





# Gamifying React Education

```
App.js
                                                                                             1 import { useState } from 'react';
                                                            Next player: X
                                                                             1. Go to game start
   function Square({ value, onSquareClick }) {
     return (
       <button className="square" onClick={onSquareClick}>
         {value}
       </button>
     );
10
   function Board({ xIsNext, squares, onPlay }) {
     function handleClick(i) {
       if (calculateWinner(squares) || squares[i]) {
13
14
         return;
15
       const nextSquares = squares.slice();
16
       if (xIsNext) {
17
         nextSquares[i] = 'X';
18
```



# MAKE 4 PURCHASES, EARN 100 BONUS STARS

Earn 100 Bonus Stars when you pick up almost anything 4 times this week.

3

**80** \*

ONILIO OTA

PURCHASES

BONUS STARS

4

100 \*

PURCHASES

BONUS STARS

Join the challenge

100 \*

You can redeem 100 Stars\* for brewed hot or iced coffee or tea, a bakery item, packaged snack and more.

### evidation

### Hi Courtney!

Keep up the good work and boost your rewards!

### Your Week on Evidation

3,367 POINTS

Congrats, you earned 198 points this week! You're 34% towards \$10.



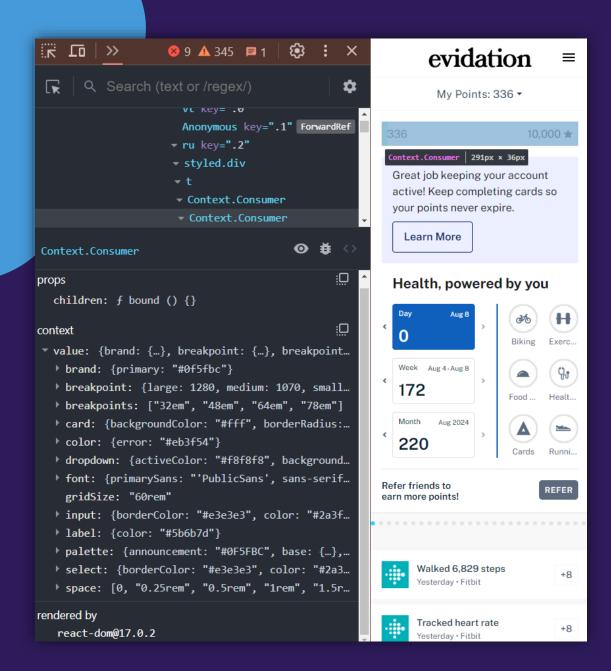
### Did you know there's an app available?

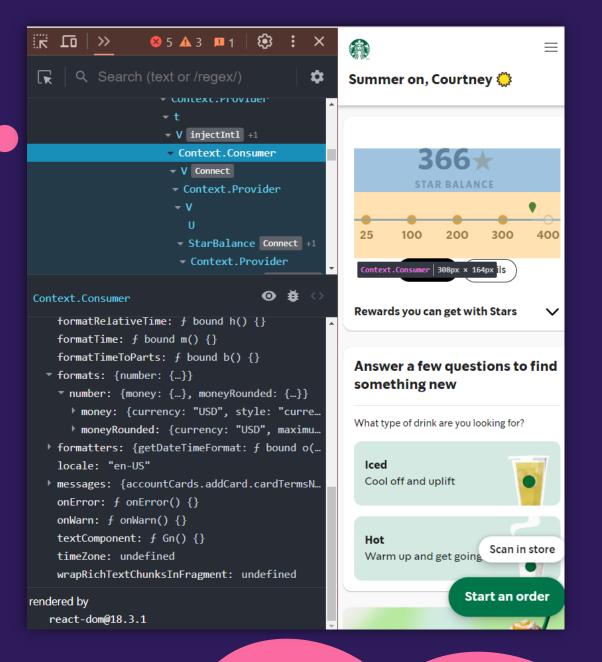
+10 pts

Discover the Evidation app - enhance your experience today!

