Development

Check out the xterm.js contribution documentation for how to work on the project.

Managing issues

Since bugs and/or features manifest themselves in both VS Code and xterm.js, it's a little unclear initially where the issue(s) should be created. After some experimentation I landed on the best way to deal with this is to create the an issue in both the Microsoft/vscode and xtermjs/xterm.js repositories. The reason this is the best workflow is because the changes will then be verified during endgame and it's much easier to compose release notes for the terminal changes. This guideline is less important for more obscure terminal issues where it's typically easier to keep a single source of truth in the xterm.js repo.

Updating `xterm` in vscode via script

1. Open a terminal in the vscode repo and run `./scripts/update-xterm.ps1` (or `node ./scripts/update-xterm.js`).
2. Add the changed files, excluding the aforementioned script files, and commit with the following message:

```
xterm@x.y.z-betaX

Diff: https://github.com/xtermjs/xterm.js/compare/91cbeec...eb25243

- Change 1...
- Change 2...
```

3. Build and run vscode to test terminal functionality. If all looks well, push.
4. Update distro as conflicts will occur

Updating `xterm` in vscode manually

Every commit that goes into the master branch of xterm.js is automatically released under the beta tag using Azure Pipelines. To update the module in vscode, follow these steps:

1. Identify the release to be used and install it. Find the latest beta and install via `yarn add xterm@x-y-z-betaX` (also do this in the `remote/` and `remote/web` folders). If you want an older commit the easiest way to do this right now is to identify the commit and then find the "Merge pull request" build on this pipeline, click into the "Release" job and view the output of the "Publish to npm" step to find the version number then install it with `yarn add xterm@x.y.z-betaX`
2. Build/test locally to make sure it works or push a branch and do a PR so that tests are run automatically. If the version change is significant it's a good idea to do a product build and verify it passes fully.
3. Write the commit message in the following format:

```
xterm@4.2.0-beta18

Diff: https://github.com/xtermjs/xterm.js/compare/91cbeec...eb25243

- Change 1...
- Change 2...
```

Updating `xterm-addon-*`

Similar to `xterm`, identify the release, pull it in and test it

Building one off builds of xterm and addons

Right now one off builds are manual.