Declared functions-adv debugging and recursion. **anonymous**-adv scope and brevity

Asynchronous-XMLHTTPRequest(old) fetch(new). Fetch is promise based

Designthinking-empathize,define,ideate,prototype,test,implement

Principles-focal point, contrast,balance,movement,rhythm,perspective,unity

Design-Metaphorical->follow real world metaphor ie bookshelf, nest thermostat. Idiomatic-building expressive interaction that users must learn. ie mapping cursor mvmnt on screen

-idioms-Primitives(atomic acts ie point/click) Compounds(complx acts, doubleClick) Idioms(high lv ie deleting text)

Imperative-expresses computation flow-> programming how to get outcome we want

declarative-expresses logic of computation-> programming what we want outcome to be

Controlled- user input elements(input, textarea) states managed by react component.

Uncontrolled-user input elements(input, textarea) use a ref() to get form values from DOM

Gestalt->similarity, similar obejcts create a connection to viewer. Closure-aligned elements are perceived connected/fill in. Continuity-incomp object feels complete. Prox-objects close togeter

Nav Principles->

1. Wayfinding-labels and signage, navigation clues and aids, maps/site maps
2. Cost-time/effort required by user. Minimize num steps/switches and heavy costs
3. Aids-global(menus,tabs) utility(settings pane) associative/inline(related links)
4. Models-commnly used patterns of nav. Hub and spoke, fully connected, multilevel, stepwise, pyramid, pan/zoom, flat nav, modal panel, clear entry points, book marks

React-immutable JS objects. Browser independent until they are rendered

-**PureComponent**->imps shouldCOmponentUpdate(), diffs and updates when rtrn true

"**Thinking in React**"- 1. Mock up Design 2. Break UI into component hierarchy 3. Build a static version 4. Identify min set of mutable state 5. Identify where state should live 6. Add inverse data flow

-**Functional components**-easier to read,write,debug. Code modularity, more efficient

-**mounting**-process of creating instance of component and inserting into DOM

-Render(no state updates, does not interact w/Browser) constructor(inherit props, initialize state, bind functions) componentDidMount(automatically called following render. Initiate API calls, request data, before browser is updated)

-updating-rerendering a component following changes to props or state. componentDidUpdate(called in an update)

-**unmounting-**rem. component fromDOM. ComponentWIllUNmount(invoked as soon as a component is unmounted. Reset counters, canceling network requests. Do not use SetState, will not be called)

Navigators- const s =createStackNavigator(). <S.Navigator>

1. **Switch Navi-** Enable show one screen at time, no "back" actions, authentication flows
2. **Stack Nav-** screens placed on stack, imp. native transition animations, back and forth between list & detail views or walk through process
3. **Tab Nav-** tabs at bottom or top, enable transitions
4. **Drawer Nav-** enables tab-like transitions through hidden drawer that exp/col. Mainly used with options and settings

-Each screen is automatically provided with a navigation prop

-Nav Actions- common-navigate, reset, goBack, SetParams stackactions-replace, push, pop, popToTop tabactions- jumpTo drawer actions-openDrawer, close Drawer, toggleDrawer, JumpTO

-useScrollToTop-automatically scrolls to top when navigated to

Gestures

-**panResponder**-reconcile several touches into single gesture which is used to recognize multi-touch gestures

-**animated**- sequence, spring(att w/o set time in motion), interpolate(maps input range to output range), easing(gradual acceleration or deceleration)

-**layoutAnimation**-animates entire screen when changes in the layout. used before setState()

-**Animated**-animates specific components without changing the layout

-Date-th,nov,7 vs date JSON.stringify(date)->2019-11-07T17:38

Microinteractions- contained product movements for single use case ie flick noti to dismiss, pull to refresh,

1. Trigger-Initiate microInteraction. Man-or-auto. Ex. Man-Flipping switch, pressing button. Auto-chime when message arrives, swoosh when email is sent
2. Rules-determine what happens/doesn't happen when microinteration is triggered
3. Feedback-info user sees, hears, feels are forms of feedback
4. Loops/modes- loops,->length of interation and whether it repeats ie bleeping fridge door. Modes-switch functioning or operation of system ie do not disturb

-Minimize scope of microinteration, supposed to support single task, single action. They are action feedback pairs for a single purpose

Patterns- s

1. Stacks-vertically organize design elements s.a. toolbar,content panes, nav bar
2. Screen Carousel- horizontal array display diff instances of same info. s.a. weather
3. Drawers- provide links for nav or controls for various settings of app
4. LISTS-involve vertically stacking large num of items. GRIDS-large num continuous panes of grids that can scroll through vert or horiz. Grids more common
5. Carousels-row of content items including images or text cards by swiping L or R
6. Swimlanes-stacked content carousels that each show row of items IE NETFLIX display
7. Card- similar to lists and grids, but put together diff comps of content. IE image, text
8. Bars- tab-placed at top, bottom, side. Toolbars,
9. Search,sort,filter- enable search and filtering to nav. Through large body of content
10. Landing pages, guided tours-landing, welcome, or home screen that serve as portal

Advanced Direct manip.-apps enable direct manipulation based controls for content creation

Panes and panels-multi-pane structures and pop-up panels and tools

AR/VR-overlay objects or info on images or video of users real environment

Prototyping-mock up/early design ideas by sketching pages/screens and simulating application

-wireframes-early to mid stage. lo-fi proto. of struct comps. Handwritten OR digital.

