SPA
(Single Page Application)

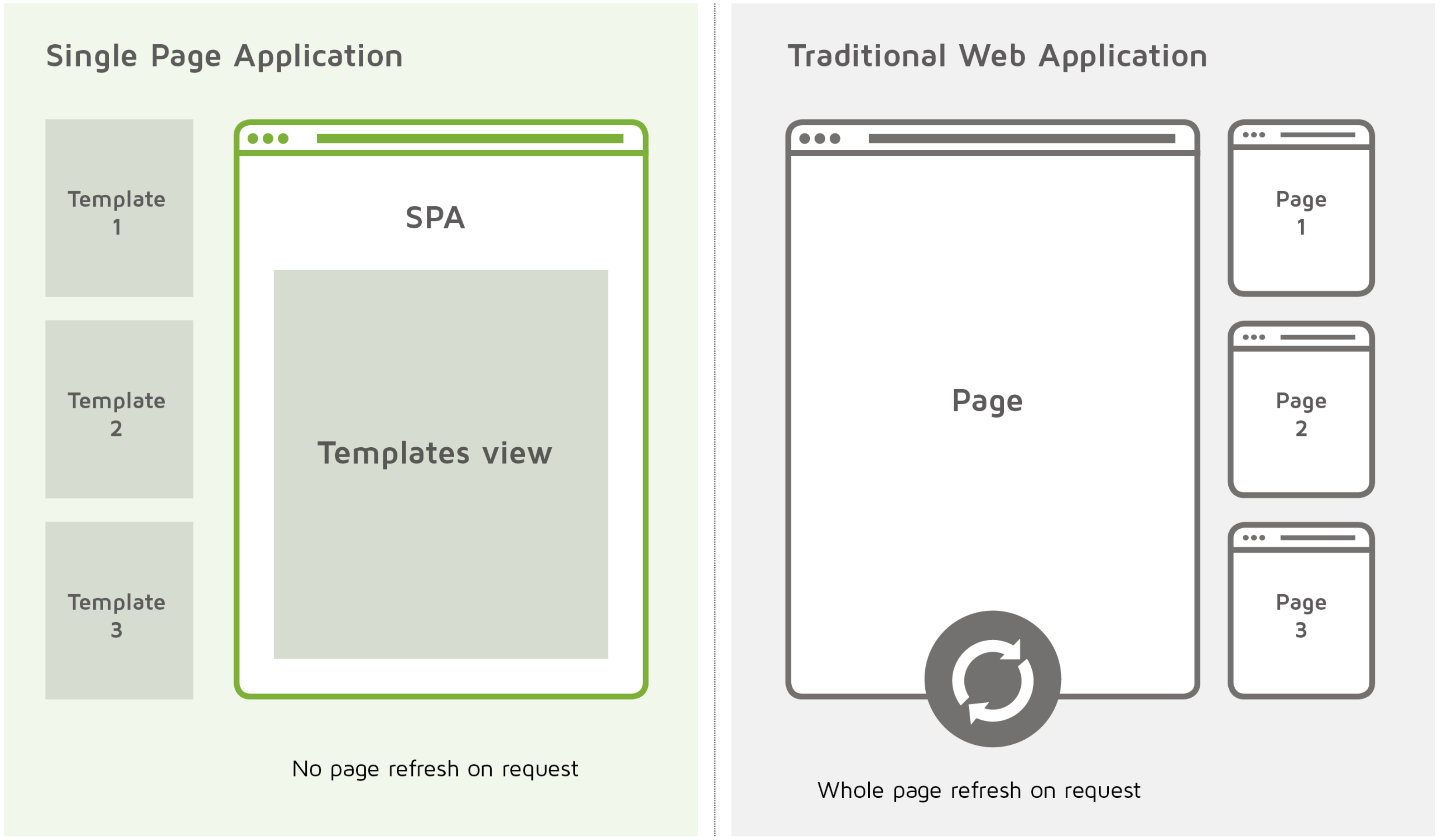
IF3110 – Web-based Application Development
School of Electical Engineering and Informatics
Institut Teknologi Bandung

