cloud connect

A cloud native company

Latihan Aplikasi Moden

Jabatan Digital Negara

9 - 12 September 2024

cloud connect

Material Latihan:
https://bit.ly/jdn-lpam

Google Meet:
https://bit.ly/jdn-lpam-meet1

Latihan Hari Ketiga

cloud connect

HARI KEDUA	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Kedua
Sesi 2	Pengenalan kepada Vue.js
Sesi 3	Pengenalan kepada Modern UI Framework
Sesi 4	Pembangunan UI Framework
Sesi 5	Pengenalan kepada Nuxt.js
Sesi 6	Pembangunan aplikasi web front-end menggunakan Nuxt.js versi 3: • Routing, Client-side / Server-side / Universal Rendering, Pages, Layouts, Components, State Management using Pinia

Ringkasan Latihan Hari Kedua

cloud connect

HARI KEDUA	
Sesi	Topik
Sesi 1	Ringkasan Latihan Hari Pertama
Sesi 2	Fastify Validation & Serialization
Sesi 3	Pengenalan kepada PrismaORM
Sesi 4	Pembangunan API untuk MySQL Database sediada
Sesi 5	Pembangunan API untuk MySQL Database baru
Sesi 6	Pengenalan kepada Jamstack Framework

https://jamstack.org/

Jamstack

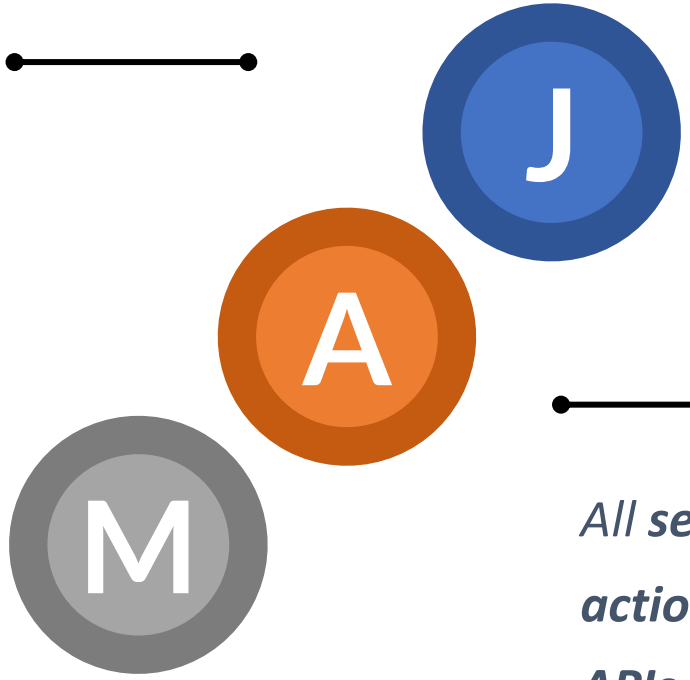
cloud connect

Javascript

*Any dynamic programming during the request/response cycle is handled by **JavaScript**, running entirely on the client.*

Markup

*Templated **markup** should be **prebuilt** at **deploy time**, usually using a site generator for content sites, or a build tool for web apps.*

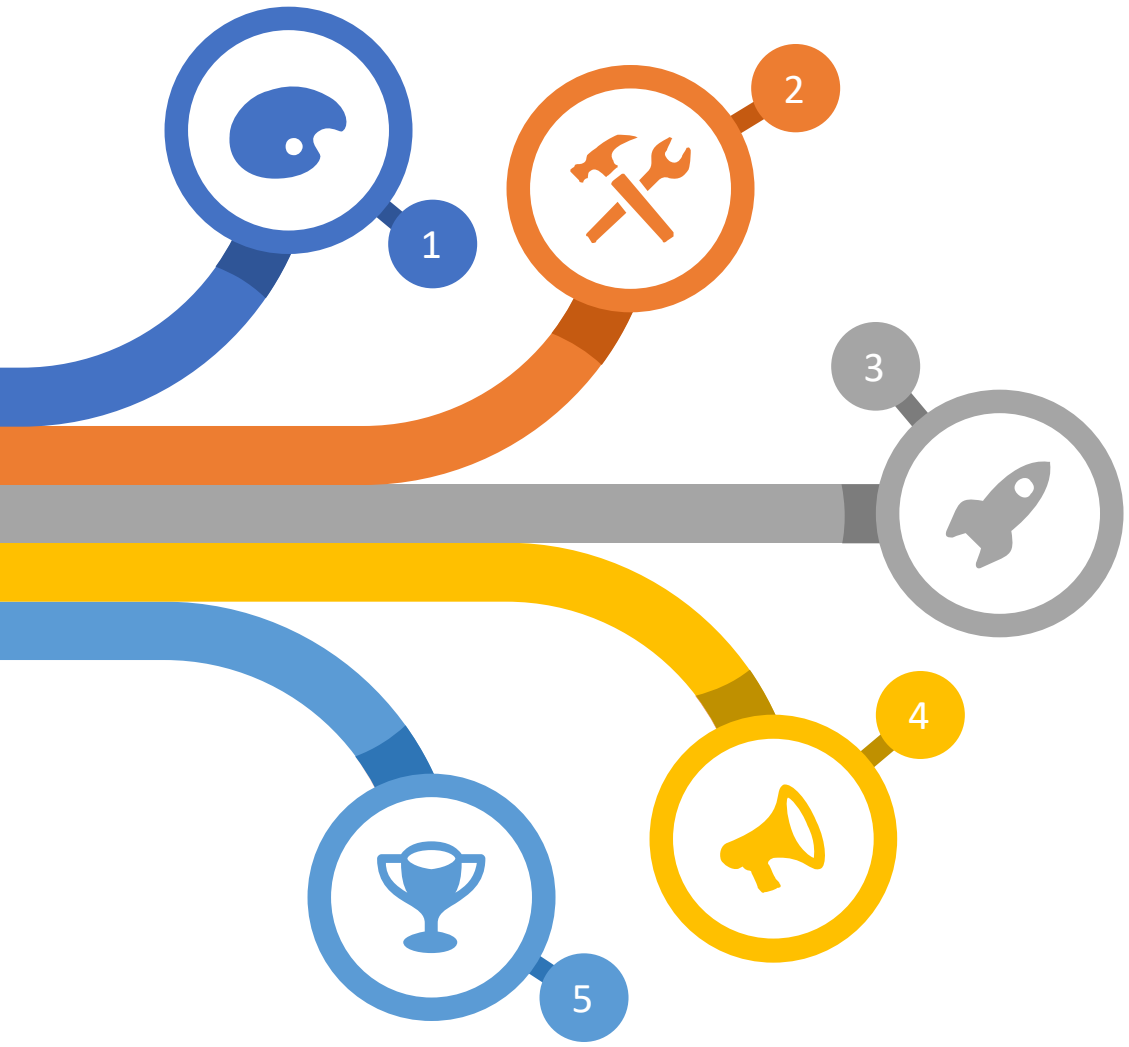


APIs

*All **server-side processes** or **database actions** are abstracted into reusable **APIs**, accessed over HTTPS with JavaScript.*

Kenapa *Jamstack*?

cloud connect



1

Faster and Better Performance

generate new pages at deploy time and serve pre-built markup and assets over a CDN.

2

Less expensive and easier to scale

lesser complexity of development reduces costs and also hosting of static files is cheap or even free

3

Higher Security

Static websites have a very low potential for vulnerabilities because it is just HTML files and external API-handling served over a CDN

4

Better developer experience

not difficult to learn. With just HTML, CSS and JS experience, developers can build complex websites