-annotations-key in addressing early proto. prov further info on content, functioning

-interactive prot.-realistic prot. Of nav or structural components of design idea. Uses series of screens/pages with design elements Can use lo-fi or hi-fi components

-native prot.-implementing and testing design ideas on target tech platform

-3 dimensions-

1. Role-Represents functions that system serves in users life

2. Look and feel-sims sensory experience while using the system. See, hear feel

3. implementation-includes tech-capabilities that enable syst to perform its func

COMBINES-represents complete user experience envisioned in concept design

-**scope-**consider space of features and functioning as everything that a system does

-Horizontal prot-provides a broad view of entire system. Focuses on user interaction rather than functionality

-vertical-focuses on single feature, provides full functioning of that feature

-**Prototyping strategies**

Throwaway-rapid prot to explore design ideas, demonstrate feasibility,

Evolutionary/breadboard-incrementally built prot of a design idea. Test idea wit users, refine prototype until it reaches maturity

Incremental-dividing system functionality into slices based on specs. For large and complex projects. Incrementally building each slice

Extreme-3phases- 1. Build static prot. 2. Build fully functional prot. 3. Implement

-**Fidelity**- level of detail in prototype. Lo/hi-fi. The more "done", narrower the feedback

-low-fi-> lower cost and smaller number of low-level specifications

Accessibility(aria)-Properties

1. Accessible- whether component is an accessible element, if so, groups children into single selectable component
2. AccessibilityLabel- attribute defines screen reader descriptions of components
3. accessibilityHint-hints what will happen if user performs action on component

AsyncStorage-simple, unencrypted key-value storage system. Returns a promise

1. Simple-core functionality involves set and get methods
2. Unencrypted-access is controlled by location access
3. Persistent: data is saved until it is explicitly deleted
4. Global: saved data is global to the app

-Saving-. AsyncStorage.setItem('key', 'value') retrieving .getItem('key')

Sensors-have listeners, best practice is to create subscribe and unsubscribe functions

AppState-provides info onnthe current state of the app

-active- indicates the app is running in the foreground

-background-indicates that the app is running in the background

-inactive-indicates the app is transitioning between foreground and background

Accessibility/Disability->Impairment, limitation in activities, restrictions in participation

1. Equitable use-design is useful and marketable to people with diversie disabilities
2. Flexibility-design accommodates a wide range of individual preferences and abilities
3. Simple and intuitive-easy to understand, regardless of users exp, knowledge, skills
4. Perceptible-communicates necessary info effectively to the user

5. Tolerance for error-design minimizes hazards and adverse consequences
6. Low physical effort-design can be used efficiently and comfortably w/ min fatigue
7. Size and space for approach and use-approp size and space is provided for approach

Principles of Design

Gricean Maxims-**Max of quality**(truthful & accurate communication**). Max of quantity(**just the right amount of information) **Max of relevance**(appropriate & relevant information**) Max of manner**(clear, cooperative communication)

Multimodality-multimodal interfaces utilitze multiple modalities like speech, touch, etc

Interaction paradigm-interfaces follow different paradigms. Based on context. **Command and control**(always on voice assistant, speech input is mapped to specific system functions ie "okay google) vs **conversational interfaces**(chatbots, interaction with system has the characteristics of human conversations)

Turn-taking-one speaker at a time, turn exchanges, interruption handling

Conversational markers-speech cues, "first" "thanks" "good job"

Confirmations-implicit(letting user know what was understood"ok, setting reminder") vs explicit(requiring user to confirm("I think you want to set reminder")

Heuristics-1. **Visibility of system Status**-system should inform users about sys status

2. **Match between sys & real world**- sys should speak users language, follow real world conventions, ie credit card display on credit card app

3. **User Control/freedom**-emergency exit to leave unwanted state 4. **Consistency and standards** 5. **Eror Prevention** 6. **Recognition rather than recall** 7.**Flexibility and efficiency of use**-accelerators for advanced users. Prov. Customiz

8. **Minimalist design** 9. **Help users recognize, diagnose, recover from errors**. Ie error mssges should be expressed in plain language. 10. **Help and Docmntation**

Experience prototyping-1. Define context 2. Develop scenarios 3. Identify design goals 4. Set up the environment 5. Act out interaction/Bodystorming 6. Develop insight

DialogFlow-development suite for conversational interfaces

-agent-virtual agent that handles conversations with users

-intents-goals of the user that are expressed to the agent

-entities-some users intents are speficit and based around an entity