Learn more







### Courtney Yatteau

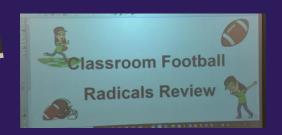
**Developer Advocate, Esri** 









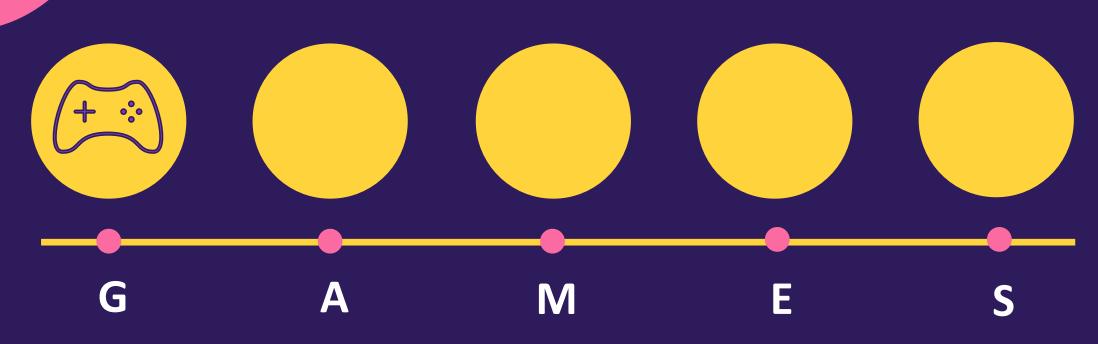








# G.A.M.E.S.



**G**amified UI

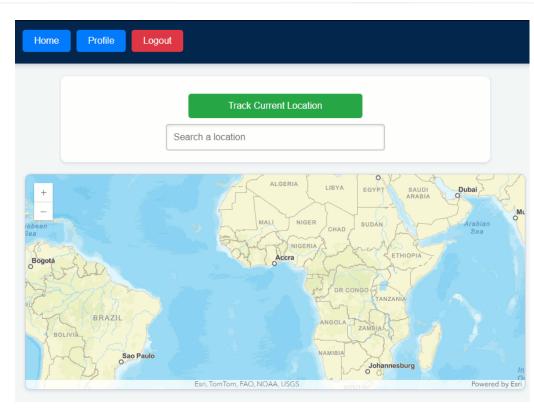
Components

```
const Badge = ({ badges }) => {
  const [earnedBadges, setEarnedBadges] = useState([]);
  const [selectedBadge, setSelectedBadge] = useState(null);
                                                                                                                Track Current Location
  const handleBadgeClick = (badge) => {
                                                                                                        Search a location
    setSelectedBadge(badge);
    setEarnedBadges((prev) => [...prev, badge]);
 };
 return (
    <div className="badges-container">
      {badges.map((badge, index) => (
        <div
          key={index}
          className={`badge ${earnedBadges.includes(badge) ? "badge-earned" :
                                                                                            BRAZIL
          style={{ backgroundColor: badge.color }}
          onClick={() => handleBadgeClick(badge)}
                                                                                                Sao Paulo
                                                                                                                                    Johannesburg
           {badge.text}
                                                                                                           Esri, TomTom, FAO, NOAA, USGS
        </div>
      ))]
      {selectedBadge && <BadgePopup badge={selectedBadge} onClose={() => setSelectedBadge(null)} />}
    </div>
  );
```

Powered by Est

```
const ProgressBar = ({ points, maxPoints }) => {
  const progress = Math.min((points / maxPoints) * 100, 100);
  return (
    <div className="progress-bar">
                                                        user1's Profile
      <div
                                                                                                 Badges
        className="progress-bar-fill"
        style={{ width: `${progress}%` }}
                                                                                             Achievements
      ></div>
      <span className="progress-bar-text">
                                                                                           Visited Locations
        {points} / {maxPoints} Points
      </span>
                                                                          Show Top Players List
    </div>
```

```
const showAchievementPopupMessage = (points, text) => {
   setAchievementMessage(`Congrats! You've earned ${points} points for ${text}`);
   setShowAchievementPopup(true);
   setTimeout(() => {
       setShowAchievementPopup(false);
   }, 3000);
};
```



```
const [isPending, startTransition] = useTransition();

useEffect(() => {
    earnedAchievements.forEach((achievement) => {
        if (!shownAchievements.includes(achievement.text)) {
            startTransition(() => {
                queueAchievementPopup(achievement.points, achievement.text);
                setShownAchievements((prev) => [...prev, achievement.text]);
        });
    }
});
}, [earnedAchievements, shownAchievements]);
```



