JavaScript Frameworks

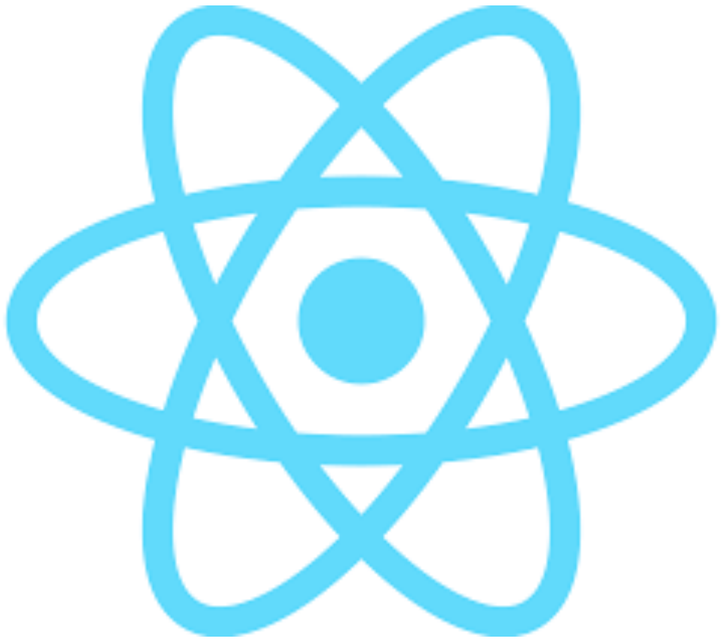
Adhithi Ravichandran

Software Consultant | Author | Speaker

@AdhithiRavi | www.adhithiravichandran.com

React Basics

React



React is a JavaScript library for building rich user interfaces.

UI is built from small units like buttons, text, images.

React lets you combine them into reusable, nestable components.

Search Twitter

Home

Explore

Notifications

Messages

Bookmarks

Lists

Profile

More

Tweet

you | #Tokyo2020 | Trending | COVID-19 | News | Sports | Entertainme

Bloomberg Quicktake · August 13, 2021

Fencing gains popularity in Hong Kong after Olympic gold win

Bloomberg Quicktake · August 12, 2021

This Olympian won gold after a #Tokyo2020 volunteer paid for his taxi when he went to the wrong venue

