SCRUM
  - DSDM
  - Adaptive Software Development
  - Crystal
  - Feature-Driven Development
  - Pragmatic Programming,

These rebels where united by a single cause: Wanting to ship great software.
Waterfall wasn’t working.
In early 2001, leaders from these movements joined together to codify what they had in common.
The umbrella term “Agile” was born.

> We are uncovering better ways of developing
> software by doing it and helping others do it.
> Through this work we have come to value:
>
> Individuals and interactions over processes and tools
> Working software over comprehensive documentation
> Customer collaboration over contract negotiation
> Responding to change over following a plan
>
> That is, while there is value in the items on
> the right, we value the items on the left more.
>
> — *agilemanifesto.org*



Suddenly, customer feedback could be integrated into the development cycle…
Developers and designers could work side by side…
Small pieces of working software could be independently improved…
Stakeholders could change course based on real-world data…
…And great software became much more achievable, in a much improved time frame.
All thanks to the power of iteration! Kaizen!

AGILE WEB DEVELOPMENT

- Most of what you see and touch of the modern web is produced with some flavor of Agile methodologies
- Our flavor will include:
  - Pair Programming
  - Standup Meetings
  - User Stories grouped into daily Sprints
- More Agile terminology, explained visually:
  - https://www.youtube.com/watch?v=OJflDE6OaSc
  - https://www.youtube.com/watch?v=XU0llRltyFM

We will integrate various aspects of agile into the flow of this course to give you some practice with it. It's implemented differently everywhere, but this should give you a taste of the essence.

AGILE WEB DEVELOPMENT: PAIR PROGRAMMING

- Pair Programming:
  - Driver:
    - hands on the keyboard
    - makes commits to the repo
  - Navigator:
    - verbalizes process
    - researches questions
    - guides driver in where to go
  - Switch off!

We'll be pairing on thurs-friday assignments. But feel free to grab a pair anytime it's needed.
TA's will act as navigators, when you get stuck.

AGILE WEB DEVELOPMENT: STANDUP

- In a Standup Meeting, each person answers 3 questions:
  - What did you do yesterday?
  - What is your plan for today?
  - **What's blocking you?**
    (Don't solve problems, just state where you are stuck)

We do indeed literally remain standing, so that the meeting doesn't go too long. Reporting on progress keeps everyone on the same page. Verbalizing goals helps. Identifying blockers is clutch.

AGILE WEB DEVELOPMENT: STANDUP

- Examples:
  - Yesterday, I added a resume upload feature. Today, I plan to add a quick-preview feature. I'm blocked until the PDF parsing engine is complete.
  - Yesterday, I setup a basic blog index page. Today, I'll add some event handling. No blockers.
  - Y: I modified the search feature to sort by last name.
    T: I'll include phone number in the display.
    B: I'm unsure how to handle accounts with no phone number.

AGILE WEB DEVELOPMENT: USER STORIES

- All User Stories must follow a template:
  - As a _____ [who]
  - I want _____ [what]
  - so that _____ [why]

AGILE WEB DEVELOPMENT: USER STORIES

AGILE WEB DEVELOPMENT: USER STORIES

- A good User Story is:
  - Independent
  - Estimable
  - Small (1-2 hours is our guide)
- User Stories solve real problems:
  - "What do I work on next?"
  - "How do I get started?"
  - "How do I know when this feature is done?"

AGILE WEB DEVELOPMENT: USER STORIES

- Examples:
  - **As a** customer, **I want** the restaurant address displayed prominently on the home page, **so that** I can quickly find it if I look it up on my phone while driving.
  - As a **Teller**, I want **to look up clients by last name**, so that I can **find them in the system faster**.
  - As a player, I want to toggle help/instructions during play, so that _____.
  - As a hiring manager, I want to give resumes 1-3 stars, so that _____.

