






Take the goRe out of the DjangoReact stack

Integrating JS apps with Django

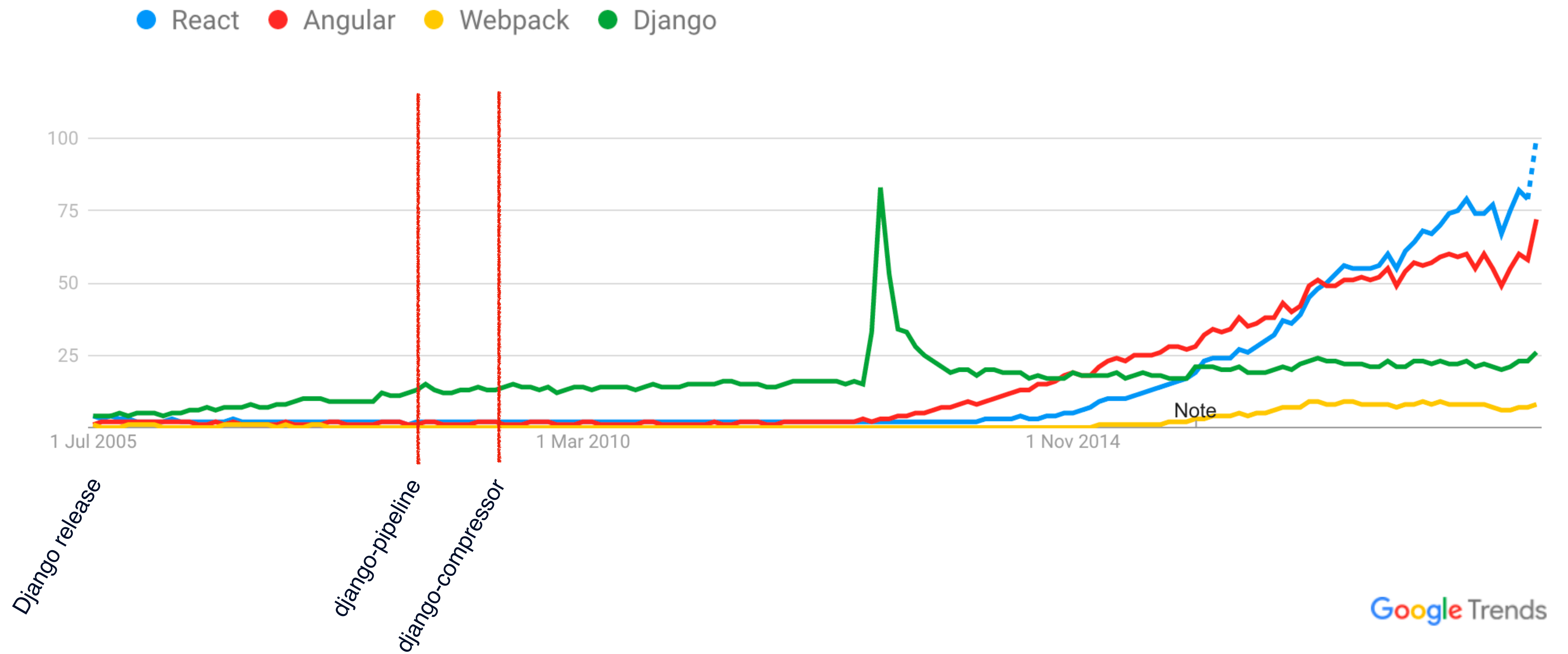
 /n6g7  /n4ng5l nathang@theodo.co.uk

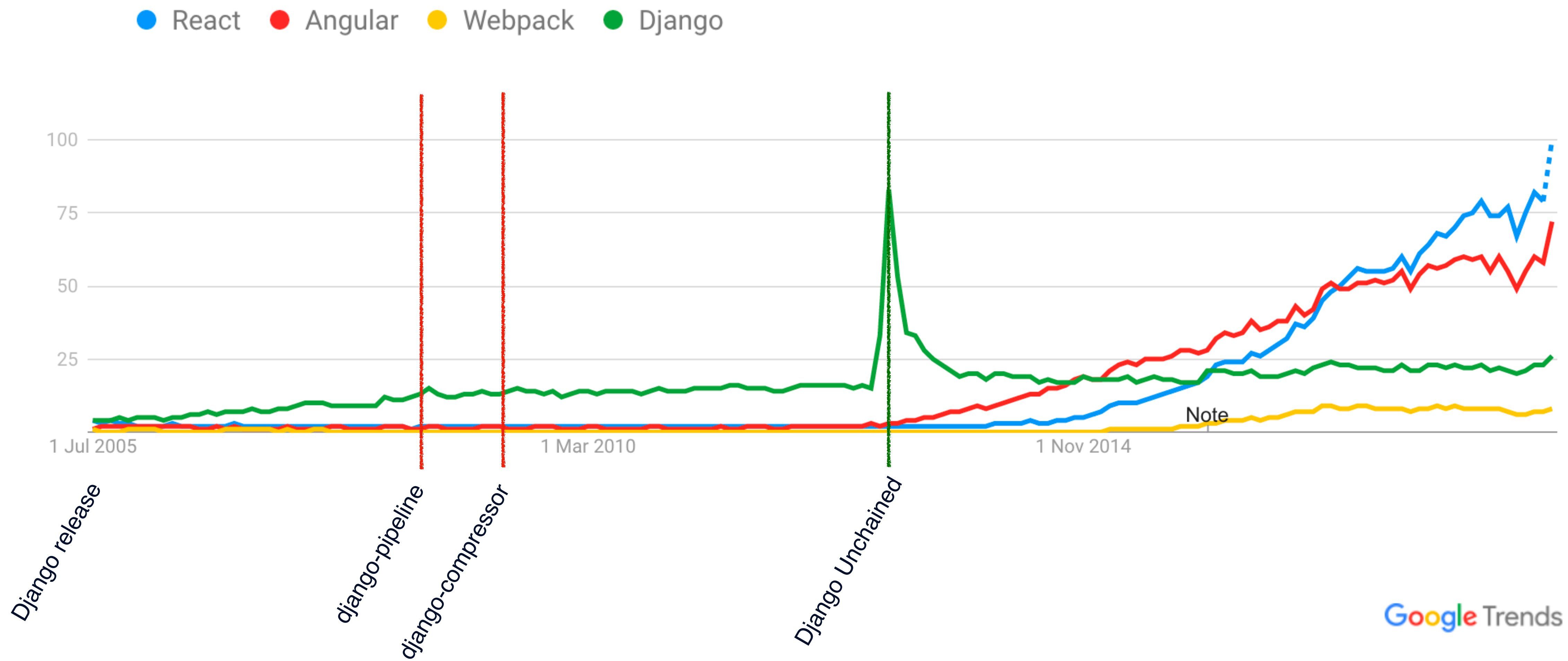
A JS toolchain primer

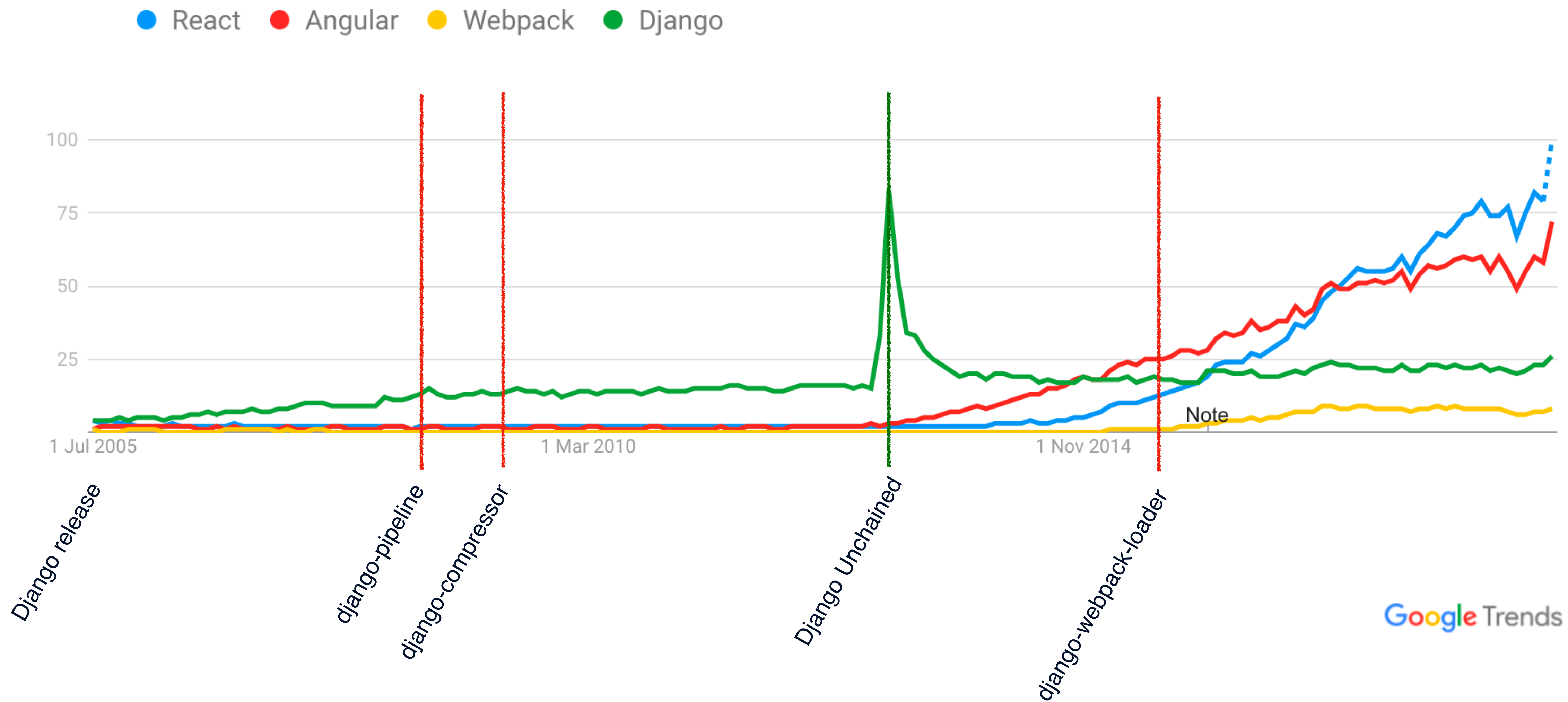
- **npm**  \Leftrightarrow PyPI & pip
- Webpack  \Leftrightarrow ??????
- Webpack dev server  \Leftrightarrow Django autoreloader



Django & JS state of the art