User earns achievement





React begins state update



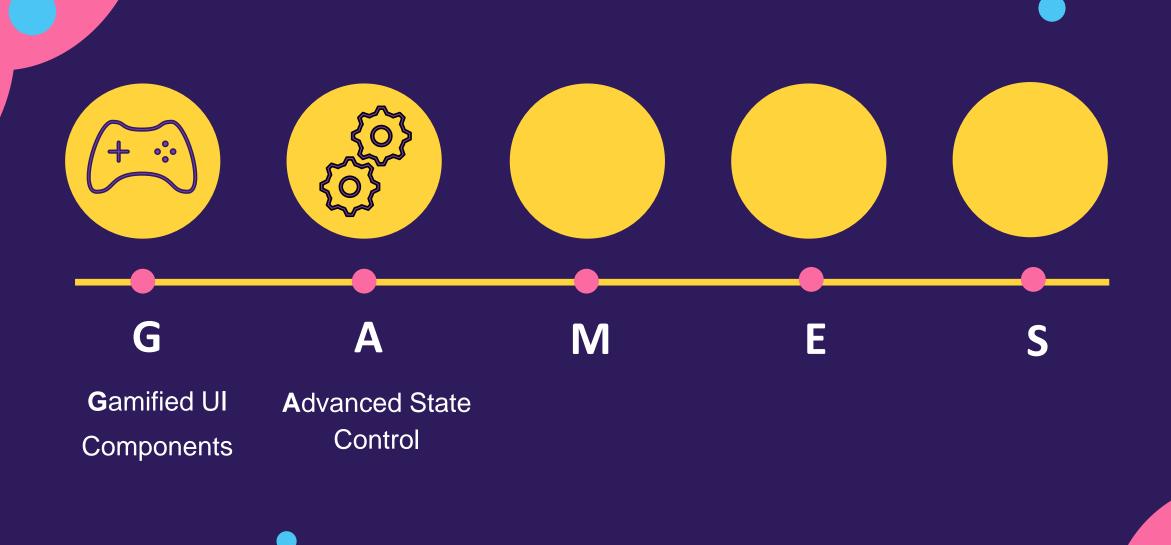


State update wrapped in useTransition





UI remains responsive



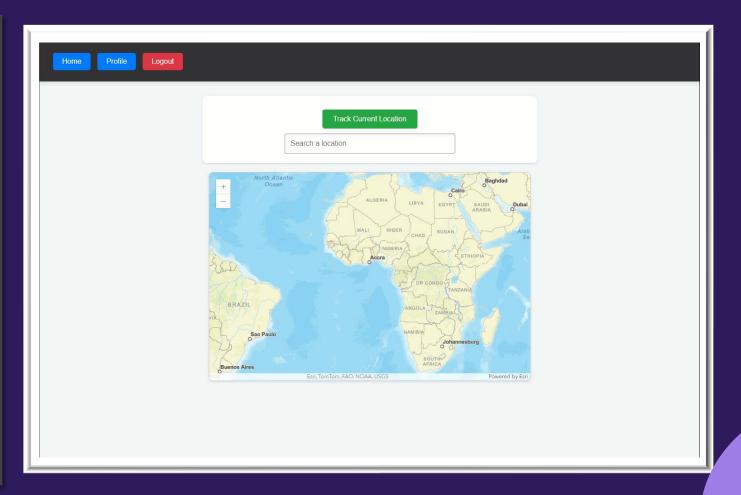
### Advanced State Control

```
import { createContext, useReducer, useContext } from "react";
 const AppContext = createContext();
const initialState = { ···
 };
 const reducer = (state, action) => {
   switch (action.type) { ···=
 };
 export const AppProvider = ({ children }) => {
   const [state, dispatch] = useReducer(reducer, initialState);
   return (
     <AppContext.Provider value={{ state, dispatch }}>
       {children}
     </AppContext.Provider>
 };
 export const useAppContext = () => useContext(AppContext);
```

```
switch (action.type) {
   case "SET_LOCATION":
      return { ...state, location: action.payload };
   case "SET_LOCATION_INPUT":
      return { ...state, locationInput: action.payload };
   case "SET_SUBMITTED":
      return { ...state, submitted: action.payload };
   case "RESET":
      return initialState;
   case "SET_SELECTED_BADGE":
      return { ...state, selectedBadge: action.payload };
   default:
      throw new Error(`Unhandled action type: ${action.type}`);
}
```

