Managing State in React 18

Deciding How to Handle State

Cory House

React Consultant and Trainer

@housecor | reactjsconsulting.com

Version Check

This course was created by using:

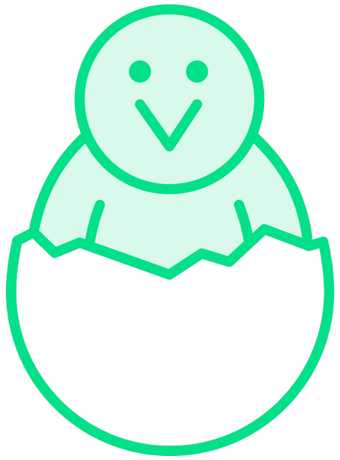
- React 18
- React Router 6

Version Check

This course is 100% applicable to:

- React 16 - 18

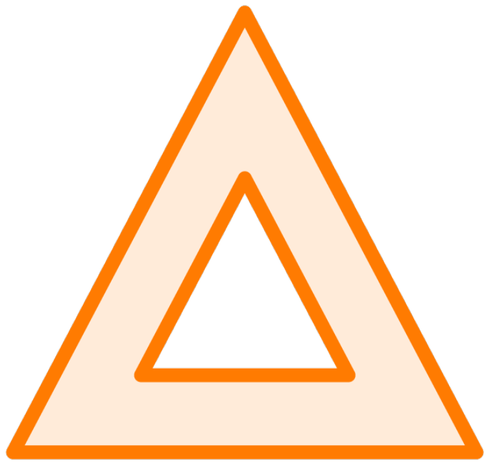
Target Audience



Relatively new to React

Want to improve state skills

Prerequisites



Understand JSX