django-webpack-loader

- 2015
- First attempt at a proper integration with Webpack
- Separates responsibilities:
 - Webpack builds the js « bundle » (no wrapper)
 - DWL provides template tags to output `<script>` tags

<https://owais.lone.pw/blog/webpack-plus-reactjs-and-django/>

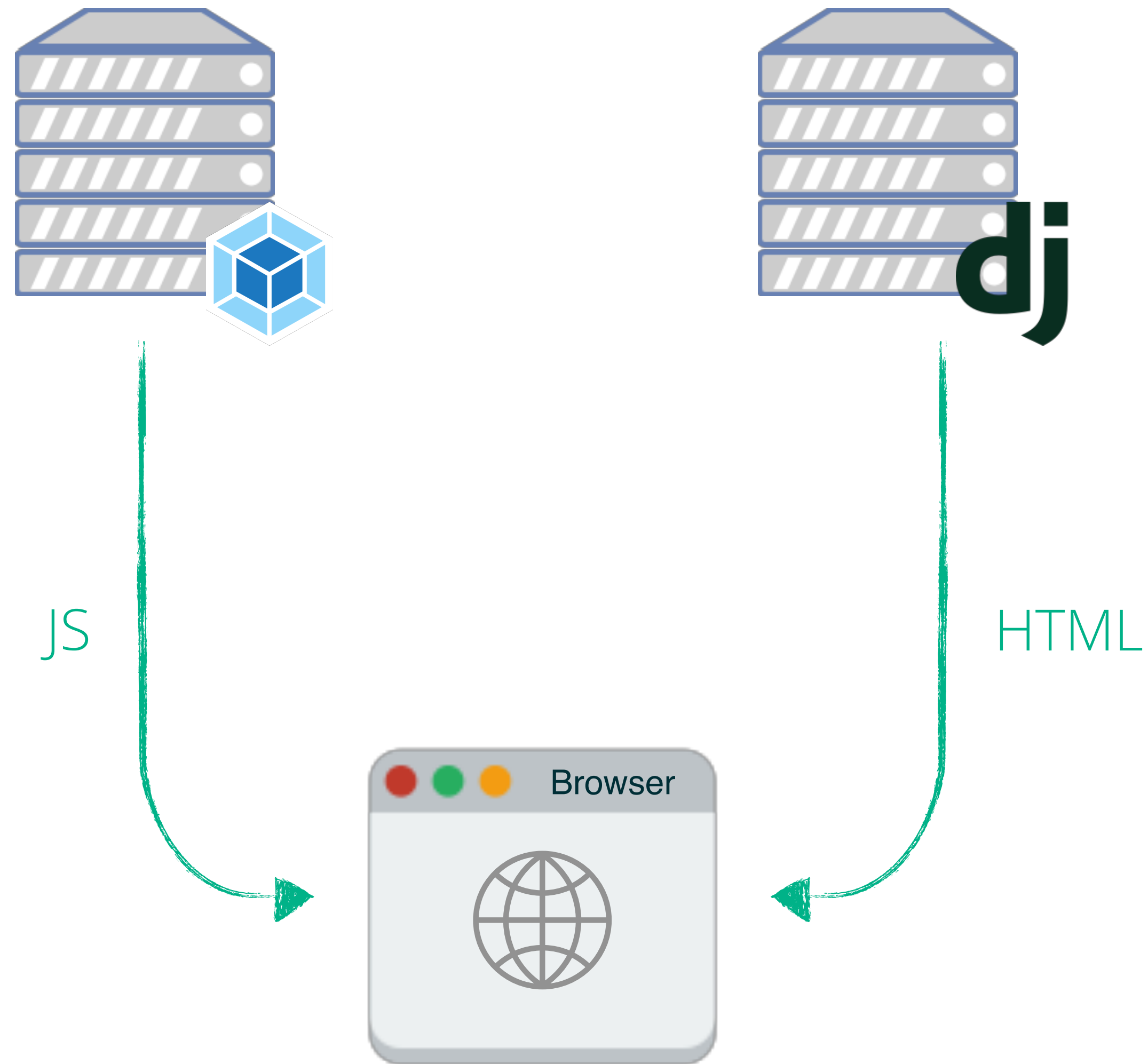
A typical dev environment (before)



HTML + JS



A typical dev environment (after)