### **A**dvanced State Control

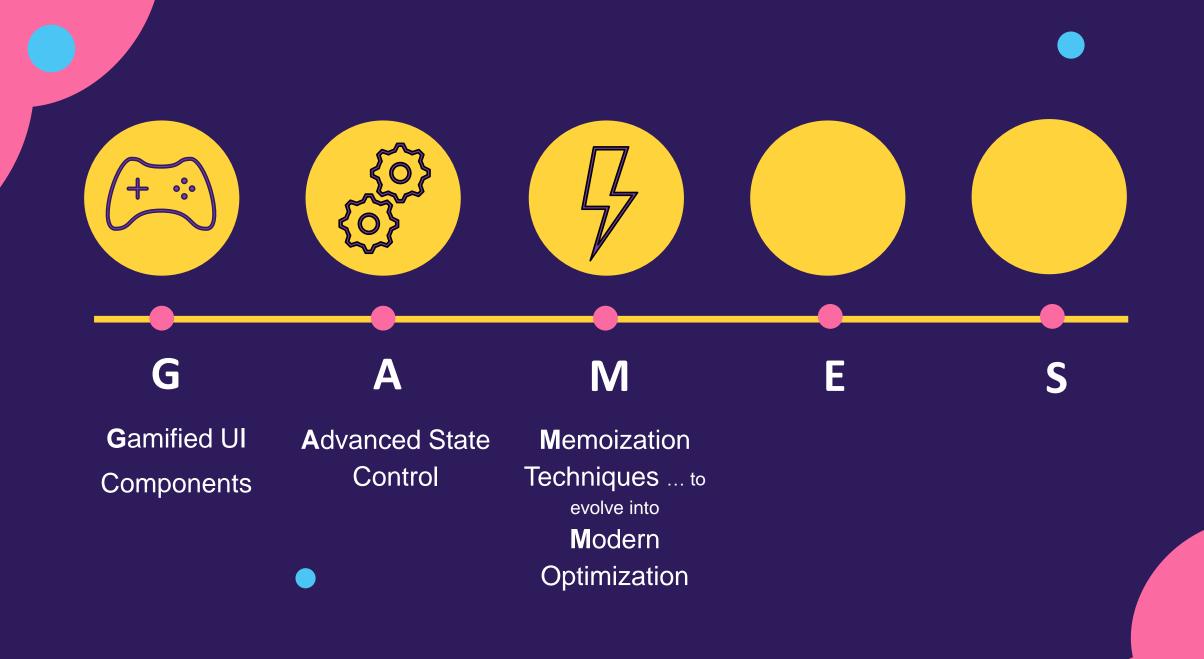
```
const handleStartQuest = () => {
  if (questEnabled) {
    setQuestStarted(true);
const handleEndQuest = () => {
 setQuestStarted(false);
 // Reset other quest-related states
};
// UI elements controlled by questStarted
{questStarted && (
  <button onClick={handleEndQuest}>
    End Quest
  </button>
```



### **A**dvanced State Control

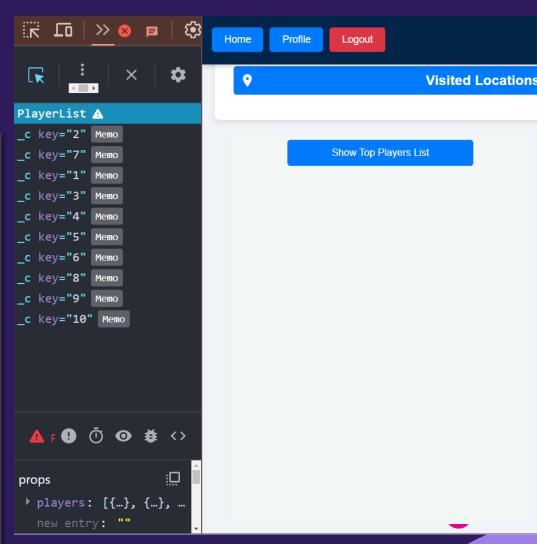
```
const handleSaveProgressAction = async (previousState) => {
  localStorage.setItem("questProgress", JSON.stringify(previousState));
  await syncProgressToServer(previousState);
  return { ...previousState, progressSaved: true };
};

const [saveProgressState, saveProgressAction] = useActionState handleSaveProgressAction, {
  questStarted: questStarted,
  foundLandmarks: foundLandmarks,
   currentLandmarkIndex: currentLandmarkIndex,
});
```



