Code Fellows

Class 21 + DSA 04

Sorting, Routing and Component Composition

seattle-javascript-401n14

Holiday Break!

Holiday Break

Monday DEC 23 2019 Co-working	~~Tuesday DEC 24 2019 Lecture~~	~~Wednesday DEC 25 2019 Lab~~	Thursday DEC 26 2019 Co-working	Friday DEC 27 2019	~~Saturday DEC 28 2019 Lecture + Lab~~	Sunday DEC 29 2019
Monday DEC 30 2019 Co-working	~~Tuesday DEC 31 2019 Lecture~~	~~Wednesday JAN 01 2020 Lab~~	Thursday JAN 02 2020 Co-working	Friday JAN 03 2020	Saturday JAN 04 2020 Lecture + Lab	Sunday JAN 05 2020

Lab 20 Review

Code Challenge 20 Review

Vocab Review!

What is a prop?

What is a state variable?

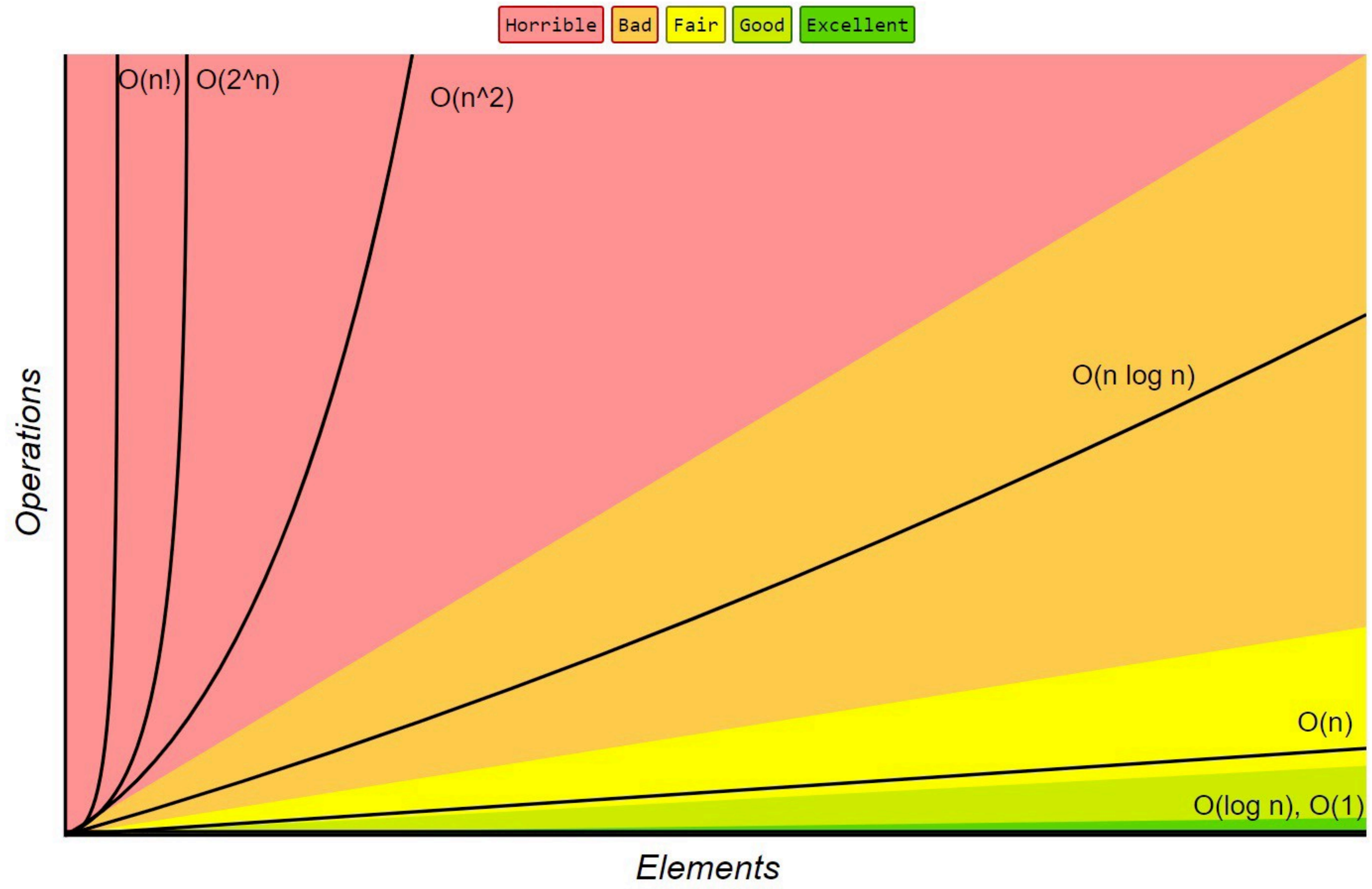
What does parent and child component mean?