Know how to declare a component

Mostly function components

Patterns presented work in classes too

Dedicated class component module

Agenda

Goal: Establish mental model

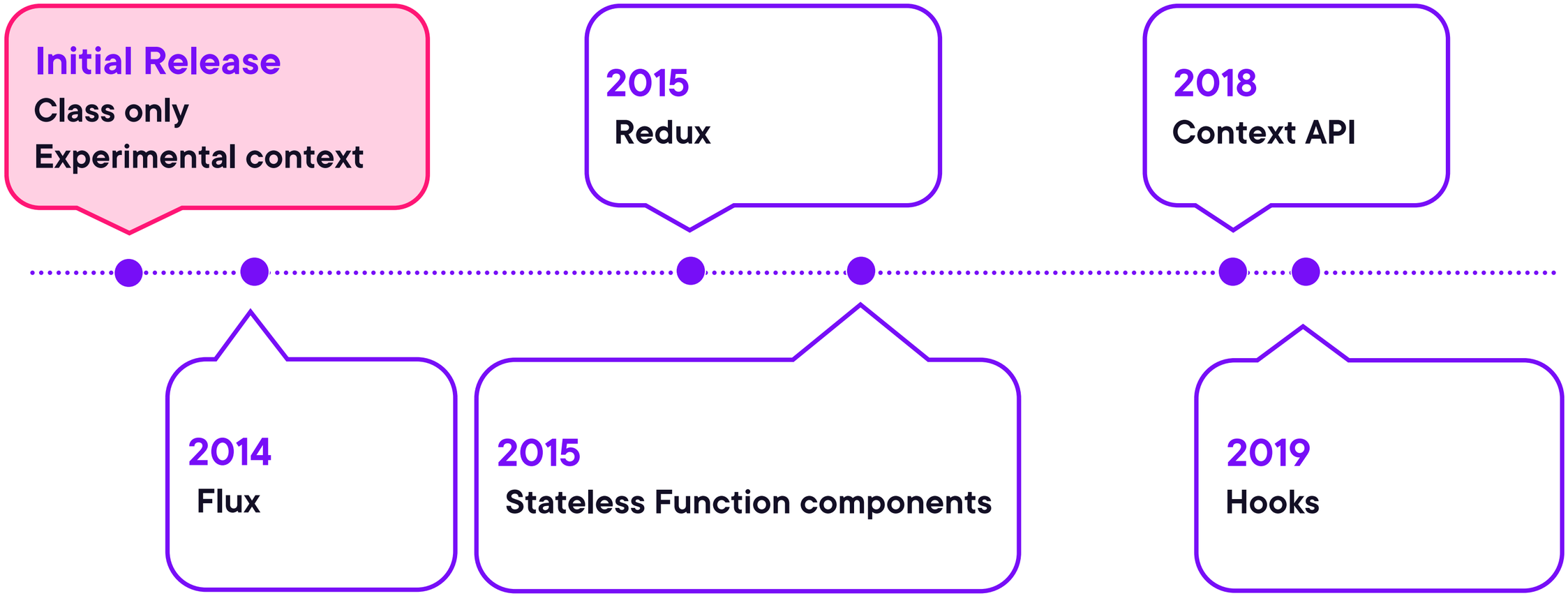
History of React State

Eight ways to handle state

- Deciding how to handle state

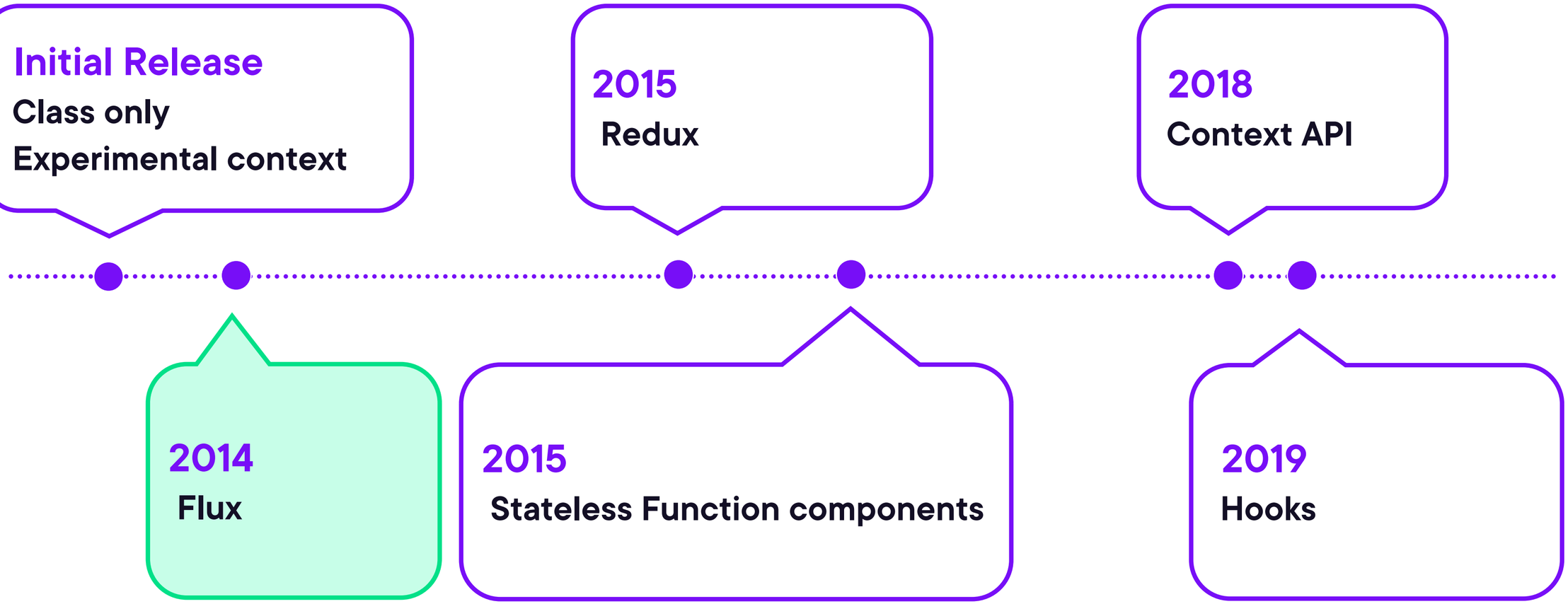
JavaScript data structures

A History of React State



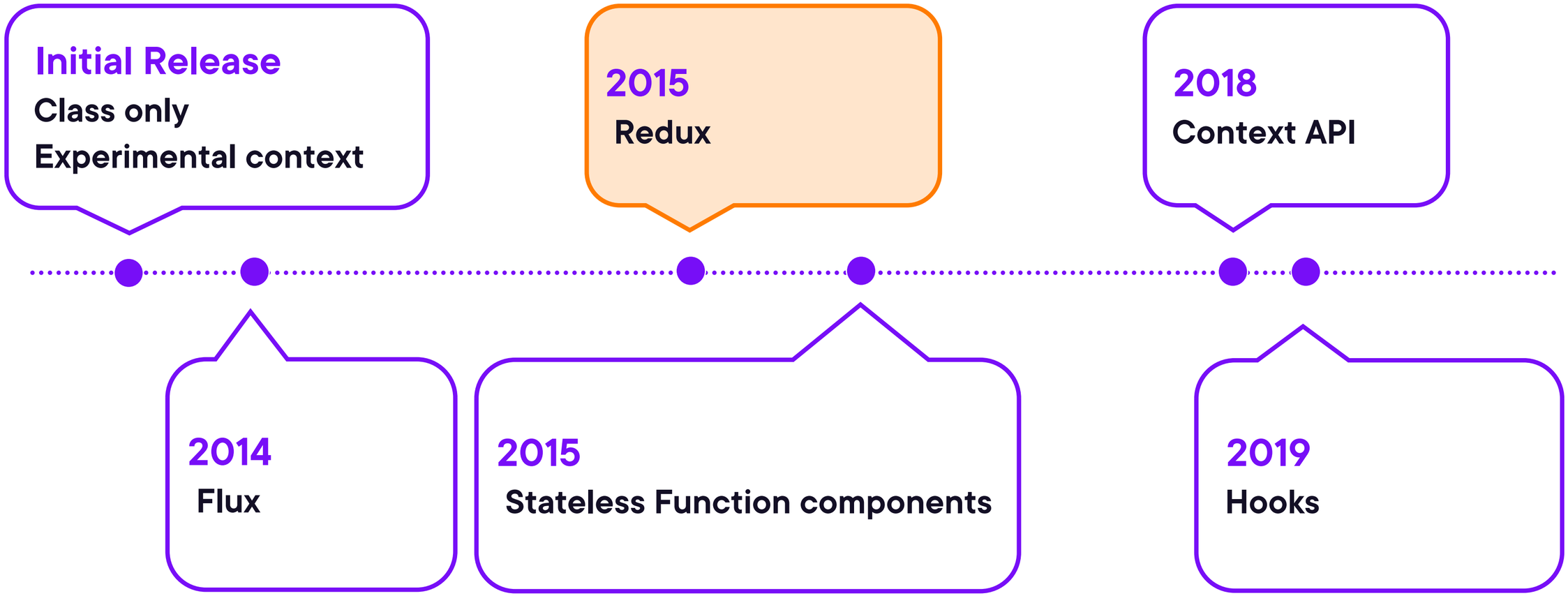
*View in presentation mode

