Functional Programming Concepts in JavaScript

What Is Functional Programming?

Adhithi Ravichandran

Software Consultant | Speaker | Author

@AdhithiRavi | www.adhithiravichandran.com

Functional Programming Concepts in JavaScript

Version Check

Version Check

This course was created by using:

- JavaScript ECMAScript 2022 Specification
- Node 16.x LTX

Version Check

This course is 100% applicable to:

- JavaScript ECMAScript 2022 Specification

Functional Programming Concepts in JavaScript

What Is Functional Programming?

Adhithi Ravichandran

Software Consultant | Speaker | Author

@AdhithiRavi | www.adhithiravichandran.com

Course Overview

What is FP?

What is and why FP?
Compare with other paradigms

Explore Core Features

First class functions, pure functions, closures, side-effects

JS Library

Functional Composition
Ramda library

Built-in JS Functions

map, filter, reduce, and other built-in functions

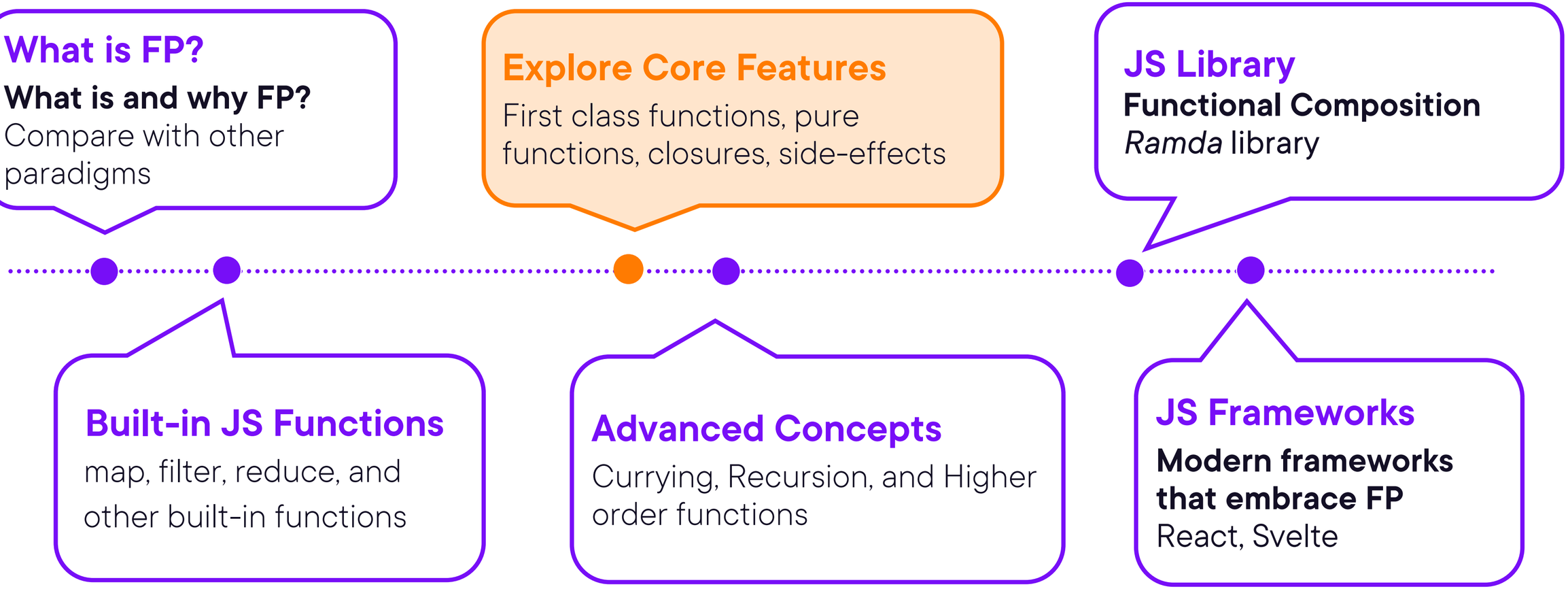
Advanced Concepts

Currying, Recursion, and Higher order functions

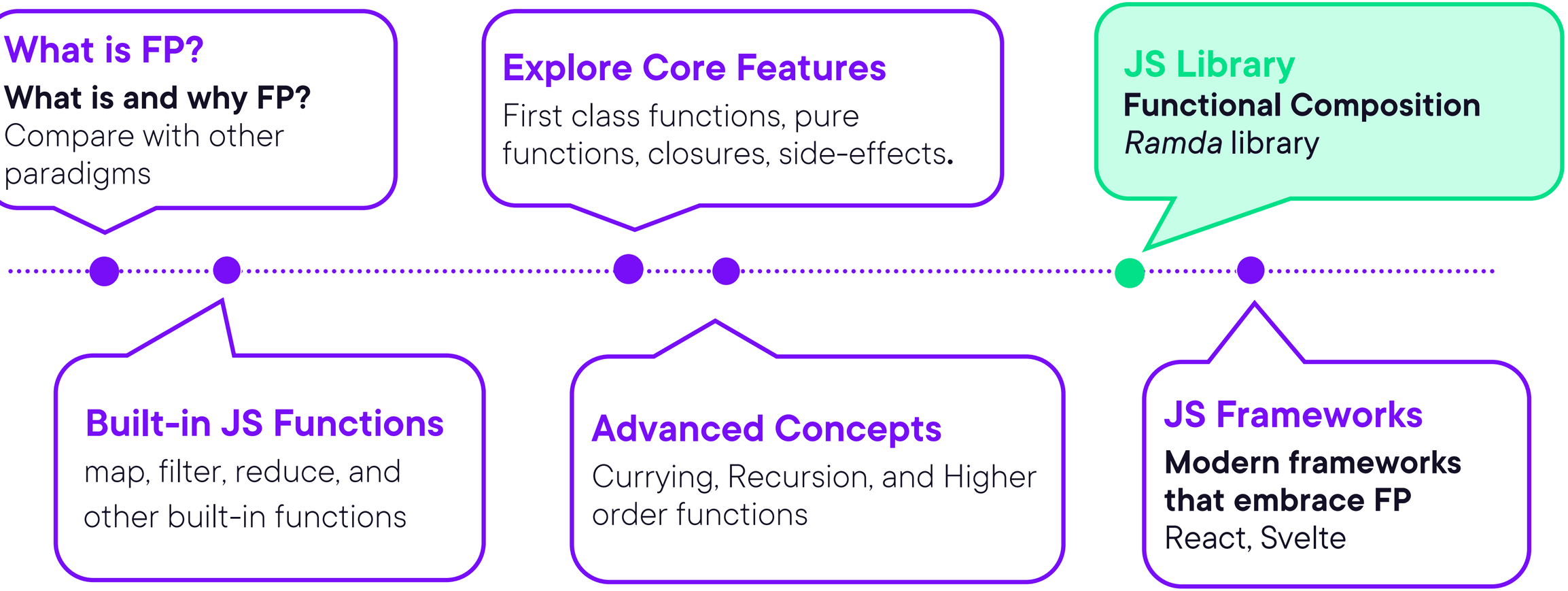
JS Frameworks

Modern frameworks that embrace FP
React, Svelte

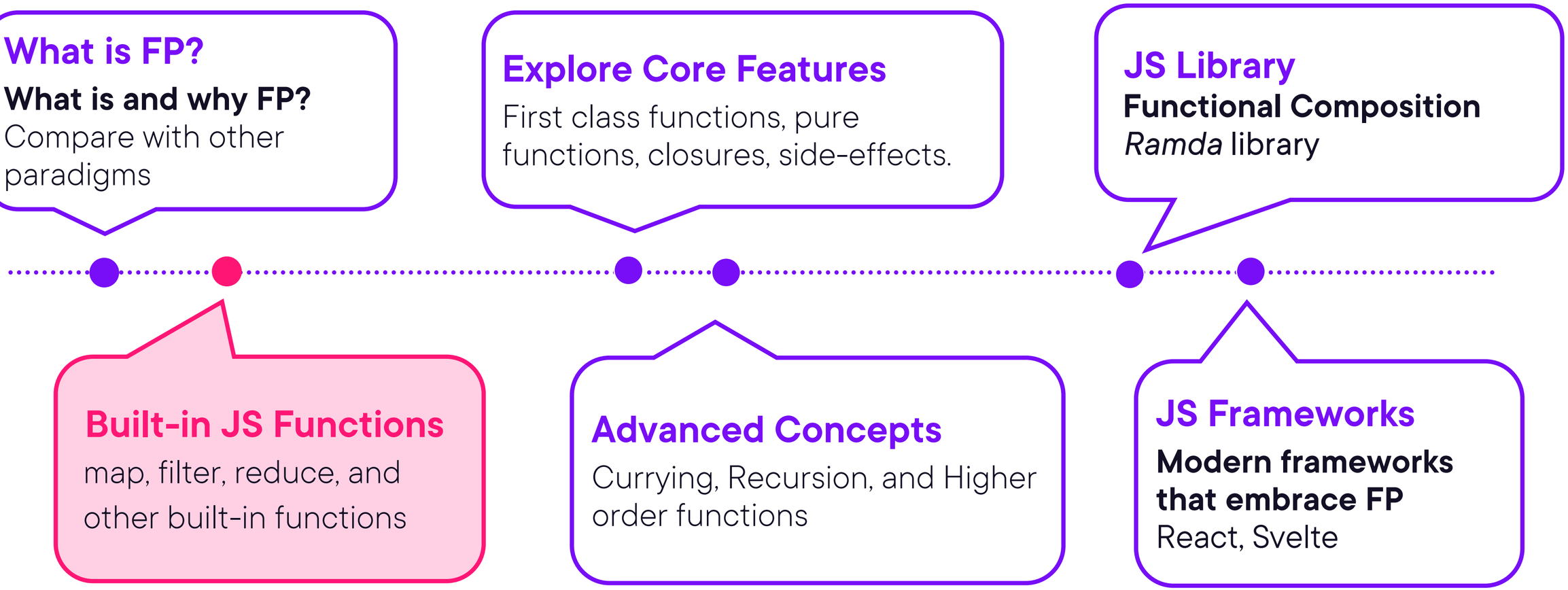
Course Overview



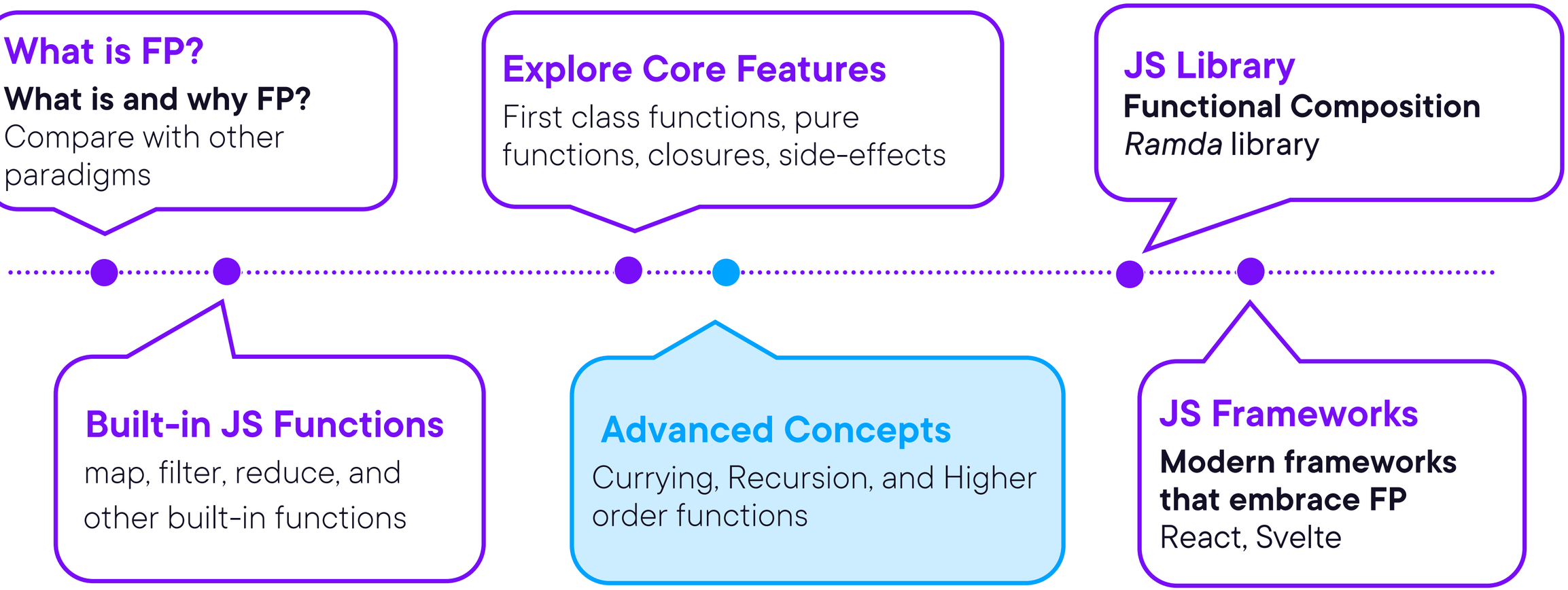
Course Overview



