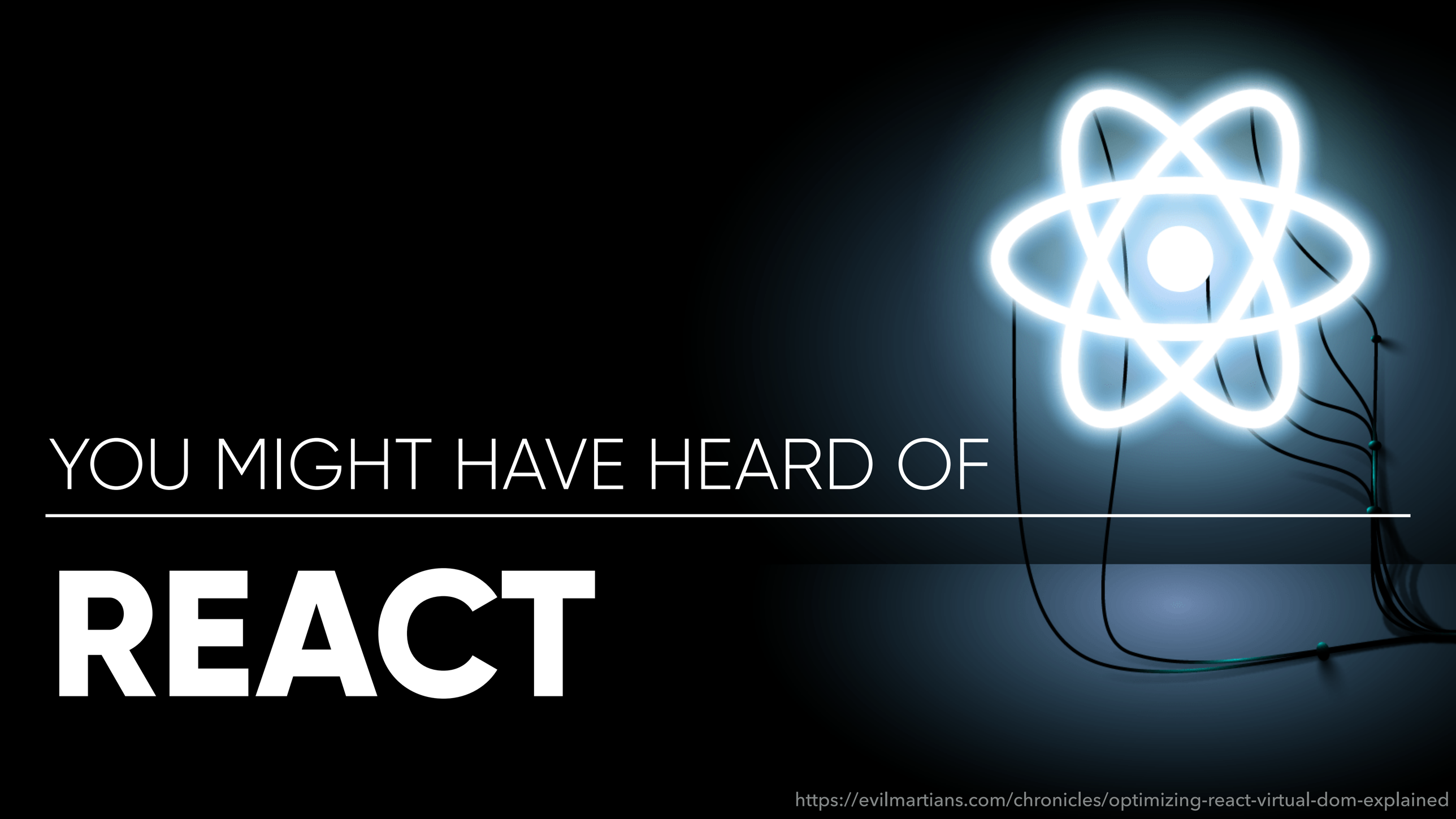
REFACTORING

OFF THE HOOKS

YOU MIGHT HAVE HEARD OF

REACT



https://evilmartians.com/chronicles/optimizing-react-virtual-dom-explained

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

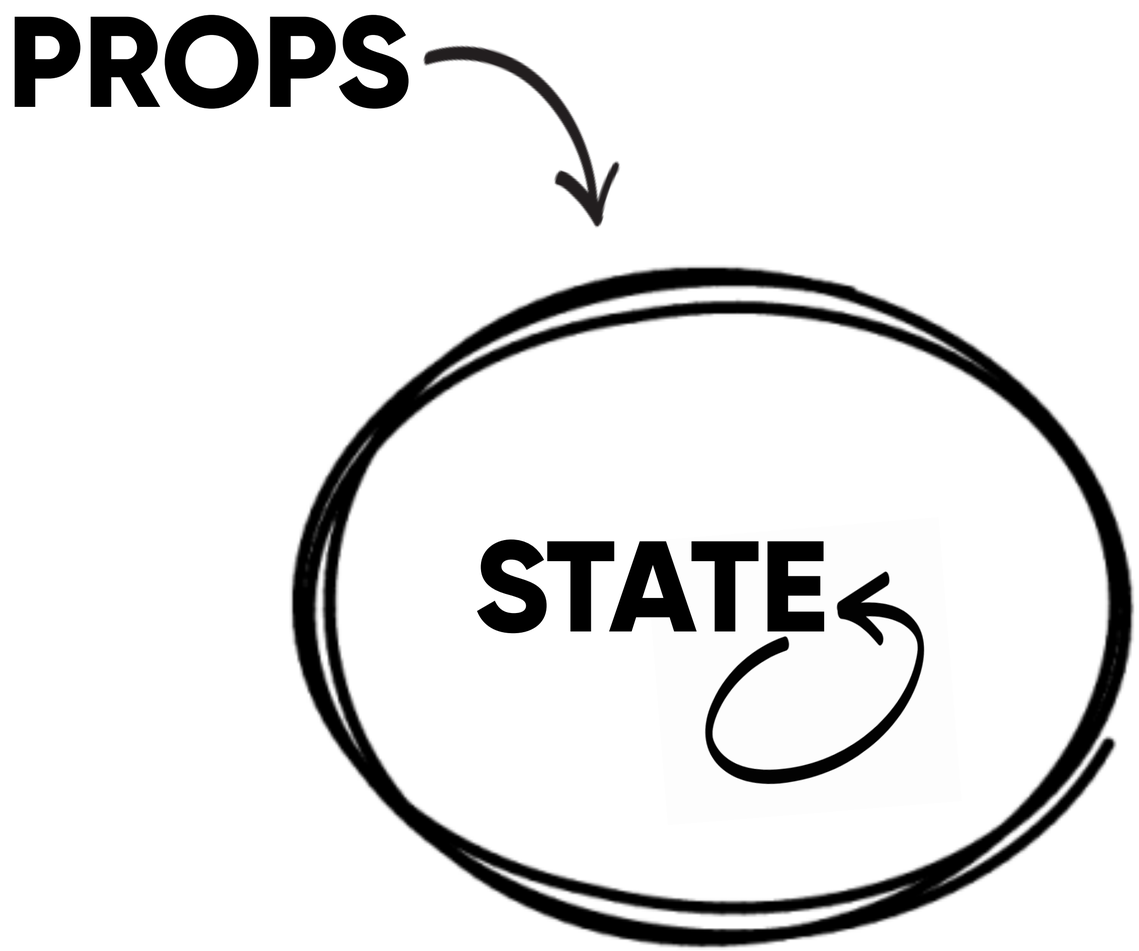
1. DECLARATIVE,
2. COMPONENT-BASED,
3. LEARN ONCE, WRITE EVERYWHERE

react docs

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

1. DECLARATIVE,
2. COMPONENT-BASED,
3. LEARN ONCE, WRITE EVERYWHERE

react docs



https://thenounproject.com/indygo/collection/hand-drawn-arrows-4/

A SIMPLE FUNCTION COMPONENT

```
const HelloMessage = (props) => (
  <div>
    Hello {props.name}
  </div>
);

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

A SIMPLE CLASS COMPONENT

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

NEITHER MODEL REALLY CAPTURES REACT.

https://dan.church

WHY ARE THESE MODELS INSUFFICIENT TO DESCRIBE REACT?

dan abramov

"PURE FUNCTION" MODEL DOESN'T DESCRIBE LOCAL STATE WHICH IS AN ESSENTIAL REACT FEATURE.

dan abramov

"CLASS" MODEL DOESN'T EXPLAIN PURE-ISH RENDER, DISAWOVING INHERITANCE, LACK OF DIRECT INSTANTIATION, AND "RECEIVING" PROPS.

dan abramov

TWITTER

EMAIL: marco@cedmax.com

NAME

WEBSITE

PRONOUNS: HE/HIM

WE'RE ARE GOING TO TALK ABOUT

COLOURS

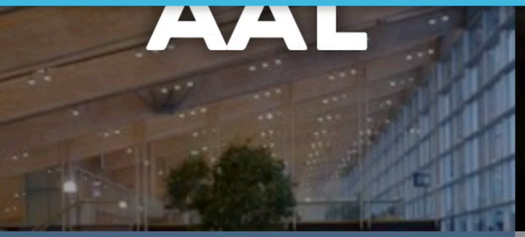
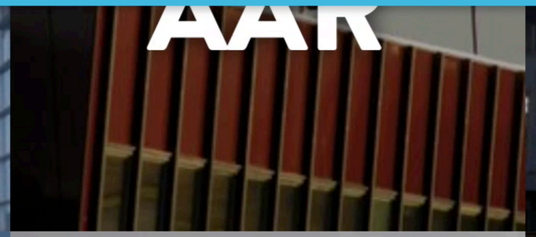
ART, THE WEB, and TINY UX

Heyo

LYNN FISHER

@LYNNANDTONIC

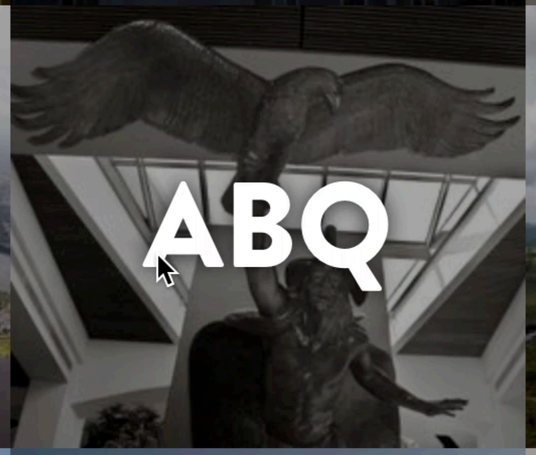
Find your airport



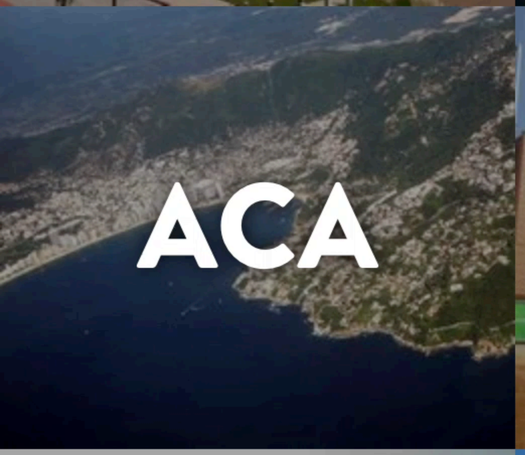
ABI


ABJ


ABQ


ABV

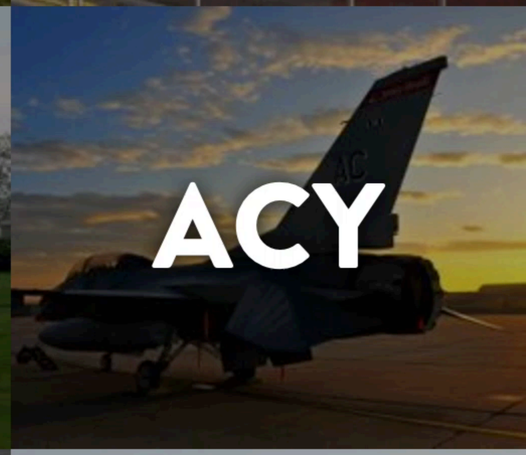
ABZ

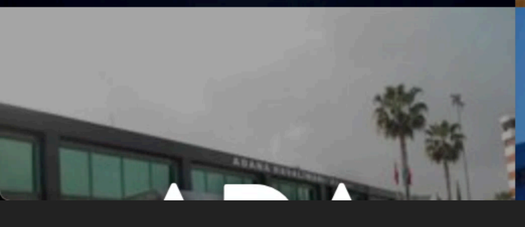
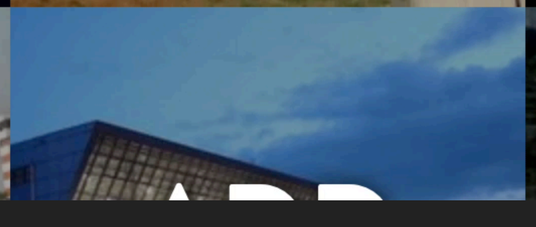
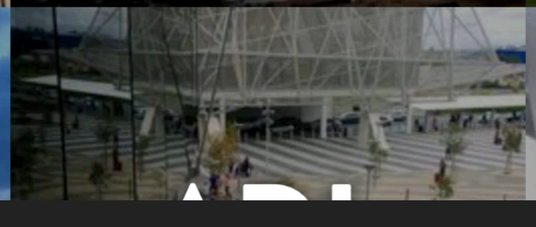