A History of React State



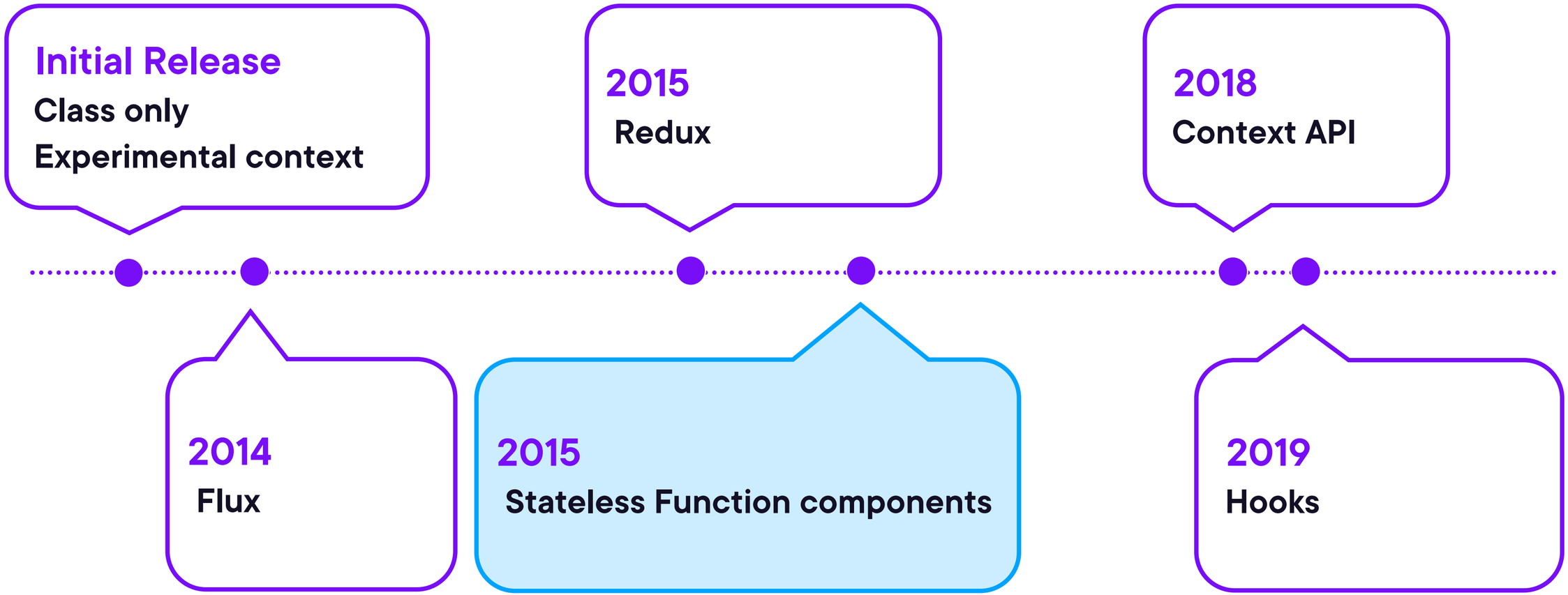
*View in presentation mode

A History of React State



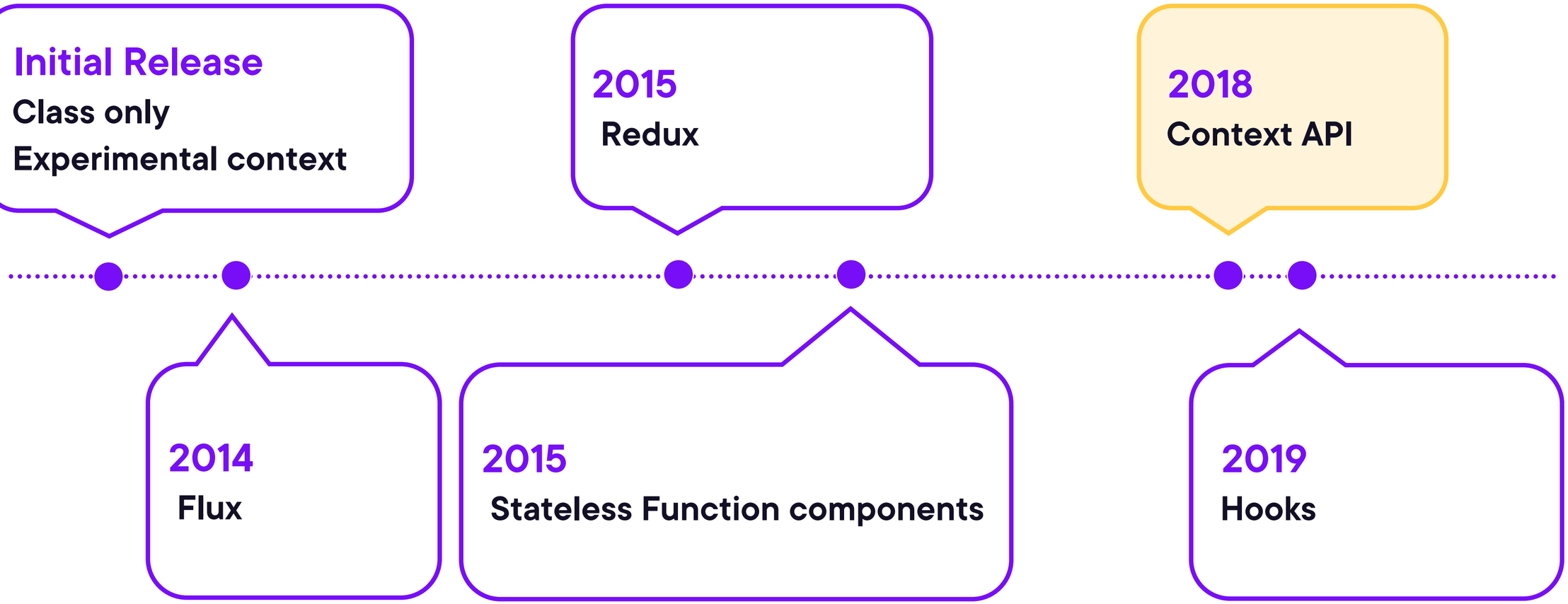
*View in presentation mode

A History of React State



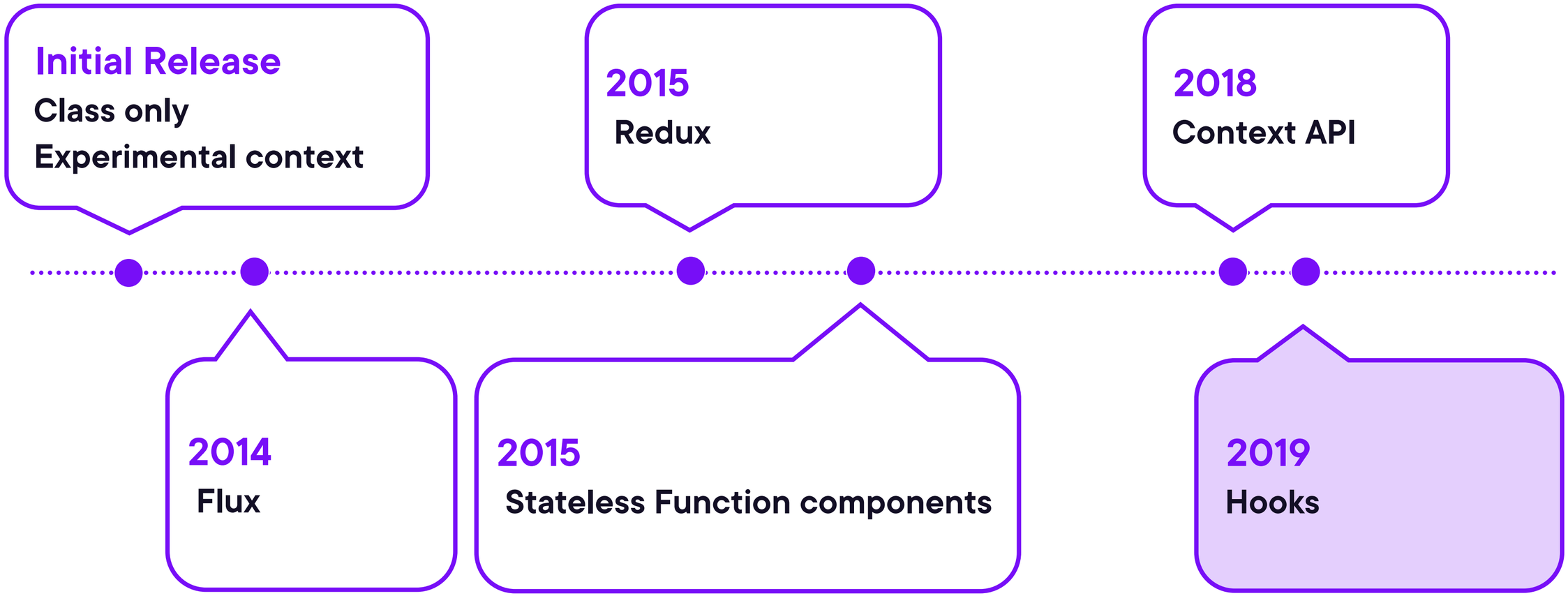
*View in presentation mode

A History of React State



*View in presentation mode

A History of React State

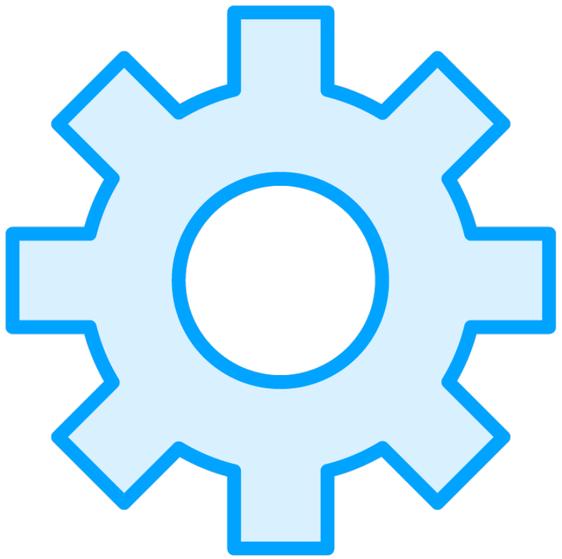


State

App data that may change

Eight Ways to Handle State