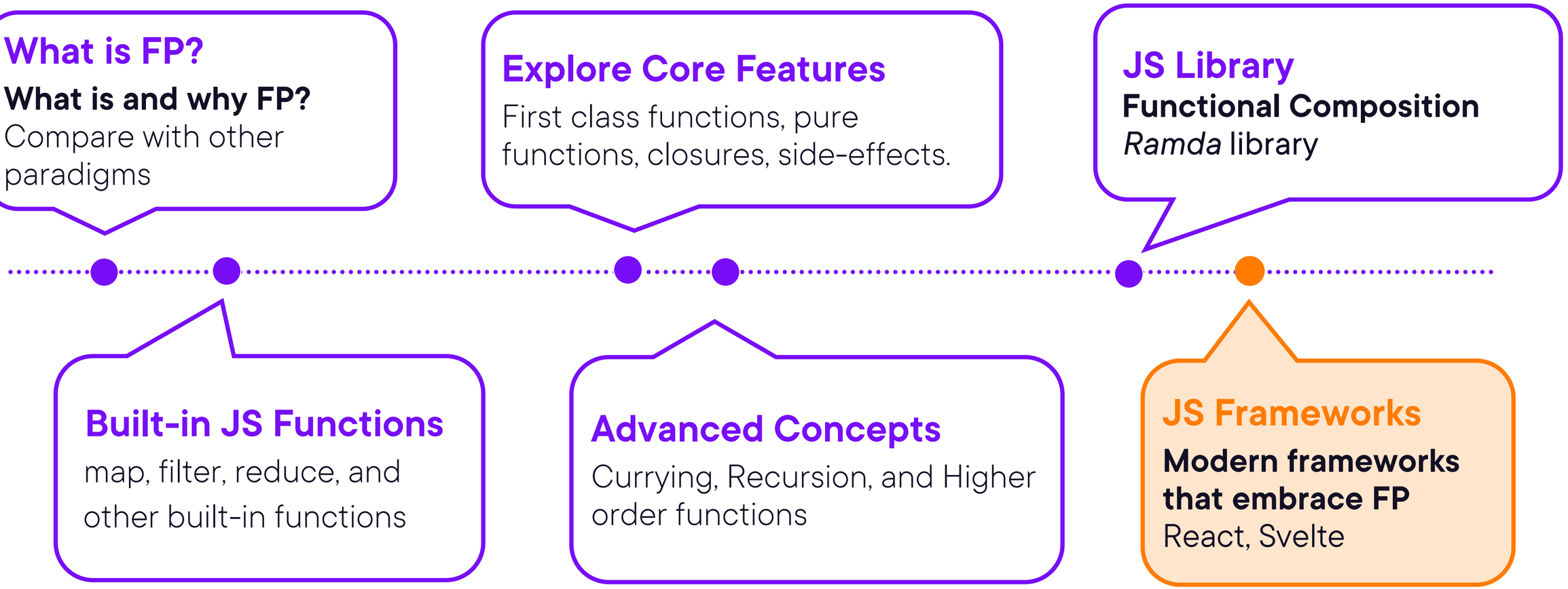
Course Overview



Course Overview



Course Overview

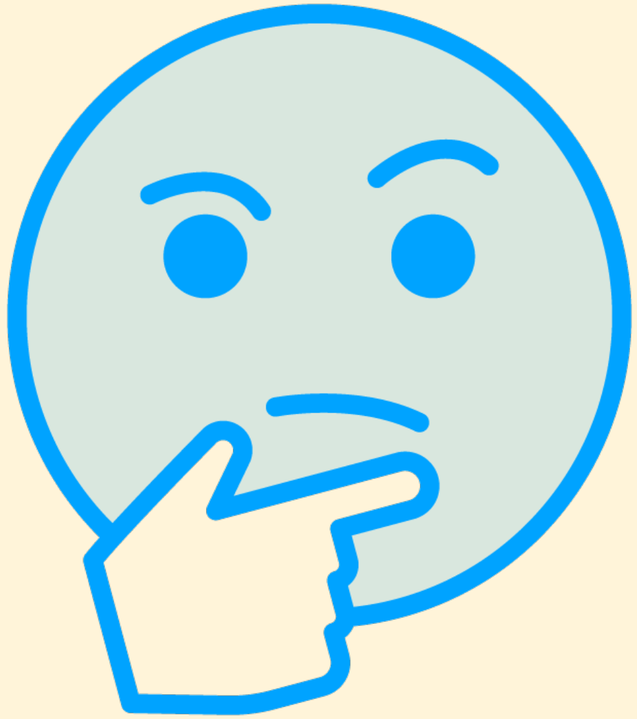


What Is Functional Programming?

What is Functional Programming?

Functional programming is a programming paradigm – a style of building the structure and elements of computer programs, that treats computation as the evaluation of mathematical functions and avoids change-state and mutable data.

Citation: https://en.wikipedia.org/wiki/Functional_programming



Programming Paradigms

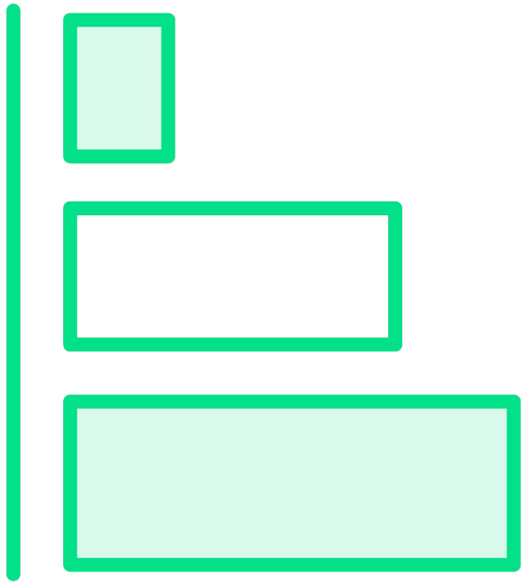
Imperative

Procedural and Object-oriented Programming

Declarative

Functional, Mathematical, and Reactive Programming

Imperative Programming



Focuses on the “How”

Procedural and Object-oriented Programming

States the order in which operations occur

Imperative Programming

```
for (var i = 0; i < courses.length; i++) {
   courses[i].lastModified = new Date()
}
```

Declarative Programming

Focuses on “What”

Functional, Mathematical, and Reactive Programming

Does not state the order in which to execute operations

Declarative Programming

```
courses.map((course) => {
    course.lastModified = new Date();
    return course
})
```

Initial Value

```
[
  {
    name: 'Fundamentals of Next.js’
  },
  {
    name: 'Fundamentals of Functional Programming’
  }
]
```

Final Value

```
[{
    name: 'Fundamentals of Next.js’,
    lastModified: Today
},
{
    name: 'Fundamentals of Functional Programming’,
    lastModified: Today
}]
```

Key Principles

In Functional Programming, focus on ensuring each function does one thing well!

Functions Doing More Than One Thing!

```
function getCourseInformation() {

  // 1. Call API and get courses information.

  // 2. Sort courses based on release date, with newest
  being first.

  // 3. Render the courses within the UI.

}
```

Function That Does One Thing Well!