Sports Insider · August 12, 2021

Meet the man in charge of timing the Olympics

Bloomberg Quicktake · August 9, 2021

Tokyo 2020 Olympians receive a hero's welcome from hometown fans

Who to follow

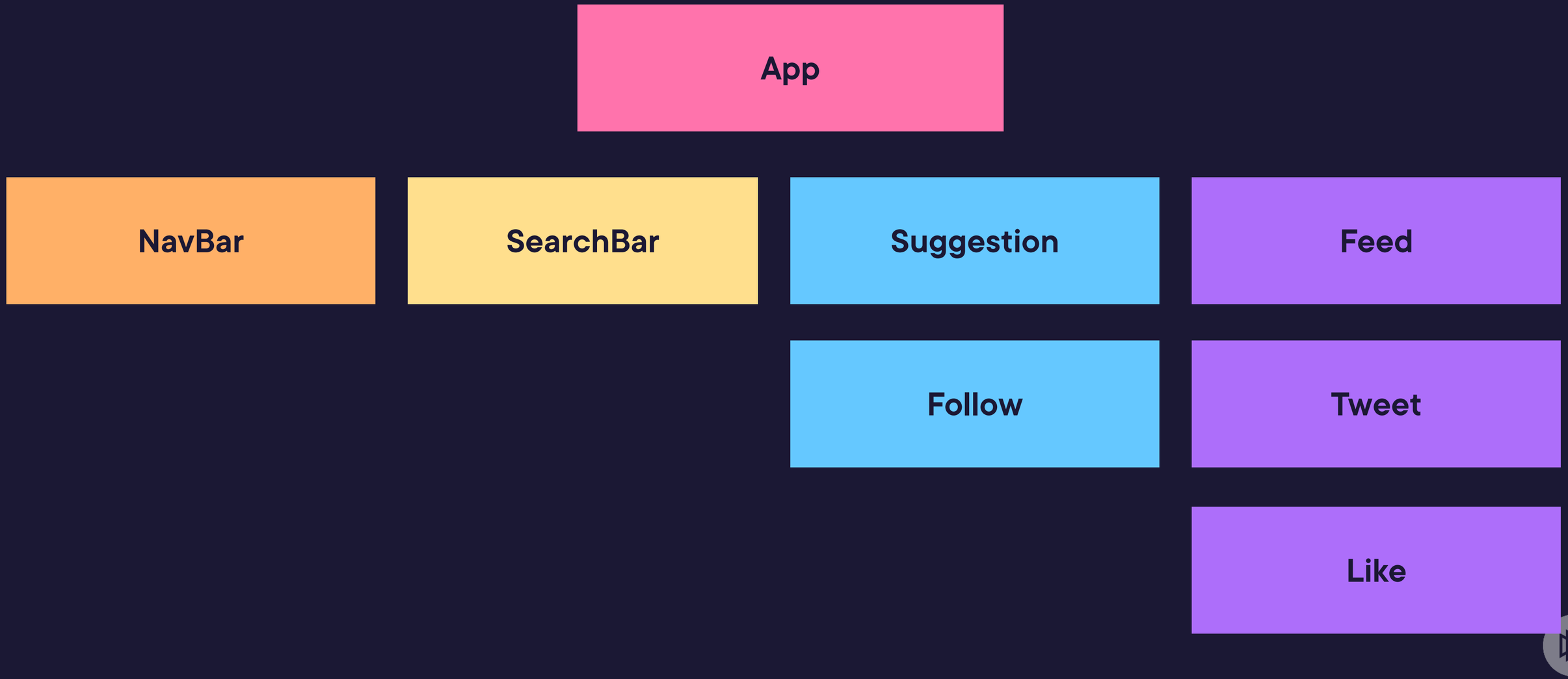
Mary Grygleski
@mgrygles
Follow

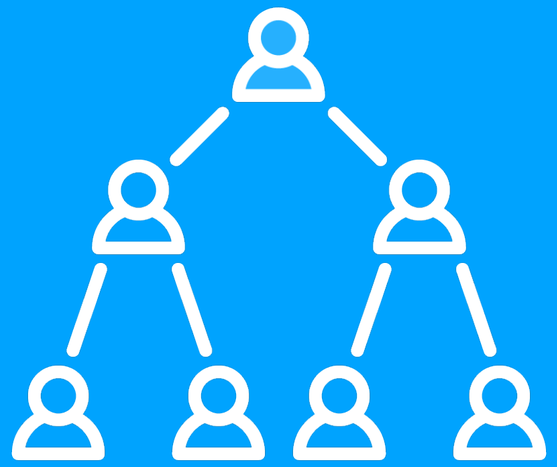
Developer Relations
@JobsInDevRel
Follow