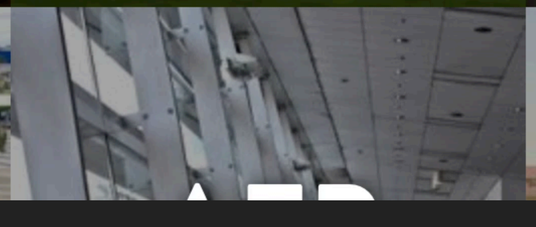

ACA


ACC


ACE


ACK


ACY


Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export

Create a book
Download as PDF
Printable version

Languages

Ελληνικά
سنڌي
اردو

Edit links

Colors in alphabetical order [edit]

Colors

Name	Hex (RGB)	Red (RGB)	Green (RGB)	Blue (RGB)	Hue (HSL/HSV)	Satur. (HSL)	Light (HSL)	Satur. (HSV)	Value (HSV)
Absolute Zero	#0048BA	0%	28%	73%	217°	100%	37%	100%	73%
Acid green	#B0BF1A	69%	75%	10%	65°	76%	43%	76%	43%
Aero	#7CB9E8	49%	73%	91%	206°	70%	70%	47%	91%
Aero blue	#C9FFE5	79%	100%	90%	151°	100%	89%	100%	89%
Air superiority blue	#72A0C1	45%	63%	76%	205°	39%	60%	41%	76%
Alabaster	#EDEAE0	93%	92%	88%	46°	27%	90%	6%	93%
Alice blue	#F0F8FF	94%	97%	100%	208°	100%	97%	6%	100%
Alien Armpit	#C5E17A	77%	88%	48%	76°	63%	68%	46%	88%
Alloy orange	#C46210	77%	38%	6%	27°	85%	42%	92%	77%
Almond	#EFDECD	94%	87%	80%	30°	52%	87%	14%	94%
Amazon	#3B7A57	23%	48%	34%	147°	35%	36%	52%	48%
Amber	#FFBF00	100%	75%	0%	45°	100%	50%	100%	100%
Amethyst	#9966CC	60%	40%	80%	270°	50%	60%	50%	80%
Anti-flash white	#F2F3F4	95%	95%	96%	210°	8%	95%	1%	96%
Antique brass	#CD9575	80%	58%	46%	22°	47%	63%	43%	80%
Antique bronze	#665D1E	40%	36%	12%	53°	55%	26%	71%	40%
Antique fuchsia	#915C83	57%	36%	51%	316°	22%	46%	37%	57%
Antique ruby	#841B2D	52%	11%	18%	350°	66%	31%	80%	52%
Antique white	#FAEBD7	98%	92%	84%	34°	78%	91%	14%	98%
Ao (English)	#008000	0%	50%	0%	120°	100%	25%	100%	50%
Apple green	#8DB600	55%	71%	0%	74°	100%	36%	100%	71%

Colour Name / *Css Name*

#Hex **Sort by:** Name **Filter:**

- Absolute Zero *#0048BA*
- Acid *#B0BF1A*
- Aero *#7CB9E8*
- Aero blue *#C9FFE5*
- Air superiority blue *#72A0C1*
- Alabaster *#EDEAE0*
