Transitions

Nuxt leverages Vue's `<Transition>` component to apply transitions between pages and layouts.

Page transitions

You can enable page transitions to apply an automatic transition for all your pages.

nuxt.config.ts

```ts
export default defineNuxtConfig({
  app: {
    pageTransition: { name: 'page', mode: 'out-in' }
  },
})
```

If you are changing layouts as well as page, the page transition you set here will not run. Instead, you should set a layout transition.

To start adding transition between your pages, add the following CSS to your `app.vue`:

app.vue pages/index.vue pages/about.vue

app.vue

```vue
<template>
  <NuxtPage />
</template>

<style>
.page-enter-active,
.page-leave-active {
  transition: all 0.4s;
```

```
}
.page-enter-from,
.page-leave-to {
  opacity: 0;
  filter: blur(1rem);
}
</style>
```

This produces the following result when navigating between pages:

To set a different transition for a page, set the `pageTransition` key in `definePageMeta` of the page:

pages/about.vue app.vue

pages/about.vue

```
<script setup lang="ts">
definePageMeta({
  pageTransition: {
    name: 'rotate'
  }
})
</script>
```

Moving to the about page will add the 3d rotation effect:

Layout transitions

You can enable layout transitions to apply an automatic transition for all your layouts.

nuxt.config.ts

```ts
export default defineNuxtConfig({
  app: {
    layoutTransition: { name: 'layout', mode: 'out-in' }
  },
})
```

To start adding transition between your pages and layouts, add the following CSS to your `app.vue`:

app.vue | layouts/default.vue | layouts/orange.vue | pages/index.vue | pages/about.vue

app.vue

```vue
<template>
  <NuxtLayout>
    <NuxtPage />
  </NuxtLayout>
</template>

<style>
.layout-enter-active,
.layout-leave-active {
  transition: all 0.4s;
}
.layout-enter-from,
.layout-leave-to {
```

```
  filter: grayscale(1);
}
</style>
```

This produces the following result when navigating between pages:

Similar to `pageTransition`, you can apply a custom `layoutTransition` to the page component using `definePageMeta`:

pages/about.vue

```
<script setup lang="ts">
definePageMeta({
  layout: 'orange',
  layoutTransition: {
    name: 'slide-in'
  }
})
</script>
```

Global settings

You can customize these default transition names globally using `nuxt.config`.

Both `pageTransition` and `layoutTransition` keys accept `TransitionProps` as JSON serializable values where you can pass the `name`, `mode` and other valid transition-props of the custom CSS transition.

nuxt.config.ts

```
export default defineNuxtConfig({
  app: {
```

```
    pageTransition: {
      name: 'fade',
      mode: 'out-in' // default
    },
    layoutTransition: {
      name: 'slide',
      mode: 'out-in' // default
    }
  }
})
```

If you change the `name` property, you also have to rename the CSS classes accordingly.

To override the global transition property, use the `definePageMeta` to define page or layout transitions for a single Nuxt page and override any page or layout transitions that are defined globally in `nuxt.config` file.

pages/some-page.vue

```
<script setup lang="ts">
definePageMeta({
  pageTransition: {
    name: 'bounce',
    mode: 'out-in' // default
  }
})
</script>
```

Disable Transitions

`pageTransition` and `layoutTransition` can be disabled for a specific route:

pages/some-page.vue

```
<script setup lang="ts">
definePageMeta({
  pageTransition: false,
  layoutTransition: false
})
</script>
```

Or globally in the `nuxt.config` :

nuxt.config.ts

```
defineNuxtConfig({
```

```
  app: {
    pageTransition: false,
    layoutTransition: false
  }
})
```

JavaScript Hooks

For advanced use-cases, you can use JavaScript hooks to create highly dynamic and custom transitions for your Nuxt pages.

This way presents perfect use-cases for JavaScript animation libraries such as GSAP or Tween.js.

pages/some-page.vue

```
<script setup lang="ts">
definePageMeta({
  pageTransition: {
    name: 'custom-flip',
    mode: 'out-in',
    onBeforeEnter: (el) => {
      console.log('Before enter...')
    },
    onEnter: (el, done) => {},
    onAfterEnter: (el) => {}
  }
})
</script>
```

Learn more about additional JavaScript hooks available in the `Transition` component.

Dynamic Transitions

To apply dynamic transitions using conditional logic, you can leverage inline middleware to assign a different transition name to `to.meta.pageTransition` .

pages/[id].vue layouts/default.vue

pages/[id].vue

```
<script setup lang="ts">
```

```
definePageMeta({
  pageTransition: {
    name: 'slide-right',
    mode: 'out-in'
  },
  middleware (to, from) {
    to.meta.pageTransition.name = +to.params.id > +from.params.id ? 'slide-left' : 'slide-right'
  }
})
</script>

<template>
  <h1>#{{ $route.params.id }}</h1>
</template>

<style>
.slide-left-enter-active,
.slide-left-leave-active,
.slide-right-enter-active,
.slide-right-leave-active {
  transition: all 0.2s;
}
.slide-left-enter-from {
  opacity: 0;
  transform: translate(50px, 0);
}
.slide-left-leave-to {
  opacity: 0;
  transform: translate(-50px, 0);
}
.slide-right-enter-from {
  opacity: 0;
  transform: translate(-50px, 0);
}
.slide-right-leave-to {
  opacity: 0;
  transform: translate(50px, 0);
}
</style>
```

The page now applies the `slide-left` transition when going to the next id and `slide-right` for the previous:

Transition with NuxtPage

When `<NuxtPage />` is used in `app.vue`, transition-props can be passed directly as a component props to activate global transition.

app.vue

```
<template>
  <div>
    <NuxtLayout>
      <NuxtPage :transition="{
        name: 'bounce',
        mode: 'out-in'
      }" />
    </NuxtLayout>
  </div>
</template>
```

Remember, this page transition cannot be overridden with `definePageMeta` on individual pages.

View Transitions API (experimental)

Nuxt ships with an experimental implementation of the **View Transitions API** (see MDN). This is an exciting new way to implement native browser transitions which (among other things) have the ability to transition between unrelated elements on different pages.

The Nuxt integration is under active development, but can be enabled with the `experimental.viewTransition` option in your configuration file:

nuxt.config.ts

```ts
export default defineNuxtConfig({
  experimental: {
    viewTransition: true
  }
})
```

If you are also using Vue transitions like `pageTransition` and `layoutTransition` (see above) to achieve the same result as the new View Transitions API, then you may wish to *disable* Vue transitions if the user's browser supports the newer, native web API. You can do this by creating `~/middleware/disable-vue-transitions.global.ts` with the following contents:

```ts
export default defineNuxtRouteMiddleware(to => {
  if (!document.startViewTransition) { return }

  // Disable built-in Vue transitions
  to.meta.pageTransition = false
  to.meta.layoutTransition = false
})
```

Known issues

- View transitions may not work as expected with nested pages/layouts/async components owing to this upstream Vue bug: https://github.com/vuejs/core/issues/5513. If you make use of this pattern, you may need to delay adopting this experimental feature or implement it yourself. Feedback is very welcome.
- If you perform data fetching within your page setup functions, that you may wish to reconsider using this feature for the moment. (By design, View Transitions completely freeze DOM updates whilst they are

Edit on Github