Kate Inyeong Kim
@kateinkim
Follow

Show more

Component Hierarchy



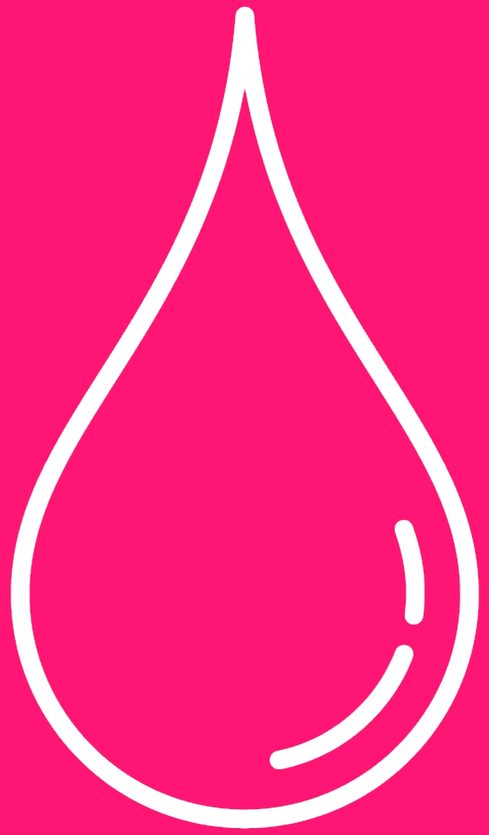


Component Based Model

One way flow of data from parent to child component.

Each component is a building block that is a reusable piece of UI. Putting them together results in a complete application.

Functional Programming in React



Pure Components

React assumes that every component you write is a pure function.

Pure Function

1. Given the same input, always returns the same output.
2. Produces no side-effects.

Pure React Component

Returns the same JSX given the same inputs.

Pure Component in React

```
function CoffeeRecipe({ guests }) {
  return (
    <ol>
      <li>Boil {guests} cups of water.</li>
      <li>Add {guests} spoons of coffee.</li>
      <li>Add {0.5 * guests} cups of milk to boil
      and sugar to taste.</li>
    </ol>
  );
}
```

Props do not change within a component! It is read-only.

Props

Avoid side-effects within a component

Impure Component

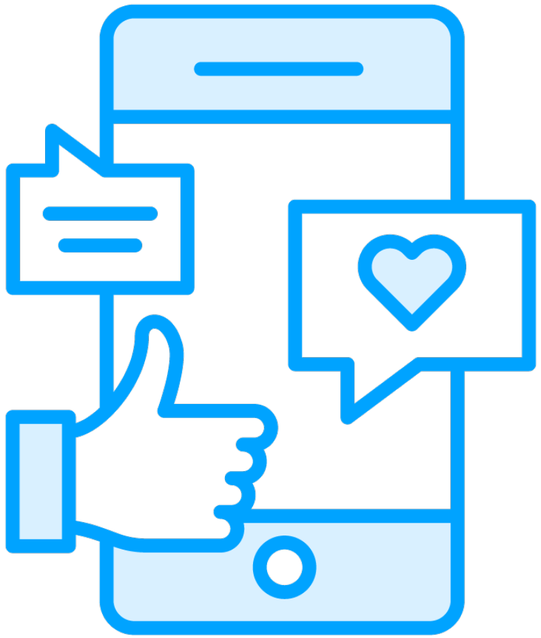
```
let guests = 0;

function CoffeeRecipe() {
   guests = guests + 1;
    return (
        <ol>
            <li>Boil {guests} cups of water.</li>
            <li>Add {guests} spoons of
            coffee.</li>
            <li>Add {0.5 * guests} cups of milk to
            boil and sugar to taste.</li>
        </ol>
    );
}
```

Pure Component

```
function CoffeeRecipe({ guests }) {
    return (
        <ol>
            <li>Boil {guests} cups of
            water.</li>
            <li>Add {guests} spoons of
            coffee.</li>
            <li>Add {0.5 * guests} cups of milk
            to boil and sugar to taste.</li>
        </ol>
    );
}
```

Where Do Side-effects Go?

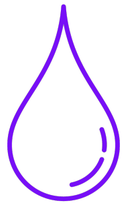


Change within an app is inevitable.

Event handlers are functions that React runs when you perform some functions.

Event handlers do not have to be pure!

Purity in React



Components can even run on server. With pure components, one component can serve many user requests.

Improve performance: Skip rendering components whose inputs have not changed. Pure functions are safe to cache.

Keeping components pure unlocks the power of React paradigm

React Code Sandbox

Code Sandbox to Pure React Component Example: Coffee Recipe

More on React Purity: https://react.dev/learn/keeping-components-pure

Svelte

Svelte

A new approach to build user interfaces.

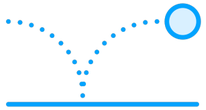
An alternative to other libraries like React and Vue.