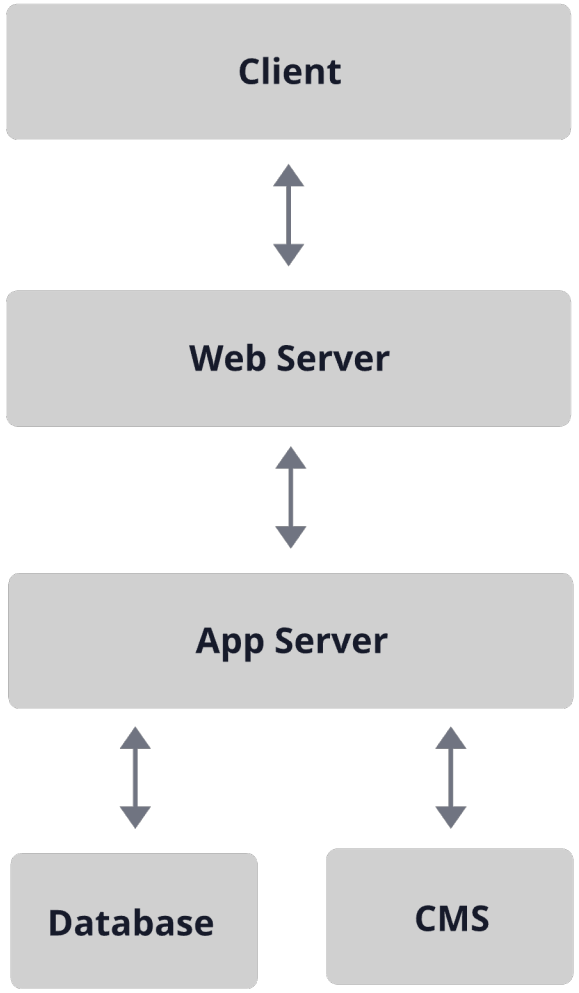
5

Great community

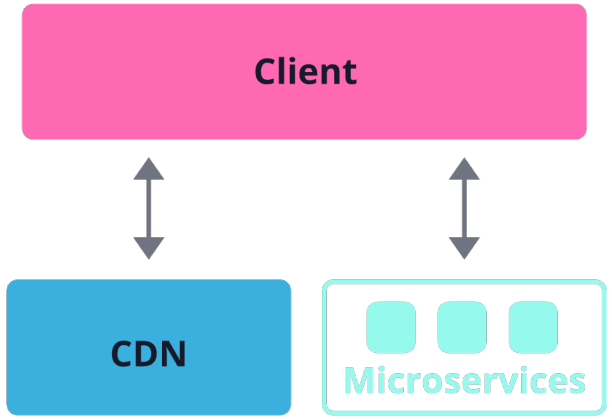
The JAMstack community is growing over time

Jamstack

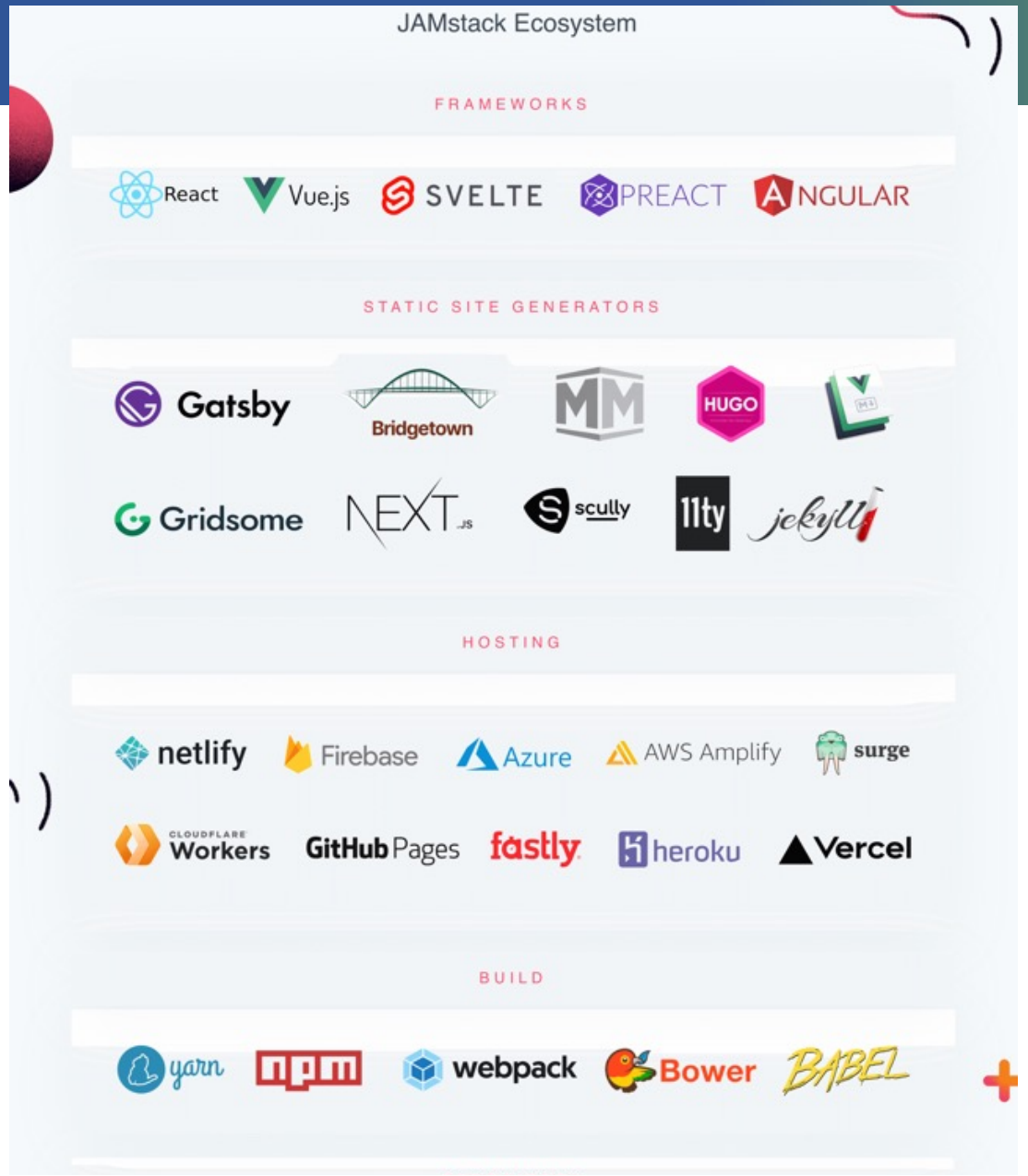
cloud connect



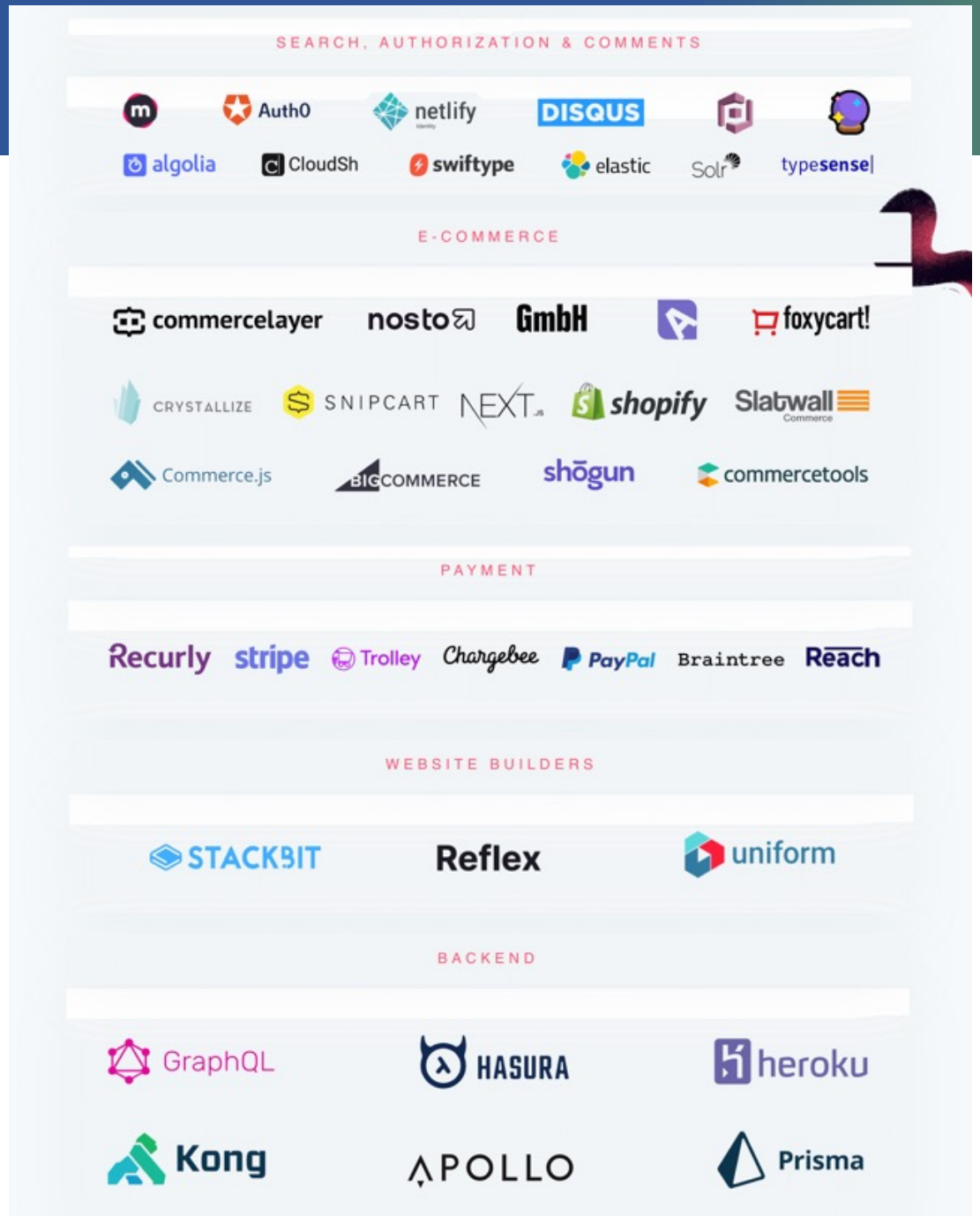
Traditional Web Apps



Jamstack


JAMstack Ecosystem
FRAMEWORKS
React
Vue.js
SVELTE
PREACT
ANGULAR
STATIC SITE GENERATORS
Gatsby
Bridgetown
MM
HUGO
Gridsome
NEXT.js
scully
11ty
jekyll
HOSTING
netlify
Firebase
Azure
AWS Amplify
surge
CLOUDFLARE Workers
GitHub Pages
fastly
heroku
Vercel
BUILD
yarn
npm
webpack
Bower
BABEL


