JPL WWW Redesign

Going Headless with Wagtail, Nuxt.js and GraphQL

183x | DesignLab

Presented by Stephanie C Smith

Jet Propulsion Laboratory
California Institute of Technology

This document has been reviewed and determined not to contain export controlled technical data.

About the Project

- Small team of on-site developers working with Torchbox to develop the redesigned JPL website
- High-profile, High-traffic
- Wide range of visitors and entry points
- Internal workflows lead to published content
- Thousands of pages

Note: This presentation is NOT a design preview. It is an abstract of our development strategy.

This document has been reviewed and determined not to contain export controlled technical data.

Key Decision: Why Headless?

- Modern approach
- Flexibility for the future: Decoupling the backend from the frontend allows us to change and adapt
- Performance: snappy and reactive front-end (if done right) can improve overall user experience

This document has been reviewed and determined not to contain export controlled technical data.

Backend

For required markings, please visit https://mh.jpl.nasa.gov

Backend Setup

- CMS: Wagtail
- API Framework: GraphQL (via Graphene)
    - Modern approach
    - Better developer experience, and can query for the exact content needed
    - Reduces over-fetching
    - Issues with monitoring and caching, but not as much an issue with SSG (more on this later)
- Headless Preview: allows us to expose the preview data to the frontend. See: https://github.com/torchbox/wagtail-headless-preview

This document has been reviewed and determined not to contain export controlled technical data.

Example data model: Mission Page

```python
class Mission(Page):
    """ Detail page for Missions """

    # Database fields
    thumbnail_image = models.ForeignKey(
        'backend.CustomImage',
        null=True,
        blank=False,
        on_delete=models.SET_NULL,
        help_text='Please provide a thumbnail image',
        related_name="+",
    )

    summary = models.TextField(
        help_text='A description of the page',
        blank=True,
    )
    body = StreamField(
        JStreamExtraBlock(required=False),
        verbose_name="Page body",
        blank=True,
        null=True,
    )

    file = models.FileField(
        upload_to='data-models/',
        blank=True,
        null=True,
        validators=[FileExtensionValidator(['glb'])],
        help_text=mark_safe("Select the <strong>.glb</strong> file type for the model"),
    )
    auto_rotate = models.BooleanField(
        default=False,
        help_text="Select checkbox to enable auto rotation of the model",
    )
```