### Memoization Techniques/Modern Optimization

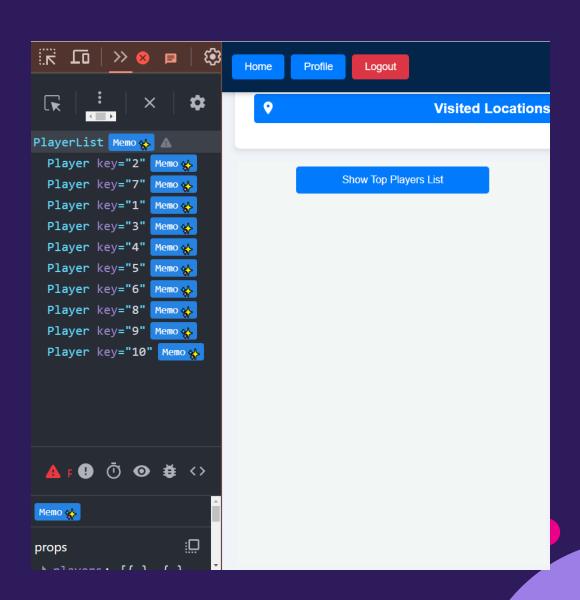
```
const Player = React.memo(({ player }) => {
  console.log(`Rendering ${player.name}`);
  return (
    <1i>>
      {player.name}: {player.score}
   import { useMemo } from 'react';
                          import Player from './Player';
});
                          const PlayerList = ({ players }) => {
export default Player;
                            const sortedPlayers = useMemo(() => {
                              return players.sort((a, b) => b.score - a.score);
                            }, [players]);
                            return (
                              <div className="player-list">
                                <h3>Top Players</h3>
                                <u1>
                                  {sortedPlayers.map(player => (
                                   <Player key={player.id} player={player} />
                                  ))}
                                </div>
                            );
                          };
                          export default PlayerList;
```