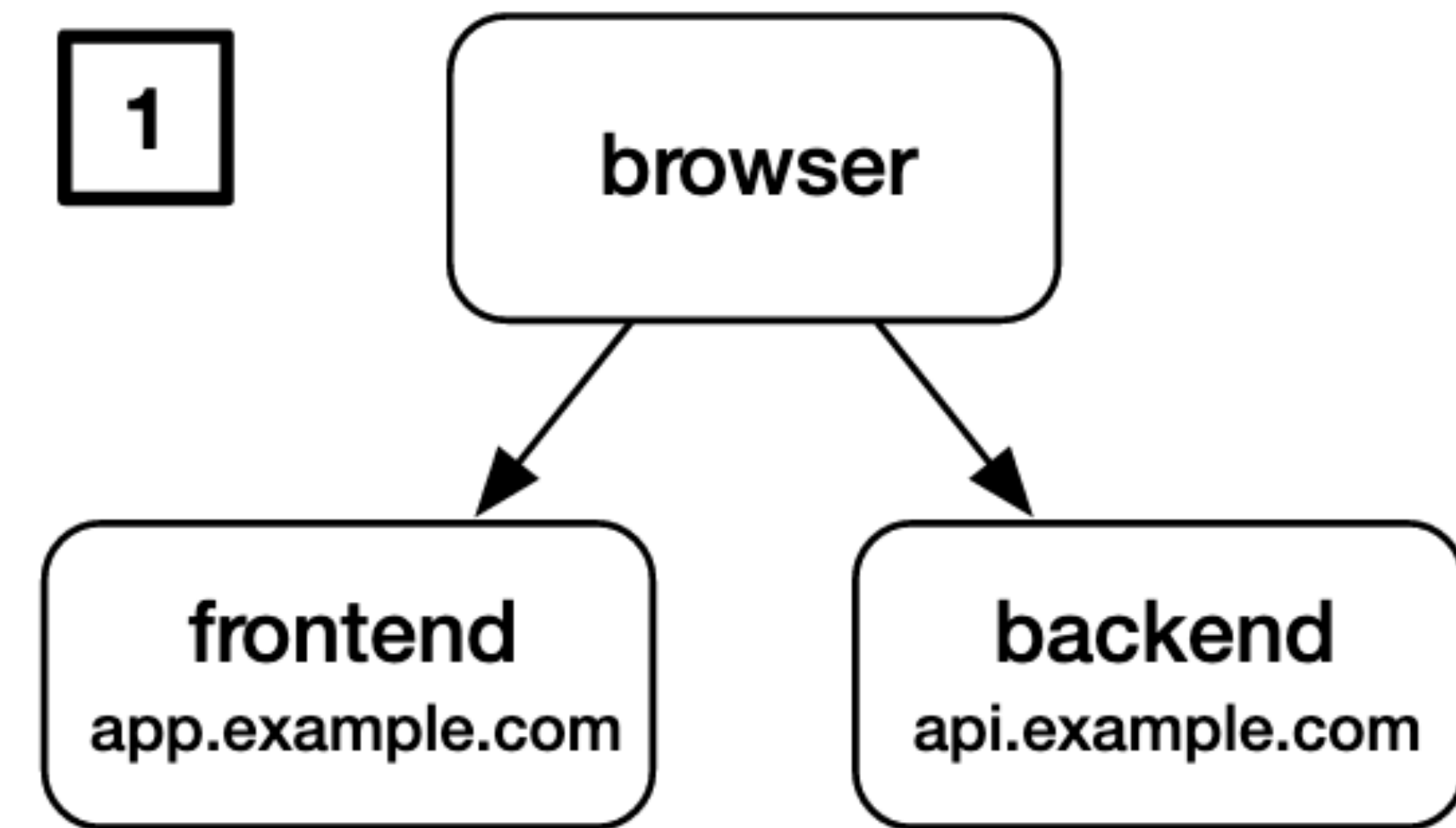
Aymeric Augustin's article series

- 2018
- Formalisation of Django+JS requirements
- Taxonomy of JS apps:
 - Single Page Application (SPA)
 - Hybrid Application
- Detailed examples of setups
 - Prod/dev parity, authentication, and more

<https://fractalideas.com/blog/making-react-and-django-play-well-together/>

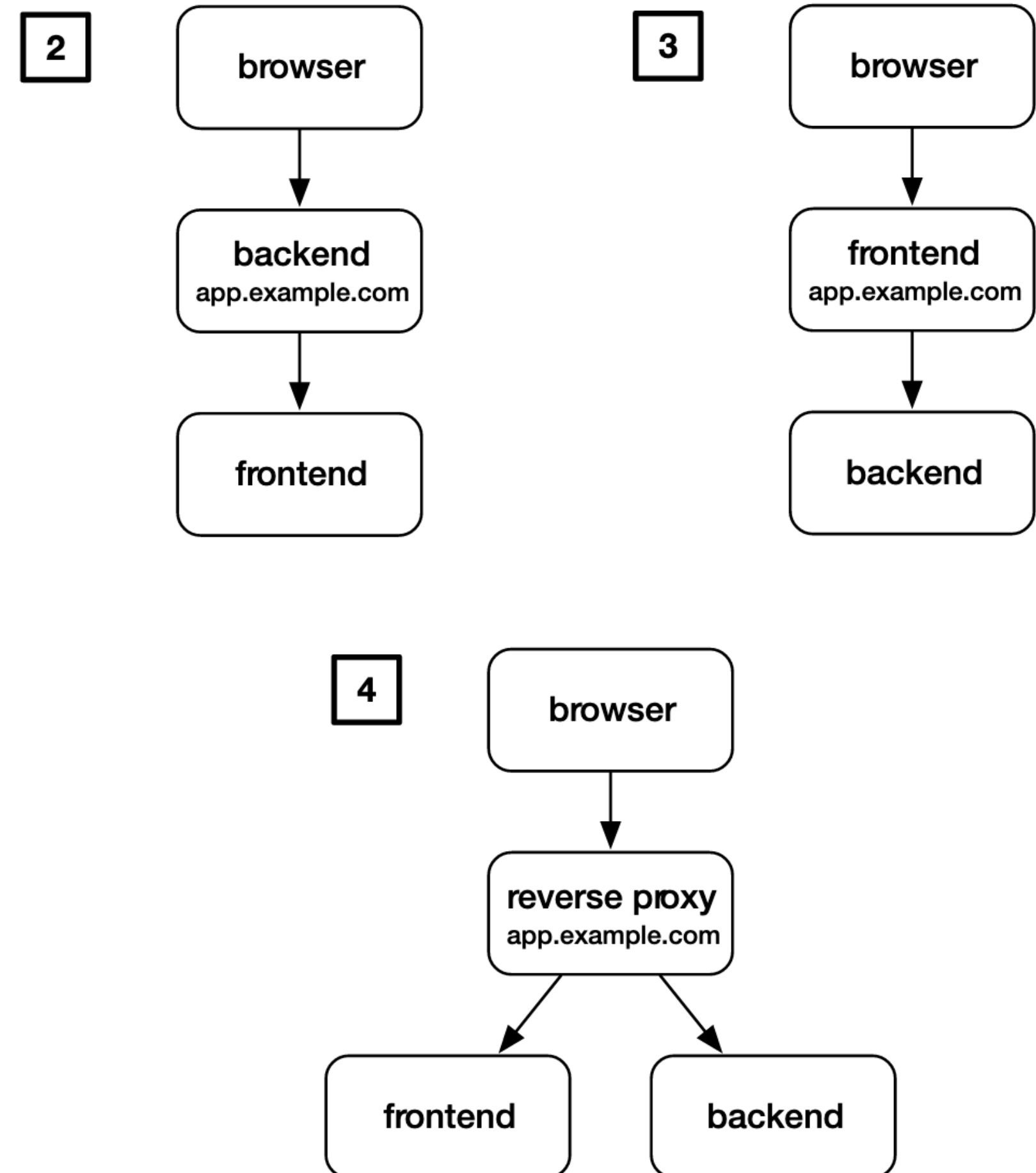
Aymeric Augustin's article series

- Single Page Application
 - HTML is static (not served by Django)
 - Django is purely an API



Aymeric Augustin's article series

- Hybrid Application
 - HTML is served by Django
 - HTML is a mix of Django template & JS





Designing our Django & JS site

Requirements

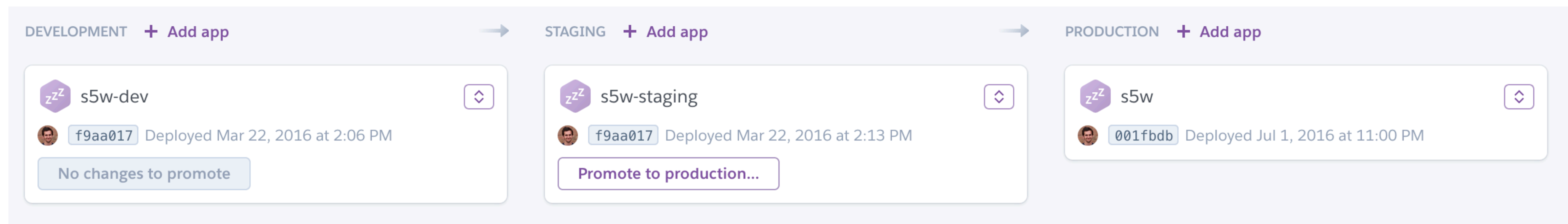
1. « Builds » are environment-independant
2. Hot reloading in development
3. Dev/prod parity

1 Builds are environment-independant

- Build = archive, docker image, commit, binary, etc
- Why?
 - Allows « version promotion »
 - Trust that what works in staging will also work in prod