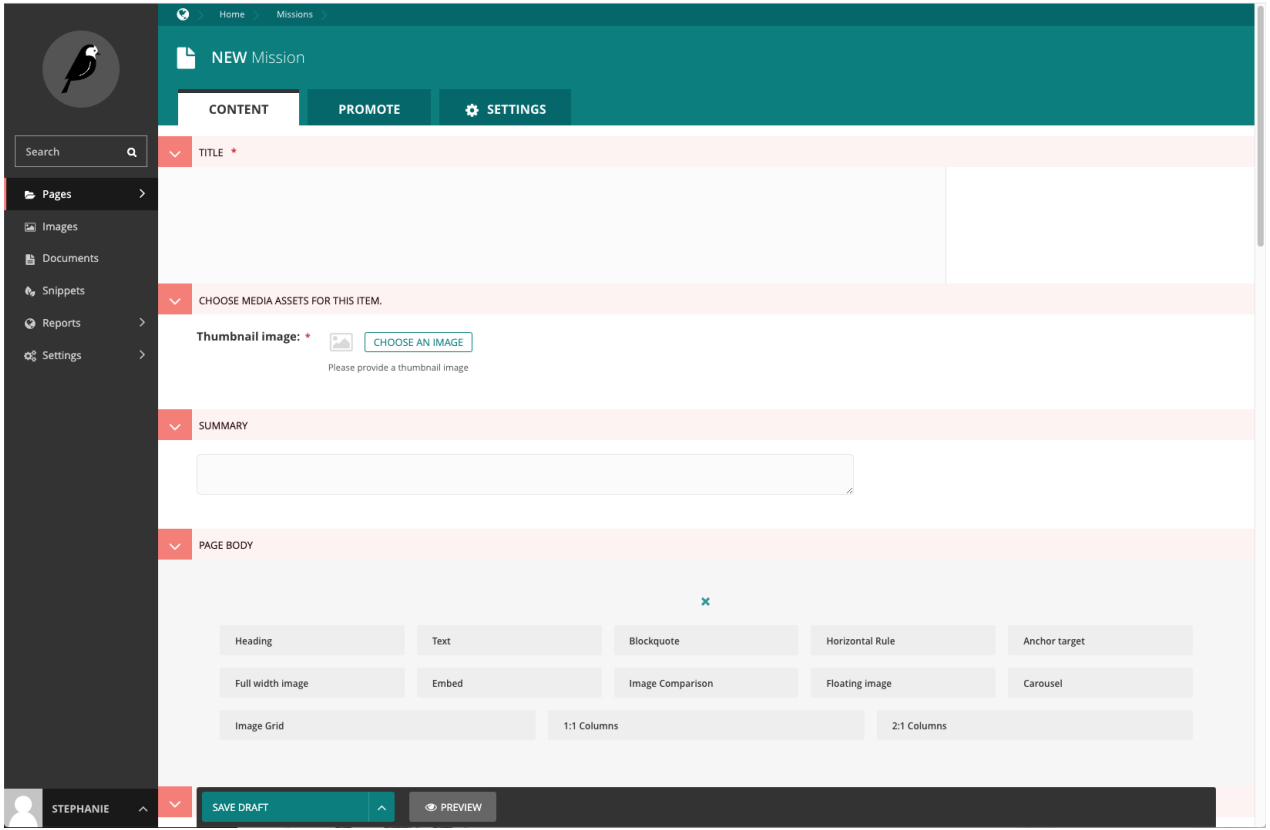
```python
    # Editor panels configuration
    content_panels = Page.content_panels + [
        ImageChooserPanel('thumbnail_image'),
        FieldPanel('summary'),
        StreamFieldPanel('body'),
        MultiFieldPanel([
            FieldPanel('file'),
            FieldPanel('auto_rotate'),
        ], heading="3D Model Settings"),
    ]

    # Search Fields
    search_fields = Page.search_fields + [
        index.SearchField('summary'),
        index.SearchField('body'),
    ]

    class Meta:
        verbose_name = "Mission"
        verbose_name_plural = "Missions"
```


This document has been reviewed and determined not to contain export controlled technical data.

Example data model: Mission Page Admin View



This document has been reviewed and determined not to contain export controlled technical data.

API: Create GraphQL nodes using Graphene

```python
# schema.py
import graphene
from django.conf import settings
from missions.models import Mission
from graphene_django import DjangoObjectType
from . import graphene_wagtail

class MissionNode(DjangoObjectType):
    thumbnail_image = graphene.String(width=graphene.Int())

    def resolve_thumbnail_image(self, info, width=860):
        if not self.thumbnail_image:
            return ""

        return self.thumbnail_image.get_rendition('width-%s' % width).url

    class Meta:
        model = Mission

class Query(graphene.ObjectType):
    mission = graphene.Field(MissionNode, slug=graphene.String())
    missions = graphene.List(MissionNode)

    def resolve_mission(self, info, slug):
        return Mission.objects.live().get(slug=slug)

    def resolve_missions(self, info):
        return Mission.objects.live().all()

schema = graphene.Schema(query=Query)
```


This document has been reviewed and determined not to contain export controlled technical data.

API: What about Streamfield?

```python
# graphene_wagtail.py
from wagtail.core.fields import StreamField
from graphene.types import Scalar
from graphene_django.converter import convert_django_field


class GenericStreamFieldType(Scalar):
    @staticmethod
    def serialize(stream_value):
        return stream_value.stream_data

@convert_django_field.register(StreamField)
def convert_stream_field(field, registry=None):
    return GenericStreamFieldType(
        description=field.help_text, required=not field.null
    )
```


This document has been reviewed and determined not to contain export controlled technical data.