Side Note: Environment Variables



Store environment-specific settings

Don’t change at runtime

Supported on all operating systems

Built into create-react-app

- REACT_APP_BASE_URL
- REACT_APP_ENABLE_FEATURE_X

Examples

- Base URLs
- Feature toggles

Eight Ways to Handle React State

URL

Web storage

Local state

Lifted state

Derived state

Refs

Context

Third party library

Option 1: URL

Current app location/settings

- Current item
- Filters
- Pagination
- Sorting

Supports deep linking

Avoid redundantly storing in your app

Consider React Router

Option 2: Web Storage

Persist state between reloads

- cookies, localStorage, IndexedDB

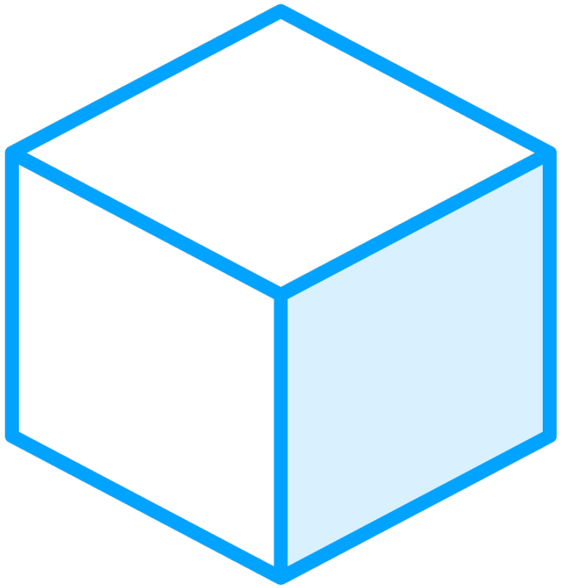
Watch out

- Tied to a single browser
- Avoid storing sensitive data

Examples

- Items in shopping cart
- Partially completed form data

Option 3: Local State



Store state inside a React component

- Useful when one component needs it