Flexible and components don’t have to be pure.

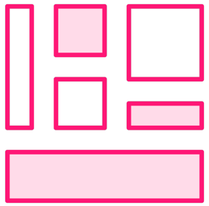
Follows reactive programming.

Reactive programming is basically a program that reacts to events over time. Svelte lets you write reactive code with JavaScript!

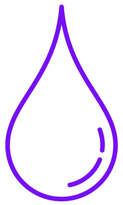
Svelte Offers Flexibility



Shift bulk of the work from the browser to the compiler.



Components are not required to embrace functional programming principles. Flexible components.



Allows you to use functional programming concepts like immutable data and pure functions if you would like to.

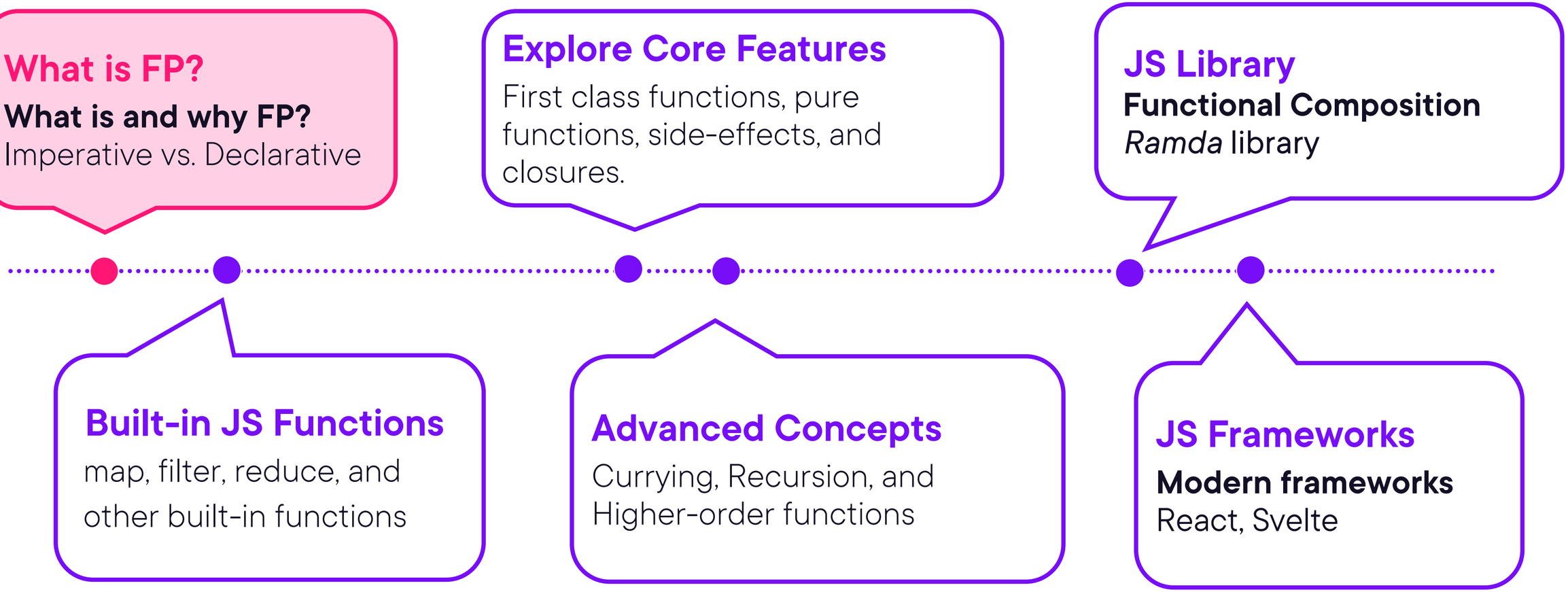
```
<svelte:options immutable={true} />;
```

