Building a new toolset

I had first thought about adopting and modifying Kent C. Dodds kcd-scripts and hade made a first pass at it. But it broke when I made custom changes to acommodate my needs beyond the package tools.

Rather than build something using CLI tools, the goal for this project is to modify `kcd-scripts` as I try to understand the code and what's the best way to expand the package and remove functionality I don't use in my projects.

Ideas for packages

These are the tools that I want to incorporate into my toolset.

WordPress related

These are the WordPress related tools that I want to incorporate into my toolset. While they can be run using npx I prefer to run them locally.

Tool	Description / Purpose	Status
@wordpress/env	Set a local WordPress development environment for testing plugins and themes	✅

Tools

These are the tools that I want to include in the toolset. Some of these were part of kcd-scripts and some of them are my own

Tool	Description	Status
Jest	A Javascript testing infrastructure	✅
Eslint	Includes the ESLint shareable config for the Google JavaScript style guide	✅
Babel	Javascript transpiler	✅
Prettier	Code Formatter	✅
Typescript	If the project uses Typescript, switch the configuration to use a	✅

Tool	Description	Status
	Typescript configuration or build one from scratch	
React	If the project uses React or Preact, switch the configuration to use a React-based configuration or build one from scratch	✅
Preact	If the project uses Preact or Preact, switch the configuration to use a React-based configuration or build one from scratch	✅
lint-staged	Will create precommit hooks to run our code against before comitting. It will also install husky to make working with precommit hooks easier	✅
Markdownlint	I could never get it to work reliably so I removed it from the toolset	✅

Potential ideas

These are some tools that I'm debating whether to include in the toolset or not.

Tool	Description	Status
@wordpress/create-block	Create plugins for Gutenberg blocks	
PostCSS	CSS Pre Processor	
SASS/SCSS	CSS Pre Processor	
Playwright	Cross Platform browser emulation tool useful for testing beyond what Jest provides	

Building the package

The first script that we'll look at is the index script (`index.js`). This till check that we're using a supported version of Node and then runs the script runner.

The script runner script (`run-script`) will run the other parts of the package.

It will parse flags and arguments and pass them to the script that we want to run.

It will also account for issues in the cross-spawn library and will normalize paths between Unix-like systems and Windows.

Once we have that in place we can look at individual scripts to run the tools we want.

I will pick one tool as an example, most tools work the same.

We first require the necessary Node modules and the functions from within out utils package.

```
const path = require('path');
const spawn = require('cross-spawn');
const yargsParser = require('yargs-parser');
const {resolveBin, hasFile} = require('../utils');
```

We then parse the arguments and options.

```
let args = process.argv.slice(2);
const here = (p) => path.join(__dirname, p);
const hereRelative = (p) => here(p).replace(process.cwd(), '.');
const parsedArgs = yargsParser(args);
```

The next step is to check and build the configuration for the tool we're using. First, we check if we have a configuration flag (`--config`) and a configuration file (`.markdownlint.json`).

```
const useBuiltinConfig =
  !args.includes('--config') &&
  !hasFile('.markdownlint.json');

const config = useBuiltinConfig ?
  ['--config', hereRelative('../config/.markdownlint.json')] :
  [];
```

```
const defaultExtensions = 'md,mdx,markdown';
const ext = args.includes('--ext') ? [] : ['--ext', defaultExtensions];
const extensions = (parsedArgs.ext || defaultExtensions).split(',');
```

```
const filesGiven = parsedArgs._.length > 0;

const filesToApply = filesGiven ? [] : ['.'];

if (filesGiven) {
  args = args.filter(
      (a) => !parsedArgs._.includes(a) || extensions.some((e) => a.endsWi
  );
}

const result = spawn.sync(
    resolveBin('markdownlint'),
    [...config, ...ext, ...args, ...filesToApply],
    {stdio: 'inherit'},
);

process.exit(result.status);
```