REPOSITORY
git
GitHub
GitLab
Bitbucket
CONTENT MANAGEMENT
GIT-based
FORESTRY
CRAFTERCMS
netlifycms
Publii
FrontAid CMS
API-FIRST
SANITY
DATO CMS
strapi
TakeShape
contentful
storyblok
agility
graphcms
ButterCMS
Kontent.
prismic.io
magnolia
FUNCTIONS
netlify Functions
Azure Functions
Cloud Function for Firebase
Architect
UTILITY APIs
DATABASE
FAUNA
DynamoDB
8base
Sheetsu
mongoDB.
Airtable
FORMS
mailchimp
Paperform
Google Forms
netlify
FORMSPREE
Typeform
FormKeep
StaticForms


SEARCH, AUTHORIZATION & COMMENTS
Auth0
netlify
DISQUS
algolia
CloudSh
swiftype
elastic
Solr
typesense
E-COMMERCE
commercelayer
nosto
GmbH
foxycart!
CRYSTALLIZE
SNIPCART
NEXT.js
shopify
Slatwall
Commerce
Commerce.js
BIGCOMMERCE
shōgun
commercetools
PAYMENT
Recurly
stripe
Trolley
Chargebee
PayPal
Braintree
Reach
WEBSITE BUILDERS
STACKBIT
Reflex
uniform
BACKEND
GraphQL
HASURA
heroku
Kong
APOLLO
Prisma

https://vuejs.org/

Apa itu Vue.js?

cloud connect

- *an open-source front end JavaScript framework for building user interfaces and single-page applications.*
- *It was created by Evan You after working for Google using AngularJS in a number of projects.*
- *Vue was first released the following February, in 2014.*
- *Vue is generally used to create single-page apps that **run on the client** but can be used to create full-stack app by making HTTP request to backend server.*
- *Vue.js is a framework for **building client-side applications***
- *Vue can run on server side by using SSR (Server Side Rendering) Framework like Nuxt.js*

cloud connect

Vue.js **Versi 3**

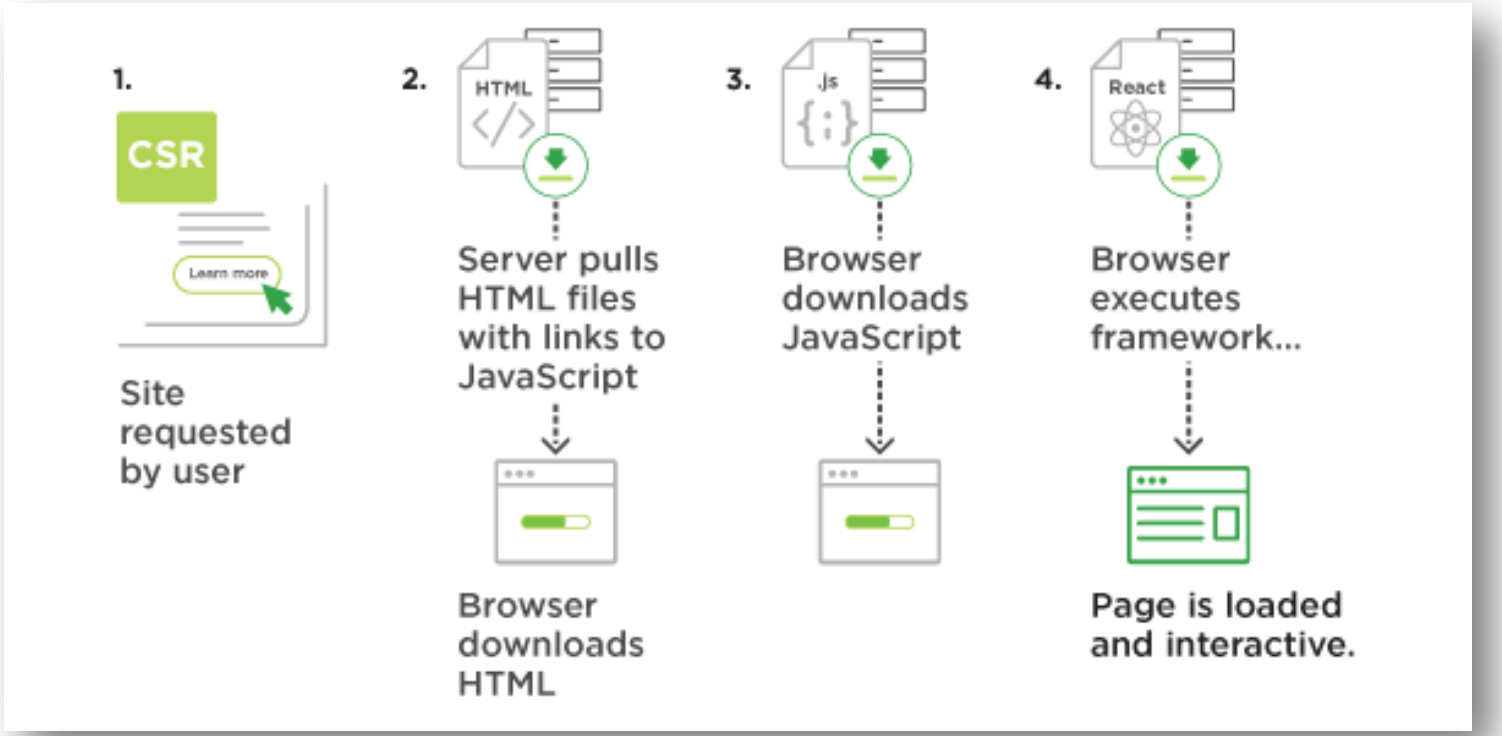
The Progressive JavaScript Framework

An approachable, performant and versatile framework for building web user interfaces.

Apa itu *Client Side Rendering*?

cloud connect

Rendering content in the browser using JavaScript. Instead of getting all of the content from the HTML document itself, you are getting a bare-bones HTML document with a JavaScript file that will render the rest of the site using the browser.



Vue - Single-File Components (SFC)

cloud connect

- *also known as *.vue files,*
- *A Vue SFC, as the name suggests, encapsulates the:*
  - *component's logic (JavaScript),*
  - *template (HTML), and*
  - *styles (CSS) in a single file*

Vue Components

cloud connect

- *Vue components can be authored in two different API styles:*
  - *Options API*
    - *define a component's logic using an object of options such as **data**, **methods**, and **mounted**. Properties defined by options are exposed on {**this**} inside functions*
    - *centered around the concept of a **"component instance"***
    - *aligns better with a class-based mental model for users coming from OOP language backgrounds.*
    - *It is also more **beginner-friendly** by abstracting away the reactivity details and enforcing code organization via option groups.*
  - *Composition API.*
    - *define a component's logic using imported API functions. In SFCs, Composition API is typically used with **<script setup>**.*
    - *centered around declaring reactive state variables directly in a function scope, and composing state from multiple functions together to handle complexity.*

Basic layout of an Options API

cloud connect

```
<template>

</template>

<script>
   export default {

   }
</script>

<style>

</style>
```

Vue page includes:

- a <template> for markup
- a <script> for javascript scripting
- a <style> for CSS styling

Apa itu *Vue Router*?

cloud connect

- *Vue Router is the official router for Vue.js. It deeply integrates with Vue.js core to make building Single Page Applications with Vue.js a breeze. Features include:*
  - Dynamic Route Matching
  - Routes' Matching Syntax
  - Named Routes
  - Nested Routes
  - Programmatic Navigation
  - Named Views
  - Redirect and Alias
  - Passing Props to Route Components
  - Active links
  - Different History modes

https://router.vuejs.org/

cloud connect

Latihan 9 – Vue.js

cloud connect

Modern UI Framework

What is Design Systems?

cloud connect

- A design system is a set of standards to manage design at scale by reducing redundancy while creating a shared language and visual consistency across different pages and channels.
- A design system is a series of components that can be reused in different combinations. Design systems allow you to manage design at scale.
- A design system is a collection of reusable components, guided by clear standards, that can be assembled together to build any number of applications. It's not just a collection of the components and assets used to build a digital product; it's a nearly infinite set of pieces that can be repurposed as strategies, markets, and trends evolve.