Immutable Option

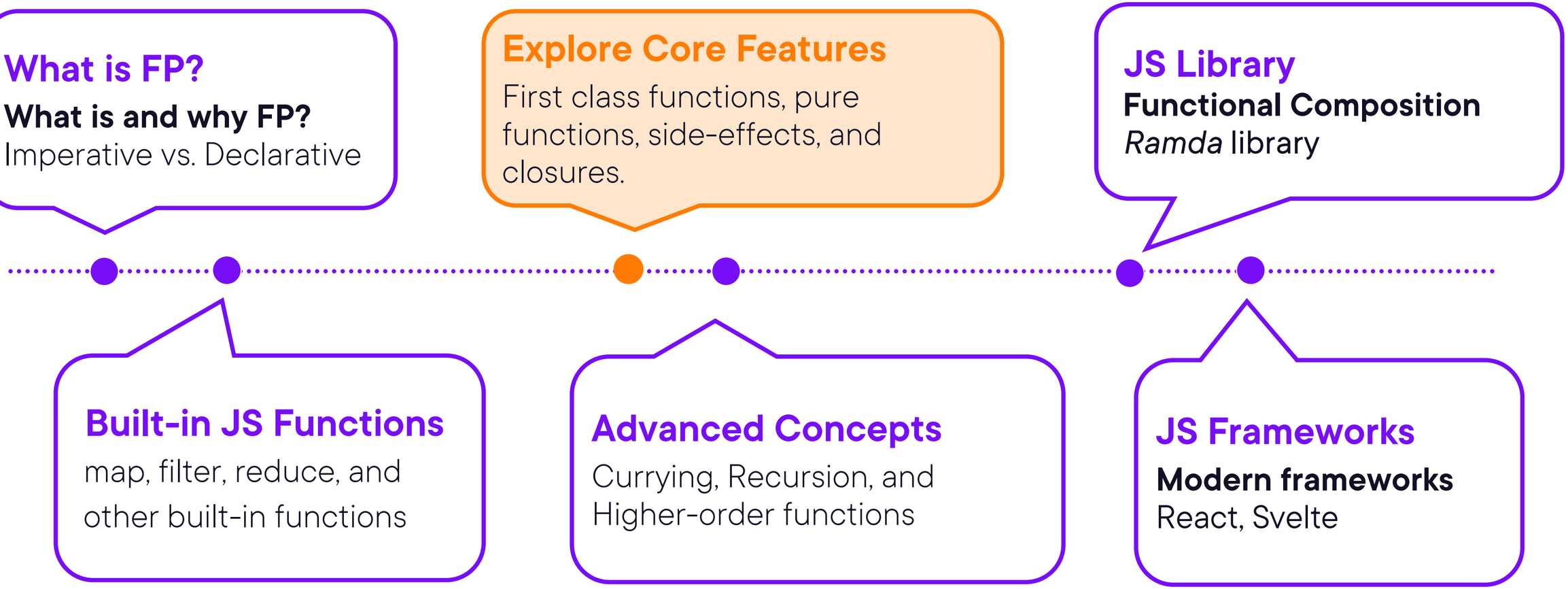
Tells the compiler to expect *immutable* data for a certain component.
By default, the *immutable* option's value is *false*.

Congratulations!

