React 3

CS571: Building User Interfaces

Cole Nelson

React Lectures

1. React 1: Introduction
2. React 2: Recap, User Input, & NPM/NodeJS
3. React 3: Managing State & Routing
4. React 4: Complex APIs & Memoization
5. React 5: Deployment & Various Topics

Midterm Exam

- Thursday, March 9th 5:45-7:15pm in the Chemistry Building Room S429. You will have **75 minutes**.
  - 24 MC (8 pts)
  - 5 SA (5 pts)
  - 1 LR (2 pts)
- F22 midterm and solution on Canvas.
- Contact me by **end of this week** about conflicts!

Academic Integrity

- **Do not share code with others!**
- **Do not use code from previous semesters!**
- You may *discuss* assignments with others, but all work must be done individually.
- Snippets taken from StackOverflow et. al. must be cited with a comment.
- We use tools like MOSS... Don't risk it!

Academic Integrity

First Offense

```
let currAssignment = 0;
const newCourseGrade = currCourseGrade * 0.9;
```

Second Offense

```
const newCourseGrade = 0;
fetch('https://conduct.students.wisc.edu/misconduct', {
  method: 'POST',
  body: JSON.stringify(currStudent)
});
```

Academic Integrity

Self-report by Sunday night. Emails go out Monday.

Only the assignment(s) will be 0'd for self-reports.

Complete Your Wordle

We'll be going through it in class. NYT Wordle

Your turn!

Create ticket tracking app! A `TicketBoard` should have three `TicketLane` which can hold many `Ticket`.

Each `Ticket` should display their name and description as well as buttons to move in to "TODO", "In Progress", and "Done" lanes.

Use `https://cs571.org/s23/week5/api/tickets`

Clone from here.

Uh-oh!

We need to talk back to our parent?

What will we learn today?

- What are React fragments?
- How can we share state in React?
- How can we handle routing in React?

Quick Note: React Fragments

A JSX component can only return *one thing*.

Sometimes we use `<></>` instead of `<div></div>`.

`<></>` is a React Fragment, a virtual separator not represented in the real DOM.

Learn More

State Management

How do we talk back to our parent? How do siblings talk to each other?

- Passing callbacks
- `useContext`
- `cookie`, `sessionStorage`, and `localStorage`
- Third-party libraries
  - Redux, Recoil, MobX, XState

Passing Callbacks

The original way to do child-to-parent communication.

```
const TodoList = (props) => {
  const [items, setItems] = useState();

  const removeItem = (itemId) => {
    // Do Remove!
  }

  return <div>
    {
      items.map(it => <TodoItem key={it.id} {...it} remove={removeItem}/>)
    }
  </div>
}
```

Passing Callbacks

This callback function is then used in the *child* to mutate the *parent*.

```
const TodoItem = (props) => {

  const handleRemove = () => {
    alert("Removing TODO item!");
    props.remove(props.id);
  }

  return <Card>
    <h2>{props.name}</h2>
    <Button onClick={handleRemove}>Remove Task</Button>
  </Card>
}
```

Ticket Management

Move tickets from lane to lane via passing callbacks.

`useContext` Hook

A useful hook for managing state across web apps with large component hierarchies.

`useContext` Hook

Motivation: How can we effectively manage state for web apps with large component hierarchies?

```
SpotifyLandingPage
 - NavBar
   - NavArrows
   - SearchBox
 - RecentSearches
   - AuthorCard
     - AuthorImage
     - AuthorName
```

`useContext` Hook

Three steps to using context.

1. Create and export a context.
2. Provide the context with some value.
3. Use the context in a child component.

Often used in combination with `useState`.