1 Builds are environment-independant

- Build = archive, docker image, commit, binary, etc
- Why?
 - Allows « version promotion »
 - Trust that what works in staging will also work in prod

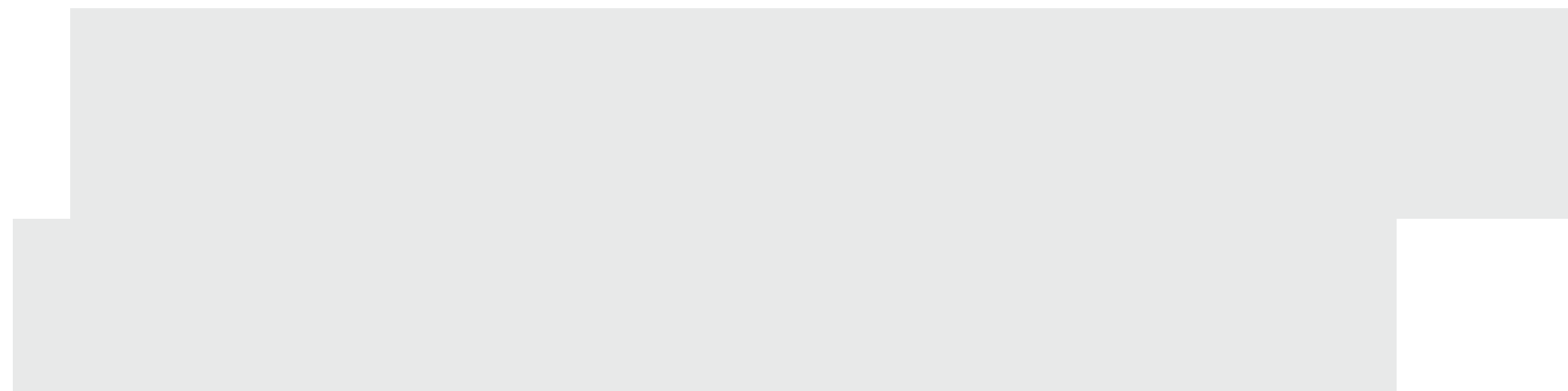


1 Builds are environment-independant

<https://12factor.net/config>


1 Builds are environment-independant

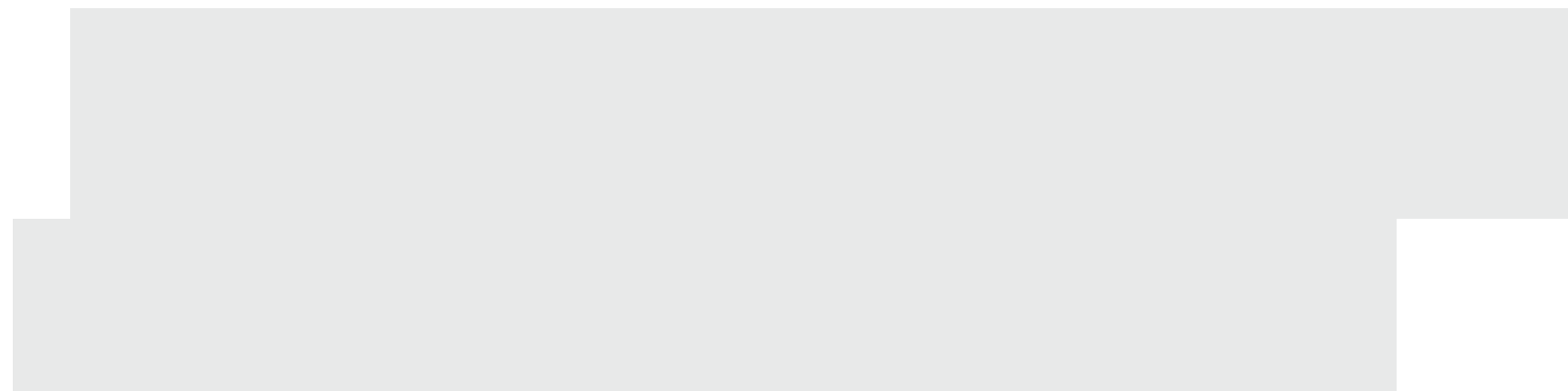
- How?
 - No environment-specific values hardcoded in the code
 - Environment variables




<https://12factor.net/config>

1 Builds are environment-independant

- How?
 - No environment-specific values hardcoded in the code
 - Environment variables
- Django is good at this 



1 Builds are environment-independant

- How?
 - No environment-specific values hardcoded in the code
 - Environment variables
- Django is good at this 
- JS example: API endpoint
 - prod: `https://api.example.com`
 - dev: `http://localhost:8000`

1 Builds are environment-independant

1 Builds are environment-independant

- JS doesn't have a concept of environment variables. 😞

1 Builds are environment-independant

- JS doesn't have a concept of environment variables. 😞
- Two options to load config:
 - Make an API call when the app starts
 - Inject config in HTML

1 Builds are environment-independant

- JS doesn't have a concept of environment variables. 😞
- Two options to load config:
 - Make an API call when the app starts
 - But where does this URL come from? ouroboros.jpg
 - Inject config in HTML

1 Builds are environment-independant

- JS doesn't have a concept of environment variables. 😞
- Two options to load config:
 - Make an API call when the app starts
 - But where does this URL come from? ouroboros.jpg
 - Inject config in HTML ✅
- So HTML needs to be a Django template
- ➡ Hybrid app

`./front/templates/front/index.html`

```
<body>
  <!-- ... -->

  <script>
    var CONFIG = {
      API_ENDPOINT: "{{ API_ENDPOINT }}",
      OPENING_HOURS: [
        {% for item in opening_hours %}
        {
          day: '{{item.day}}',
          openingTime: '{{item.opening_time}}',
          order: '{{item.order}}',
        },
        {% endfor %}
      ]
    }
  </script>

  <!-- ... -->
</body>
```

ERRATA

Depending on where those config values come from, this may be vulnerable to XSS injections. Use the `json_script` tag instead (since Django 2.1).

2 Hot reloading in development

2 Hot reloading in development

- Why?
 - Shortens feedback loop:
Write code > Run code > Detect bug

2 Hot reloading in development

- Why?
 - Shortens feedback loop:
Write code > Run code > Detect bug
- Relies on Webpack Dev Server (WDS)
 - Live reloading: full page refresh
 - Hot reloading: in-place replacement (no page refresh, uses websocket)