What is recursion?

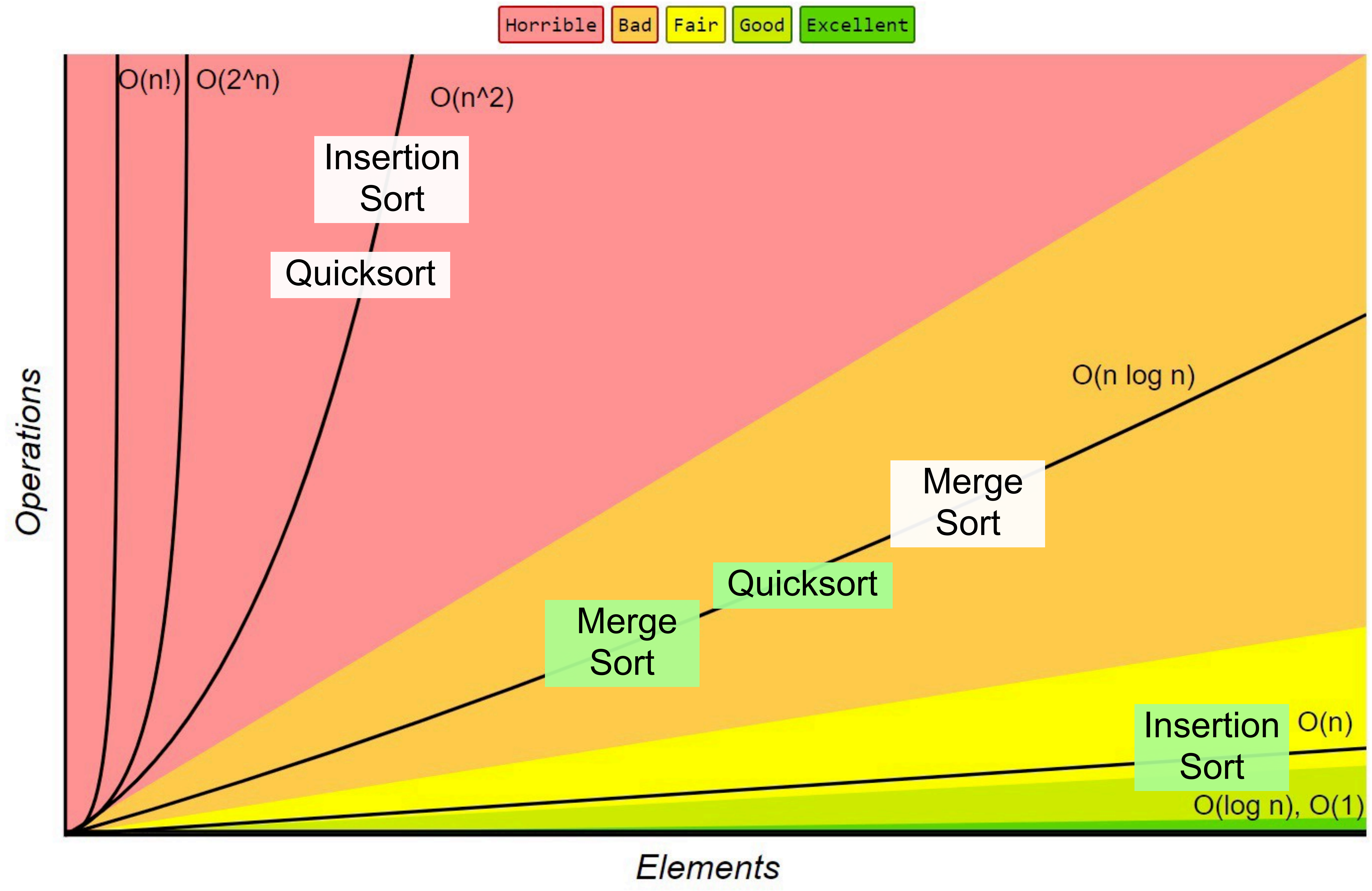
What is a recursive base case?

What is big-o?