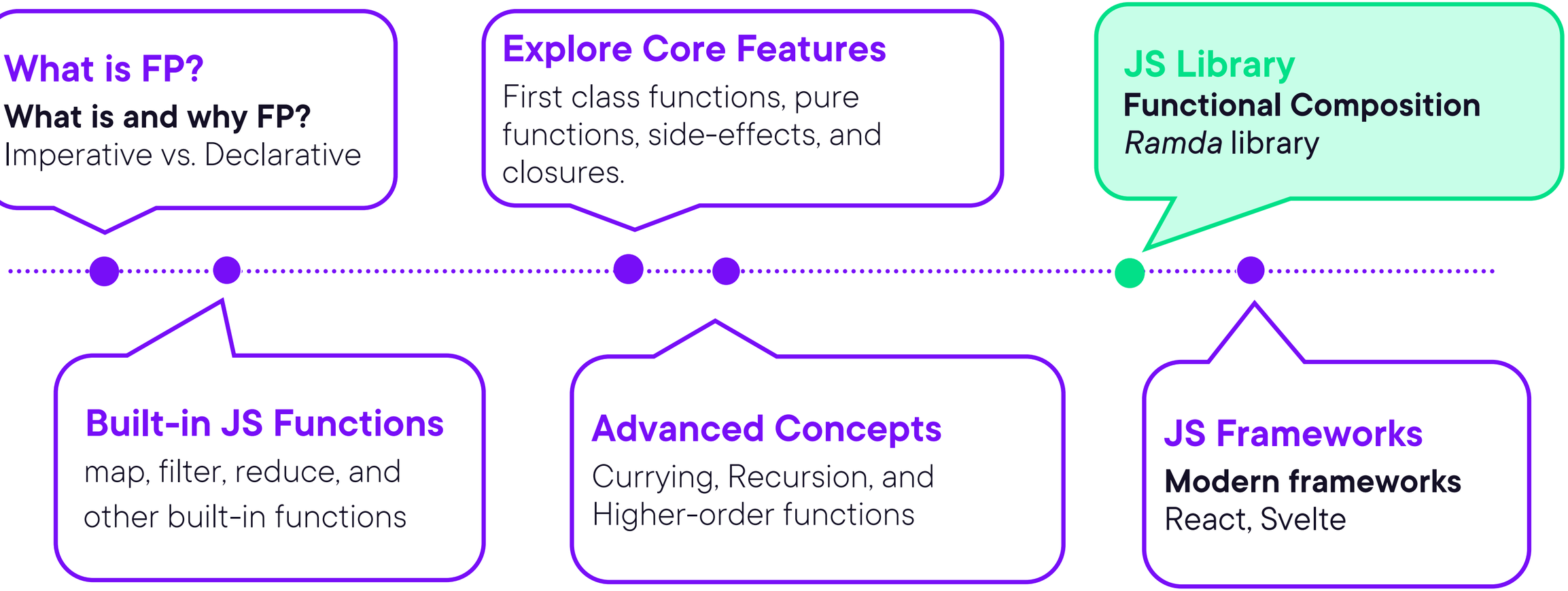
Course Summary



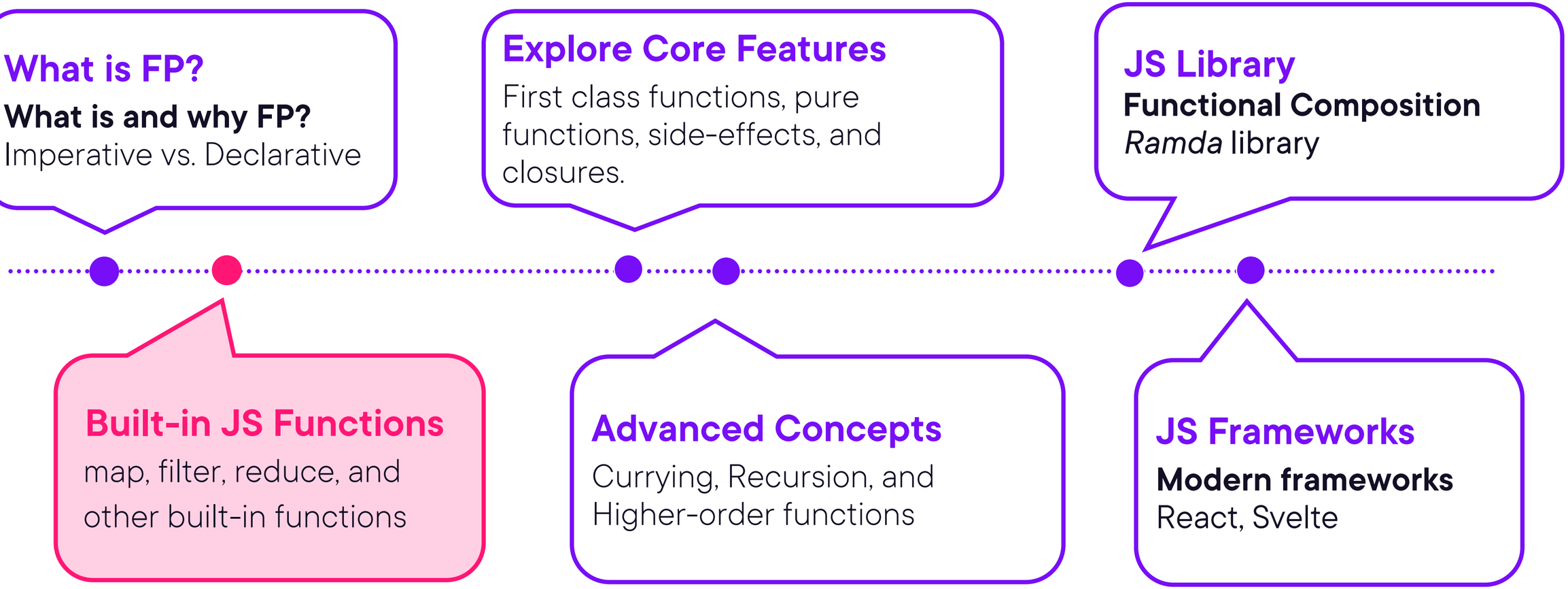
Course Summary



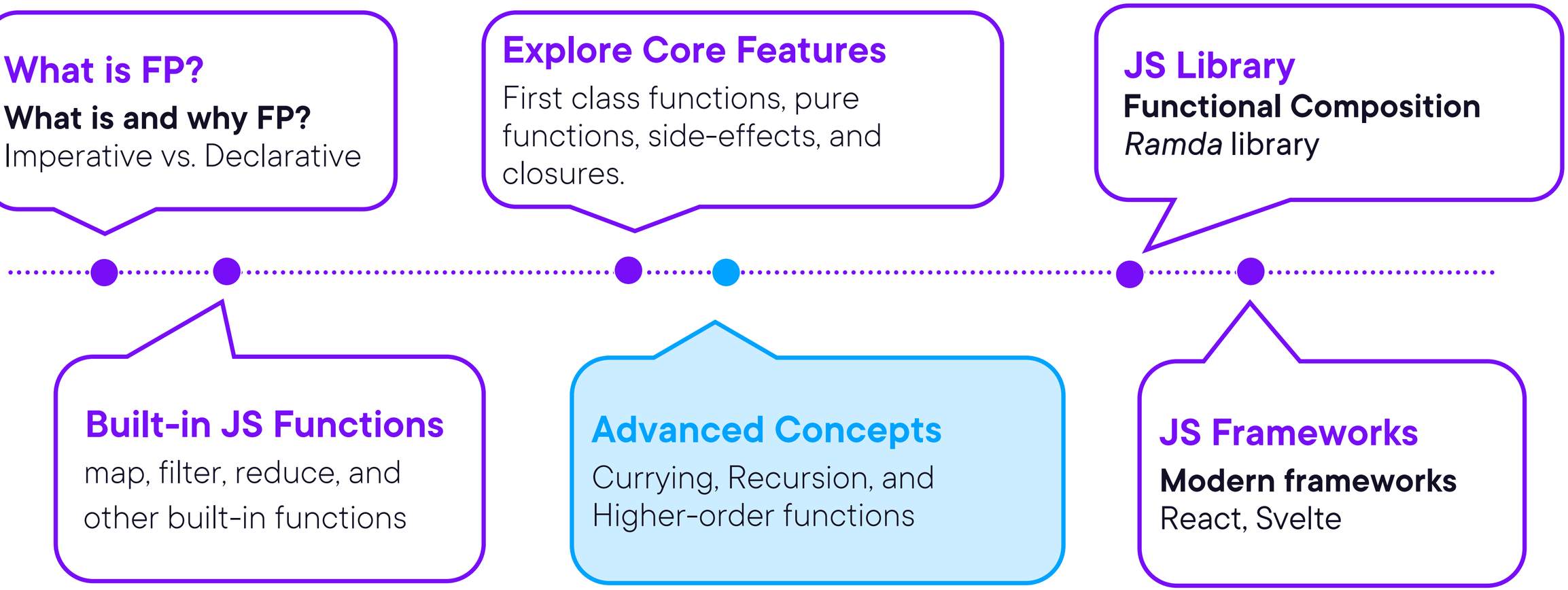
