The Web as the Primary Control System User Interface

Rich Neswold
ICALEPCS - User Interfaces, Perspective, and Experience
October 9th, 2019
Why Web Technologies?*

- All major browser cores track standards
  - Bleeding-edge JavaScript
  - Support latest HTML/CSS specs
  - Great developer tools
- JavaScript engines are fast
- Benefit from huge collaboration in JS ecosystem
  - Companies have a vested interest in the web
- Apps are extremely portable
  - Across browsers, operating systems, and form factors (i.e. desktop and mobile devices)
- Progressive Web Apps

*We have a JavaScript module providing full bandwidth to our control system
Built-in Developer Tools

• Debuggers
  - Code debugger (breakpoints, single-step, etc.)
  - Layout debugger (manipulate DOM tree.)
  - Style debugger (tweak CSS attributes.)

• Profilers
  - Code profiler
    • Shows hot spots
    • Measures function execution time
    • Measures GPU rendering time
  - Memory profiler
    • Monitor heap and garbage collector behavior
  - Network profiler
    • Show network packet timing and contents
    • Measures load time of page resources
TypeScript

- “Transpiler” created by Microsoft
- Uses JavaScript syntax with extensions
  - Adds type annotation to function arguments, variables, and object properties
  - Adds new types to language (i.e. tuples)
- Converts TypeScript to JavaScript
  - Annotations are stripped
  - During conversion, extensive type analysis is done
- Finds many silly mistakes at compile-time
- Many 3rd party libraries include TypeScript declaration files
- Highly recommended
React (w/JSX)

- Light-weight JavaScript library to build “components”
- Components are stand-alone JavaScript modules that:
  - Render themselves as HTML elements
  - Manage state associated with their elements
- Components containing components build complex behavior
  - Applications are a tree of nested components with glue logic to manage state
- JSX allows HTML-like syntax in source
  - Gets converted into equivalent DOM calls
  - Expressions can be injected in generated elements
import React, { useState } from 'react'
import './ReactiveInput.css'

interface ReactiveInputProps {
  label: string,
  maxLength?: number
}

const ReactiveInput: React.FunctionComponent<ReactiveInputProps> =
  ({ label, maxLength = Infinity }) => {
    const [currInput, setCurrInput] = useState('');
    return (
      <div className='reactiveInput'>
        <label htmlFor='reactiveInput'>{label}</label>
        <input
          type='text'
          name='reactiveInput'
          value={currInput}
          onChange={(event) => {setCurrInput(event.target.value)}}
        />
        <p className={currInput.length > maxLength ? 'invalid' : ''}>{currInput}</p>
      </div>
    );
  };
export default ReactiveInput;

ReactDOM.render(<ReactiveInput
  label='Hello ICALEPCS 2019'
  maxLength={10} />, document.getElementById('root'));

Hello ICALEPCS 2019

Hello world

Hello world

Available at https://github.com/fermi-controls/icalepcs2019
• Not typical, familiar edit/compile/link cycle
• Build tools have a different focus
  - Compatibility between browsers
  - Minimizing final bundle size
  • Dead code elimination
  • Remove comments and unnecessary whitespace
  • Shorten identifier names
Example - Charts

Chart demo using Nivo Charts (https://nivo.rocks/)
Example - 3D Modeling

WebGL demo using ThreeJS (https://threejs.org/)
Still To Do

• Deployment details
  - Setting up an area to host applications
  - Build system that can properly add new apps
  - Directory / Index page to help find available apps

• Security
  - Access currently requires client to be on-site or using VPN
  - Need authentication credentials for settings, etc.
    • Two-factor / YubiKey?
    • Kerberos / GSSAPI?
Conclusions

• Modern browsers provide a powerful and compelling environment for hosting acceleration applications.
  - There are comprehensive development tools in browsers to handle all aspects of web development.

• Frameworks provide a professional, intuitive experience for users, and they hide browser differences from programmers.

• Tools, like TypeScript and JSX, move many run-time issues to compile-time, making it easier to produce correct code.

• All these technologies are backed by huge companies (Google, Apple, Microsoft, Facebook, …) that have a stake in the success of the web.