- Alice blue / *AliceBlue* *#F0F8FF*
- Alien Armpit *#C5E17A*
- Alloy orange *#C46210*
- Almond *#EFDECD*
- Amazon *#3B7A57*
- Amber *#FFBF00*
- Amethyst *#9966CC*
- Anti-flash white *#F2F3F4*
- Antique brass
- Fiery rose *#FF5470*
- Firebrick / *FireBrick* *#B22222*
- Fire engine red *#CE2029*
- Fire opal *#E95C4B*
- Flame *#E25822*
- Flax *#EEDC82*
- Flesh *#FFE9D1*
- Flickr Blue *#0063DC*
- Flickr Pink *#FB0081*
- Flirt *#A2006D*
- Floral white / *FloralWhite* *#FFFAF0*
- Fluorescent blue *#15F4EE*
- Forest green (Crayola) *#5FA777*
- Forest green (traditional) *#014421*
- Forest green (web) / *ForestGreen* *#228B22*
- French bistre *#856D4D*
- French blue
- Orchid / *Orchid* *#DA70D6*
- Orchid pink *#F2BDCD*
- Orchid (Crayola) *#E29CD2*
- Outer space (Crayola) *#2D383A*
- Outrageous Orange *#FF6E4A*
- Oxford blue *#002147*
- OU Crimson red *#841617*
- Pacific blue *#1CA9C9*
- Pakistan green *#006600*
- Palatinate purple *#682860*
- Pale cerulean *#9BC4E2*
- Pale pink *#FADADD*
- Pale purple (Pantone) *#FAE6FA*
- Pale silver *#C9C0BB*
- Pale spring bud *#ECEBBD*
- Pansy purple *#78184A*
- Paolo Veronese green

COMPONENT STRUCTURE

```
export default class App extends Component {
  state = {
    colors: originalList,
    currentFilter: "",
    currentSortBy: "name",
    style: { ...constants, color: "black", background: "white" },
  };
  Type a quote here.

  sortBy = sortBy => {
    this.setState({
      currentSortBy: sortBy,
      colors: sorters[sortBy](this.state.colors),
    });
  };

  filter = currentFilter=> {
    const colors = getFilteredColours(originalList, currentFilter);
    this.setState({ currentFilter, colors });
  };
```

COMPONENT STRUCTURE