Result: Parse-able mission object

```
query getMission {
  mission(slug: "mars-2020") {
    id
    title
    slug
    summary
    thumbnailImage
    body
  }
}
```


```
{
  "data": {
    "mission": {
      "body": [
        {
          "id": "ce09ceac-17d0-489b-b16d-4d13cbd54146",
          "type": "text",
          "value": "<p>Lorem ipsum page body</p>"
        },
        {
          "id": "36a9e4a1-2535-4315-b262-08bfffd13c80",
          "type": "heading",
          "value": {
            "heading_text": "A heading!",
            "size": "h2"
          }
        },
        {
          "id": "72b3d404-4a1f-4baf-b943-5274812f9e58",
          "type": "text",
          "value": "<p>More text</p>"
        }
      ],
      "id": "5",
      "slug": "mars-2020",
      "summary": "Lorem ipsum summary info",
      "thumbnailImage": "http://127.0.0.1:9000/media/images/mars2020-PIA23719-1920x1200.width-860.jpg",
      "title": "Mars 2020"
    }
  }
}
```


This document has been reviewed and determined not to contain export controlled technical data.

Querying for all missions returns an array

```
query getMissions {
  missions {
    id
    title
    slug
    summary
  }
}
```


```
{
  "data": {
    "missions": [
      {
        "id": "5",
        "slug": "mars-2020",
        "summary": "Lorem ipsum summary info",
        "title": "Mars 2020"
      },
      {
        "id": "6",
        "slug": "example-mission",
        "summary": "Nunc augue est, porta sed nisl ac, dictum dignissim tellus.",
        "title": "Example Mission"
      }
    ]
  }
}
```


This document has been reviewed and determined not to contain export controlled technical data.

Frontend

This document has been reviewed and determined not to contain export controlled technical data.

Frontend Stack

- Vue.js – great developer experience, solid user-base
- Nuxt.js to leverage its features (there are many!)
- Apollo Client to manage data with GraphQL

This document has been reviewed and determined not to contain export controlled technical data.

Using GraphQL queries in Nuxt.js:

Mission landing page

```
<template>
  <div class="lg:container mx-auto my-5">
    <h1 class="text-4xl">{{ missionIndexPage.title }}</h1>
    <p class="lead">
      {{ missionIndexPage.summary }}
    </p>
    <Cards :data="missions" path="missions" />
  </div>
</template>

<script lang="ts">
import Vue from 'vue'
import gql from 'graphql-tag'

export default Vue.extend({
  async asyncData({ app }) {
    const client = app.apolloProvider.defaultClient
    const missionIndexPageResponse = await client.query({
      query: gql`
        query getMissionIndexPage {
          missionIndexPage {
            title
            summary
          }
          missions {
            id
            title
            slug
            thumbnailImage
            summary
          }
        }
      `,
    })
    const missionIndexPageData = await missionIndexPageResponse.data
    return { ...missionIndexPageData }
  },
  head() {
    return {
      title: 'Missions',
    }
  },
})
</script>
```


This document has been reviewed and determined not to contain export controlled technical data.

Cards Component

```
<template>
  <div class="cards">
    <nuxt-link :to="'/' + path + '/' + item.slug"  v-for="item in data" :key="item.id">
      <img :src="item.thumbnailImage" class="w-full" :alt="item.title">
      <div>
        <h3 v-html="item.title"></h3>
        <p v-html="item.summary"></p>
      </div>
      <button>
        Learn more
      </button>
    </nuxt-link>
  </div>
</template>

<script>
export default {
  name: 'Cards',
  props: {
    data: {
      type: Array,
      default: () => [],
    },
    path: {
      type: String,
      default: '',
    },
  },
}
</script>
```


This document has been reviewed and determined not to contain export controlled technical data.