`useContext` Hook

A context must be exported.

```
const MyDataContext = createContext([]);
export default MyDataContext;
```

`useContext` Hook

A context must be provided to child component(s).

```
function ParentComponent() {
  const [data, setData] = useState([]);
  return (
    <MyDataContext.Provider value={[data, setData]}>
        <SomeChildComponent />
        <SomeOtherChildComponent />
    </MyDataContext.Provider>
  );
}
```

`useContext` Hook

The context can be used by any of child, grandchild, great-grandchild, etc. component(s).

```
function SomeChildComponent() {
  const [data, setData] = useContext(MyData);
  return (
    { /* Do something interesting with data here! */ }
  );
}
```

See StackBlitz

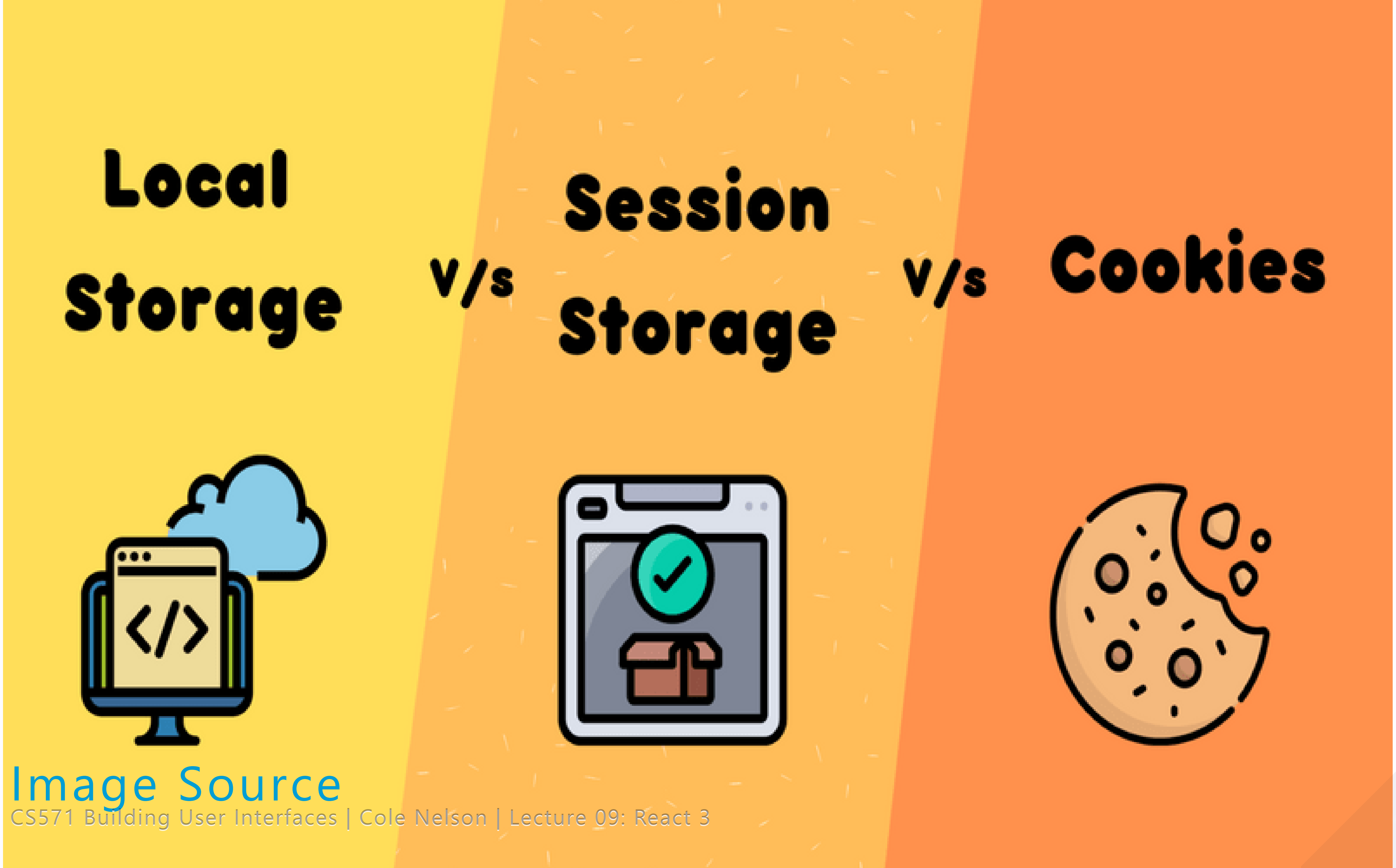
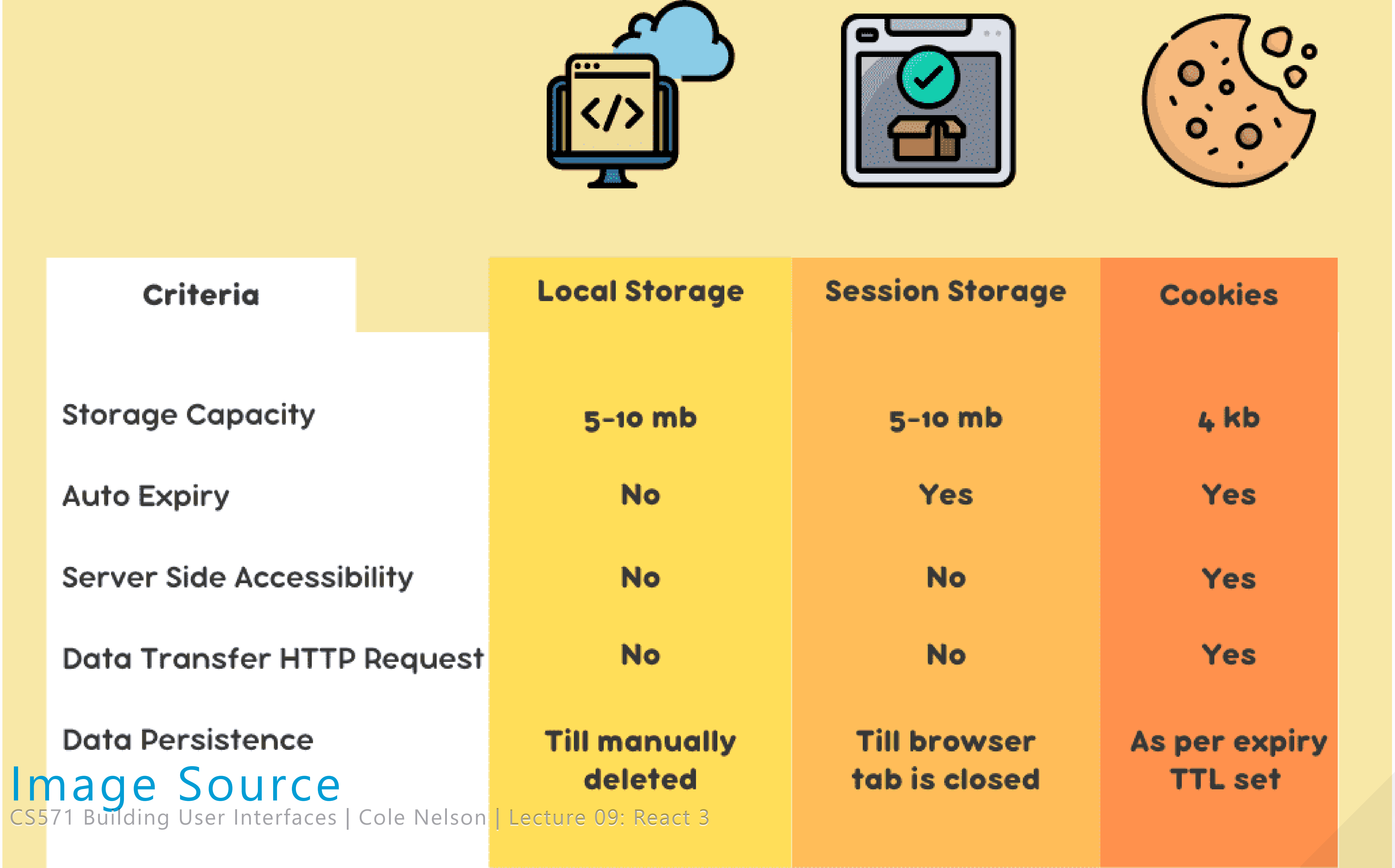


Image Source

Cookies vs. Session vs. Local

These exist in your browser!