```
  onColorChange = hex => {
    this.setState({
      style: {
        ...this.state.style,
        background: hex,
        color: getMostReadable(hex),
      },
    });
  };

  return (
    <AppUI
      {...state}
      sortBy={sortBy}
      filter={filter}
      onColorChange={onColorChange}
    />
  );
};
```

LET'S CODE

HOOKS

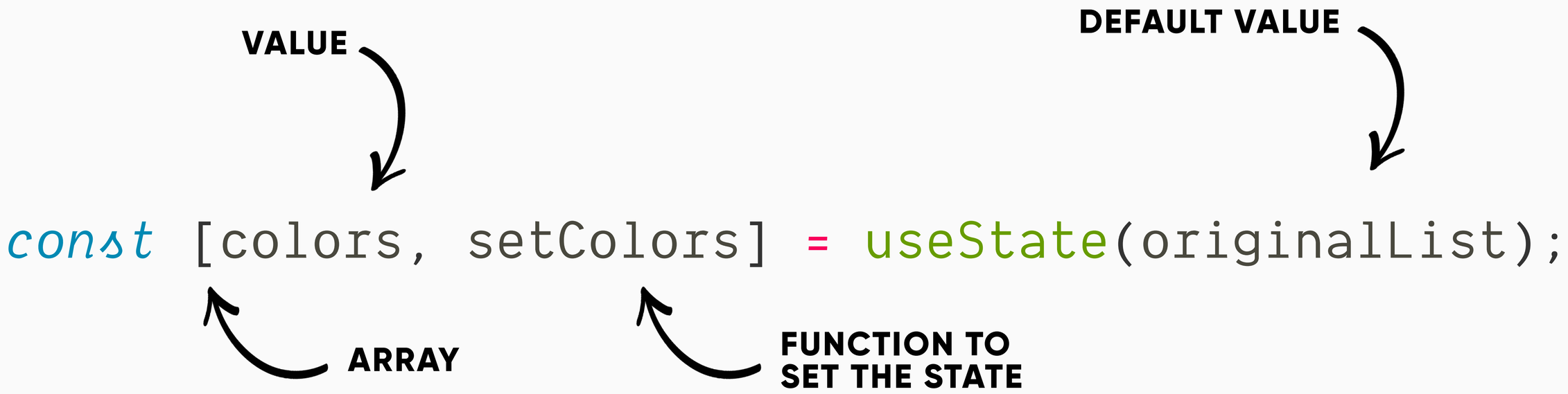
A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

1. OPT-IN
2. 100% BACKWARDS-COMPATIBLE
3. AVAILABLE NOW
   HOOKS WERE RELEASED WITH REACT V16.8.0.

react docs

USESTATE

USESTATE


VALUE
DEFAULT VALUE
const [colors, setColors] = useState(originalList);
ARRAY
FUNCTION TO
SET THE STATE

USESTATE

VALUE

DEFAULT VALUE

```
const [colors, setColors] = useState(originalList);
```

ARRAY

FUNCTION TO SET THE STATE CAN TAKE A FUNCTION...

```
setColors(sortingFunction);
```

USESTATE

VALUE

DEFAULT VALUE

```
const [colors, setColors] = useState(originalList);
```

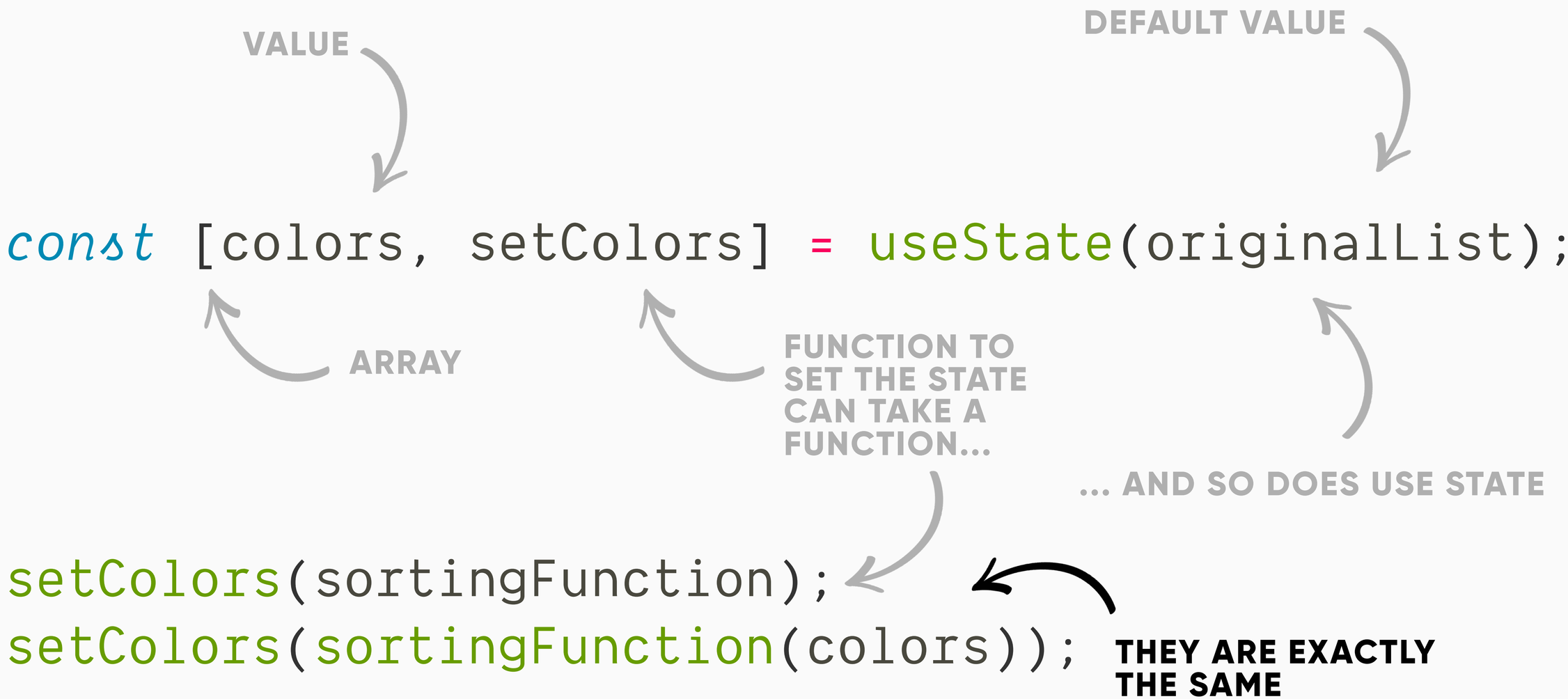
ARRAY

FUNCTION TO SET THE STATE CAN TAKE A FUNCTION...

... AND SO DOES USE STATE