Detail page using Nuxt.js' dynamic routes based on slug

```
<template>
  <main role="main" class="position-relative">
    <div>
      <div class="mission-header flex items-center justify-center">
        <div class="w-full md:w-2/5 lg:w-1/5">
          <h1 class="text-4xl">{{ mission.title }}</h1>
          <div class="lead" v-html="mission.summary"></div>
        </div>
        <div class="w-full md:w-3/5 lg:4/5">
          <img class="w-full my-5" :src="mission.thumbnailImage">
        </div>
      </div>
    </div>
    <div class="lg:container mx-auto">
      <Streamfield :data="mission.body" />
    </div>
  </main>
</template>
<script>
import gql from 'graphql-tag'

export default {
  async asyncData({ params, app }) {
    const client = app.apolloProvider.defaultClient
    const response = await client.query({
      query: gql`
        query getMission {
          mission(slug: "${params.slug}") {
            slug
            title
            thumbnailImage
            summary
            body
          }
        }
      `,
    })
    const data = await response.data
    return {
      ...data,
    }
  },
}
</script>
```

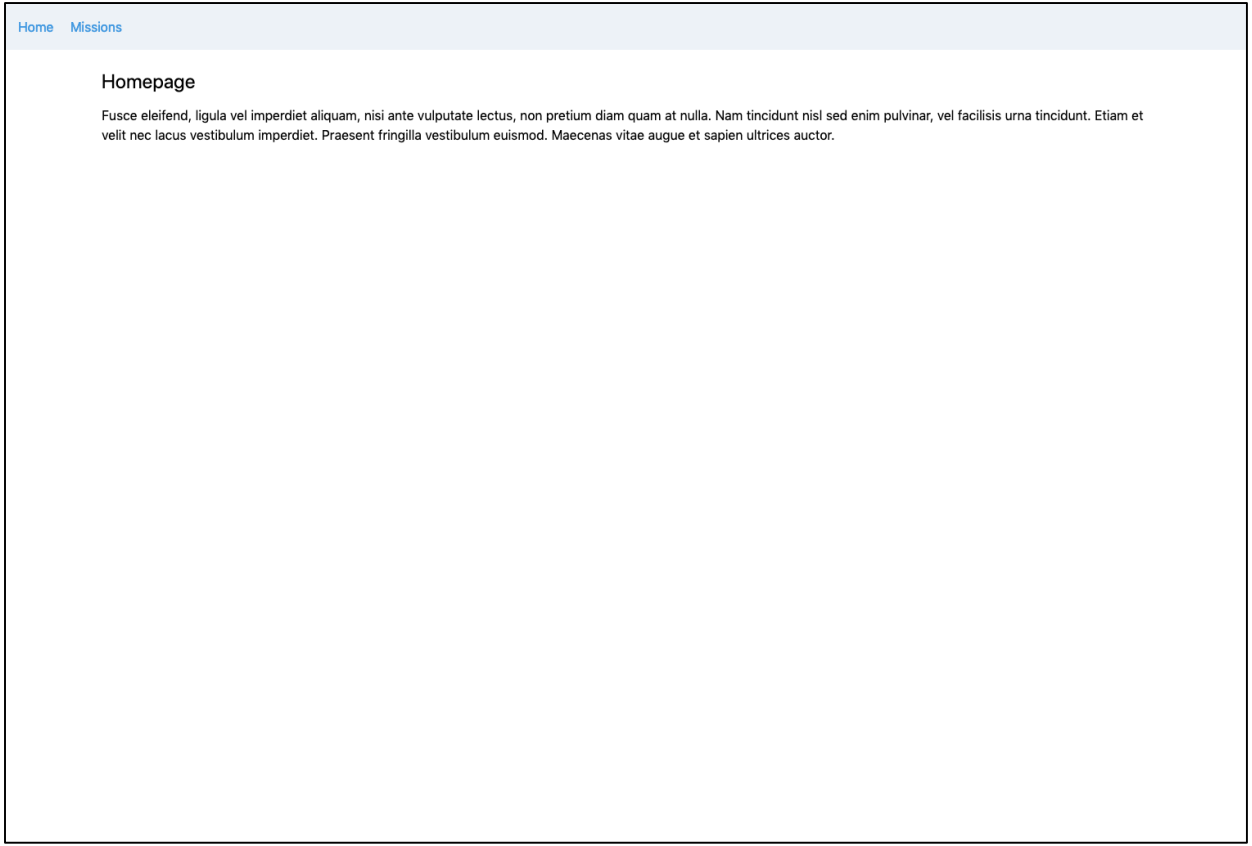

This document has been reviewed and determined not to contain export controlled technical data.