Big-O Complexity Chart



Big-O Complexity Chart



Quicksort

- Step 1 - Pick an item in the array as the **pivot**
- Step 2 - Swap the pivot with the item at the end of the array
- Step 3 - Find the **first item larger** than the pivot, searching left to right
- Step 4 - Find the **first item smaller** than the pivot, searching right to left
- Step 5 - Swap these **two values**
- Repeat Step 3-5, until **first item larger** is at a greater index than **first item smaller**, or out of bounds
- Step 6 - Swap the last found **item larger than pivot** with the **pivot**
- Step 7 - The pivot should now be in the correct position. Recursively repeat this with the left half of the array before the pivot, and the right half of the array after the pivot

0	1	2	3	4
E	D	B	C	A

Step 1: Pick a Pivot

0	1	2	3	4
E	D	B	C	A

pivot = 3

Step 2: Swap the pivot with the last item in the array

0	1	2	3	4
E	D	B	A	C

pivot = 4

Step 3: Find the first item larger than the pivot, left → right

0	1	2	3	4
E	D	B	A	C

fil = 0 pivot = 4

Step 4: Find the first item smaller than the pivot, right → left

0	1	2	3	4
E	D	B	A	C

fil = 0 pivot = 4 fis = 3

Step 5: Swap the two values

0	1	2	3	4
A	D	B	E	C

pivot = 4

Repeat steps 3 - 5 until fil > fis

0	1	2	3	4
A	D	B	E	C

pivot = 4 fil = 1 fis = 2

0	1	2	3	4
A	B	D	E	C

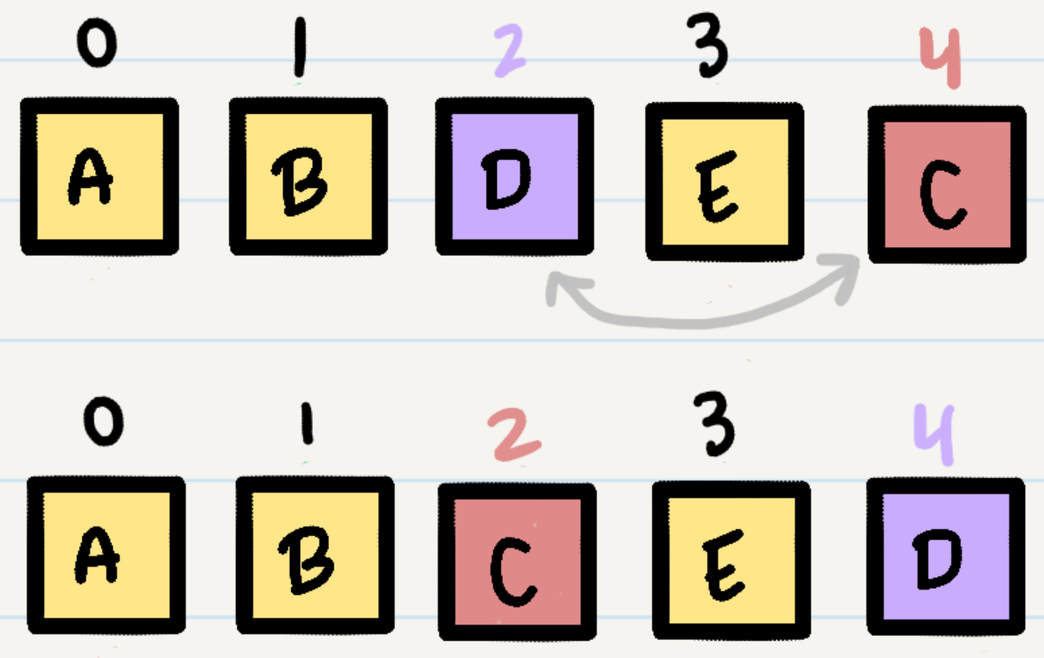
swap!