Example

- Forms
- Toggles
- Local lists

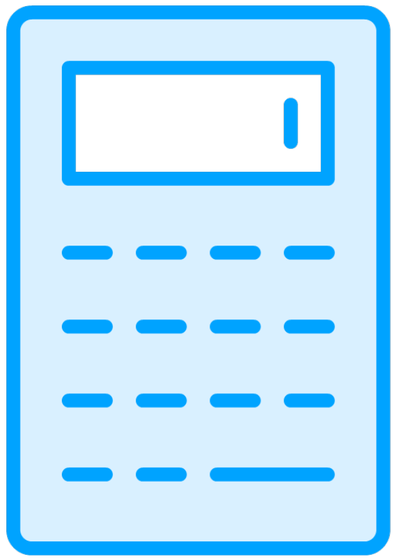
Option 4: Lifted State

Lift state to a common parent

- Declare state in a parent component
- Pass state down via props

Use: Related components need same state

Option 5: Derived State



Derive state from existing state/props

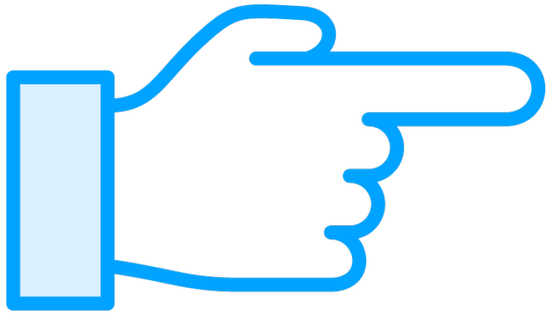
Examples

- Call .length on an array in render
- Derive errorsExist by checking if the errors array is empty.

