# Contributing

## Web Server

### Simple PHP Server

For a simple local development environment running on PHP, you will need:

\* [A supported version of PHP](http://php.net/supported-versions.php) with:

\* `php-cli`

\* `php-curl`

\* `php-intl`

\* `php-json`

\* `php-mbstring`

\* `php-sqlite3`

\* `php-xml`

\* `composer`

\* [Node.js](https://nodejs.org/) and `npm`

\* packages installed with `npm ci`

These can be most easily installed on elementaryOS 5.0 (Ubuntu 18.04) with this script:

```

sudo apt install php-cli php-curl php-intl php-json php-sqlite3 php-mbstring php-xml composer &&

sudo apt install nodejs npm &&

npm ci

```

Then inside the project directory, run `npm run build && npm run start`. Next,

just navigate to [localhost:8000](http://localhost:8000/) to view the site.

If you are working on CSS and would like an easier time developing, you can run

the `npx gulp watch` command. This will watch for any CSS and image changes,

and rebuild on the fly.

### Nginx Web Server

For a full web-server environment, which includes more redirect and permissions

you may find useful, you will need:

\* Everything required for "Simple PHP Server" (above)

\* The latest stable version of [Nginx](http://nginx.org)

\* `php7.0-fpm`

Then, we need to configure Nginx. To start, open up a configuration file in

Nano.

```bash

sudo nano /etc/nginx/sites-enabled/mvp.conf

```

Then, paste in required configuration in, modifying the root, include and

error\_log paths.

```

server {

listen 80;

server\_name mvp.localtest.me;

root /path/to/mvp;

include /path/to/mvp/nginx.conf;

error\_log /path/to/error.log;

}

```

You can test the configuration with Nginx.

```bash

sudo nginx -t

```

Now we just need to restart the service.

```bash

sudo service nginx restart

```

Then we need to build the static assets.

```bash

npm ci && npm run build

```

Finally, navigate to [mvp.localtest.me](http://mvp.localtest.me)

## Code Style

- Four space indentation

- Remove trailing whitespaces and add an empty line at the end of each file

- Compatibility with the latest versions of popular browsers (chrome, firefox,

safari, edge, midori)

### PHP

- `include` templates, not `require` or `\_once`

- Use full PHP tags, not short ones

- Don't close PHP tags on PHP only files

- Correctly format assignments for readability

```php

<?php

require\_once \_\_DIR\_\_.'/\_backend/preload.php';

$page['title'] = 'HTML Safe Title';

include $template['header'];

$foo = bar($para, $param);

$second\_foo = 42;

$third\_foo = 'hey';

?>

```

### HTML

- Include `alt` attribute for all images

- Include `title` attribute for all links

- Close all your tags properly

- `a` elements with `target="\_blank"` should include a `rel="noopener"`

### CSS

- Try to use classes instead of IDs unless things are absolutely unique

- One selector per line

- Care with fallbacks and browsers compatibilities. Using only official syntax

and let the build process add prefixes as needed.

```css

.class {

color: #fefe89;

font-size: 1.1em;

}

.second-class,

.third-class {

backgound-color: white;

}

```

## Proposing Changes

### Make a new branch and push it to GitHub.

```bash

git checkout -b feature\_branch\_name

git push -u origin feature\_branch\_name

```

### Updating from Master

```bash

git pull origin master

```

### Merge from master

```bash

git checkout feature\_branch\_name

git merge master

```