Single Page Application

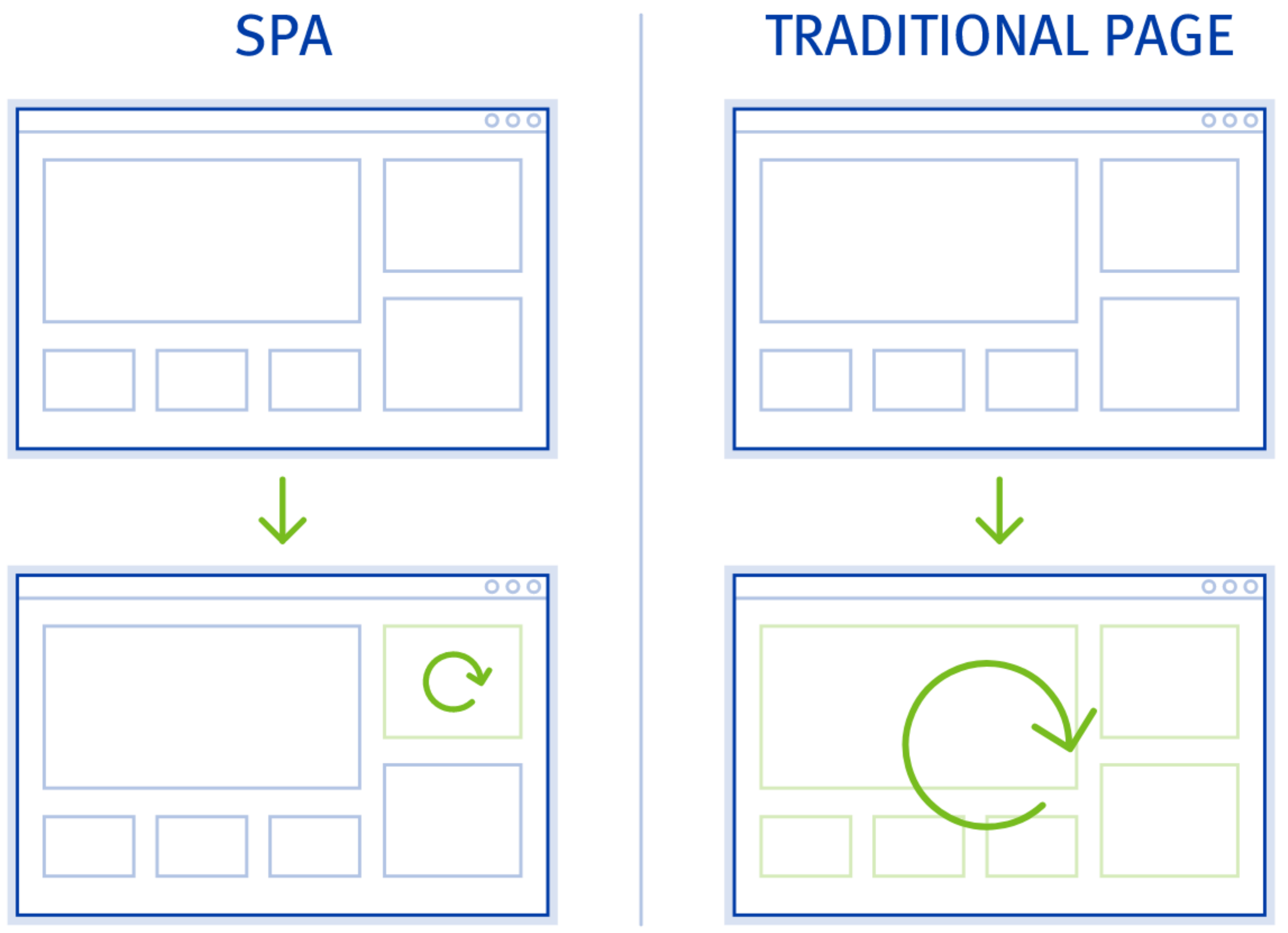
A **single-page application** (**SPA**) is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages.

The goal is faster transitions that make the website feel more like a native app.

SPA vs. Traditional Page



SPA vs. Traditional Page



SPA Website Examples

- Google Workspace (GSuite)
  Gmail, Maps, Drive, Calendar, Meet, Classroom, etc
- Facebook
- Whatsapp web
- Instagram
- Twitter
- GitHub
- Netflix

Advantages of SPA

- Quick, majority of resources (HTML+CSS+Scripts) only loaded once
- Development is usually quicker and more efficient
- Optimised for debugging, developers can monitor network operations, investigate page elements and data associated
- Easier to create a mobile application based on it, developers can reuse the same backend code
- Efficient in caching any local storage, even offline

Disadvantages of SPA

- Need special tricks for SEO
- Slow to download, heavy client frameworks are required to be loaded to the client
- Require JavaScript in its active mode in users' browsers
- Less secure due to Cross-Site Scripting (XSS), it enables attackers to inject client-side scripts
- Memory leaks in JavaScript can lead to productivity drop for even powerful systems

Technical/Pattern approaches

- **Document Hashes (URI fragments)**
  HTML/CSS authors can leverage element IDs or #target to show or hide different sections of the HTML document.
- **History API (new approach, more SEO friendly)**
  The History API provides access to the browser's session history. It exposes useful methods and properties that let you navigate back and forth through the user's history and manipulate the contents of the history stack.
- **AJAX**
  AJAX using asynchronous requests to a server for XML or JSON data using XMLHttpRequest or Fetch API.
- **WebSockets**
  bidirectional real-time client-server communication technology
- **Server-sent events**
  servers can initiate data transmission to browser clients.
- **Server-Side Rendering**
  Rendering initial HTML on server for performance and SEO, then hydrate on client.
- **Micro-frontend Architecture**
  Split large apps into independently deployable frontend module
- **Code splitting and Lazy Loading**
  Load routes/components on demand