- Facebook uses `cookies` to track you.
- Vanguard uses `cookies` to store temporary session/login credentials.
- Wordle uses `sessionStorage` and `localStorage` to store game state.



Criteria	Local Storage	Session Storage	Cookies
Storage Capacity	5-10 mb	5-10 mb	4 kb
Auto Expiry	No	Yes	Yes
Server Side Accessibility	No	No	Yes
Data Transfer HTTP Request	No	No	Yes
Data Persistence	Till manually deleted	Till browser tab is closed	As per expiry TTL set

Image Source

Cookies vs. Session vs. Local

Type	Notes
Cookies	Can be set programmatically, but typically set by server through a `Set-Cookie` header
Session	Set programmatically via `sessionStorage` , typically used with form data.
Local	Set programmatically via `localStorage` , typically used with long-lasting data.

Cookies vs. Session vs. Local

These are all just key-value pairs of *strings*!

Type	Example
Cookies	`document.cookie = 'lang=en'`
Session	`sessionStorage.setItem('name', 'Cole')`
Local	`localStorage.getItem('lastLogin')`

Let's Persist Some Data!

Using `sessionStorage` or `localStorage`.

StackBlitz Solution | Inspitation from WDS

Third Party Libraries

Each have their own unique way of managing state. Examples include...

- Redux
- Recoil
- MobX
- XState

Note! These come and go. See flux.

Client Vs. Server-Side Storage

These are all examples of doing client-side storage.

What if we want to persist data long-term?

Server-side storage with `PUT`, `POST`, and `DELETE`!

Multi-Page Apps

Knowing how to do state management, how do we manage apps with many pages?

React is a *library*, not a *framework*!

This means that *batteries are not included*. You'll be choosing many of your own tools and libraries!

- **Layout & Design:** Bootstrap React-Bootstrap, Reactstrap, Material, Elemental, Semantic
- **Routing & Navigation:** React Router, React Navigation, React Location
- **State Management:** Redux, Recoil, MobX, XState

Navigation w/ React Router

See StackBlitz

Types of Routers

- `BrowserRouter` : What you typically think of!
- `MemoryRouter` : Same as `BrowserRouter` , but the path is hidden from the browser in memory! 🤫
- `HashRouter` : Support for older browsers.
- `StaticRouter` : Used for server-side rendering.
- `NativeRouter` : We'll use react-navigation instead!

Routing

Using a `Router`, `Routes`, and `Route`!

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Layout />}>
      <Route index element={<Home />} />
      <Route path="about-us" element={<AboutUs />} />
      <Route path="other-info" element={<OtherInfo />} />
      <Route path="*" element={<Home />} />
    </Route>
  </Routes>
</BrowserRouter>
```

Browser `outlet`

`<Outlet/>` shows the component returned by the child route! e.g. in `Layout` we may see...

```
function Layout() {
  return (
    <>
      <Navbar bg="dark" variant="dark">
        { /* Some navigation links...*/ }
      </Navbar>
      <Outlet />
    </>
);
}
```

Navigable Components

Notice how each route maps to a component.

```
function Home() {
  return <h2>Home</h2>
}
function AboutUs() {
  return <h2>About Us :)</h2>
}
function OtherInfo() {
  return <h2>Other Info!</h2>
}
```

`useNavigate` Hook

Useful for programmatic navigation!

```
export default function OtherInfo() {

  const navigate = useNavigate();

  const handleClick = () => {
    navigate('/home');
  }

  return <div>
    <h2>Other Info!</h2>
    <Button onClick={handleClick}>Back to Home</Button>
  </div>
}
```

Navigation

Navigation for a `BrowserRouter` is done via URLs.

```
<>
  <Navbar bg="dark" variant="dark">
    <Nav className="me-auto">
      <Nav.Link as={Link} to="/">Home</Nav.Link>
      <Nav.Link as={Link} to="/about-us">About Us</Nav.Link>
      <Nav.Link as={Link} to="/other-info">Other Info</Nav.Link>
    </Nav>
  </Navbar>
  <Outlet />
</>
```

Questions?