# Day 2 Review

Daily Standup

1. Something you learned yesterday.

2. What you're looking forward to learning today.

3. Blockers. (confusion, questions, etc).

JavaScript

-- const and let

-- arrow function ()=>{}

const myFunction = (param1, number)=> {

let param2 = param1 + 1;

return param2;

}

-- rest and spread operators (...)

myFunction(x,y,…arg){

// x === 4

// y ===2

// arg === [6,4,3]

}

myFuction(4,2,6,4,3);

let props = {prop1:'test',prop2:'test2'}

myFunc(...props)

const [thing1, thing2] = things

const {prop1,prop2} = props;

===

const prop1 = props.prop1;

const prop2 = props.prop2;

-- template literals ` ${props.prop1}`

-- method notation

const theObj = {

doSomething () {

...

}

}

-- array and object destructuring

const [a,b] = [1,2,3]

const {name,address} = myArray;

-- Class definition

class MyClass extends React.Component{

constructor(params){

super(params);

}

...

}

-- understanding 'this'

-- call (invokes a function with a context, pass args invididually)

logName(role){

return this.name

}

const Person = {

name: 'Chris'

}

logName.call(Person, role);

-- apply (invokes a function with a context, pass arg in array)

logName.apply(Person,roles);

-- bind (returns a function with a context)

const makeChrisAdmin = logName.bind(Person,'admin');

makeChrisAdmin()

function processData(){

return (this.name + " is now processed.")

}

const myObj = {

name: 'the data',

data: [3,5,3]

}

processData.call(myObj);

const processMyObj = processData.bind(myObj);

processMyObj();

-- array.map - runs a function for each element, returns array

-- array.filter - runs a test for each element, returns array

-- array.reduce - runs a reducer function, returns a single value

-- JavaScript modules

import, export

1. named

export const PI = 3.14;

import {PI} from './file.js';

import {Component} from 'react';

2. default

function myFunc(){}

export default myFunc;

import myFunc from './myFunc';

import importantFunction from './myFunc';

HTML DOM

classes

fetch

Promises

event Listeners

DOM methods

One-way data flow

React Props

props.children

<MyComponent>

<Product />

</MyComponent>

return {props.children}

React State

Setting Initial State

constructor(){

this.state = {}

}

Updating State

--this.setState - merges objects (class)

--useState hook - replaces the old state (function)

const [statefulValue,setStatefulValue] = useState(init);

const stateArray = useState(init);

const statefulValue = stateArray[0];

const setStatefulValue = stateArray[1];

Setting state that depends on previous state

this.setState((current)=>{message: current.message + "!"})

this.setState((current)=>{message: current.message + "!"})

this.setState((current)=>{message: current.message + "!"})

What to put in state

-- data that causes changes

-- user input

-- timer

-- changing data from API

Where to put state

-- in a component that's a common parent to every component that needs that state

App

|

Main????

|

ProductList - Cart

Shallow copy of arrays and objects

currentState = [1,2,3]

const newState = [...currentState, 4]

setState(newState)

class components

class React.Component {

constructor(){

this.state = {}

}

setState(objectToMerge){

this.state = {...this.state, ...objectToMerge};

render();

}

}

class MyComponent extends React.Component {

constructor(){

this.state = {

message: 'Welcome'

}

}

translateMessage(lang){

const translation = googleTranslate(this.state.message, lang);

this.setState({message: translation});

}

render(){

{this.state.message}

}

}