```
setColors(sortingFunction);
```

USESTATE


VALUE
DEFAULT VALUE
const [colors, setColors] = useState(originalList);
ARRAY
FUNCTION TO SET THE STATE CAN TAKE A FUNCTION...
... AND SO DOES USE STATE
setColors(sortingFunction);
setColors(sortingFunction(colors));
THEY ARE EXACTLY THE SAME

```
setCurrentFilter(currentFilter);
setColors(colors);
```

COMPONENTS UPDATES
GET ENQUEUED

USECALLBACK

USECALLBACK

WRAPS A FUNCTION RETURNING
A MEMOIZED VERSION

```
const sortBy = useCallback(e => {
  const { value: sortBy } = e.target;
  const sortingFunction = sorters[sortBy];
  setColors(sortingFunction);
}, []);
```

USEREDUCER

STILL QUITE BUSY

```
export default () => {
  const [colors, setColors] = useState(originalList);
  const [currentFilter, setCurrentFilter] = useState("");
  const [currentSortBy, setCurrentSortBy] = useState("name");
  const [style, setStyle] = useState({
    ...constants,
    color: "black",
    background: "white",
  });

  const sortBy = useCallback(sortBy => {
    const sortingFunction = sorters[sortBy];
    setCurrentSortBy(sortBy);
    setColors(sortingFunction);
  }, []);

  const filter = useCallback(currentFilter => {
    const colors = getFilteredColors(originalList, currentFilter);
    setCurrentFilter(currentFilter);
    setColors(colors);
  }, []);
```

STILL QUITE BUSY

```
  const onColorChange = useCallback(hex => {
    setStyle({
      ...style,
      background: hex,
      color: getMostReadable(hex),
    });
  }, []);

  return (
    <AppUI
      {...state}
      sortBy={sortBy}
      filter={filter}
      onColorChange={onColorChange}
    />
  );
};
```

USEREDUCER


VALUE
DEFAULT VALUE
const [state, dispatch] = useReducer(reducer, defaultState);
ARRAY
REDUCERS

USEREDUCER

```
export default (state, { type, payload }) => {
  switch (type) {
    case "filter":
      return {
        ...state,
        currentFilter: payload,
        colors: getFilteredColors(state.allColors, payload),
      };
    case "sort":
      return {
        ...state,
        currentSortBy: payload,
        colors: sorters[payload](state.colors),
      };
    case "change":
      return {
        ...state,
        style: {
          ...state.style,
          background: payload,
          color: getMostReadable(payload),
        },
      };
    default:
      return state;
  }
};
```

USEREDUCER

```
const [state, dispatch] = useReducer(reducer, defaultState);

const emit = useCallback((type, payload) => dispatch({type, payload}), []);
const sortBy = useCallback(sortBy => emit("sort", sortBy), []);
const filter = useCallback(filter => emit("filter", filter), []);
const onColorChange = useCallback(hex => emit("change", hex), []);
```

USEREDUCER

```
const [state, dispatch] = useReducer(reducer, defaultState);

const emit = useCallback((type, payload) => dispatch({type, payload}), []);
const sortBy = useCallback(sortBy => emit("sort", sortBy), []);
const filter = useCallback(filter => emit("filter", filter), []);
const onColorChange = useCallback(hex => emit("change", hex), []);
```

USEREDUCER

```
export default () => {
  const [state, dispatch] = useReducer(reducers, defaultState);

  const emit = useCallback((type, payload) => dispatch({ type, payload }), []);
  const sortBy = useCallback(sortBy => emit("sort", sortBy), []);
  const filter = useCallback(filter => emit("filter", filter), []);
  const onColorChange = useCallback(hex => emit("change", hex), []);

  return (
    <AppUI
      {...state}
      sortBy={sortBy}
      filter={filter}
      onColorChange={onColorChange}
    />
  );
};
```

USEEFFECT

ACCEPTS A FUNCTION THAT CONTAINS IMPERATIVE, POSSIBLY EFFECTFUL CODE.

react docs

THINK OF EFFECTS AS AN ESCAPE HATCH FROM REACT'S PURELY FUNCTIONAL WORLD INTO THE IMPERATIVE WORLD.

react docs

NETWORK (DATA FETCHING...), **DOM OR WINDOW** (TITLE UPDATES, SUBSCRIBE TO WINDOW RESIZE OR MOUSE EVENTS, ACCESS LOCAL STORAGE), **LOGGING** (ANALYTICS...)

FETCHING DATA

```
useEffect(() => {
  const fetchData = async () => {
    const { data } = await axios.get(`/api/colors/${props.id}`);
    setColor(data);
  };
  fetchData();
}, [props.id]);
```

FETCHING DATA

NOT BEING ABLE TO USE AN ASYNC FUNCTION WAS QUITE A GOTCHA FOR ME

```
useEffect(() => {
  const fetchData = async () => {
    const { data } = await axios.get(`/api/colors/${props.id}`);
    setColor(data);
  };
  fetchData();
}, [props.id]);
```

FETCHING DATA

NOT BEING ABLE TO USE AN ASYNC FUNCTION WAS QUITE A GOTCHA FOR ME

```
useEffect(() => {
  const fetchData = async () => {
    const { data } = await axios.get(`/api/colors/${props.id}`);
    setColor(data);
  };
  fetchData();
}, [props.id]);
```

LIST OF DEPENDENCIES

THE EMPTY DEPENDENCY LIST GUARANTEES IT'S EXECUTED ONLY ONCE NO MATTER THE NUMBER OF RE-RENDERS

```
//componentDidMount
useEffect(() => console.log('mounted'), []);


//componentDidUnmount
useEffect(() => () => {
  console.log('will unmount');
}, []);
```

THE EMPTY DEPENDENCY LIST GUARANTEES IT'S EXECUTED ONLY ONCE NO MATTER THE NUMBER OF RE-RENDERS

```
//componentDidMount
useEffect(() => console.log('mounted'), []);
```

IF THE CALLBACK RETURNS A FUNCTION, IT WILL BE CALLED BEFORE THE COMPONENT IS REMOVED FROM THE UI.