Why derive?

- Avoids out of sync bugs
- Simplifies code

Option 6: Refs



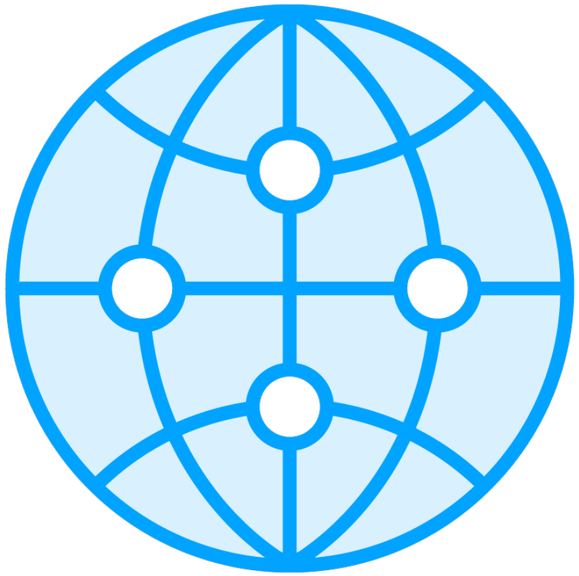
1. DOM reference

- Uncontrolled components
- Interfacing with non-React libraries

2. State that isn't displayed

- Track if component is mounted
- Hold timer
- Store previous state value

Option 7: Context



Global / broadly used state and functions

- Avoids “prop drilling”

Examples

- Logged in user
- Authorization settings
- Theming
- Internationalization settings

Option 8: Third Party State

General options

- Redux
- Mobx
- Recoil

Remote State

- react-query
- Swr
- Relay
- Apollo

Eight Ways to Handle React State

 **URL**	Sharable app location
 **Web storage**	Persist between sessions, one browser
 **Local state**	Only one component needs the state
 **Lifted state**	A few related components need the state
 **Derived state**	State can be derived from existing state
 **Refs**	DOM reference, state that isn't rendered
 **Context**	Global or subtree state
 **Third party library**	Global state, Remote state

JavaScript Data Structures

Primitives

Immutable

Boolean	true/false
String	text
Number	numbers
BigInt	big numbers
Symbol	unique id

Collections

Mutable reference

Objects	properties
Array	list
Map	key/value pairs
Set	unique values

Type	Value Range	Size in bytes	Description
`Int8Array`	`-128` to `127`	1	8-bit two's complement signed integer
`Uint8Array`	`0` to `255`	1	8-bit unsigned integer
`Uint8ClampedArray`	`0` to `255`	1	8-bit unsigned integer (clamped)
`Int16Array`	`-32768` to `32767`	2	16-bit two's complement signed integer
`Uint16Array`	`0` to `65535`	2	16-bit unsigned integer
`Int32Array`	`-2147483648` to `2147483647`	4	32-bit two's complement signed integer
`Uint32Array`	`0` to `4294967295`	4	32-bit unsigned integer
`Float32Array`	1.2×10^{-38} to 3.4×10^{38}	4	32-bit IEEE floating point number (7 significant digits e.g., `1.1234567`)
`Float64Array`	5.0×10^{-324} to 1.8×10^{308}	8	64-bit IEEE floating point number (16 significant digits e.g., `1.123...15`)
`BigInt64Array`	-2^{63} to $2^{63}-1$	8	64-bit two's complement signed integer
`BigUint64Array`	`0` to $2^{64}-1$	8	64-bit unsigned integer

JavaScript data types and data structures

Programming languages all have built-in data structures, but these often differ from one language to another. This article attempts to list the built-in data structures available in JavaScript and what properties they have; these can be used to build other data structures. Wherever possible, comparisons with other languages are drawn.