Hot reloading in development

- **Hot reloading**
- Websocket connection to Webpack Dev Server

localhost:3000



localhost:8000



localhost:3000

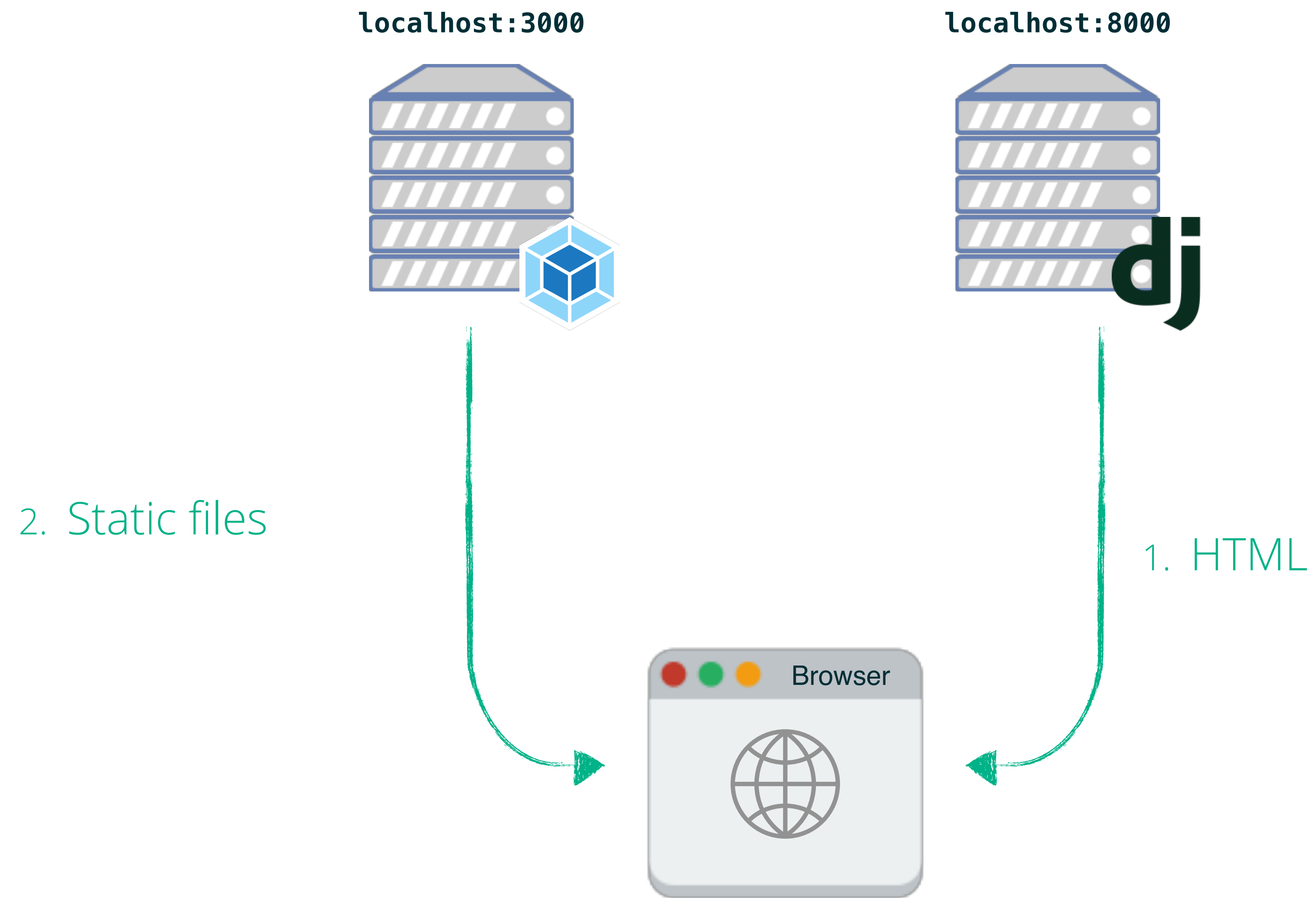


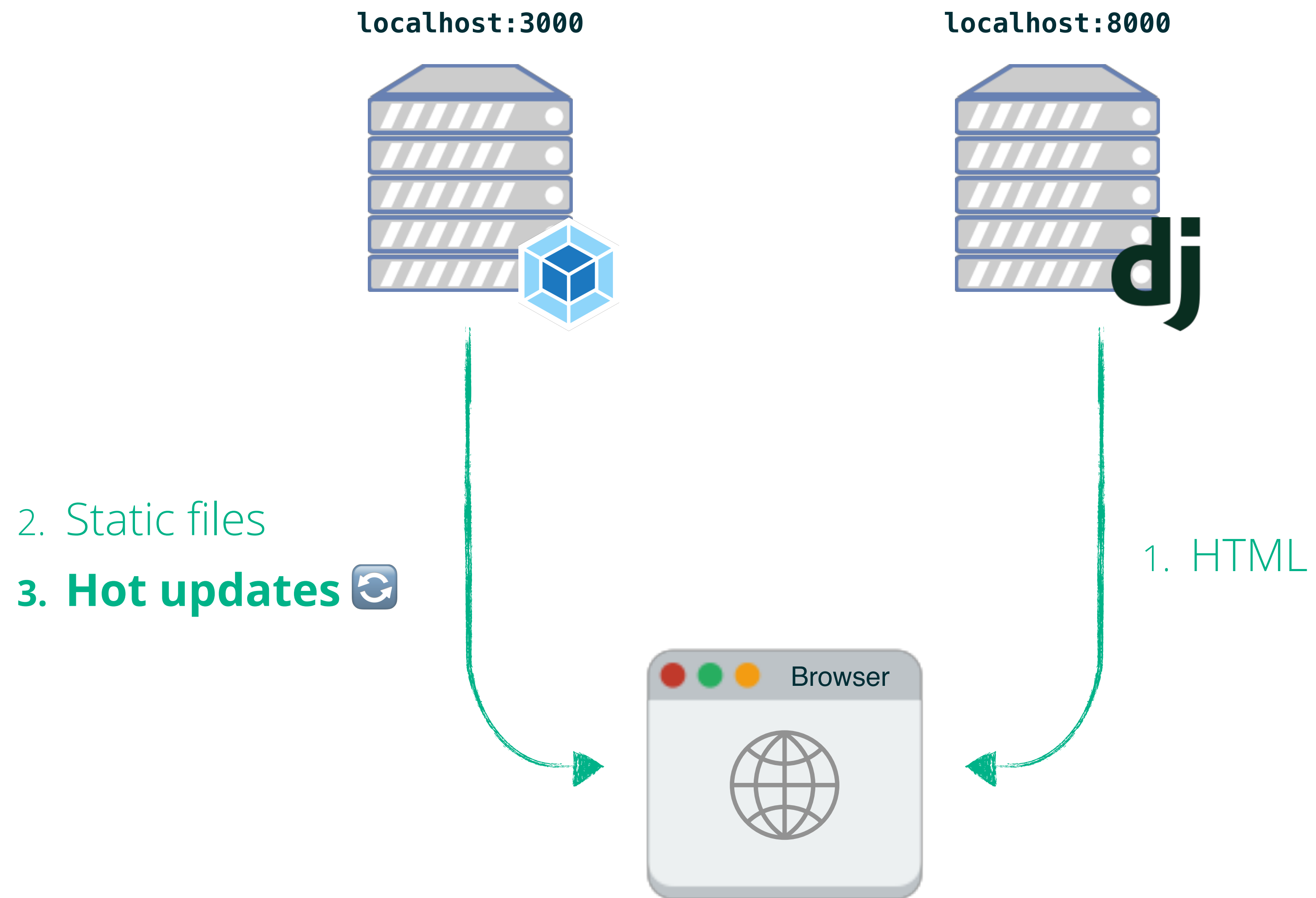
localhost:8000



1. HTML







2 Hot reloading in development

- **Hot reloading**
- Websocket connection to Webpack Dev Server
 - Possible if you're using vanilla WDS
 - But not if you're using create-react-app ❌
 - GitHub issue and PR

2 Hot reloading in development

- **Live reloading**
- Built in webpack dev server:
 1. Browser loads files dynamically from WDS
 2. Django renders `<script>` tags in app HTML (django-webpack-loader)

3 Dev/prod parity

- **Live reloading**
- Built in webpack dev server:
 1. Browser loads files dynamically from WDS
 2. Django renders `<script>` tags in app HTML (django-webpack-loader)

3 Dev/prod parity

- **Live reloading**
- Built in webpack dev server:
 1. Browser loads files dynamically from WDS
 2. Django renders `<script>` tags in app HTML (django-webpack-loader)
- Django renders `<script>` tags in prod ➡ same in dev

3 Dev/prod parity

- **Live reloading**
- Built in webpack dev server:
 1. Browser loads files dynamically from WDS
 2. Django renders `<script>` tags in app HTML (django-webpack-loader) ✓
- Django renders `<script>` tags in prod ➡ same in dev

`./front/templates/front/index.html`

```
<head>
  <!-- ... -->
  {% if not debug %}
    {% for style in initial_stylesheets %}
      <link href="{% static style %}" rel="stylesheet" />
    {% endfor %}
  {% endif %}
</head>

<body>
  <!-- ... -->

  {% if debug %}
<script type="text/javascript">
  // In development, we dynamically load the entry chunks into the dom.
  // To do so, we load the webpack-generated manifest, filter it to only
  // keep the initial chunks, and insert them in the dom.
  fetch("http://localhost:3000/static-manifest.json")
    .then(function(response) {
      return response.json();
    })
    .then(function(manifest) {
      for (var chunkName in manifest) {
        var chunk = manifest[chunkName];
        if (chunk.isInitial && chunk.path.match(/.*\.js$/)) {
          var script = document.createElement("script");
          script.type = "text/javascript";
          script.src = "http://localhost:3000/" + chunk.path;
          document.getElementsByTagName("head")[0].appendChild(script);
        }
      }
    });
</script>
  {% else %}
    {% for script in initial_scripts %}
      <script src="{% static script %}"></script>
    {% endfor %}
  {% endif %}
</body>
```