```
//componentDidUnmount
useEffect(() => () => {
  console.log('will unmount');
}, []);
```

THE QUESTION IS NOT "WHEN DOES THIS EFFECT RUN"

THE QUESTION IS "WITH WHICH STATE DOES THIS EFFECT SYNCHRONIZE WITH"

ryan florence

WHAT STATE THE EFFECT SYNCS TO?

```
useEffect(() => console.log('mounted'));
```

ALL STATE CHANGES

```
useEffect(() => console.log(''), []);
```

```
useEffect(
  () => console.log(`fetch ${props.id}`),
  [props.id]
);
```

WHAT STATE THE EFFECT SYNCS TO?

```
useEffect(() ⇒ console.log('mounted'));
```

ALL STATE CHANGES

```
useEffect(() ⇒ console.log(''), []);
```

NO STATE CHANGES

```
useEffect(
  () ⇒ console.log(`fetch ${props.id}`),
  [props.id]
);
```

WHAT STATE THE EFFECT SYNCS TO?

```
useEffect(() => console.log('mounted'));
```

ALL STATE CHANGES

```
useEffect(() => console.log(''), []);
```

NO STATE CHANGES

```
useEffect(
  () => console.log(`fetch ${props.id}`),
  [props.id]
);
```

PROP ID CHANGES

UPDATE THE URL WITH USEEFFECT

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

MATCH URL TO STATE

```
export const listenToHistory = callback => {
  const getQs = () => qs.parse(window.location.search);
  window.addEventListener("popstate", () => callback(getQs()));
  callback(getQs());
};
```

THE CALLBACK GETS INVOKED IMMEDIATELY AND IN RESPONSE OF ANY POPSTATE EVENT WITH THE PARAMETERS IN THE QUERYSTRING

```
listenToHistory(data => {
  const { currentFilter, currentSortBy } = data; // from the URL
  sortBy(currentSortBy || defaultSortBy);
  filter(currentFilter || defaultFilter);
});
```

MATCH URL TO STATE

```
export const listenToHistory = callback => {
  const getQs = () => qs.parse(window.location.search);
  window.addEventListener("popstate", () => callback(getQs()));
  callback(getQs());
};
```

THE CALLBACK GETS INVOKED IMMEDIATELY AND IN RESPONSE OF ANY POPSTATE EVENT WITH THE PARAMETERS IN THE QUERYSTRING

```
listenToHistory(data => {
  const { currentFilter, currentSortBy } = data; // from the URL
  sortBy(currentSortBy || defaultSortBy);
  filter(currentFilter || defaultFilter);
});
```

IN THE CALLBACK WE SET THE STATE FOR ANY CORRESPONDING PARAMETER IN THE URL

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

AND WE MAKE SURE IT HAPPENS ONLY THE FIRST TIME

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

AND WE MAKE SURE IT HAPPENS ONLY THE FIRST TIME

AFTER THAT SETUP, WE CAN PUSH THE STATE CHANGES TO THE HISTORY

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

MATCH URL TO STATE

```
window.history.pushState({}, "", `?${qs.stringify(qsObj)}`);
```

MATCH URL TO STATE

```
const [urlRestored, setUrlRestored] = useState(false);

useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

LET'S SEE IT WORKING

CUSTOM HOOKS

CUSTOM HOOK

```
useQueryString(
  { currentSortBy, currentFilter },
  { currentSortBy: sortBy, currentFilter: filter },
  { currentSortBy: defaultSortBy, currentFilter: defaultFilter }
);
```

CUSTOM HOOK

```
useQueryString(
  { currentSortBy, currentFilter },
  { currentSortBy: sortBy, currentFilter: filter },
  { currentSortBy: defaultSortBy, currentFilter: defaultFilter }
);
```

CUSTOM HOOK


```
useQueryString(
  { currentSortBy, currentFilter },
  { currentSortBy: sortBy, currentFilter: filter },
  { currentSortBy: defaultSortBy, currentFilter: defaultFilter }
);
```

CUSTOM HOOK

```
useQueryString(
  { currentSortBy, currentFilter },
  { currentSortBy: sortBy, currentFilter: filter },
  { currentSortBy: defaultSortBy, currentFilter: defaultFilter }
);
```

VALUES

SETTERS, MATCHING THE VALUES KEYS

DEFAULT VALUES, MATCHING THE VALUES KEYS

CUSTOM HOOK

```
useQueryString(
  { currentSortBy, currentFilter },
  { currentSortBy: sortBy, currentFilter: filter },
  { currentSortBy: defaultSortBy, currentFilter: defaultFilter }
);
```

```
useQueryString(
  [currentSortBy, sortBy, defaultSortBy],
  [currentFilter, filter, defaultFilter],
);
```

CUSTOM HOOK

```
const [urlRestored, setUrlRestored] = useState(false);
useEffect(() => {
  if (!urlRestored) {
    listenToHistory(data => {
      const { currentFilter, currentSortBy } = data; // from the URL
      sortBy(currentSortBy || defaultSortBy);
      filter(currentFilter || defaultFilter);
    });
    setUrlRestored(true);
  } else {
    objectToHistory({
      currentSortBy,
      currentFilter,
    });
  }
}, [currentSortBy, currentFilter]);
```

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        const { currentFilter, currentSortBy } = data; // from the URL
        sortBy(currentSortBy || defaultSortBy);
        filter(currentFilter || defaultFilter);
      });
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, [currentSortBy, currentFilter]);
};
```

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        const { currentFilter, currentSortBy } = data; // from the URL
        sortBy(currentSortBy || defaultSortBy);
        filter(currentFilter || defaultFilter);
      });
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, [currentSortBy, currentFilter]);
};
```

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        const { currentFilter, currentSortBy } = data; // from the URL
        sortBy(currentSortBy || defaultSortBy);
        filter(currentFilter || defaultFilter);
      });
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, Object.values(values));
};
```

OUR DEPENDENCIES ARE THE VALUES WE WANT TO PERSIST IN THE URL

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        const { currentFilter, currentSortBy } = data; // from the URL
        sortBy(currentSortBy || defaultSortBy);
        filter(currentFilter || defaultFilter);
      });
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, Object.values(values));
};
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k]; // in the URL
          setters[k](value || defaults[k]);
        })});
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, Object.values(values));
};
```