What is a Design System?

BUILDING BLOCKS

- Typographic scales
- Color schemes
- Grid definitons
- Icons
- Image assets

UI PATTERNS

- Templates
- Modules
- Components
- Elements
- ...

RULES

- Design principles
- Implementation guidelines
- Do's and Dont's
- Editorial guidelines

Core components of a Design System

COLORS

ICONOGRAPHY

ELEVATION

COPY

TYPOGRAPHY

ANIMATION

LAYOUT

ILLUSTRATION

Component-Driven UI

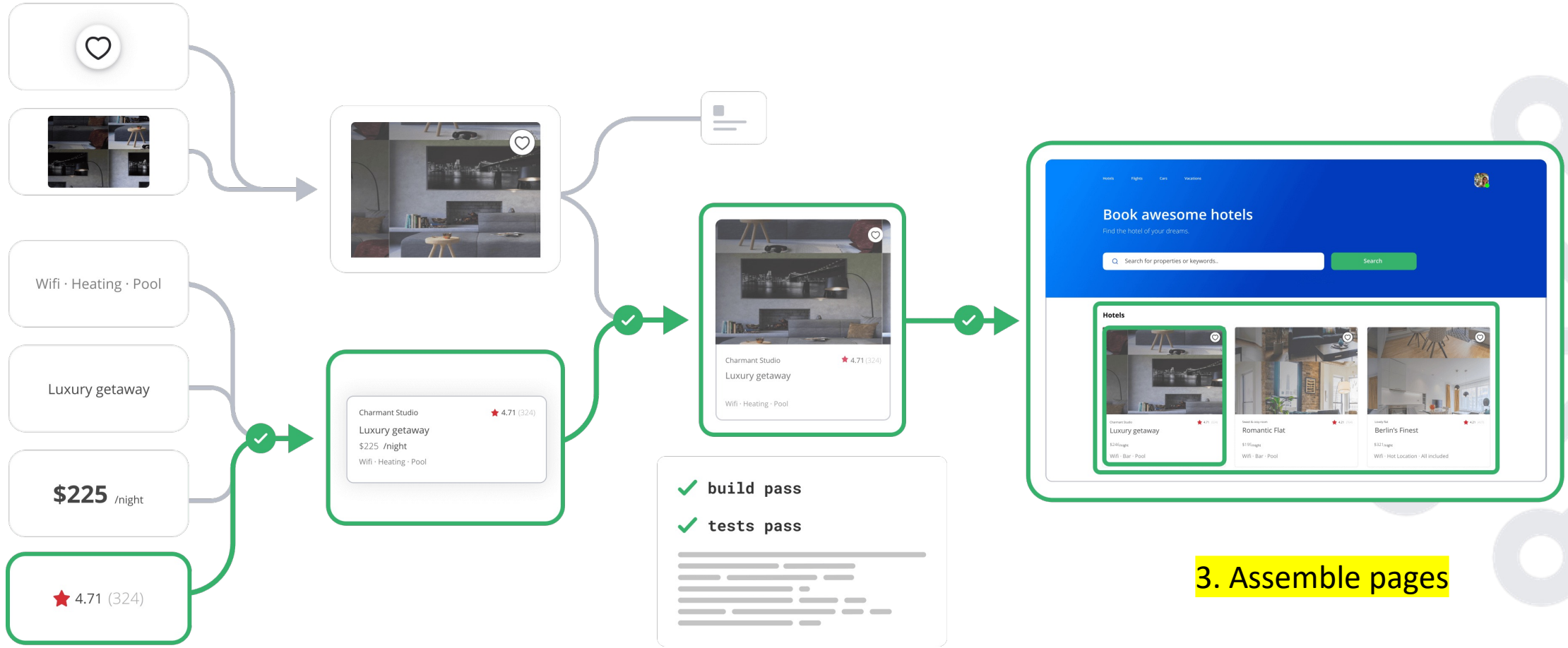
www.componentdriven.org

cloud connect

- The development and design practice of building user interfaces with modular components.
- UIs are built from the "bottom up" starting with basic components then progressively combined to assemble screens.
- Benefits:
  - **Quality**: Verify that UIs work in different scenarios by building components in isolation and defining their relevant states.
  - **Durability**: Pinpoint bugs down to the detail by testing at the component level. It's less work and more precise than testing screens.
  - **Speed**: Assemble UIs faster by reusing existing components from a component library or design system.
  - **Efficiency**: Parallelize development and design by decomposing UI into discrete components then sharing the load between different team members.

Components-Driven Development

cloud connect



1. Build one component at a time

2. Combine components

3. Assemble pages

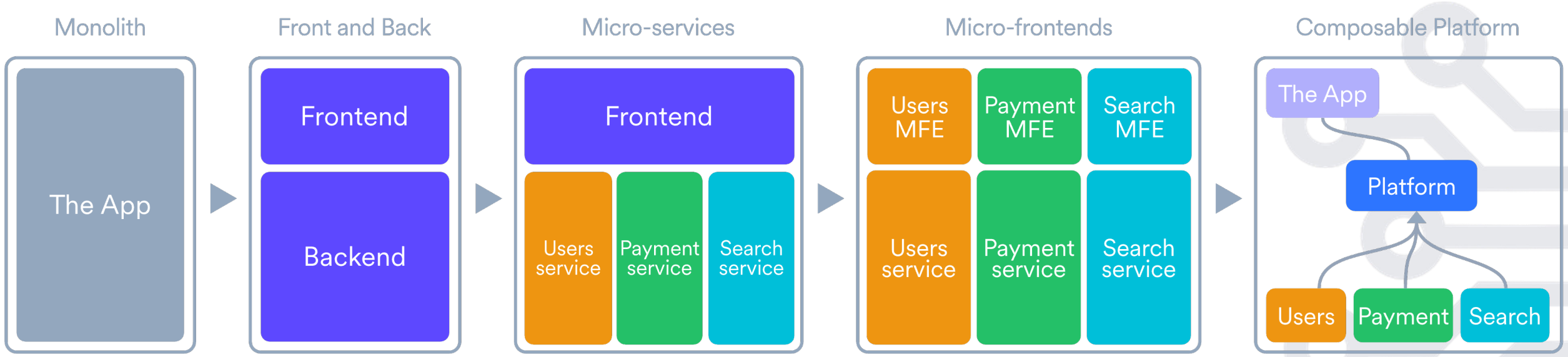
Atomic Design

cloud connect

- Atomic Design is atoms, molecules, organisms, templates, and pages concurrently working together to create effective interface design systems.

Composable Architecture

cloud connect



Modern UI Framework

cloud connect

Users choice	• **Vuetify** - mobile and desktop ready, material design • **Quasar** - material design as well but with cross-platform capabilities
Any platform	• **Quasar** - full framework, option of Cordova or Capacitor for mobile builds • **Ionic** - cross-platform UI toolkit, uses Capacitor for native mobile
Desktop focused	• **Element UI** - extensive, desktop oriented, widely popular in asia • **Tailwind**
Easy if familiar	• **Bootstrap** - Bootstrap based • **Buefy** - Bulma based
Most flexible	• **Tailwind**

Apa itu Tailwind?

cloud connect

- Tailwind CSS is a utility-first CSS framework designed to help developers build custom user interfaces quickly and efficiently.
- Instead of writing custom CSS, you use utility classes directly in your HTML to control layout, color, spacing, typography, shadows, and more

https://tailwindcss.com/

Tailwind Features

cloud connect

- **Utility-First Approach**: Provides low-level utility classes that can be combined to create custom designs without writing CSS.
- **Highly Customizable**: Allows extensive customization to fit your design needs.
- **Responsive Design**: Facilitates the creation of responsive websites with ease.
- **Pre-Designed Classes**: Offers a comprehensive set of pre-designed classes for common tasks.

Kelebihan Tailwind

cloud connect

- **Reduced Custom CSS:** Minimizes the need to write custom CSS, making development faster.
- **Smaller CSS Files:** Keeps CSS files small and manageable by reusing utility classes.
- **Consistent Design:** Ensures a consistent design system by using predefined classes.
- **Efficiency:** Speeds up the development process by allowing you to build directly in HTML.