./front/templates/front/index.html

```
<head>
  <!-- ... -->
  {% if not debug %}
    {% for style in initial_stylesheets %}
      <link href="{% static style %}" rel="stylesheet" />
    {% endfor %}
  {% endif %}
</head>

<body>
  <!-- ... -->

  {% if debug %}
    <script type="text/javascript">
      // In development, we dynamically load the entry chunks into the dom.
      // To do so, we load the webpack-generated manifest, filter it to only
      // keep the initial chunks, and insert them in the dom.
      fetch("http://localhost:3000/static-manifest.json")
        .then(function(response) {
          return response.json();
        })
        .then(function(manifest) {
          for (var chunkName in manifest) {
            var chunk = manifest[chunkName];
            if (chunk.isInitial && chunk.path.match(/.*\.js$/)) {
              var script = document.createElement("script");
              script.type = "text/javascript";
              script.src = "http://localhost:3000/" + chunk.path;
              document.getElementsByTagName("head")[0].appendChild(script);
            }
          }
        });
    </script>
  {% else %}
    {% for script in initial_scripts %}
      <script src="{% static script %}"></script>
    {% endfor %}
  {% endif %}
</body>
```

```
<head>
  <!-- ... -->
  {% if not debug %}
    {% for style in initial_stylesheets %}
      <link href="{% static style %}" rel="stylesheet" />
    {% endfor %}
  {% endif %}
</head>

<body>
  <!-- ... -->

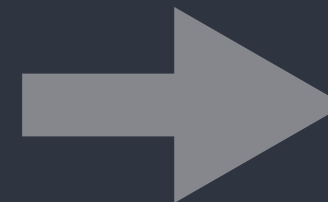
  {% if debug %}
    <script type="text/javascript">
      // In development, we dynamically load the entry chunks into the dom.
      // To do so, we load the webpack-generated manifest, filter it to only
      // keep the initial chunks, and insert them in the dom.
      fetch("http://localhost:3000/static-manifest.json")
        .then(function(response) {
          return response.json();
        })
        .then(function(manifest) {
          for (var chunkName in manifest) {
            var chunk = manifest[chunkName];
            if (chunk.isInitial && chunk.path.match(/.*\.js$/)) {
              var script = document.createElement("script");
              script.type = "text/javascript";
              script.src = "http://localhost:3000/" + chunk.path;
              document.getElementsByTagName("head")[0].appendChild(script);
            }
          }
        });
    </script>
  {% else %}
    {% for script in initial_scripts %}
      <script src="{% static script %}"></script>
    {% endfor %}
  {% endif %}
</body>
```

./front/templates/front/index.html

```
<head>
  <!-- ... -->
  {% if not debug %}
    {% for style in initial_stylesheets %}
      <link href="{% static style %}" rel="stylesheet" />
    {% endfor %}
  {% endif %}
</head>

<body>
  <!-- ... -->

  {% if debug %}
    <script type="text/javascript">
      // In development, we dynamically load the entry chunks into the dom.
      // To do so, we load the webpack-generated manifest, filter it to only
      // keep the initial chunks, and insert them in the dom.
      fetch("http://localhost:3000/static-manifest.json")
        .then(function(response) {
          return response.json();
        })
        .then(function(manifest) {
          for (var chunkName in manifest) {
            var chunk = manifest[chunkName];
            if (chunk.isInitial && chunk.path.match(/.*\.js$/)) {
              var script = document.createElement("script");
              script.type = "text/javascript";
              script.src = "http://localhost:3000/" + chunk.path;
              document.getElementsByTagName("head")[0].appendChild(script);
            }
          }
        });
    </script>
  {% else %}
    {% for script in initial_scripts %}
      <script src="{% static script %}"></script>
    {% endfor %}
  {% endif %}
</body>
```



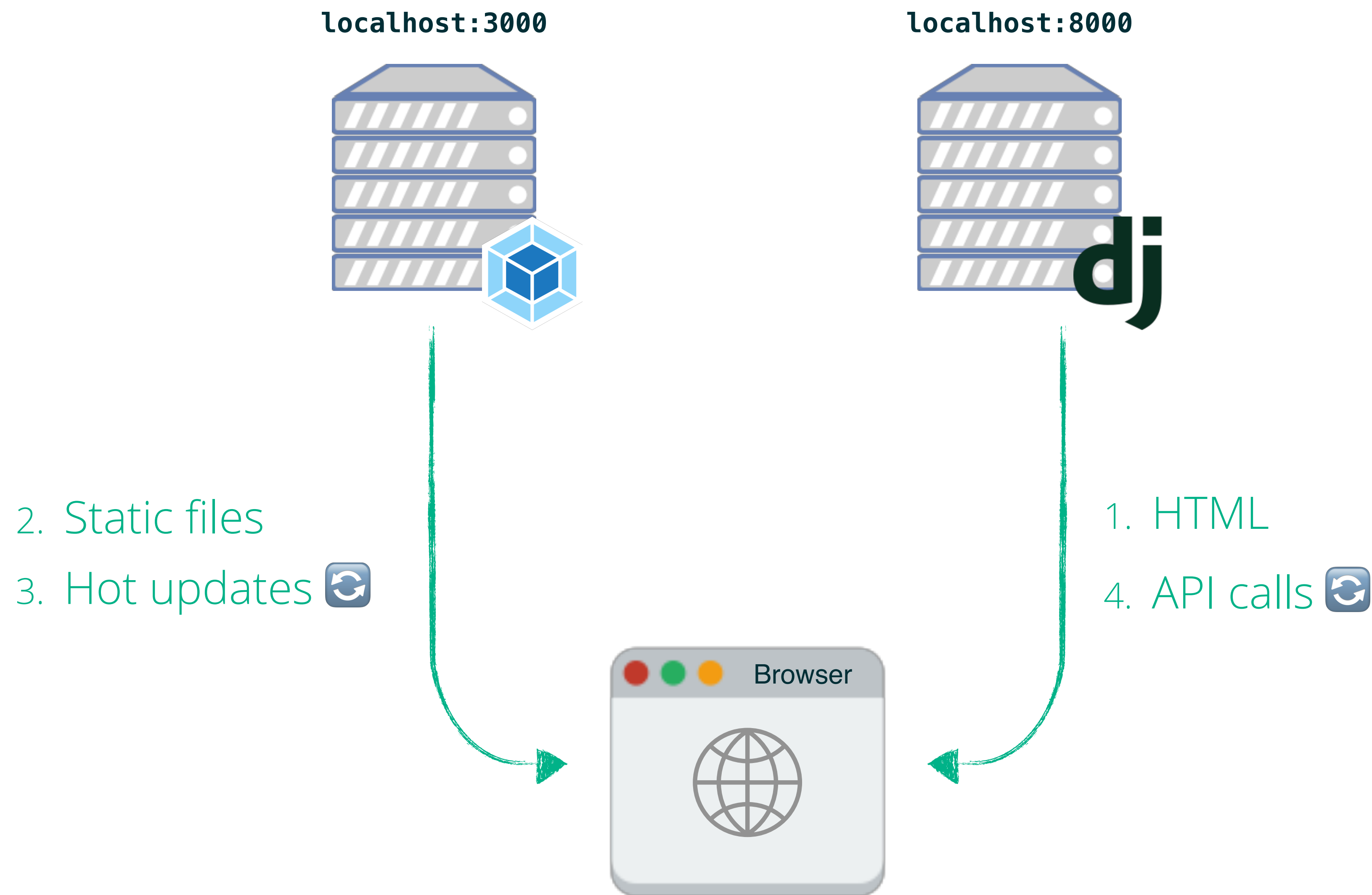
```
<head>
  <!-- ... -->
  {% render_bundle 'main' 'css' %}
</head>

<body>
  <!-- ... -->

  {% render_bundle 'main' 'js' %}
</body>
```




A concrete example





`./front/templates/front/index.html`

```
{% load render_bundle from webpack_loader %}

<!DOCTYPE html>
<html lang="en">
  <head>
    {% render_bundle 'main' 'css' %}
  </head>

  <body>
    <div id="app"></div>

    <script>
      var CONFIG = {
        API_ENDPOINT: "{{ API_ENDPOINT }}",
        OPENING_HOURS: [
          {% for item in opening_hours %}
          {
            day: '{{item.day}}',
            openingTime: '{{item.opening_time}}',
            order: '{{item.order}}',
          },
          {% endfor %}
        ]
      }
    </script>

    {% render_bundle 'main' 'js' %}
  </body>
</html>
```

ERRATA

Depending on where those config values come from, this may be vulnerable to XSS injections. Use the `json_script` tag instead (since Django 2.1).