0	1	2	3	4
A	B	D	E	C

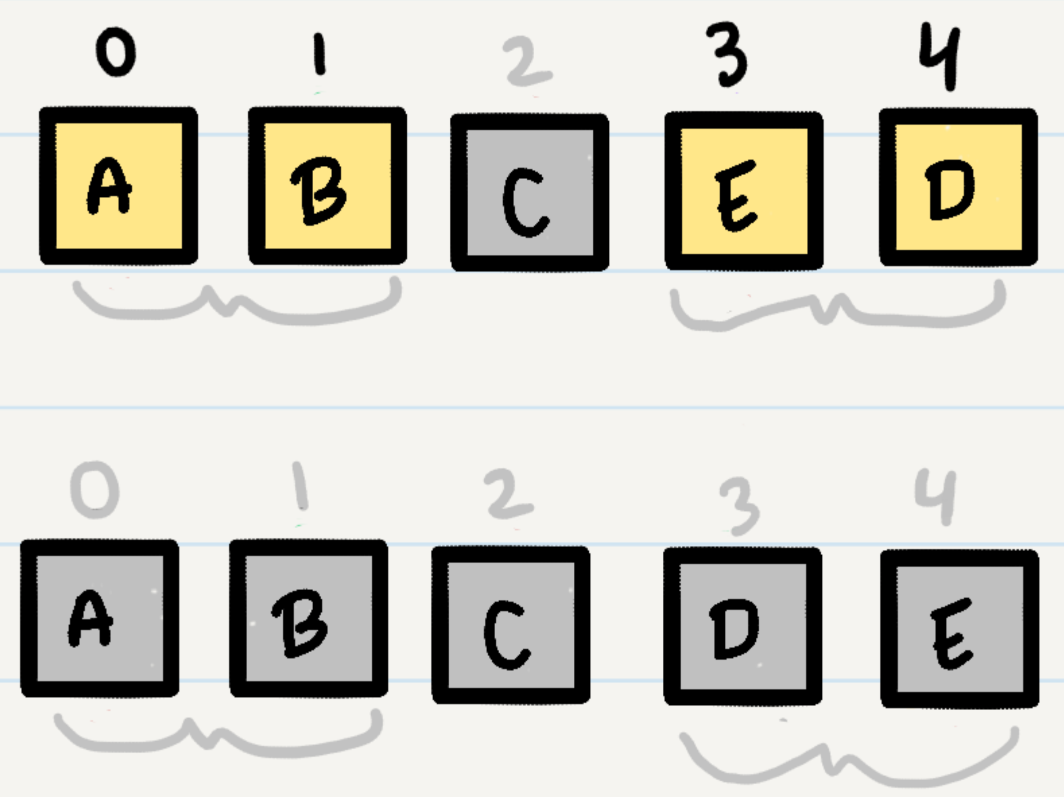
now, fil = 2 fis = 1

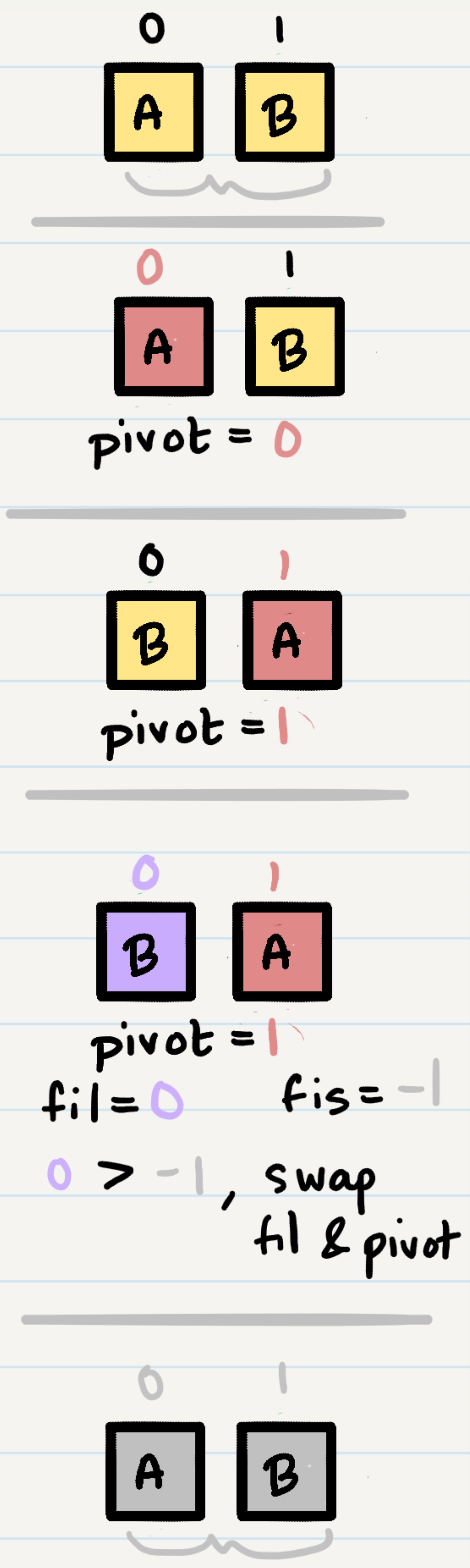
fil > fis!
2 > 1

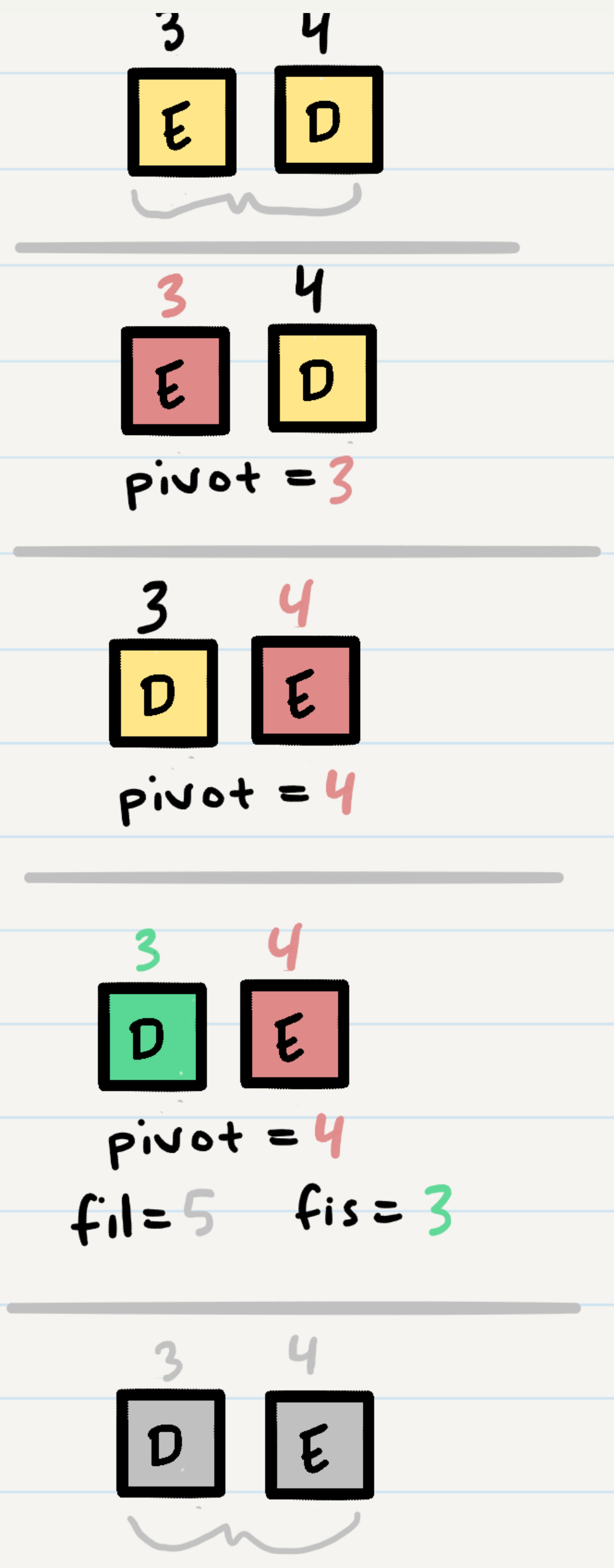
Step 6: Swap the last found item larger than pivot with the pivot



Step 7: The pivot should now be in the correct position.
Recursively repeat with each partition