TailwindCSS Libraries (open source)

cloud connect

https://tailwindcss.com/

Flowbite

https://flowbite.com/

preline

https://preline.co/

daisyUI

https://daisyui.com/

The Intuitive Vue Framework

Nuxt is an open source framework that makes web development intuitive and powerful.
Create performant and production-grade full-stack web apps and websites with confidence.

Nuxt 3

cloud connect

- Nuxt 3 release on November 16, 2022 – Nuxt 2 release on October 2016
- Nuxt 3 is a modern rewrite of the Nuxt framework based on **Vite**, **Vue3**, and **Nitro** with first-class TypeScript support
- Nuxt 3 is the new version of the Nuxt.js framework for building universal Vue.js applications.
- Nuxt is a powerful tool that makes it easy to create server-rendered Vue.js applications, providing out-of-the-box features such as **server-side rendering**, **automatic code splitting**, and **hot module replacement**.
- Nuxt 3 promises to bring several new features and improvements, including **faster startup times**, **better developer experience**, **improved performance**, and **enhanced scalability**.

Nuxt 3 – nuxt.com

cloud connect

Routing & Layouts
File based routing system to build complex views and interfaces with a powerful and conventional approach.

Data Fetching
Composables that run on the server to fetch data for your components and enable you to render content in different ways.

Assets & Style
Image, Font and Script optimizations with a built-in support for CSS Modules, Sass, PostCSS, CSS-in-JS and more.

SEO & Meta Tags
Production ready and indexable by search engines while giving the feeling of an app to the end-users.

Middleware
Run custom code such as authentication, localization or A/B testing before rendering a page or a group of pages.

State Management
Nuxt provides a simple way to share a reactive and SSR-friendly state between components.

Transitions
Create smooth transitions between layouts, pages and components with a built-in support for Vue & browser transitions.

Error Handling
Built-in error handling and logging to help you debug your application and provide a better user experience.

Layers
Extend your Nuxt application with another Nuxt application to reuse components, composables, layouts, pages and more.

Server Routes
Create API endpoints and server routes to securely connect with third party services and consume from your frontend.

Auto Imports
Nuxt auto-imports helpers, composables, and Vue APIs to use across your app without explicitly importing them.

TypeScript
Nuxt provides helpful shortcuts to ensure you have access to accurate type information when you are coding.

Perbezaan Vue3 and Nuxt3

cloud connect

Feature/Aspect	Vue 3	Nuxt 3
Purpose	JavaScript framework for building UIs and SPAs	Framework built on Vue 3 for SSR, SSG, and hybrid apps
Rendering Modes	Primarily Client-Side Rendering (CSR)	Supports SSR, SSG, and CSR
File Structure	Flexible, no strict conventions	Enforced, conventional structure with automatic routing
Features & Ecosystem	Core libraries (Vue Router, Vuex); highly modular	Extended features (SSR, SSG, built-in routing); integrated modules
Development Experience	Lightweight, flexible, more manual configuration	Streamlined with many out-of-the-box features, less manual setup
Community & Ecosystem	Large, active community with a vast ecosystem	Built on Vue 3's ecosystem; growing community focused on SSR/SSG
Project Setup Complexity	More manual setup required	Automatic setup with conventions for routing, state management, etc.
Use Cases	Suitable for SPAs and projects needing flexibility	Ideal for projects requiring SSR, SSG, or a quick setup with best practices

Nuxt 3 Architecture

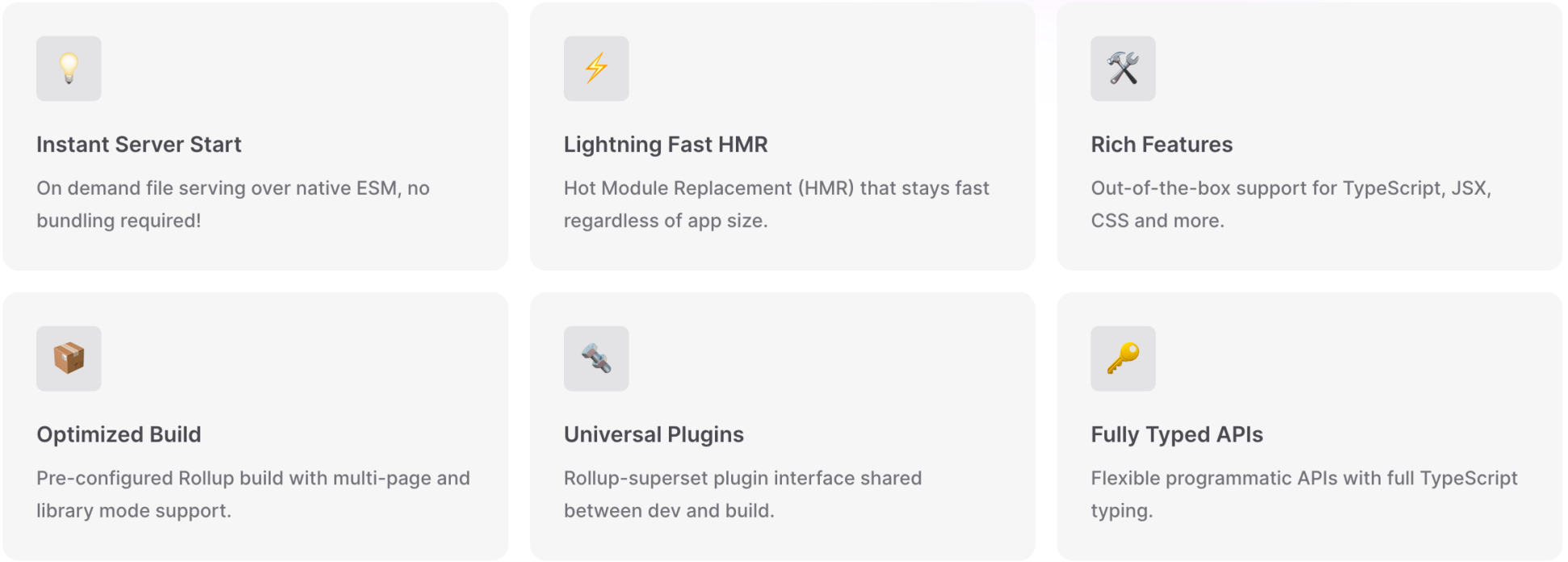
cloud connect

- Core Engine: **nuxt**
- Bundlers: **@nuxt/vite-builder** and **@nuxt/webpack-builder**
- Command line interface: **nuxi**
- Server engine: **nitro**
- Development kit: **@nuxt/kit**
- Nuxt 2 Bridge: **@nuxt/bridge**

Nuxt Bundler - Vite

cloud connect

- Vite (pronounced '**veet**') is a no-bundler alternative to webpack made by Vue.js author, Evan You.
- https://vitejs.dev/



Nuxt CLI - Nuxi

cloud connect

- Nuxi - New Nuxt Command Line Interface (CLI)
- npx nuxi [command]
- Command
  - init – to create nuxt 3project
  - add
    - page – to add new page
    - component – to add new component
    - layouts – to add new layout
    - etc - https://nuxt.com/docs/api/commands/add

Server Engine - Nitro

cloud connect

- https://nitro.unjs.io/
- Cross-platform support for Node.js, Browsers, service-workers and more.
- Serverless support out-of-the-box.
- API routes support.
- Automatic code-splitting and async-loaded chunks.
- Hybrid mode for static + serverless sites.
- Development server with hot module reloading.