Course Summary



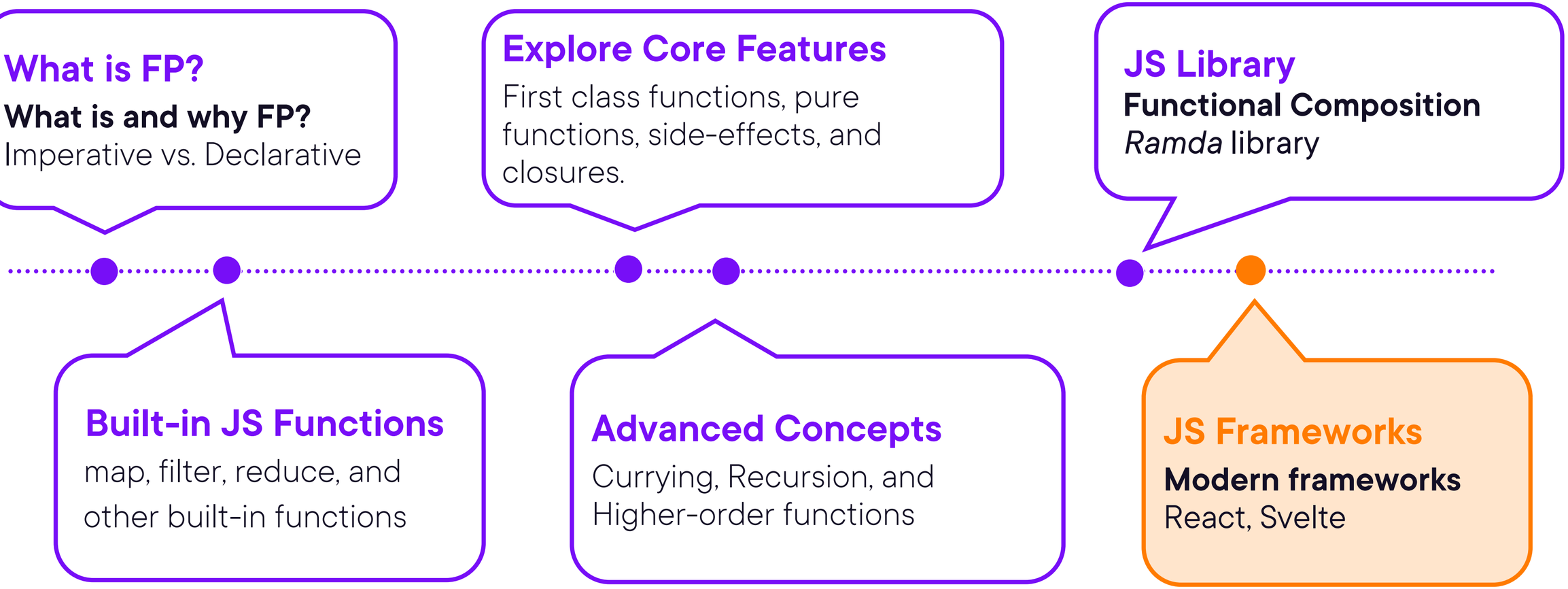
Course Summary



Course Summary



Course Summary



GitHub Repository

https://github.com/adhithiravi/Functional-JS

Thank you!

@AdhithiRavi

app.pluralsight.com/profile/author/adhithi-ravichandran