`./front/views.py`



```
from django.conf import settings
from django.views.generic import TemplateView

from app.models import OpeningHours

class FrontendView(TemplateView):
    template_name = "index.html"

    def get_context_data(self, **kwargs):
        return super().get_context_data(
            API_ENDPOINT=settings.API_ENDPOINT,
            opening_hours=OpeningHours.objects.all(),
            **kwargs
        )
```

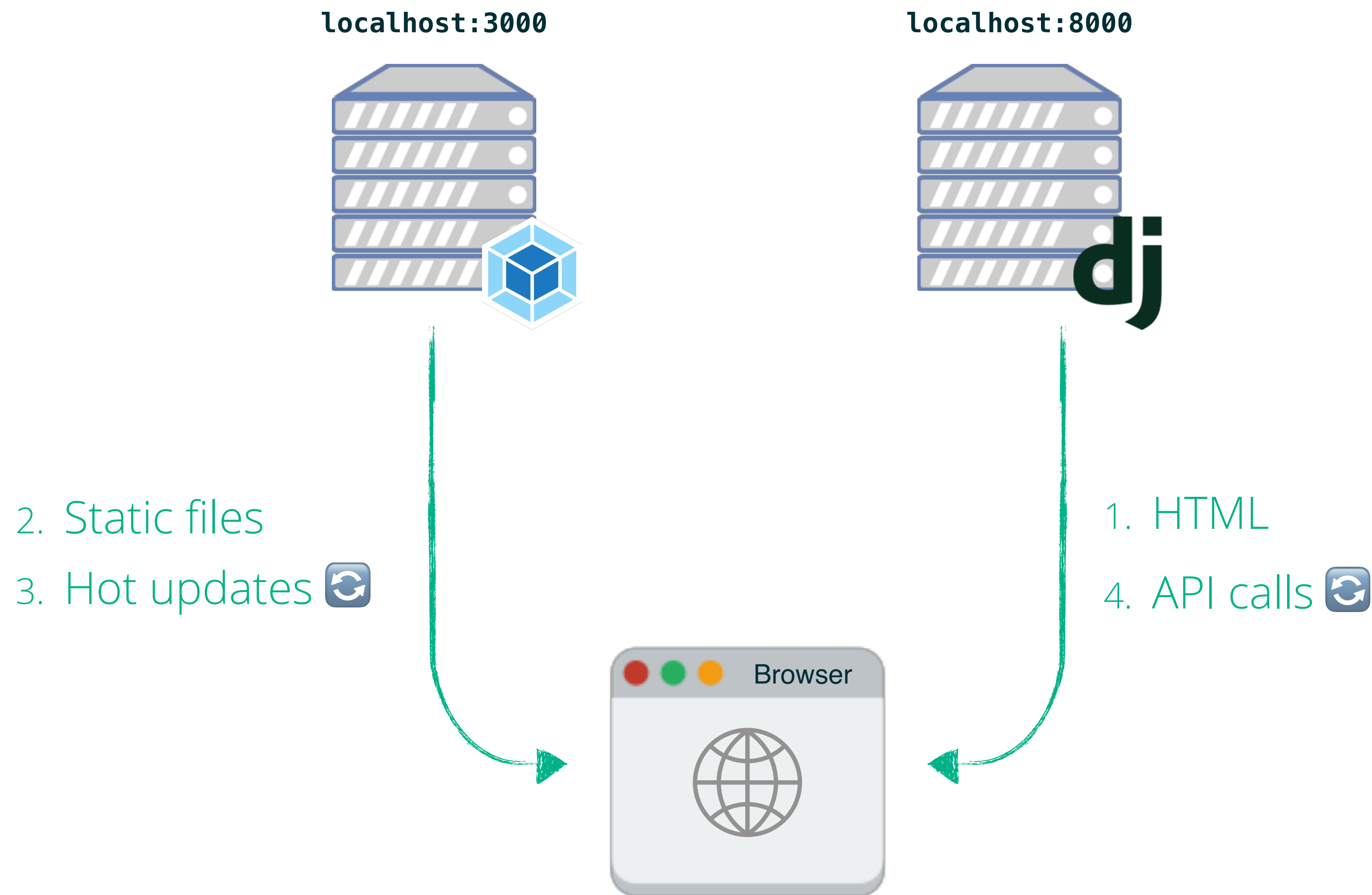
`./my_project/urls.py`



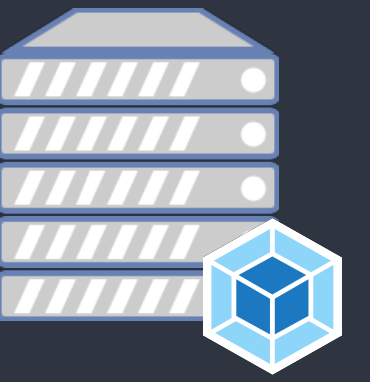
```
from django.conf import settings
from django.contrib import admin
from django.urls import path, re_path

from front.views import FrontendView

urlpatterns = [
    path("admin/", admin.site.urls),
    re_path(r"^[a-zA-Z0-9/-]*$", FrontendView.as_view(), name="app")
]
```



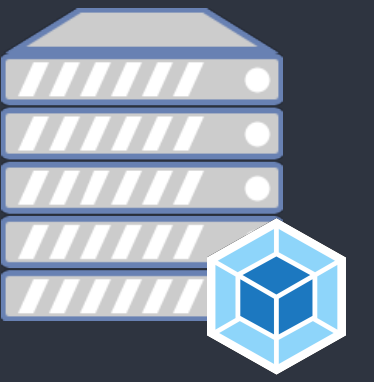
./frontend/webpack.config.js



```
const BundleTracker = require('webpack-bundle-tracker');

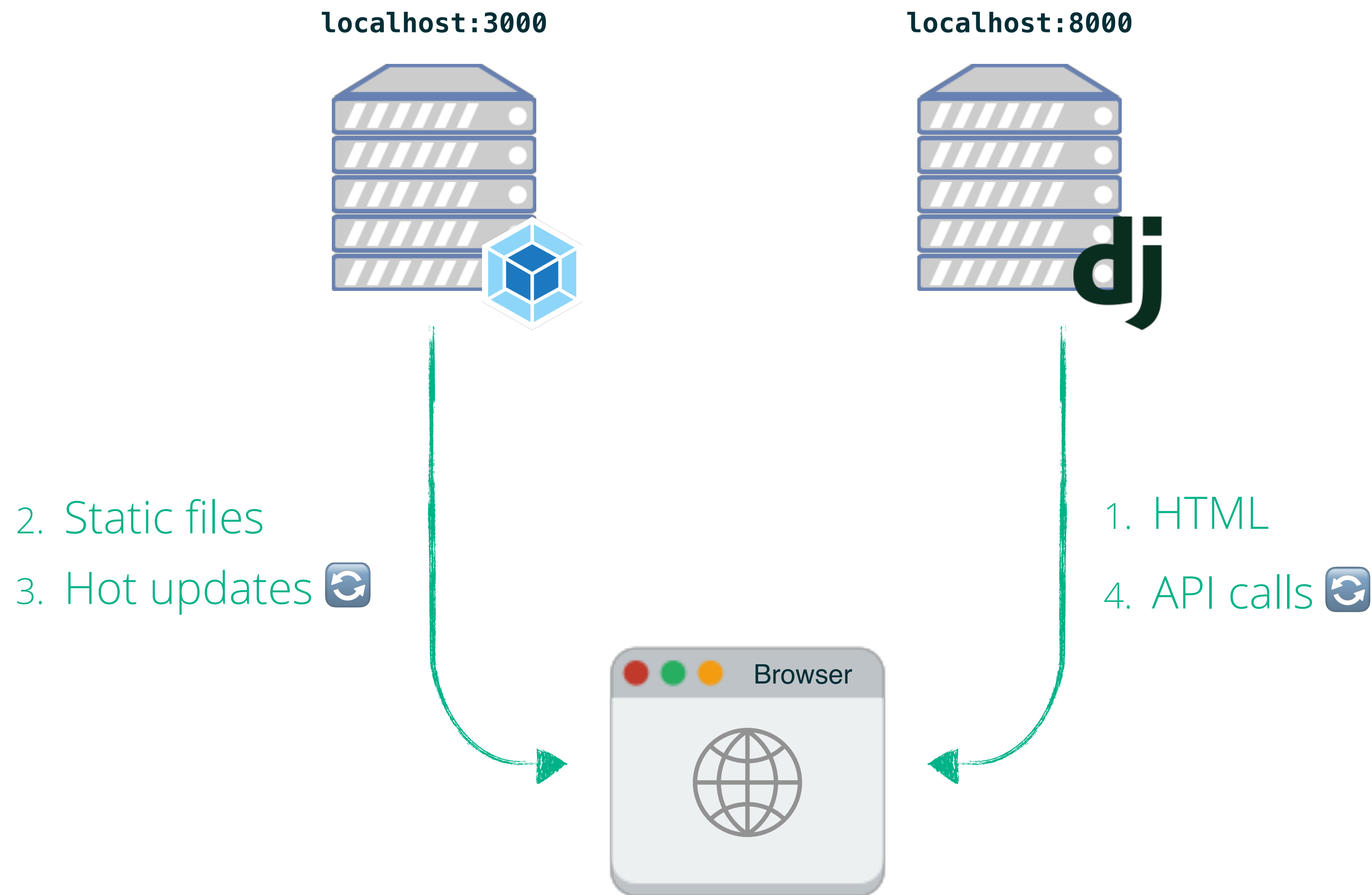
module.exports = {
  context: __dirname,
  entry: {
    app: ['./app']
  },
  output: {
    path: require("path").resolve('./assets/bundles/'),
    filename: "[name]-[hash].js",
    publicPath: 'http://localhost:3000/assets/bundles/',
  },
  plugins: [
    new BundleTracker({path: __dirname, filename: './assets/webpack-stats.json'})
  ]
}
```