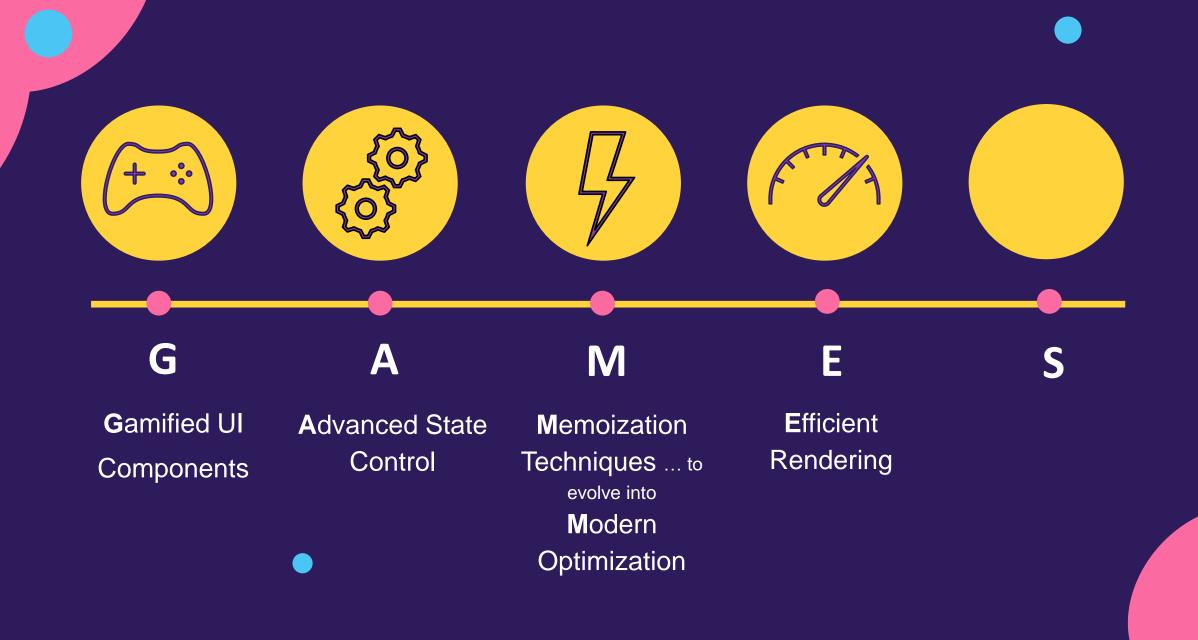


### Memoization Techniques/Modern Optimization

```
import { Ds=Marko } from 'react';
import Player from './Player';

const PlayerList = ({ players }) => {
   const sortedPlayers = uscMerro(() => {
     return players.sort((a, b) => b.score - a.score);
   }, [players]);
```





# **E**fficient Rendering

```
const MapViewComponent = lazy(() => import("./components/MapViewComponent"));
const SimpleMapComponent = lazy(() => import("./components/SimpleMapComponent"));
const DemographicData = lazy(() => import("./components/DemographicData"));
const DemographicData = lazy(() => import("./components/D
```

# **E**fficient Rendering

```
{showLeaderboard && <Leaderboard />}
{selectedBadge && (
   <BadgeModal</pre>
      badge={selectedBadge}
      onClose={() => dispatch({ type: 'SET_SELECTED_BADGE', payload: null })}
                                                                                              Statistics:
                                                                                          Total Population 2: 245270
                                                                                       Volunteers 77122 volunteered last year
                                                                                  Pet Owners : 47874 (out of 133716) households have pets
                                                                             Married Adults &: 105533 are married (out of the 217083 who are 15+ years old)
```

Educated Residents : 74571 have at least a bachelor's degree (out of the 196039 who

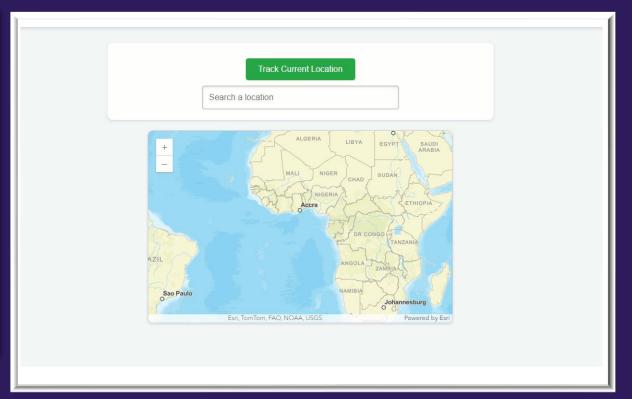
Badges Earned: Large Population Area

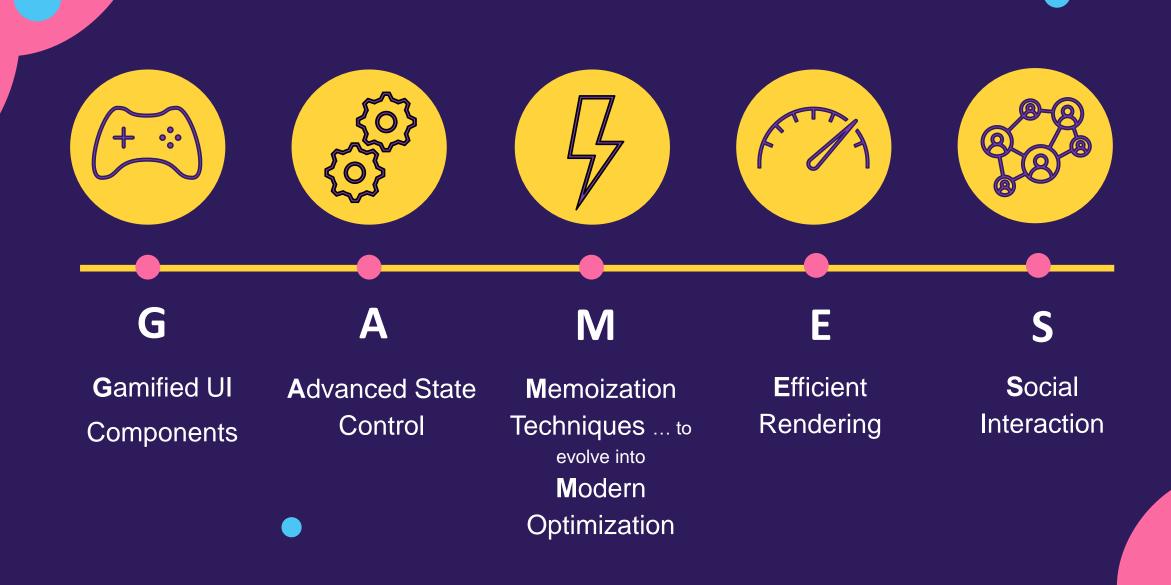
### Efficient Rendering

```
useEffect(() => {
  if (!location || !mapDiv.current) return;

const map = new Map({ basemap: 'streets' });
const view = new MapView({
    container: mapDiv.current,
    map: map,
    center: [location.longitude, location.latitude],
    zoom: 12,
  });

return () => view.destroy();
}, [location, landmarks]);
```