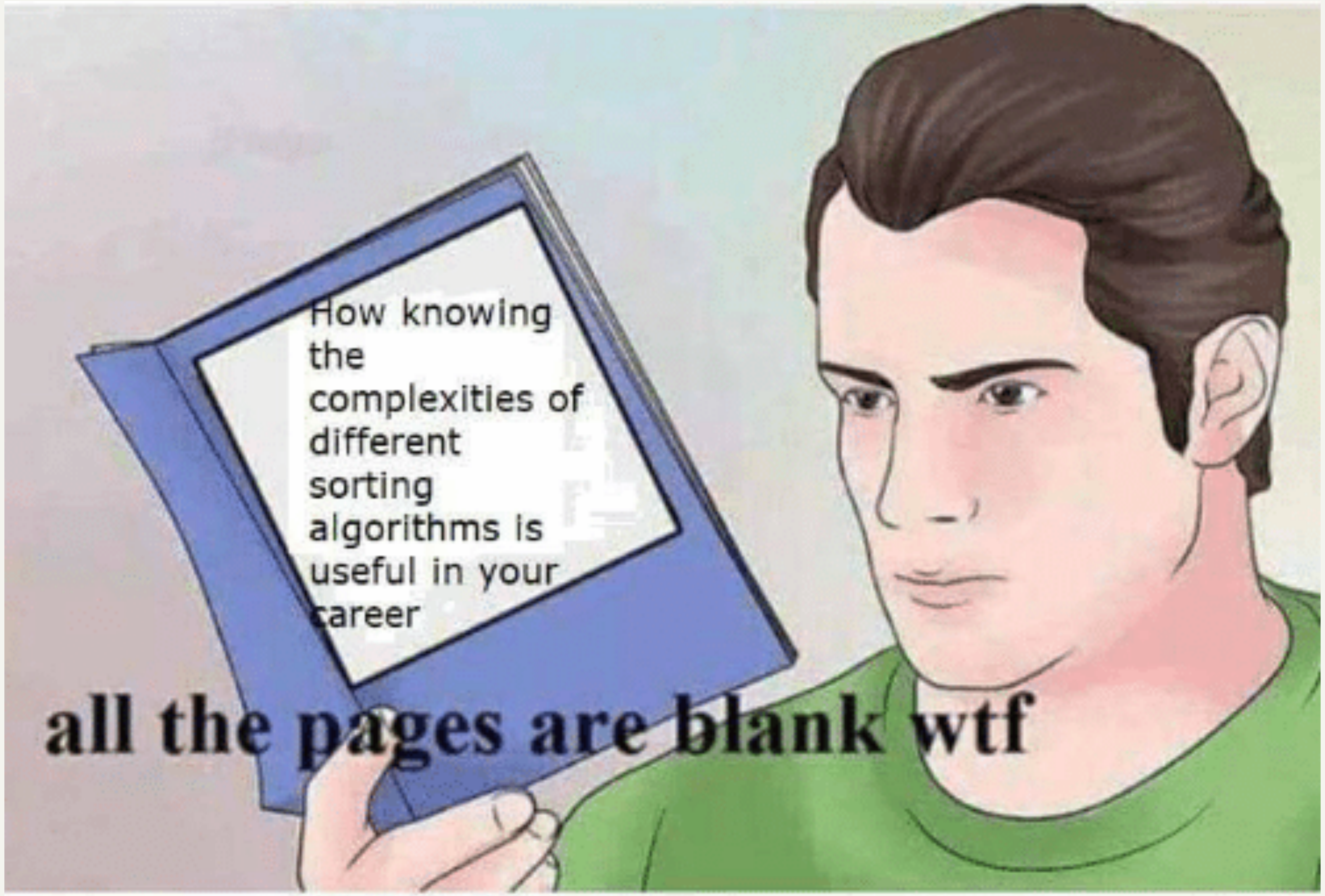
Quicksort

```
quickSort(arr, sInd, eInd)
   if (sInd < eInd)
       pivot = partition(arr, sInd, eInd);
       quickSort(arr, sInd, pivot - 1);  // Before pivot
       quickSort(arr, pivot + 1, eInd); // After pivot


partition (arr, sInd, eInd)
   pivot = eInd; // Pick our pivot
   while (pivot != fil && fil <= eInd && fis >= sInd)
      fil = firstItemLargerThan(arr, pivot, sInd);
      fis = firstItemSmallerThan(arr, pivot, eInd - 1);
      if (fil < fis)
         swap(arr[fil], arr[fis])
      else if (fil > fis)
         swap(arr[fil], arr[pivot]);
         pivot = fil;
   return pivot;
```