Here you can see the structure. It's what gets filled in the blanks that really matters.
What do you think the missing "so that" part is?
#3: "I can pull them up when needed"
#4: "I can glance at them once and and track which ones I need to come back to"

AGILE WEB DEVELOPMENT: USER STORIES

- Your user stories for today:
  - As a developer, I want my site to use valid and semantic markup, so that employers will love me.
  - As a reader, I want the blog to show most recent articles on top so that I can easily read the latest.
  - As a reader, I want relative timestamps to give me a idea of how many days ago something was written.
  - As an author, I want my name linked to my social feed, so that readers can follow me, and I can build my audience.
  - As a reader, I want the site to look reasonable, so that I can read it on any device.

Here's the stories you'll be working on today, for your new blog. :]

AGILE WEB DEVELOPMENT: SUMMARY

- We'll leverage a few key pieces of Agile in Code 301:
  - Pairing
  - Standup Meetings
  - User Stories

AGILE WEB DEVELOPMENT: RESOURCES

- Check out these videos:
  - https://www.youtube.com/watch?v=OJflDE6OaSc
  - https://www.youtube.com/watch?v=XU0llRltyFM
- Read more how the movement started:
  - http://agilemanifesto.org/

MVC

MVC

- "Design pattern" for organizing your code
- **Model**: Source material or data
- **View**: Presentation scheme
- **Controller***: Behavior management

- * the pattern has other incarnations: MVP, MVVM, MVT, etc…

What do I mean, when I say "Design Pattern"? It means this captures the "core of the solution".

MVC

- MVC is everywhere
- The modern web follows an MVC-like pattern:
  - Model: Source material or data
    - HTML
  - View: Presentation scheme
    - CSS
  - Controller: Behavior management
    - JavaScript

Guess what? You already know MVC!
This is a fairly natural separation, that you'll come to see all around you. In fact, in a way, you've already been doing this in your code.

MVC

- MVC is everywhere
- At the theater:
  - Model: Source material or data
    - Actresses and actors
    - The story
  - View: Presentation scheme
    - Rehearsed dramatic retelling of the story
  - Controller: Behavior management
    - Ticket booth, ushers, post-play Q&A

A play takes the raw material of story & actresses and actors, presents through the rehearsed production, controlled by the ticket booth

MVC

- MVC is everywhere
- What examples can you think of?

Think about it for a moment. Where else do you see the MVC pattern around you?

MVC

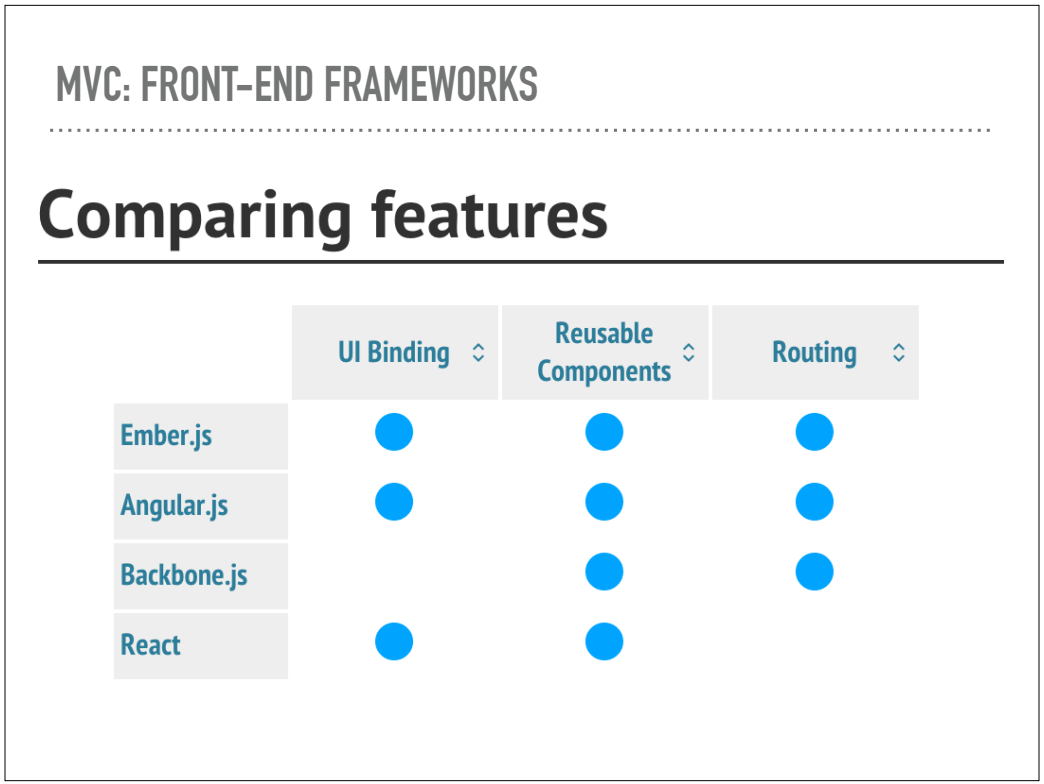
- Benefits
  - Code at a higher level of abstraction
  - Organize your code base
  - Leverage best practices
- Challenges
  - Learning curve
  - Inter-file spaghetti code
  - Hard to extend when the metaphor fails