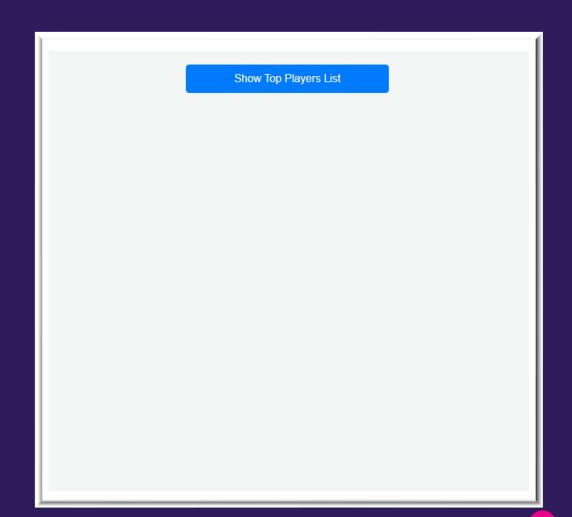


### **S**ocial Interaction

```
const ShareButtons = ({ url, message }) => {
  console.log("ShareButtons rendered with URL:", url, "and message:", message);
  return (
    <div className="share-buttons">
      <TwitterShareButton url={url} title={message}>
        <TwitterIcon size={32} round />
                                                                      user1's Profile
      </TwitterShareButton>
                                                                                                            Badges
      <FacebookShareButton url={url} quote={message}>
        <FacebookIcon size={32} round />
                                                                                                        Achievements
      </FacebookShareButton>
                                                                                                       Visited Locations
      {/* other share buttons */}
    </div>
                                                                                       Show Top Players List
```

### **S**ocial Interaction

```
const PlayerList = ({ players }) => (
 <div className="player-list">
   <h3>Top Players</h3>
   <l
    {players.map(player => (
      {player.name} - {player.score}
      ))}
   </div>
```



G

Streak Counters & XP Bars

A

Adaptive Learning System

M

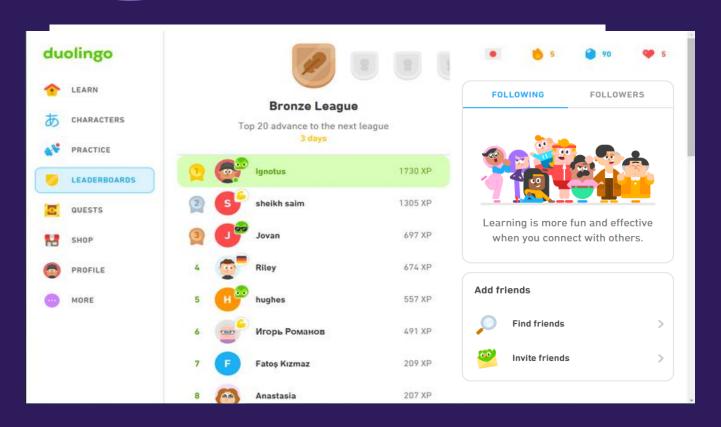
Optimized Lesson Rendering

Ε

**Efficient Component Rendering** 

S

Leaderboards & Challenges





# GAIMES

# Thank you, React Rally!

### Courtney Yatteau

- in courtneyyatteau

