Browsers and Clients

Minimal requirements

- Render (display) HTML
- Cookie interaction: accept, return cookies from server to allow sessions
- Text-mode browsers (lynx, elinks etc) may not do anything more!

Text-mode and Accessibility

- Browse from command line - only text displayed
- No images, limited styling

Accessibility:

- Page should not rely on colours or font sizes/styles to convey meaning
- W3 accessibility guidelines

Page styling

- Cascading Style Sheets (CSS) most popular now
- Difficult in text, accessible browsers
    - But has many features to help even with those!
- Proper separation of HTML and styling gives best freedom to browser, user

Interactivity

- Some form of client-side programmability needed
- JavaScript most popular - de facto standard
- Can interact with basic HTML elements (buttons, links, forms etc.)
- Can also be used independently to create more complex forms

Performance of JS depends on browser and choice of scripting engine

JavaScript engines

- Chrome/Chromium/Brave/Edge: V8
- Firefox: SpiderMonkey
- Safari, older versions of IE use their own

Impact

- Performance: V8 generally best at present
- JS standardization means differences in engines less important

Client load

- JS engines also use client CPU power
  - Complex page layouts require computation
- Can also use GPU: extensive graphics support
  - Images
  - Video
- Potential to load CPU
  - Wasteful - block useful computations
  - Energy drain! - https://www.websitecarbon.com/

Machine clients

- Client may not always be a human!
- Machine end-points: typically access APIs
- Embedded devices: post sensor information to data collection sites
    - Especially for monitoring, time series analysis etc.
- Typically cannot handle JS - only HTTP endpoints

Alternative scripting languages

Python inside a browser? - Brython!

https://brython.info/

Problems with alternatives

- JS already included with browsers - why alternative?
- Usual approach: **transpilation**
  - Translation - Compilation
- Some older browsers tried directly including custom languages - now mostly all convert

WASM

- WebAssembly
- Binary instruction format
- Targets a stack based virtual machine (similar to Java)
- Sandboxed with controlled access to APIs
- “Executable format for the Web”
- Handles high performance execution - can translate graphics to OpenGL etc.

Emscripten

- Compiler framework: compile C or C++ (or any other language that can target LLVM) to WebAssembly
- Potential for creating high performance code that runs inside browser
- Limited usage so far

https://emscripten.org/index.html

Native Mode

- File system
- Phone, SMS
- Camera object detection
- Web payments

Functionality can be exposed through suitable APIs: requires platform support

- Adds additional security concerns!