Your application becomes a set of API methods, that provide a layer of abstraction over common tasks. You write the language of your problem domain into your code. OOP defines how to set up your objects, MVC defines where you objects live, and what you can do with them.

Code stays organized: You know where to go, when you need to fix a problem or add a feature. Disciplined following of the constraints makes code that is scalable.

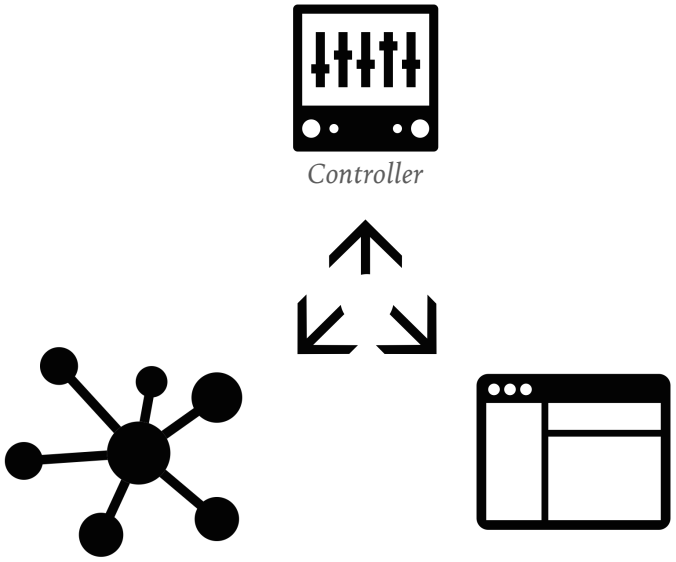
Lots of good OSS work has been done to build frameworks according to these patterns.

But it takes a little getting used to. You'll have things spread between files, so get good at jumping around in Atom. If you build a big app, eventually there will be something you want to do that doesn't fit in the MVC pattern you are using.



UI Binding: How data is shown to the user (VIEW)
Components: Small independent objects (MODEL)
Routes: What urls an app is able to respond to (CONTROLLER)
In the JS world, you MVC means different things to different folks, so even the most popular frameworks don't have a strict MVC approach.

MVC FLOW



MVC DEMO

Basic web page code structure

We'll use the MVC paradigm to build a blog.
DEMO: What would MVC look like in a blog? Discuss!
- Draw wireframe of blog page on whiteboard…
- index.html file with basic structure, including a template for what an article will look like
- presentation in CSS
- actions in a js file: blog.js, and article.js (which has constructor)

MVC

- Follow web page MVC patterns to get started
- Create a standard file structure for your files
- We'll dig deeper into the View this week, starting right now, with jQuery

RECAP

RECAP

- Agile
  - Power of fast iteration makes better software
  - Vocab words:
    - Standup, User Stories, Backlog, Sprint
- MVC
  - Structure your code for fun and profit
  - Follow best practices for greater interoperability
  - Every MVC is a little different

RESOURCES AND ATTRIBUTIONS

- Vectors from Noun Project:
  - https://thenounproject.com/search/?q=agile&i=204943
  - https://thenounproject.com/search/?q=gantt&i=6299
- http://dilbert.com/strip/2005-11-16
- http://dilbert.com/strip/2003-01-10
- http://www.developereconomics.com/feature-comparison-of-4-popular-js-mv-frameworks/