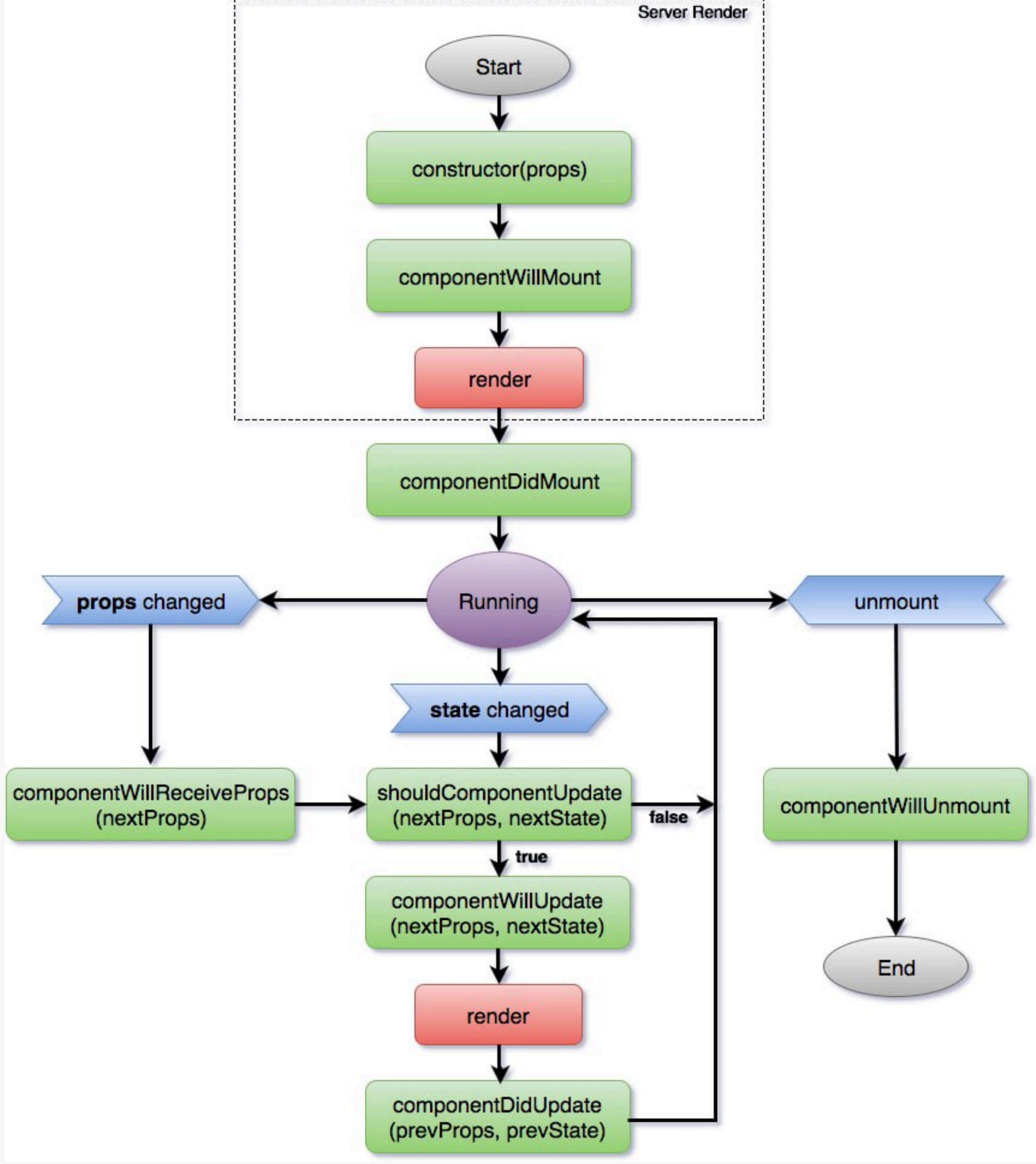
Quicksort Complexity

- The complexity of Quicksort depends on how you pick the pivot
- Best performance when the pivot is the median value of the array
    - Some people always choose the right-most element
    - Some people randomly pick an element
    - Some people pick three random elements, find the middle of those three, and use that
- Worst case is `O(n²)`, best case is `O(nlogn)`
- Why is this better than Insertion Sort, whose best case is `O(n)`?
    - Insertion sort *almost always* gets worse as `n` increases
    - Quicksort is *usually* `O(nlogn)`; it only changes to `O(n²)` when we pick a bad pivot, and that's actually pretty rare, as most pivots will result in `O(nlogn)`


How knowing the complexities of different sorting algorithms is useful in your career
all the pages are blank wtf

Component Lifecycle

- Each component has life events that we can listen in on with pre-defined handler events
- Components are born when they render, and once born they are "**mounted**"
- When components change, they re-render or "**update**"
- When components are about to die (they are removed from the page), they prepare to "**unmount**"


Server Render
Start
constructor(props)
componentWillMount
render
componentDidMount
Running
props changed
unmount
state changed
componentWillReceiveProps (nextProps)
shouldComponentUpdate (nextProps, nextState)
false
true
componentWillUnmount
componentWillUpdate (nextProps, nextState)
End
render
componentDidUpdate (prevProps, prevState)

Props (Again)

