1) **Intro**
    a) Welcome!
    b) Animations are here for fun, but also to illustrate that one of the things we do as developers is think about time. And what happens at various points in time.
        i) Graphic artists concern themselves with this less
2) **Psychology of the industry**
    a) Enjoy the ride
        i) This industry is, to paraphrase Shakespeare, a never fixéd mark.
            My advice is to enjoy the roller coaster - don't think of yourself as perfecting "a" language but perfecting your ability to perfect a language.
    b) Parallel techs - the "new new" is not all there is.
        i) For example, when node first started reddit announced PHP's death - this never happened -- (Mark Twain quote "the reports of my death are greatly exaggerated.")
            A lot of techs are running parallel to one another - the "new new" just becomes a new thread in the tapestry.
            Be wary of internet opinions. They are worth what you paid for them.
        ii) Yes, React is huge, but Ruby on Rails, Vue, and Angular are still very viable.
        iii) Exact same thing happened when Mongo started.
    c) CareerFoundry is teaching you both substantive and procedural knowledge:
        i) Substantive:
            React, React Native, Node are all huge employment opportunities.
        ii) Procedural
            Learning how to learn languages is essential,
                (a) As discussed above, your future work may be in a tech that you haven't heard of yet.
                (b) Once again, enjoy the ride
        iii) Pretend your mentors are your supervisors
    d) Opinion: aim for Hospitality industry level customer service
        i) Cleaning up other people's messes is a huge $$ maker
        ii) Clients are infinitely grateful for this
3) **Importance of coding challenges**
    a) Your resume, portfolio, and cover letter will only get you through the door, the live-coding challenges get you the job
    b) During the challenge:
        i) Anxiety can cause your IQ to plummet
        ii) Food, coffee, hour of the day -- huge factors in your ability to think clearly.
        iii) Always map out your logic and verbally describe your process
    c) Start by doing one a day. Don't panic. Each one you do will help you with the next. There are patterns.
    d) Continuing to do this throughout your career is, imo, a really good thing

i) No matter what language you code them in, it will help
ii) In your team, it is a really good feeling to be the one that has seen so many repeated patterns that you have valuable insight to give=

e) Average job length in this industry isn't long
i) https://hackerlife.co/blog/san-francisco-large-corporation-employee-tenure
ii) Be prepared to interview again

4) Continuing education

a) An ugly truth is that your employer may not be looking out for your long term employability -- you have to be your own champion.
b) Look at job boards for languages that are currently being used.
i) If the job looks like one you'd like to eventually have, learn it.
c) Mozilla css videos -- highly recommend
d) Always be aware of new versions of languages coming out -- THESE ARE PLAYING FIELD EQUALIZERS!
i) AS3

5) **Ugly but true: Scam Employer**s

a) Be skeptical. There are clients out there that won't pay and there are companies out there engaged in fraud.
b) From my experience, these manifest themselves in three ways:
i) They're doing something illegal and have the money to pay you but you could be liable.
Pyramid company
ii) They're scamming clients by promising future work to be done by you
Heritage
Ashland Coding Company
iii) They are hustlers trying to make it. I think this category can kind of go either way but it is a gamble
San francisco company who "ran out of checks"
c) Lastly: Please please please: always get paid for your work.

6) **Finally practices that took me too long to incorporate**:

a) Never show clients unfinished work
b) Tack % onto estimates
c) Consider doing css first
i) All endpoint clients care about is what it looks like
d) Beware underbilling to impress clients
i) Will set up false expectations
ii) Undermine long term relationship
e) Always always always always include steps for replication when asking for help.
f) If you have a huge application and a piece of it doesn't work--
i) pull it out -- solve it in isolation and the integrate it
g) Reverse engineering
i) Jquery
ii) Bootstrap
h) When an artist gives you a wireframe --- it isn't a suggestion, it is law

i) COUNT PIXELS

i) If you are completely stuck:
   i) bring it back to a moment where it worked and build it back piece by piece.
   ii) If need be, start from scratch in one window with the broken code in another window and move everything over piece by piece.

j) Github --
   i) If you have to do something complicated with github (reversions for example) and you are not comfortable with this:
      Make a dummy repo that mirrors the structure of your project
      Test, test, test on the dummy repo

k) Avoid solution spewing -- never underestimate a calm mental state
   i) “Maybe I’ll try this, maybe I’ll try this, maybe I’ll try this”
      (a) Very common with css
      (b) This almost never works out to be an efficient use of time
      (c) Even if this works, you may not understand why it works

l) Do not be intimidated by error messaging or server logs
   i) Error messaging will ofter reflect the stacks of the origin of the issue and many of the lines will be code from the node_modules directory, keep reading them until you get to the actual issue
   ii) Server logs are your friends

m) If you don’t know something
   i) Please say “I don’t know”
   ii) Identify guesses as guesses

n)

o) Ghosts in the machine
   i) 99% of the time there is a logical code-based reason why something is going wrong BUT corrupted files do exist.
   ii) So never start with the assumption that there is a ghost in the machine. But always remember that it is a possibility.

p) Extra:
   i) Don’t forget OG images
   ii) Common coding errors:
      (1) Always be thinking mutation
         (a) Arrays
      (2) Be careful with “this”

   iii) Unexpected behavior:
      (1) isNaN
      (2) Be careful with shortcuts (++)

7) Bonus:
   a) Toughest interview questions;
      i) Write a recursive fibonacci sequence
      ii) Define restful architecture

iii) What are the four principals of OOP
iv) Discuss your thought process on whether to put code client side or server side
(1) This one not so much hard but interesting

b) Shallower
i) CMSs

c) Deeper
i) Object pooling
ii) Recursive functions
iii) Permutation
iv) Maze Handling
v) Big “O” notation
vi) Sorting Algorithms
vii) Searching Algorithms
viii) Stacks, queues
ix) Hash maps
x) (super fun) code your own tweeting engine

d) Final note:
i) Remember all you have to be is smart enough in the moment
(1) Geodesic dome emulator
ii) Self-presentation
(1) A lot of this is rooted in bigotry, so let me just say

Exhibits:

1) Codepen with ++ examples
2) Mdn array methods
3) Button example with the keyword “this”
4) OG images link
5) OG images summary