{ Latihan 10.1 – Pengenalan Nuxt 3 }

cloud connect

Nuxt UI Framework

Flowbite - TailwindCSS UI Framework

Flowbite TailwindCSS UI Framework

cloud connect

Flowbite https://flowbite.com/

- an open-source library of UI components built on top of the utility-first Tailwind CSS framework
- Utility-First Approach: Flowbite leverages Tailwind CSS's utility classes, making it easy to build and customize components.
- Interactive Elements: It includes JavaScript to enable interactive components like modals, dropdowns, and tooltips.
- Figma Design System: Flowbite provides Figma design files to help prototype and design your website before coding.
- Dark Mode Support: It supports dark mode, allowing for modern and accessible design.
- Easy Integration: You can easily integrate Flowbite into any existing Tailwind CSS project via NPM.

cloud connect

{ Latihan 2 – Nuxt UI Framework }

Nuxt 3 Rendering

cloud connect

- Client-side only rendering (CSR)
- Server-side rendering (SSR)
- Universal rendering (Server-side rendering and hydration)
- Hybrid rendering (per-routes caching strategy)

https://nuxt.com/docs/guide/concepts/rendering

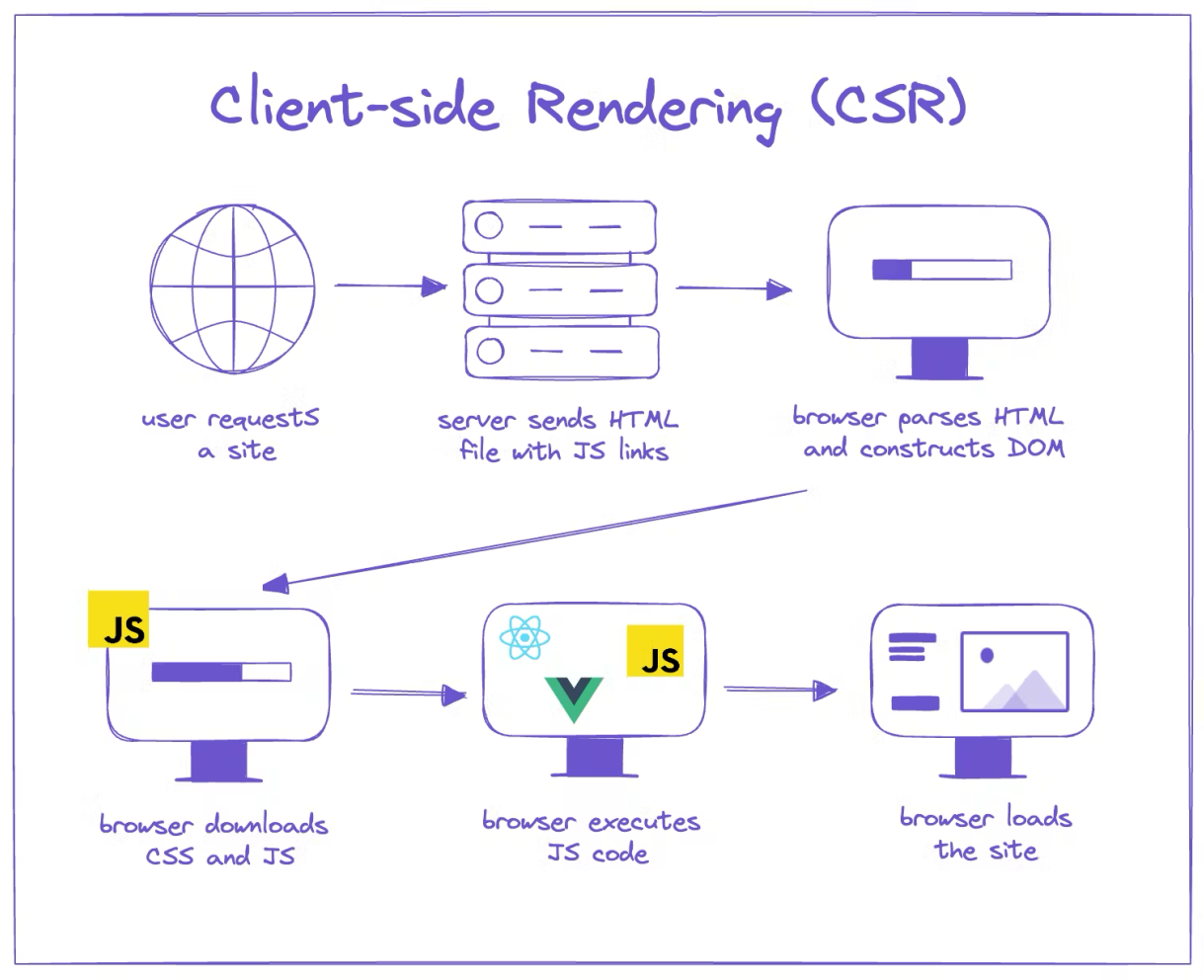
Client-Side Only Rendering

cloud connect

- Out of the box, a traditional Vue.js application is rendered in the **browser** (or **client**).
- Then, Vue.js generates HTML elements after the browser downloads and parses all the JavaScript code containing the instructions to create the current interface.

Client-Side Only Rendering

cloud connect



Client-Side Only Rendering – Pros & Cons

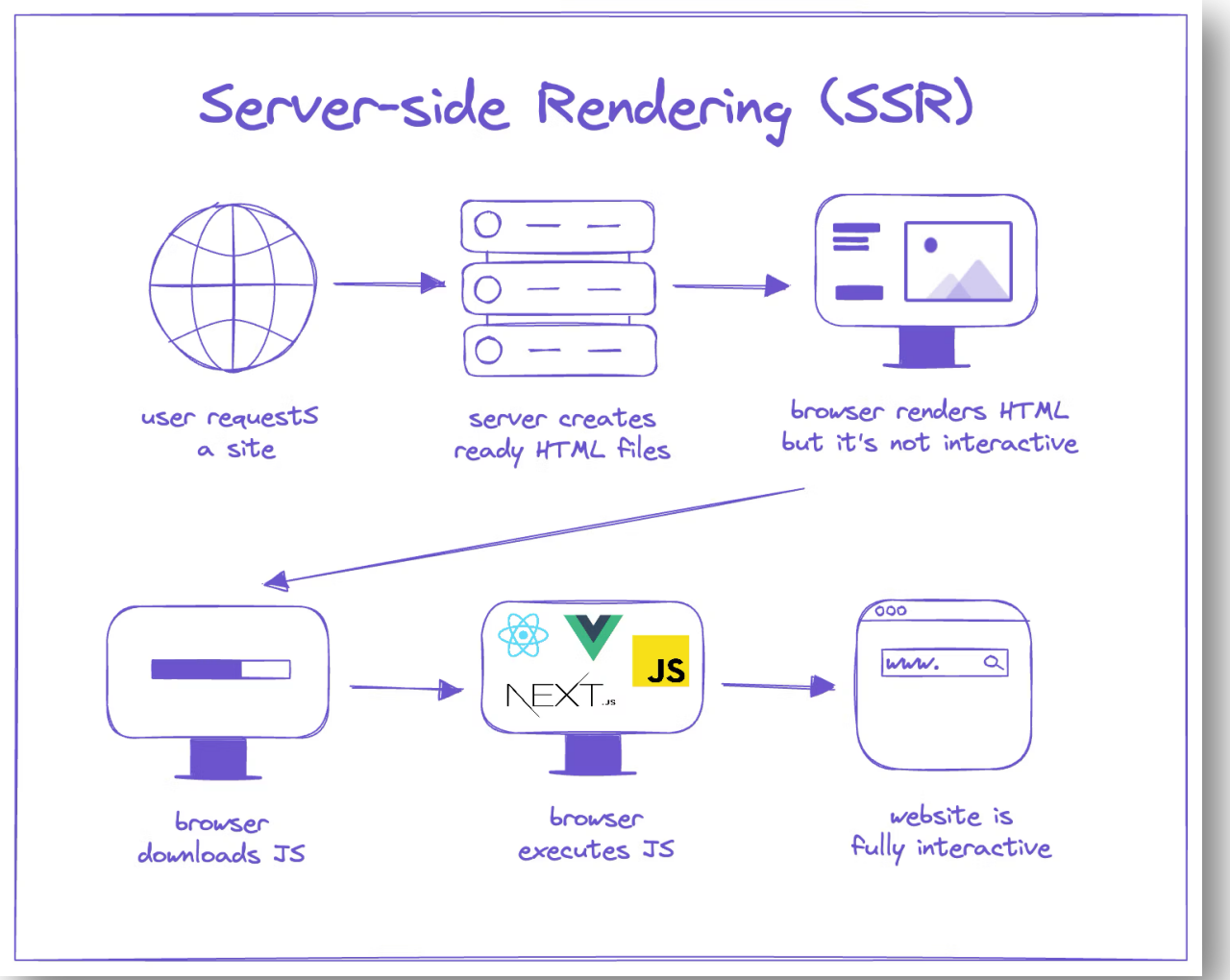
cloud connect

- Pros
  - Development speed: When working entirely on the client-side, we don't have to worry about the server compatibility of the code, for example, by using browser-only APIs like the window object.
  - Cheaper: Running a server adds a cost of infrastructure as you would need to run on a platform that supports JavaScript. We can host Client-only applications on any static server with HTML, CSS, and JavaScript files.
  - Offline: Because code entirely runs in the browser, it can nicely keep working while the internet is unavailable.
- Cons
  - Performance: The user has to wait for the browser to download, parse and run JavaScript files. Depending on the network for the download part and the user's device for the parsing and execution, this can take some time and impact the user's experience.
  - Search Engine Optimization: Indexing and updating the content delivered via client-side rendering takes more time than with a server-rendered HTML document. This is related to the performance drawback we discussed, as search engine crawlers won't wait for the interface to be fully rendered on their first try to index the page. Your content will take more time to show and update in search results pages with pure client-side rendering.

