



IIT Madras BSc Degree

Copyright and terms of use

IIT Madras is the sole owner of the content available in this portal - onlinedegree.iitm.ac.in and the content is copyrighted to IIT Madras.

- Learners may download copyrighted material for their use for the purpose of the online program only.
- Except as otherwise expressly permitted under copyright law, no use other than for the purpose of the online program is permitted.
- No copying, redistribution, retransmission, publication or exploitation, commercial or otherwise of material will be permitted without the express permission of IIT Madras.
- Learner acknowledges that he/she does not acquire any ownership rights by downloading copyrighted material.
- Learners may not modify, publish, transmit, participate in the transfer or sale, create derivative works, or in any way exploit, any of the content, in whole or in part.

Frontend

Application Frontends

- Mechanisms
- Asynchronous updates
- Browser / Client options
- Client-side computation
- Security implications

Mechanisms

What is the application frontend?

- User-facing interface
 - General GUI application on desktop
 - Browser based client
 - Custom embedded interface
- Device / OS specific controls and interfaces
- Web browser standardization
 - Common conventions among multiple browsers on how to render, what to render
- Browser vs. Native
 - Look and feel
 - APIs, interfaces, interaction

Web applications

- Browser based: HTML + CSS + Javascript
 - HTML - what to show
 - CSS - how to show it
 - Javascript - bonus interaction (not core UI but essential for dynamic experience)
- Frontend mechanisms?
 - How to generate the HTML, CSS, JS?
 - Functional reuse, common frameworks
 - Server/Client load implications
 - Security implications

Fully static pages

- All (or most) pages on site are statically generated
 - Compiled ahead of time
 - Not generated at run-time
- Excellent for high performance
 - Server just picks up file and delivers
- How do you adapt to run-time conditions?
 - User login, user specific information, time-of-day
 - Javascript can help - more later
- Increasingly popular: Static site generators
 - Jekyll, Hugo, Next.js, Gatsby
 - Javascript allows very interesting variants

Run-time HTML generation

- Traditional CGI / WSGI based apps
 - Python (Flask, Django,...), Ruby (RoR)
 - PHPs core concept: server-side run-time generation of HTML
 - Wordpress, Drupal, Joomla - traditional CMS applications
- Great flexibility:
 - common layouts, adaptation and theming easy
 - run-time changes, user login, time-of-day etc easy
- Server load!
 - Every page has to be generated dynamically
 - May involve database hits
 - Cost
 - Speed
- Caching and other technologies can help, but complex

Client Load?

- Typical web-browser:
 - issue requests, wait for response
 - render HTML
 - wait for user input: most time spent waiting here
- Why not let client do more?
 - Also allows more fancy interactions
- Client-side scripting
 - Javascript de facto standard
 - Component frameworks allow reuse, complex interactions
 - Server-side Javascript! NodeJS

Tradeoffs

- Server-side rendering
 - Very flexible
 - May be easier to develop
 - Less security issues on client

- Server-side rendering
 - Load on server!
 - More security issues on server

Tradeoffs

- **Server-side rendering**
 - Very flexible
 - May be easier to develop
 - Less security issues on client
- **Static**
 - Cache-friendly
 - VERY fast

- **Server-side rendering**
 - Load on server!
 - More security issues on server
- **Static**
 - Interaction difficult / impossible?
 - Compilation phase: small changes require recompile

Tradeoffs

- **Server-side rendering**
 - Very flexible
 - May be easier to develop
 - Less security issues on client
- **Static**
 - Cache-friendly
 - VERY fast
- **Client-side**
 - Can combine well with static pages
 - Less load on server but still dynamic

- **Server-side rendering**
 - Load on server!
 - More security issues on server
- **Static**
 - Interaction difficult / impossible?
 - Compilation phase: small changes require recompile
- **Client-side**
 - More resources needed on client
 - Potential security issues, data leakage

Estimating performance

<https://serverguy.com/comparison/apache-vs-nginx/>

- Static pages:
 - Apache: ~ 10,000 req/s - 512 parallel requests
 - Nginx - ~ 20,000 req/s - 512 parallel requests
- Dynamic (call out to PHP - limited by page rendering in PHP):
 - Both ~ 100 req/s @ 16 parallel
- Dynamic occupies more resources for longer - harder to scale
- Severe impact on server