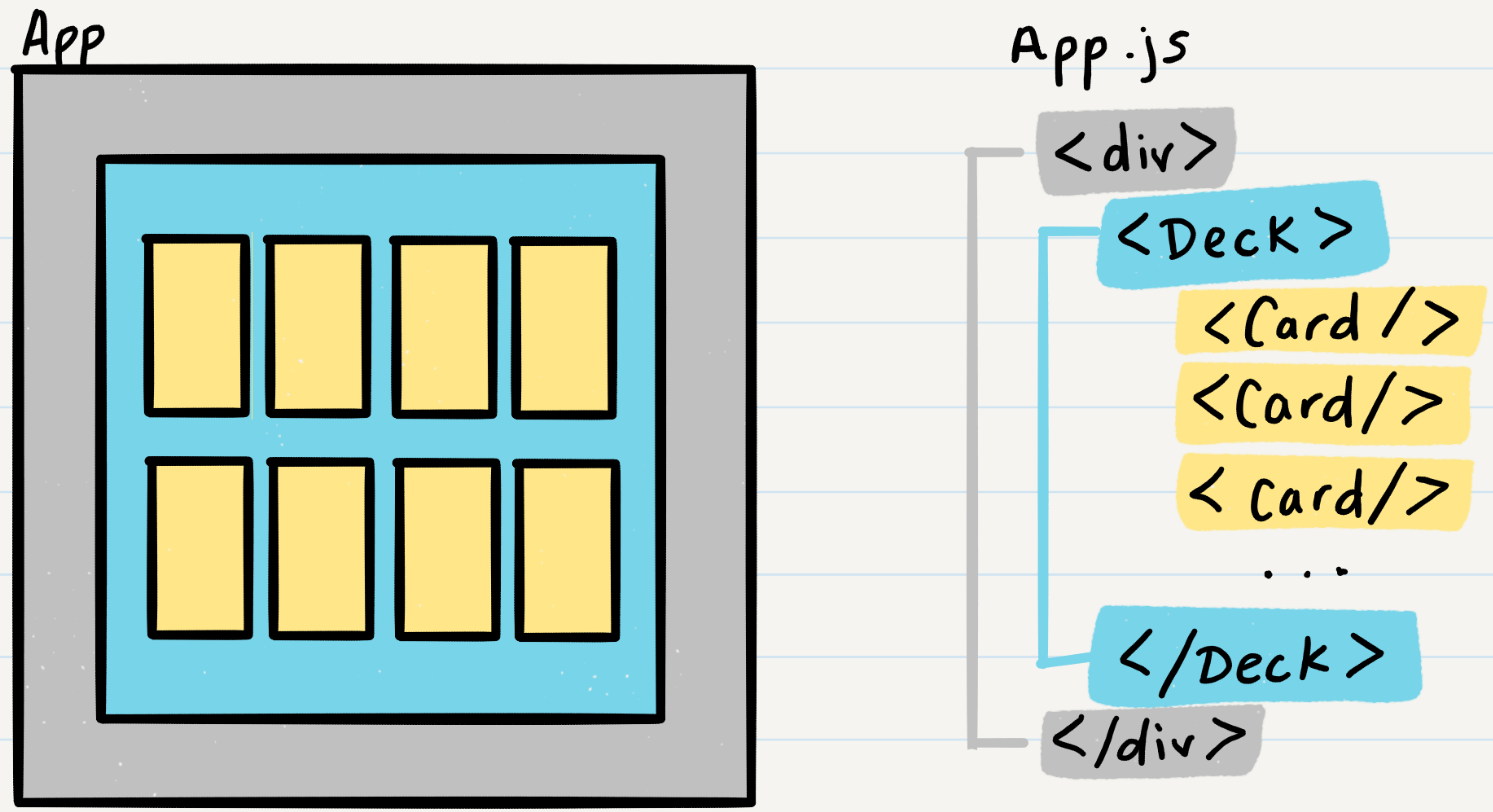
- We know that props are like function parameters, and we can define as many as we like
- What does the props object look like at default?
- There is a default property called **children** that is reserved

Demo

Let's play around with component lifecycle and learn about `props.children`

Cards and Decks

- Common UI design concept
- Cards are small components that display unique data in a specific way
- Decks are components whose main purpose is to contain multiple cards and display them in a specific way (grid layout, single column, etc.)
- In React, we would implement a Deck component that renders some **props.children**, which are Card components



App.js

```
<div>
  <Deck>
    <Card/>
    <Card/>
    <card/>
    ...
  </Deck>
</div>
```

Using Props.Children

- You can define container components that take arbitrary children (0-infinity, any type)
- You can move child components to a specific area of its parent component
- You can create custom lists or same-element containers and let devs fill those containers with more custom HTML
- You can check that children match certain criteria (length, type, etc.)

Routing

- React so far has been running on just `index.html`
- How do we have new “pages”?
- We don’t!
- We tell React to render different components in the `<body>` tag of `index.html`
- We do this via a package called React Router (`npm install react-router` & `npm install react-router-dom`)

Demo

Let’s play around with making links using React Router

Lab 21 Preview

What’s Next:

- Due by Midnight Tonight:
  - **Learning Journal 21**
  - **Partner Power Hour Report**
- Due by Midnight Sunday:
  - **Feedback Week 13**
  - **Career Coaching Workshop #2 Prep**
- Due by Saturday January 4th, 9am:
  - **Lab 21**
  - **Reading Class 22**
- Next Class on Saturday: **Class 22 - Hooks API**

Questions?