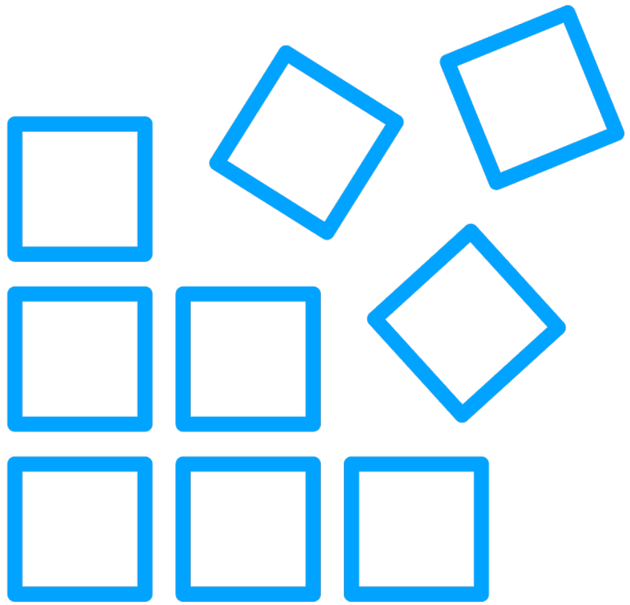
```
function getCourseInfoQuery() {
 // Call API and get courses information.
}

function sortCourses() {
 // Sort Courses based on release date, with newest being
first.
}

function renderCourses() {
 // Render the courses within the UI.
}
```

Function That Does One Thing Well!

Readable

Maintainable

Testable



Functional Programming is all about building functions for immutable variables

Immutable data is data that cannot be changed after it is created!

```
const initialArray = ['c', 'd', 'a', 'e', 'b'];

const sortedArray = sortFunction(initialArray);

// sortedArray = ['a', 'b', 'c', 'd', 'e']
// initialArray = ['c', 'd', 'a', 'e', 'b']
```

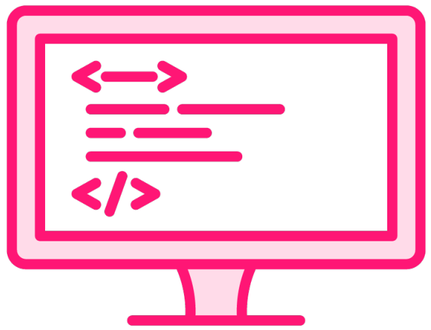
Immutable Sort

- *sortFunction* does not replace first array.
- Function returns a new array *sortedArray* that contains the new copy of the sorted array.
- There are two different arrays now, and the *sortFunction* is immutable and did not change any data.

Why Functional Programming?

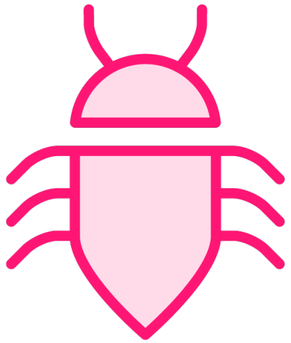

Why should I use Functional Programming?

Why Functional Programming?

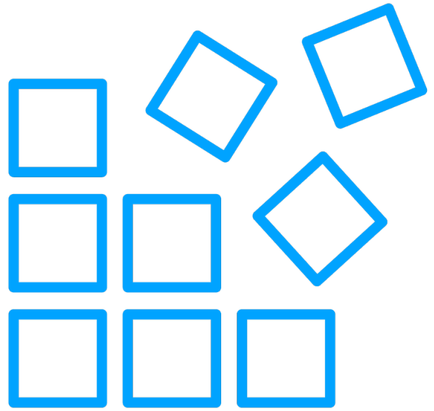


Code is concise, predictable, and readable

Easier to test functional code



Results in fewer bugs



Modular code with smaller and independent units

A Paradigm For JavaScript

Functional vs. Object-oriented Programming

Functional Programming	VS	Object-oriented Programming
Function is the main unit		Object is the primary unit
Data is immutable		Data is mutable
Fewer data structures		More data structures
Functions with no side effects		Methods with side effects

What type of language is JS?

What type of language is JavaScript?

JavaScript supports both Object-oriented and Functional programming paradigms.

A paradigm for JavaScript

You can choose whichever paradigm suits your application's use case or use a combination of the paradigms!

Up Next:

Explore Core Features of Functional Programming