./frontend/webpack.config.js



```
const BundleTracker = require('webpack-bundle-tracker');

module.exports = {
  context: __dirname,
  entry: {
    app: ['./app']
  },
  output: {
    path: require("path").resolve('./assets/bundles/'),
    filename: "[name]-[hash].js",
    publicPath: 'http://localhost:3000/assets/bundles/',
  },
  plugins: [
    new BundleTracker({path: __dirname, filename: './assets/webpack-stats.json'})
  ]
}
```



./devops/Dockerfile

```
# First stage: build front app
FROM node:10 AS node
WORKDIR /code

COPY ./frontend /code/
RUN yarn
RUN yarn build

# Second stage: build base backend
FROM python:3.7

ENV DJANGO_SETTINGS_MODULE myproject.settings.prod
ENV PYTHONPATH /code

WORKDIR /code

RUN pip install pipenv gunicorn

COPY ./backend /code/
COPY --from=node /code/build /code/front/static/front

RUN pipenv install --system --deploy

# Collect statics
RUN python ./manage.py collectstatic
```

./devops/Dockerfile

```
# First stage: build front app
FROM node:10 AS node
WORKDIR /code

COPY ./frontend /code/
RUN yarn
RUN yarn build

# Second stage: build base backend
FROM python:3.7

ENV DJANGO_SETTINGS_MODULE myproject.settings.prod
ENV PYTHONPATH /code

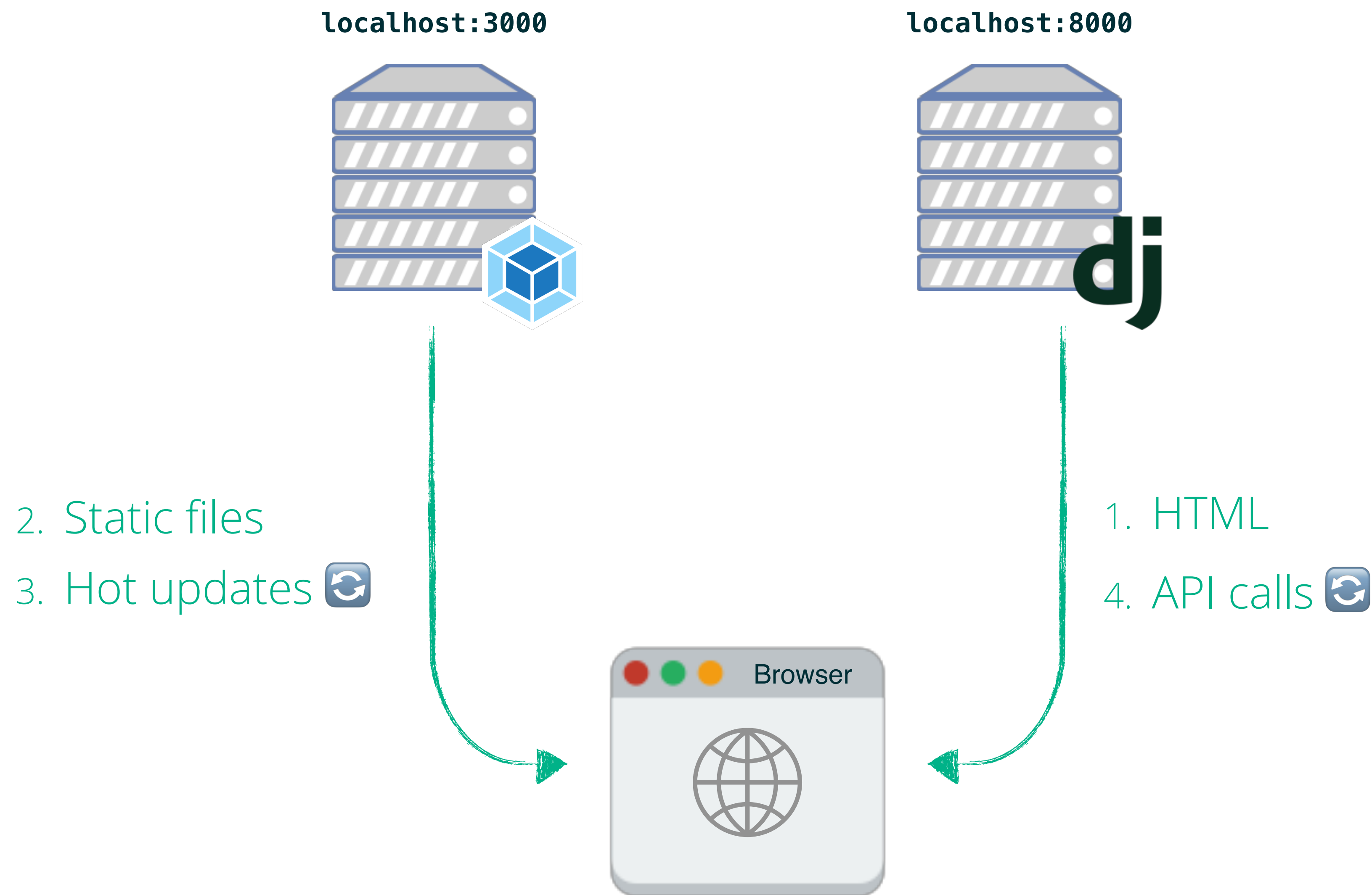
WORKDIR /code

RUN pip install pipenv gunicorn

COPY ./backend /code/
COPY --from=node /code/build /code/front/static/front

RUN pipenv install --system --deploy

# Collect statics
RUN python ./manage.py collectstatic
```

Other interesting topics

- Authentication (JWT / cookies)
- CORS
- CSRF
- SEO
- PWA support

Thank you!

 /n6g7  /n4ng5l nathang@theodo.co.uk

Thank you!

 /n6g7  /n4ng5l nathang@theodo.co.uk