Server-side Rendering

cloud connect

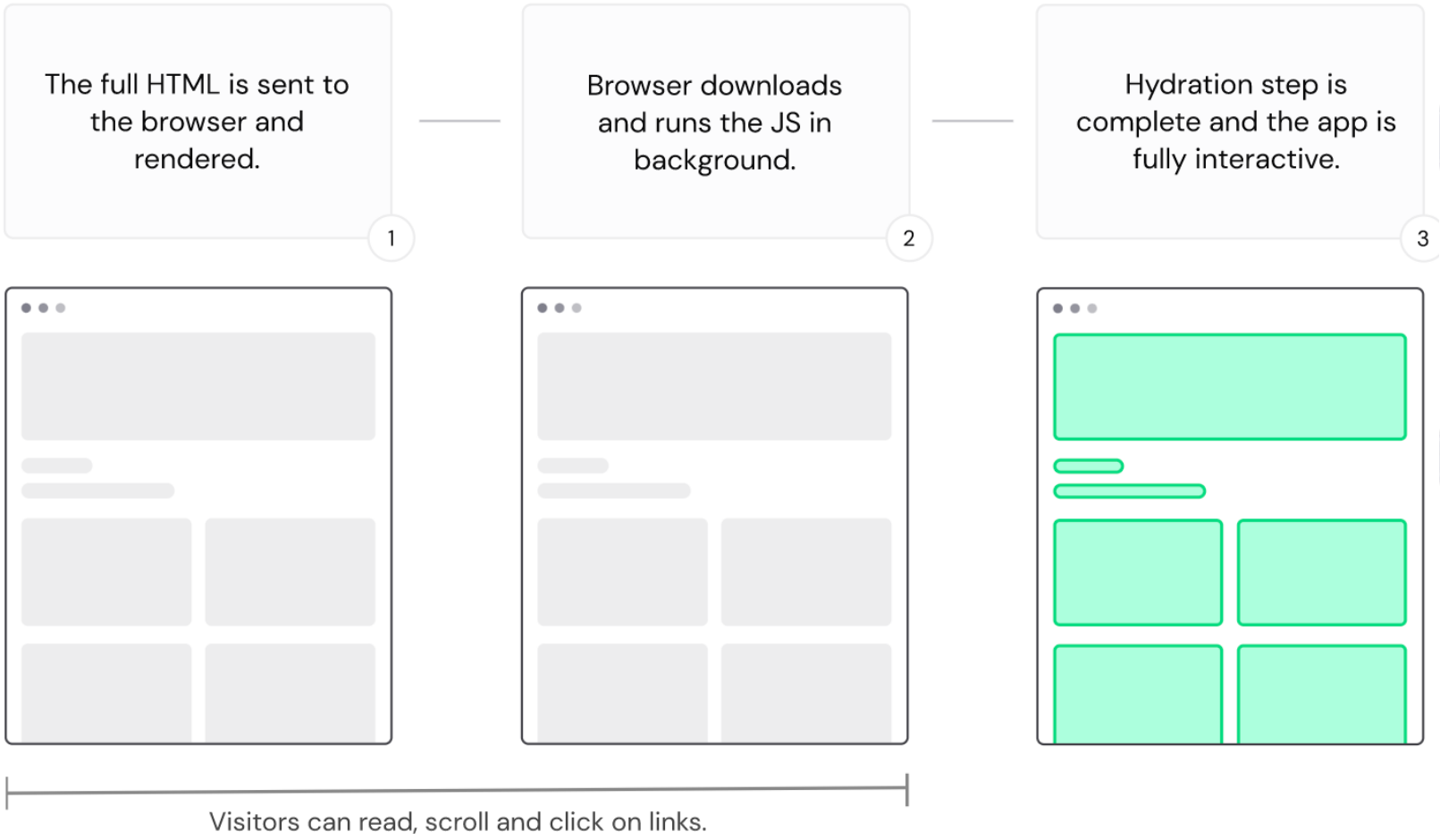
- Nuxt comes with built-in server-side rendering (SSR) capabilities by default, without having to configure a server yourself.



Universal Rendering

cloud connect

- Nuxt's universal rendering refers to the ability of Nuxt.js to render the application on both the server and the client.



Universal Rendering

cloud connect

- Pros
  - Performance: Users can get immediate access to the page's content because browsers can display static content much faster than JavaScript-generated one. At the same time, Nuxt preserves the interactivity of a web application when the hydration process happens.
  - Search Engine Optimization: Universal rendering delivers the entire HTML content of the page to the browser as a classic server application. Web crawlers can directly index the page's content, which makes Universal rendering a great choice for any content that you want to index quickly.
- Cons
  - Development constraints: Server and browser environments don't provide the same APIs, and it can be tricky to write code that can run on both sides seamlessly. Fortunately, Nuxt provides guidelines and specific variables to help you determine where a piece of code is executed.
  - Cost: A server needs to be running in order to render pages on the fly. This adds a monthly cost like any traditional server. However, the server calls are highly reduced thanks to universal rendering with the browser taking over on client-side navigation.

Nuxt Universal Rendering

cloud connect

Client Side Rendering

- Initial page load is slow
- No round trip to the server

Server Side Rendering

- ***Very fast startup***
- *Interactions are slow*

Universal Rendering

- Very fast startup
- No round trip to the server

Hybrid Rendering

cloud connect

- Hybrid rendering allows different caching rules per route using **Route Rules** and decides how the server should respond to a new request on a given URL.
- Previously every route/page of a Nuxt application and server must use the same rendering mode, universal or client-side. **But** some pages could be generated at build time, while others should be client-side rendered.

Hybrid Rendering Rules

cloud connect

- Can create different rendering rules

```
export default defineNuxtConfig({
  routeRules: {
    // Homepage pre-rendered at build time
    '/': { prerender: true },
    // Products page generated on demand, revalidates in background, cached until API response changes
    '/products': { swr: true },
    // Product page generated on demand, revalidates in background, cached for 1 hour (3600 seconds)
    '/products/**': { swr: 3600 },
    // Blog posts page generated on demand, revalidates in background, cached on CDN for 1 hour (3600 seconds)
    '/blog': { isr: 3600 },
    // Blog post page generated on demand once until next deployment, cached on CDN
    '/blog/**': { isr: true },
    // Admin dashboard renders only on client-side
    '/admin/**': { ssr: false },
    // Add cors headers on API routes
    '/api/**': { cors: true },
    // Redirects legacy urls
    '/old-page': { redirect: '/new-page' }
  }
})
```

https://nuxt.com/docs/guide/concepts/rendering#route-rules

Nuxt Pages

Nuxt Pages

cloud connect

- Nuxt file-system routing creates a route for every file in the **pages/** directory, based on their filename.
- Every Vue file inside the pages/ directory creates a corresponding URL automatically (or route) that displays the contents of the file.
  Whereas in Vue, manually.
- Directory Structure:

```
| pages/
---| about.vue
---| index.vue
---| posts/
-----| [id].vue
```

Nuxt Pages

cloud connect

app.vue file

- Nuxt 3 provides a central entry point to your app via ~/app.vue.
- If you don't have an app.vue file in your source directory, Nuxt will use its own default version.

/pages directory

- All *.vue files will be created in this directory
- If you have a pages/ directory, to display the current page, use the <NuxtPage> component:

```
<template>
 <div>
  <NuxtLayout>
   <NuxtPage/>
  </NuxtLayout>
 </div>
</template>
```

Nuxt Pages

cloud connect

- Nuxt pages can be created using 2 approaches:
  1. Manual creation of .vue file, or
  2. Using Nuxi CLI command

  nuxi add page mypage

  * **mypage** is the vue file

{ Latihan 10.3 – *Nuxt Pages* }

Nuxt Layouts

Nuxt Layouts

cloud connect

- Nuxt provides a layouts framework to extract common UI patterns into reusable layouts.
- Layouts are a great help when you want to change the look and feel of your Nuxt app.
- Whether you want to include a sidebar, navigation bar, header, footer or have distinct layouts for mobile and desktop.