Streamfield block component

```
<template>
  <div class="block-streamfield">
    <div v-for="block in data">
      <div v-if="block.type == 'heading'">
        <Heading :data="block.value" />
      </div>
      <div v-else-if="block.type == 'text'">
        <div class="block-text" v-html="block.value"></div>
      </div>
      <div v-else>
        <pre>{{ block }}</pre>
      </div>
    </div>
  </div>
</template>
<script>
import Heading from '@/components/Blocks/Heading/Heading.vue'
export default {
  name: 'WagtailStreamfield',
  components: {
    Heading,
  },
  props: {
    data: {
      type: Array,
      required: true,
    },
  },
}
</script>
```

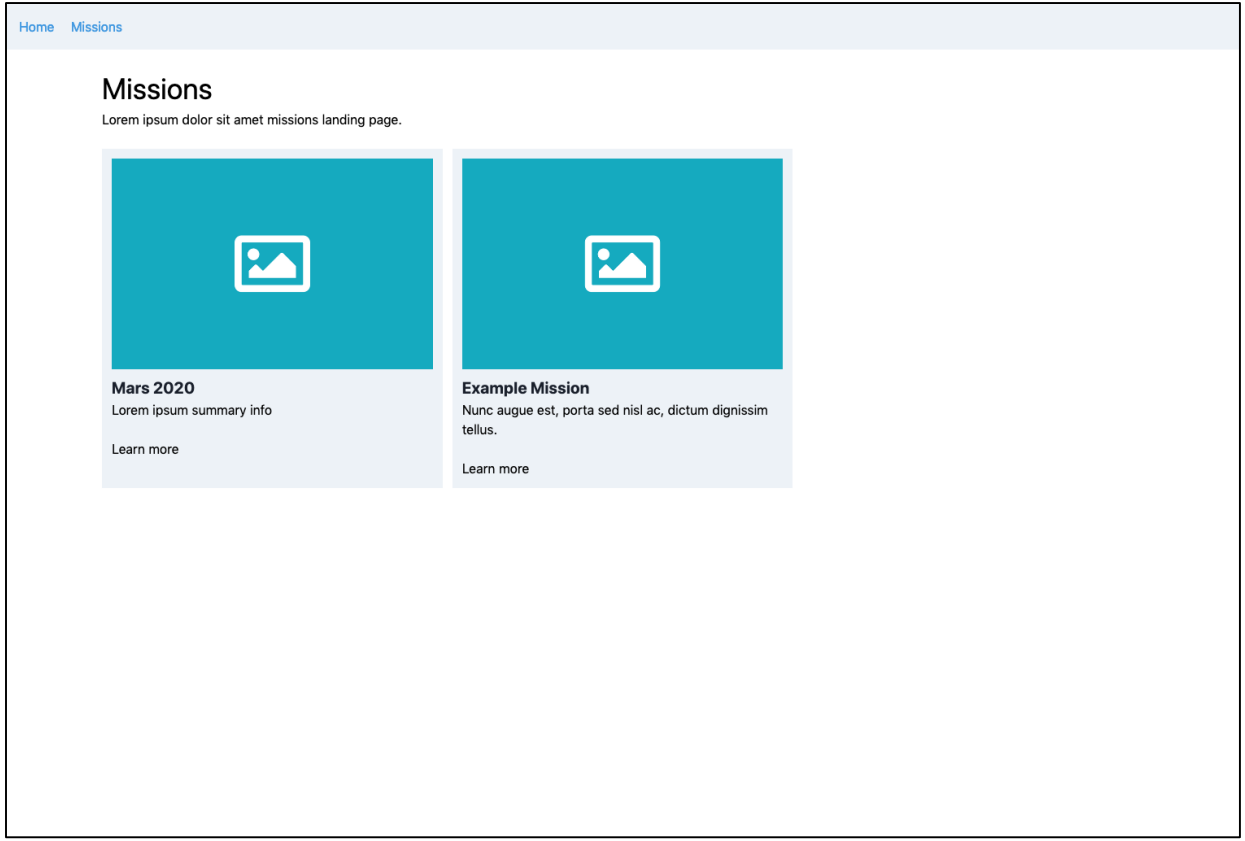

This document has been reviewed and determined not to contain export controlled technical data.