ON FIRST LOAD WE NEED TO RESTORE THE STATE FROM THE URL AND SUBSCRIBE TO THE HISTORY EVENTS

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k]; // in the URL
          setters[k](value || defaults[k]);
        })
      });
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, Object.values(values));
};
```

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k]; // in the URL
          setters[k](value || defaults[k]);
        })
      });
      setUrlRestored(true);
    } else {
      objectToHistory({
        currentSortBy,
        currentFilter,
      });
    }
  }, Object.values(values));
};
```

AFTER THAT SETUP, WE CAN PUSH THE STATE CHANGES TO THE HISTORY

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);
  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k]; // in the URL
          setters[k](value || defaults[k]);
        })
      });
      setUrlRestored(true);
    } else {
      objectToHistory(values);
    }
  }, Object.values(values));
};
```

AFTER THAT SETUP, WE CAN PUSH THE STATE CHANGES TO THE HISTORY

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);

  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k]; // in the URL
          setters[k](value || defaults[k]);
        })
      });
      setUrlRestored(true);
    } else {
      objectToHistory(values);
    }
  }, Object.values(values));
};
```

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [urlRestored, setUrlRestored] = useState(false);

  useEffect(() => {
    if (!urlRestored) {
      listenToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k]; // in the URL
          setters[k](value || defaults[k]);
        })
      });
      setUrlRestored(true);
    } else {
      objectToHistory(values);
    }
  }, Object.values(values));
};
```

LET'S MAKE SURE IT'S STILL WORKING

A STEP FORWARD

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

WE COULD STORE THE LISTENER INSTEAD OF A BOOLEAN

HAVE THE SUBSCRIBER RETURNING THE LISTENER

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

WE COULD STORE THE LISTENER INSTEAD OF A BOOLEAN

HAVE THE SUBSCRIBER RETURNING THE LISTENER

STORE THE LISTENER IN THE STATE

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

WE COULD STORE THE LISTENER INSTEAD OF A BOOLEAN

HAVE THE SUBSCRIBER RETURNING THE LISTENER

STORE THE LISTENER IN THE STATE

RETURN A FUNCTION TO CLEAR THE EFFECT, UNSUBSCRIBING FROM THE HISTORY EVENTS

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

USEREDUCER

```
export default () => {
  const [state, dispatch] = useReducer(reducers, defaultState);
  const emit = useCallback((type, payload) => dispatch({ type, payload }), []);
  const sortBy = useCallback(sortBy => emit("sort", sortBy), []);
  const filter = useCallback(filter => emit("filter", filter), []);
  const onColorChange = useCallback(hex => emit("change", hex), []);

  useQueryString(
    { currentSortBy: state.currentSortBy, currentFilter: state.currentFilter },
    { currentSortBy: sortBy, currentFilter: filter },
    defaultState
  );

  return (
    <AppUI
      {...state}
      sortBy={sortBy}
      filter={filter}
      onColorChange={onColorChange}
    />
  );
};
```

CUSTOM HOOK

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);
  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

CUSTOM HOOKS

TESTING

TESTING

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

TESTING

```
export const useQueryString = (values, setters, defaults) => {
  const [historyListener, setHistoryListener] = useState(null);

  useEffect(() => {
    if (!historyListener) {
      const listener = subscribeToHistory(data => {
        Object.keys(values).forEach(k => {
          const value = data[k];
          setters[k](value || defaults[k]);
        });
      });
      setHistoryListener(() => listener);
    } else {
      objectToHistory(values);
    }
    return () => unsubscribeFromHistory(historyListener);
  }, Object.values(values));
};
```

TESTING

```
jest.mock("./utils", () => ({
  objectToHistory: jest.fn(),
  subscribeToHistory: jest.fn(() => "listener function"),
  unsubscribeFromHistory: jest.fn(),
}));

const stateSetter = jest.fn();

const Component = ({ value }) => {
  useQueryString(
    { value },
    { value: stateSetter },
    { value: "defaultValue" }
  );
  return null;
};
```

TESTING

```
jest.mock("./utils", () => ({
  objectToHistory: jest.fn(),
  subscribeToHistory: jest.fn(() => "listener function"),
  unsubscribeFromHistory: jest.fn(),
}));

const stateSetter = jest.fn();

const Component = ({ value }) => {
  useQueryString(
    { value },
    { value: stateSetter },
    { value: "defaultValue" }
  );
  return null;
};
```

TESTING

```
test("the first time: subscribe to history with the right callback", () ⇒ {
  mount(<Component value="a" />);
  expect(subscribeToHistory).toBeCalledTimes(1);

  const popStateListener = subscribeToHistory.mock.calls[0][0];
  popStateListener({ value: "a" });
  expect(stateSetter).toBeCalledWith("a");

  callback({ value: "" });
  expect(stateSetter).toBeCalledWith("defaultValue");
});
```

TESTING

```
test("the first time: subscribe to history with the right callback", () => {
  mount(<Component value="a" />);
  expect(subscribeToHistory).toBeCalledTimes(1);

  const popStateListener = subscribeToHistory.mock.calls[0][0];
  popStateListener({ value: "b" });
  expect(stateSetter).toBeCalledWith("b");

  callback({ value: "" });
  expect(stateSetter).toBeCalledWith("defaultValue");
});
```

TESTING

```
test("the first time: subscribe to history with the right callback", () => {
  mount(<Component value="a" />);
  expect(subscribeToHistory).toBeCalledTimes(1);

  const popStateListener = subscribeToHistory.mock.calls[0][0];
  popStateListener({ value: "b" });
  expect(stateSetter).toBeCalledWith("b");

  callback({ value: "" });
  expect(stateSetter).toBeCalledWith("defaultValue");
});
```

TESTING

```
const wrapper = mount(<Component value="a" />);

test("further executions should invoke objectToHistory", () => {
  wrapper.setProps({ value: "b" });
  expect(subscribeToHistory).toBeCalledTimes(0);
  expect(objectToHistory).toBeCalledTimes(1);
});

test("further executions with the same props should do nothing", () => {
  wrapper.setProps({ value: "b" });
  expect(subscribeToHistory).toBeCalledTimes(0);
  expect(objectToHistory).toBeCalledTimes(0);
});

test("unmounting should invoke unsubscribeFromHistory", () => {
  wrapper.unmount();
  expect(unsubscribeFromHistory).toBeCalledTimes(1);
  expect(unsubscribeFromHistory).toBeCalledWith("listener function");
});
```

TESTING