Nuxt Layouts

cloud connect

- Layouts are enabled by adding **<NuxtLayout>** to your **app.vue**:

app.vue

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

{ Latihan 10.4 – *Nuxt Layouts* }

Nuxt Components

Component-Driven UI

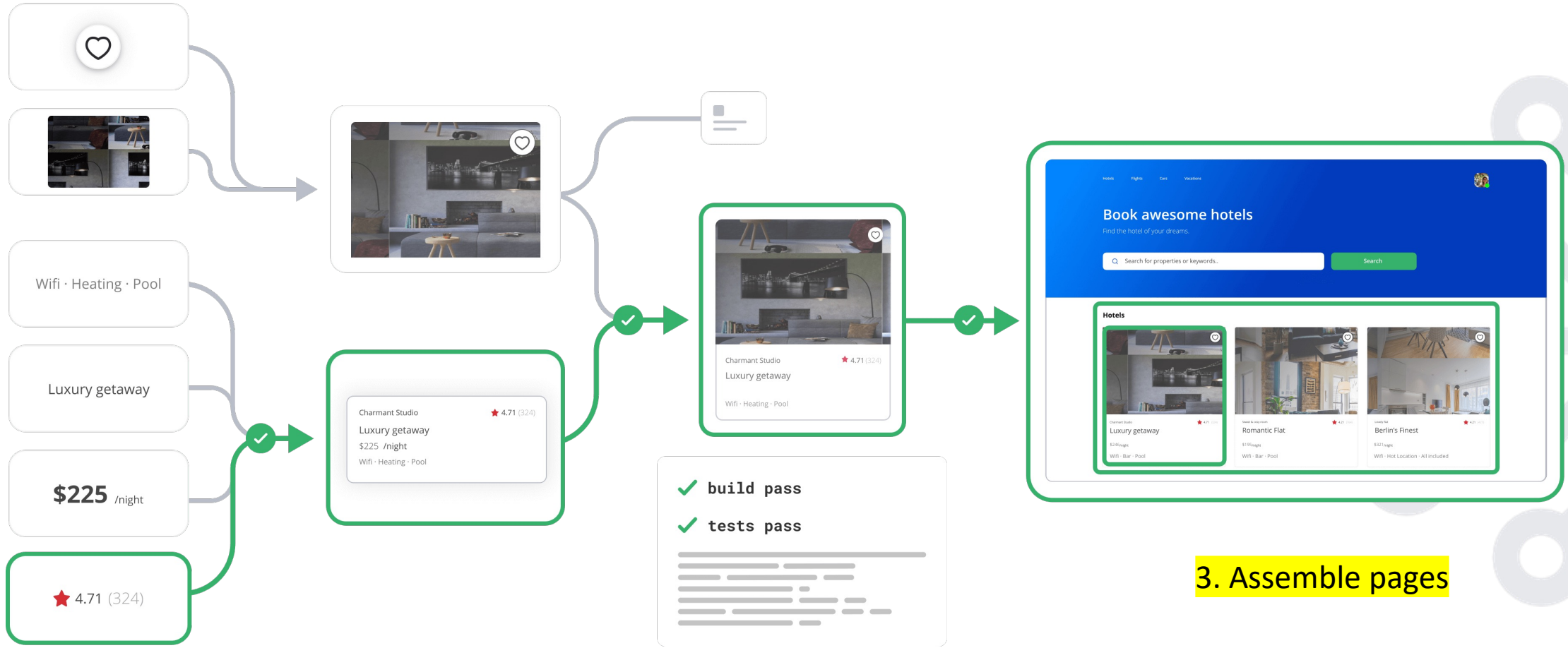
www.componentdriven.org

cloud connect

- The development and design practice of building user interfaces with modular components.
- UIs are built from the "bottom up" starting with basic components then progressively combined to assemble screens.
- Benefits:
  - **Quality**: Verify that UIs work in different scenarios by building components in isolation and defining their relevant states.
  - **Durability**: Pinpoint bugs down to the detail by testing at the component level. It's less work and more precise than testing screens.
  - **Speed**: Assemble UIs faster by reusing existing components from a component library or design system.
  - **Efficiency**: Parallelize development and design by decomposing UI into discrete components then sharing the load between different team members.

Components-Driven Development

cloud connect



1. Build one component at a time

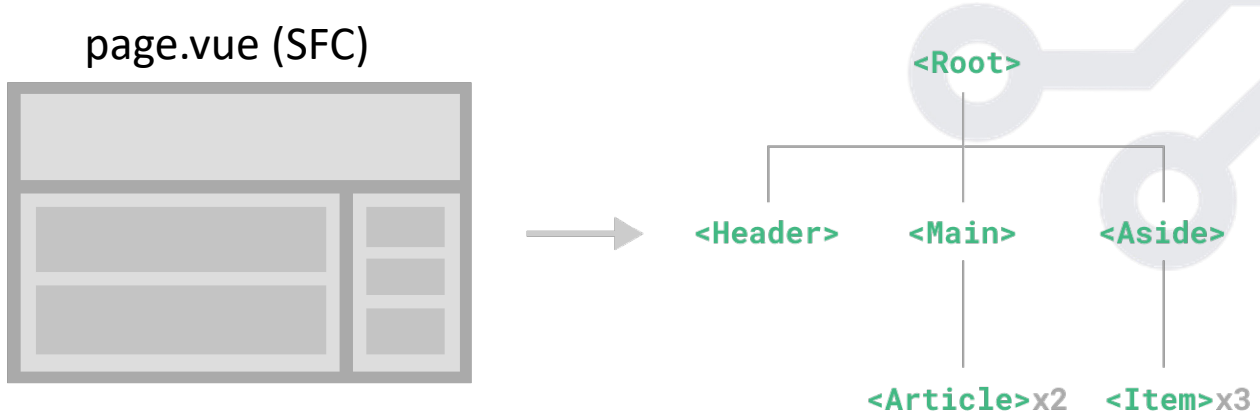
2. Combine components

3. Assemble pages

Vue Components

cloud connect

- Components allow us to split the UI into independent and reusable pieces.
- define each Vue component in a dedicated file using the **.vue extension** - known as a **Single-File Component** (SFC)



Nuxt Components

cloud connect

- Similar like Vue.js, components are what makes up the different parts of your page and can be reused and imported into your pages, layouts and even other components.
- The components directory (**components/**) contains your Vue.js components.

Nuxt Components

cloud connect

Directory Structure

```
| components/
--| AppHeader.vue
--| AppFooter.vue
```

app.vue

```
<template>
  <div>
    <AppHeader />
    <NuxtPage />
    <AppFooter />
  </div>
</template>
```

Nuxt Components

cloud connect

- Components Name Guideline

Directory Structure

```
| components/
--| base/
----| foo/
------| Button.vue
```

```
<BaseFooButton />
```

{ Latihan 10.5 – *Nuxt Components* }

cloud connect

Pembangunan Nuxt Front-end

Mengabungkan Pages, Components dan Layouts untuk membentuk aplikasi Nuxt

{ Latihan 10.6 – *Pembangunan Nuxt Front-end* }

cloud connect

State Management using Pinia

Apa itu *State Management*?

cloud connect

- Vue state management refers to the **process of managing the state or data** of a Vue.js application.
- In Vue.js, **the state is the data that represents the current state of the application**, including **user input, server responses, and other variables** that may change over time.
- Vue state management is an **important aspect of building scalable and maintainable** Vue.js applications.
- Vue state management:
  - Vuex
  - Pinia

Apa itu *Pinia*?

cloud connect

- **Pinia** is a **state management** library for Vue.js applications. A simple and reactive way to manage the state of your application, making it easier to build complex and scalable applications.
- Pinia uses the **composition API** introduced in Vue.js 3 to create **stores**, which are objects that hold the application's state and provide methods to update it. With Pinia, you can create multiple stores to manage different parts of your application's state, and easily share state between components.
- One of the advantages of using Pinia is that it **offers type safety and auto-completion for your state, mutations, and actions**, which can save you time and reduce the chances of introducing errors.
- Overall, Pinia provides a simple and effective way to manage the state of your Vue.js applications, helping you build scalable and maintainable code.

- Pinia also uses the **state**, **getters**, and **actions** concepts, which are equivalent to data, computed, and methods in components:
  - The **state** is defined as a function returning the initial state
  - The **getters** are functions that receive the state as a first argument
  - The **actions** are functions that can be asynchronous

- You can think of **state** as the **data** of the store, **getters** as the **computed properties** of the store, and **actions** as the **methods**.

Pinia

cloud connect

stores/config.js

```
export const useConfigStore = defineStore(
'ConfigStore',{
    state: () => {
        return {
          strapiURL: "http://abc.com:1337",
          identifier: "nama",
          password: "katalaluan",
          jwttoken: null,
        };
    },
    getters: {},
    actions: {},
});
```

Declare Pinia State Management

pages/myvuefile.vue

```
<script setup>
import { useConfigStore } from '~~/stores/config';

const configStore = useConfigStore();

// read
const strapiURL = configStore.strapiURL
const identifier = configStore.identifier
const katalaluan = configStore.password

// update
let configStore.jwttoken = "mytoken"



</script>
```

Using Pinia State Management

cloud connect

{ Latihan 10.7 - *State Management with Pinia* }