App Organization and History API

Michael Chang

Spring 2020

Full stack topics

(Today) Web app organization

(Thursday) Authentication and users

CSS animations and mobile styling

Accessibility

Frontend frameworks, deployment, wrap-up

Plan for today

- **Web app structure**
  - Single page applications
- **location and history API**
  - Changing URL without reloading page
- **Aside: including external JS libraries**
- **Quick intro to Git**
  - Generally useful for projects
  - Needed for many deployment tools

Note: debugging backends

Syntax errors, errors in calling `app.get()`

- Server won't start, error logged to console (terminal, not browser!)
- `fetch()` will fail ("NetworkError", "failed to fetch")

JS error in a route handler

- Code throws an exception, logged to console
- Express returns error in text or HTML
- `.json()` on fetch response fails ("Unexpected token")

Note: route ordering

- **Express checks routes one by one**
  - First added -> first checked
  - Uses first match
- **Example**
  - `api.post("/foo", ...);`
  - `api.use(bodyParser.json());`
    - `req.body` not available in POST /foo route
  - `api.all("/*", ...); /* handle all requests */`
  - `api.get("/", ...);`
    - Second handler never called

Web app structure

How many HTML pages?

Traditional websites: one page per "view"

- Pages might have dynamic content filled in by server

Single page application (SPA)

- Only one HTML file; elements shown/hidden via JS
- Dynamic content via fetch
- Popular with frontend frameworks

Modern non-SPAs

- Use fetch for dynamic content
- Some showing/hiding to respond to user/server
- If page sufficiently different, can make a new HTML file

Browser URL bar

Current model

- URL shows exact file name the server will return
  - E.g. `http://localhost:1930/rest.html`
- Links point to specific HTML pages
  - E.g. `<a href="3_social.html">Part 3</a>`

Limitations

- No dynamic URLs, sharable links
  - E.g. `http://localhost:1930/profile/mchang`
- Changing URL -> reload from server
  - No smooth/fancy transitions, loading bars...
- Awkward combining with SPAs: no "deep links"
  - Can't link to parts of app; can't use back button

Multi-step approach

0. Don't. It's fine, really.

- For informational pages, pointing at HTML files is fine
- If only a couple views, no need for deep links

0.5. Use URL hash

- Hash part never sent to server; changing won't reload
  - E.g. `http://localhost:1930/#foo`

1. Server maps multiple URLs -> same HTML

- Use `location` in frontend to fetch/display content

2. Use history API to change URL

- Change URL without reloading page

Frontend: location

`location`: info about the loaded URL

`.href`: "http://localhost:1930/index.html?foo=bar#baz

- `.protocol`: "http:"
- `.host`: "localhost:1930"
- `.pathname`: "/index.html"
- `.search`: "?foo=bar"
- `.hash`: "#baz"

`.assign(url)`

Navigate to this URL (loaded from server)

`.replace(url)`

Replace URL (load from server, can't "Back" to current URL)

Backend: sending files

```
const PUBLIC_PATH = path.join(__dirname, "public");
app.get("/profile/:id", (req, res) => {
  res.sendFile("profile.html", { root: PUBLIC_PATH });
});
```

`res.sendFile(path, options)`

- Send the file as a response
- `path` must be absolute, or `options.root` must be set
- Careful: if path is coming from user, could introduce security issues
  - E.g. "send me the file `../../../secret.txt`"

Frontend: history API

`history`: interact with browser's URL bar and history

`.pushState(state, title, url)`

- Change URL bar without loading page
- No one uses title (pass null)
- Can pass "state" that isn't shown in URL
  - Accessed through history.state

`.replaceState(state, title, url)`

- Replace history entry with new one (no load)

`.back, .forward, .go`

- Programmatically move through browser history

Frontend: history API

`popstate` event on `window`

- Fired when user (or program) moves through history (back, forward, etc.)
- NOT fired when pushState or replaceState

Example

```
window.addEventListener("popstate", (event)
=> {
  console.log(`Location: $
{location.pathname}`);
  reloadPage();
});
```

Plan for today

- **Web app structure**
  - Single page applications
- **location and history API**
  - Changing URL without reloading page
- **Aside: including external JS libraries**
- **Quick intro to Git**
  - Generally useful for projects
  - Needed for many deployment tools

Aside: external (frontend) JS libraries

Most don't use modules

- `import Foo from "https://...";`
  - Will fail with "import not found: default"

Two options

- `<script src="..." defer></script>`
- Or: `import "...";`
  - For both, access through global variable

Example graph/chart library: Chart.js

- IMO, looks very convenient
- ...But maybe replace their use of var in their examples

Intro to Git

Version control system

- Stores revisions of your files
- Can compare files to old revisions, track changes through history
- Can merge changes between multiple contributors

Terminology

- Repository (repo): project managed by Git
- Commit: a set of changes to the repo
  - Associates an author, date/time, and message
- Remote: somewhere with a copy of your repo
  - E.g. GitHub

Some git commands

`git init`

Create a repo in current directory

`git add <files...>`

Mark files to be committed

`git status`

Show what will be committed, and what has changed

`git commit [-m <message>]`

Make a commit

`git push`

Send changes to a remote for safe keeping/publishing

More git commands

`git log`

Show a list of commits; each has a "hash" (big string) that uniquely identifies it

`git show <hash>`

Show what changed in a commit

`git diff`

Show differences that haven't been committed

`git pull`

Get changes from a remote (e.g. made by a collaborator)

If changes "conflict" (same part of a file), you will have to "merge"

Summary

Today

- Assorted things that may be relevant to your project
- Should be able to settle on design and structure

Before next time: assign4

Next time: auth and users

- Tracking who's logged in, API tokens
- Using 3rd-party auth (like Google), Oauth
- Security considerations