```
const wrapper = mount(<Component value="a" />);

test("further executions should invoke objectToHistory", () => {
  wrapper.setProps({ value: "b" });
  expect(subscribeToHistory).toBeCalledTimes(0);
  expect(objectToHistory).toBeCalledTimes(1);
});

test("further executions with the same props should do nothing", () => {
  wrapper.setProps({ value: "b" });
  expect(subscribeToHistory).toBeCalledTimes(0);
  expect(objectToHistory).toBeCalledTimes(0);
});

test("unmounting should invoke unsubscribeFromHistory", () => {
  wrapper.unmount();
  expect(unsubscribeFromHistory).toBeCalledTimes(1);
  expect(unsubscribeFromHistory).toBeCalledWith("listener function");
});
```

TESTING

```
const wrapper = mount(<Component value="a" />);

test("further executions should invoke objectToHistory", () => {
  wrapper.setProps({ value: "b" });
  expect(subscribeToHistory).toBeCalledTimes(0);
  expect(objectToHistory).toBeCalledTimes(1);
});

test("further executions with the same props should do nothing", () => {
  wrapper.setProps({ value: "b" });
  expect(subscribeToHistory).toBeCalledTimes(0);
  expect(objectToHistory).toBeCalledTimes(0);
});

test("unmounting should invoke unsubscribeFromHistory", () => {
  wrapper.unmount();
  expect(unsubscribeFromHistory).toBeCalledTimes(1);
  expect(unsubscribeFromHistory).toBeCalledWith("listener function");
});
```

FAQS

WHY HOOKS AGAIN?

WHY HOOKS AGAIN?

THEY PROVIDE A BETTER MENTAL MODEL OF WHAT A COMPONENT IS.

WHY HOOKS AGAIN?

WITHOUT HOOKS: **46.15KB**
WITH HOOKS: **45.69KB**
~0.5KB REMOVING 1 CLASS

ARE THEY STABLE?

ARE THEY STABLE?

YES AND NO: THE BASIC ONES PROBABLY YES, BUT THEY MIGHT EVOLVE A BIT

Andrew Clark @acdlite · 27 Nov 2018

Intentionally underspecifying dependencies passed to `useEffect`/`useMemo` is the `any` of React Hooks.

You think you're being clever by passing an empty array, but you're probably wrong.

5 10 60

tom @tomjfinney · 27 Nov 2018

Why does the react docs for hooks then mention passing an empty array if you desire the effect to only be ran on mount and cleanup

2 2

Andrew Clark @acdlite · 27 Nov 2018

Where do the docs say that? If they do, then they're wrong :(

1 1

tom @tomjfinney · 27 Nov 2018

reactjs.org/docs/hooks-ref… the last bit of that block

> **Hooks API Reference – React**
> A JavaScript library for building user interfaces
> reactjs.org

1 2

Daveo @inflammatorydev · 27 Nov 2018

Tom is spot on. Thats how I read that line in the official docs as well. Are you saying that the paragraph is wrong, and if so, how would we accomplish the same thing (ie only have it run once and not on any re-renders)

Dan Abramov @dan_abramov · 27 Nov 2018

We have this note on the effect page. There should be an equivalent one in the API reference too. In general I suggest to read the main content *before* the API reference, you'll miss plenty of details otherwise. reactjs.org/docs/hooks-eff…

2 6

Peer @Twitt_Peer · 28 Nov 2018

I still don't get it: This note states the opposite as @acdlite says here:

twitter.com/acdlite/status…

Which one is correct?

> **Andrew Clark** @acdlite
> We're considering ignoring the dependencies array on every nth render (firing the effect regardless) in dev mode, to surface potential bugs
> Show this thread

1 3

Dan Abramov @dan_abramov · 28 Nov 2018

Working on it.

> **Dan Abramov** @dan_abramov
> Sorry, I don't think we have a broad agreement on the team about it either, and that's why our messaging is inconsistent. There is also some temptation to see Hooks+Suspense as one package, and ignore shortcomings of "fetch in effect" scenario that's common today. We'll fix.

1

Andrew Clark @acdlite · 27 Nov 2018

Intentionally underspecifying dependencies passed to `useEffect`/`useMemo` is the `any` of React Hooks.

You think you're being clever by passing an empty array, but you're probably wrong.

Dan Abramov @dan_abramov · 27 Nov 2018

We have this note on the effect page. There should be an equivalent one in the API reference too. In general I suggest to read the main content *before* the API reference, you'll miss plenty of details otherwise. reactjs.org/docs/hooks-eff…

2 6

Dan Abramov @dan_abramov · 28 Nov 2018

Working on it.

> **Dan Abramov** @dan_abramov
>
> Sorry, I don't think we have a broad agreement on the team about it either, and that's why our messaging is inconsistent. There is also some temptation to see Hooks+Suspense as one package, and ignore shortcomings of "fetch in effect" scenario that's common today. We'll fix.

1

Daveo @inflammatorydev · 27 Nov 2018

Tom is spot on. Thats how I read that line in the official docs as well. Are you saying that the paragraph is wrong, and if so, how would we accomplish the same thing (ie only have it run once and not on any re-renders)

temptation to see Hooks+Suspense as one package, and ignore shortcomings of "fetch in effect" scenario that's common today. We'll fix.

1

ARE THERE MORE HOOKS?

ARE THERE MORE HOOKS?

YES! A FEW MORE COME WITH REACT & CUSTOM ONES, COMMUNITY DRIVEN

SHOULD I START USING THEM NOW?

SHOULD I START USING THEM NOW?

UP TO YOU, BUT THEY ARE AVAILABLE TO USE SINCE REACT V16.8.0

ARE CLASSES DISAPPEARING?

ARE CLASSES DISAPPEARING?

NOT IN THE FORESEEABLE FUTURE: THE REACT TEAM WAS CLEAR ABOUT IT

WHAT'S THE MOST IMPORTANT TAKE AWAY?

WHAT'S THE MOST IMPORTANT TAKE AWAY?

YOU *CANNOT* STACK DRAW 2 & DRAW 4 CARDS.

Yerndi Redd @RealNYRedd · 11h

Settle this game night family argument...Can you put a Draw 4 on top of a Draw 4?

1

UNO

@realUNOgame

Follow

Replying to @RealNYRedd

No, you cannot stack to make the following player draw 8 cards.

1:53 PM - 8 May 2019

colours.dsgn.it

github.com/**cedmax**/**colours**

reactjs.com
official documentation

hooks.guide
collection of React hooks

overreacted.io
dan abramov's blog

usehooks.com
code examples

marco@cedmax.com
http://cedmax.com
@cedmax

つづく