Rendered: Homepage



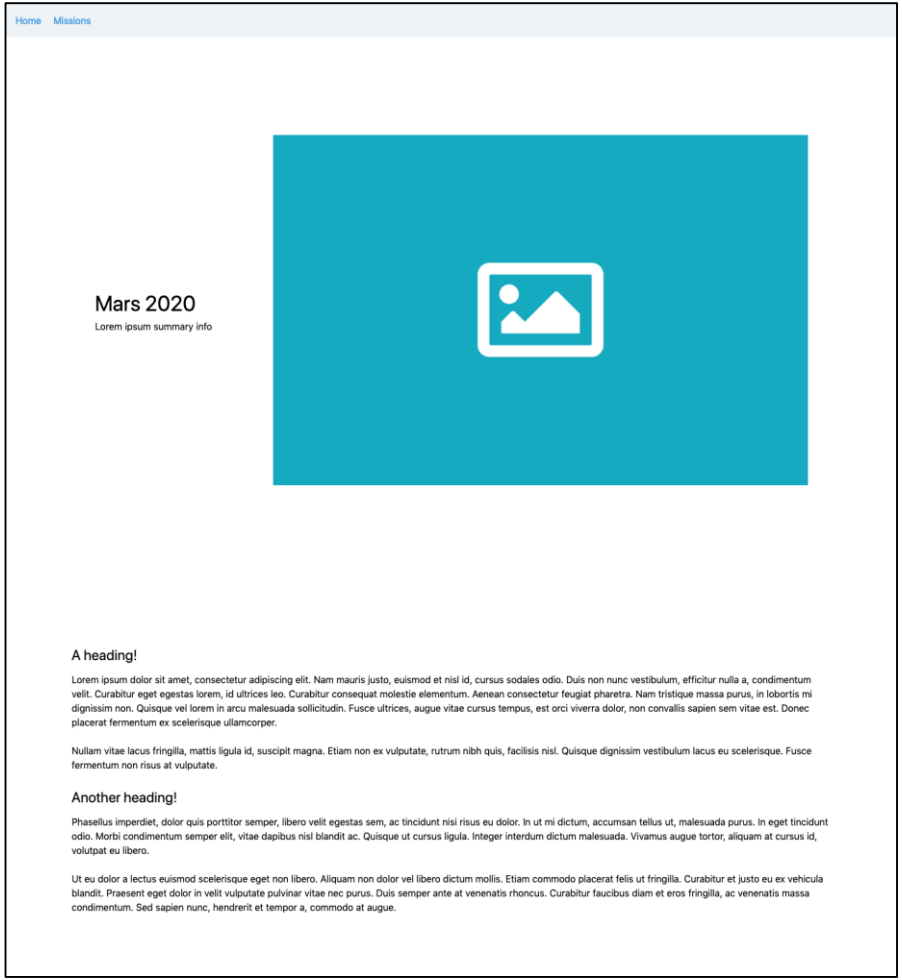
This document has been reviewed and determined not to contain export controlled technical data.

Rendered: Landing page



This document has been reviewed and determined not to contain export controlled technical data.

Rendered:
A mission page



This document has been reviewed and determined not to contain export controlled technical data.

Leveraging Static Site Generation

- Nuxt.js has this option built in
- Super-fast page loading
- The API is required only during site generation. It is never accessed client-side.

This document has been reviewed and determined not to contain export controlled technical data.

Bigger questions

- Optimizing static site generation
  - Selectively rebuild only changed pages rather than the entire site
- Site search
  - How to leverage Wagtail's built-in capability?
  - 3rd party services may be the answer (AWS Elasticsearch)

This document has been reviewed and determined not to contain export controlled technical data.

Always a work in progress

- Web development is a rapidly evolving field
- Even as we're working on initial proof-of-concepts, new technologies are available every day, continuously shaping our approach
- Watch for the open-source release!

This document has been reviewed and determined not to contain export controlled technical data.

Jet Propulsion Laboratory
California Institute of Technology

This document has been reviewed and determined not to contain export controlled technical data.