Dynamic typing

JavaScript is a *loosely typed* and *dynamic* language. Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types:

```
let foo = 42;    // foo is now a number
foo     = 'bar'; // foo is now a string
foo     = true;  // foo is now a boolean
```

Data and Structure types

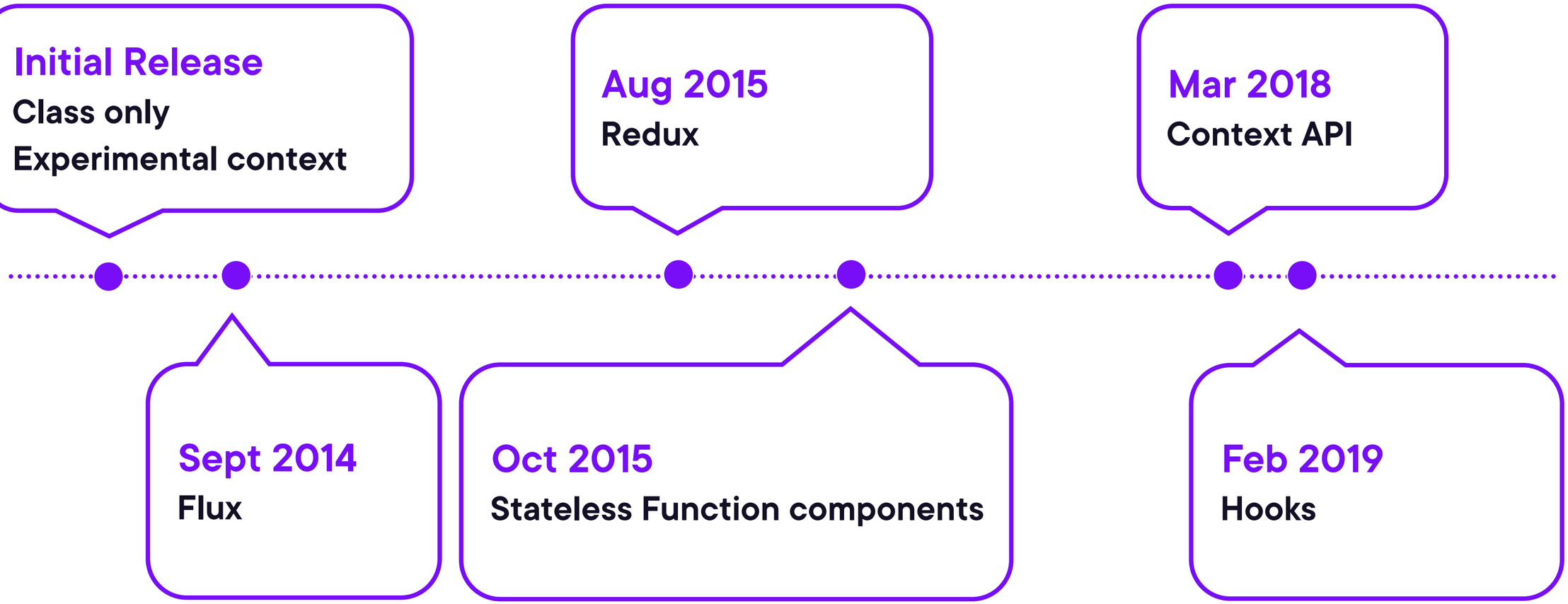
The latest ECMAScript standard defines nine types:

- Six **Data Types** that are primitives, checked by typeof operator:

Summary

History of React State

A History of React State



*View in presentation mode

Summary

History of React State

Eight ways to handle state

Eight Ways to Handle React State

 **URL**	Sharable app location
 **Web storage**	Persist between sessions, one browser
 **Local state**	Only one component needs the state
 **Lifted state**	A few related components need the state
 **Derived state**	State can be derived from existing state
 **Refs**	DOM reference, state that isn't rendered
 **Context**	Global or subtree state
 **Third party library**	Global state, Remote state

Summary

History of React State

Eight ways to handle state

JS data structures

JavaScript Data Structures

Primitives

Immutable

Boolean	true/false
String	text
Number	numbers
BigInt	big numbers
Symbol	unique id

Collections

Mutable reference

Objects	properties
Array	list
Map	key/value pairs
Set	unique values

Up Next:

Managing Local and Remote State