JAVASCRIPT

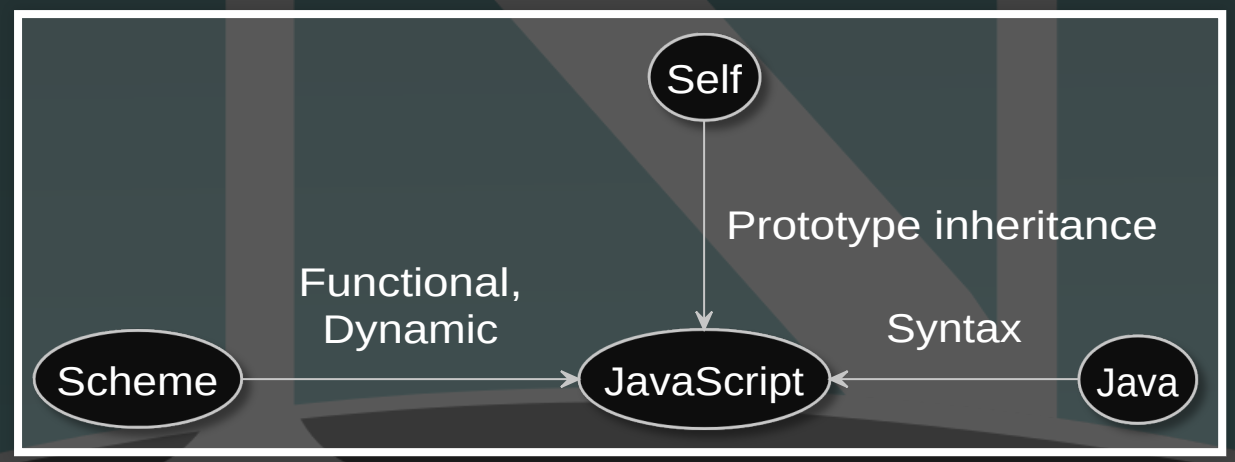
Morten Hindsholm
morten@hindsholm.dk

AGENDA

1. History of JavaScript
2. Building Stuff
3. UI Frameworks and Libraries
4. Wrap-up

HISTORY OF JAVASCRIPT

1995



Speaker notes

- Mocha was to become a scripting language for the web. Simple, dynamic, and accessible to non-developers.
- Brendan Eich was contracted by Netscape Communications to develop a "Scheme for the browser".
- Java was not suited for the type of audience that would consume Mocha: scripters, amateurs, designers. Java was just too big, too enterprisy for the role.
- Python, Tcl, Scheme itself were all possible candidates.
- Lots of important decisions had to be made and very little time was available to make them.
- "I’m not proud, but I’m happy that I chose Scheme-ish first-class functions and Self-ish (albeit singular) prototypes as the main ingredients. The Java influences, [...] the primitive vs. object distinction (e.g., string vs. String), were unfortunate."

FUNCTIONS AS FIRST-CLASS OBJECTS

```
var a = [1, 2, 3];
a.forEach(function(e) {
  console.log(e);
});
```

```
const a = [1, 2, 3];
a.forEach(e => {
  console.log(e);
});
```

`class` IS JUST SYNTACTICAL SUGAR

```
class User {
  constructor(name) {
    this.name = name;
  }
  sayHi() {
    alert(this.name);
  }
}
let user = new User("John");
user.sayHi();
```

```
function User(name) {

  this.name = name;
}
User.prototype.sayHi = function() {
  alert(this.name);
}

let user = new User("John");
user.sayHi();
```

JavaScript still uses prototype-based objects

PRIMITIVES VS OBJECTS

CONSOLE

Speaker notes

```
typeof "abc"
typeof new String("abc")
```

NOV. 1996: ECMASCRIPT

... more than 300,000 JavaScript-enabled pages on the Internet today according to www.hotbot.com.

- ECMAScript 1: June 1997
  - based on Netscape Navigator 4
- ECMAScript 2: June 1998
  - alignment with ISO

DEC. 1999: ECMASCRIPT 3

- Regular expressions
- The do-while block
- Exceptions and the try/catch blocks
- More built-in functions for strings and arrays
- Formatting for numeric output
- The in and instanceof operators

Speaker notes

This version of ECMAScript spread far and wide. It was supported by all major browsers at the time, and continued to be supported many years later.

2000-8: "ECMASCRIPT 4"

Classes
Interfaces
Namespaces
Packages
Optional type annotations
Optional static type checking
Structural types
Type definitions
Multimethods

Iterators
Generators
Instrospection
Type discriminating exception handlers
Constant bindings
Proper block scoping
Destructuring
Succint function expressions

Speaker notes

- Adobe, Mozilla, Opera, Microsoft and later Yahoo.
- Doug Crockford, an influential JavaScript developer, was the person sent by Yahoo
- Crockford and Microsoft opposed the standard
- All in all, ECMAScript 4 took almost 8 years of development and was finally scrapped. A hard lesson for all who were involved.
- ActionScript (Flash) remains the closest look to what ECMAScript 4 could have been if it had been implemented by popular JavaScript engines

2005: AJAX

- XMLHttpRequest
- First introduced by Internet Explorer 5
  - as an ActiveX control(!)
- Later incorporated in the standard

DEC. 2009: ECMASCRIPT 5

- Getters/setters
- Reserved words as property names
- New `Object` and `Array` methods
- New `Date` methods (`toISOString`, `now`, `toJSON`)
- Function `bind`
- JSON support
- Immutable global objects (`undefined`, `NaN`, `Infinity`)

Speaker notes

- Array methods improve certain functional patterns (map, reduce, filter, every, some).
- Strict mode is preventing many common sources for errors.
- JSON: a JavaScript-inspired data format that is now natively supported through JSON.stringify and JSON.parse

JUNE 2015:

ECMASCRIPT 2015 (ES6)

- Constants
- Block scope
- Arrow functions
- Template literals
- Destructuring assignment
- Modules
- Classes
- Map & Set collections
- Promises
- ...

2016 -

- June 2016: ECMAScript 2016 (ES7)
  - exponentiation operator (**)
- June 2017: ECMAScript 2017 (ES8)
  - `async` / `await`
- June 2018: ECMAScript 2017 (ES9)
  - rest/spread properties, asynchronous iteration

JAVASCRIPT ASSASSINATION ATTEMPTS

- Java - Sun Microsystems
- VBScript - Microsoft
- ActionScript/Flash - Adobe
- Silverlight - Microsoft
- Dart - Google
- TypeScript - Microsoft

JAVASCRIPT ENGINES

Provider	Engine	Usage
Opera	Carakan	Opera browser (until v. 15)
Microsoft	Chakra	Edge
Apple	JavaScriptCore	Safari
Mozilla	SpiderMonkey	Firefox
Google	V8	Chrome, Chromium, Opera, Node.js

BUILDING STUFF

NODE.JS

- An asynchronous event driven JavaScript runtime designed to build scalable network applications.
- Created 2009, 13 years after server-side JavaScript
- Combines V8, an event loop and a low-level I/O API

NODE.JS EXAMPLE

```
// Synchronous
const fs = require('fs');
const data = fs.readFileSync('/file.md');
console.log(data);
moreWork();  // will run after console.log

// Asynchronous
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
  console.log(data);
});
moreWork();  // will run before console.log
```

MODULE SYSTEMS AND IMPLEMENTATIONS

- Immediately-Invoked Function Expression (IIFE)
- CommonJS - Webpack
  - Node.js - Browserify
- Asynchronous Module Definition (AMD) - require.js, Dojo
- ES2015 - all major browsers
- All of the above - System.js

Speaker notes

- Client side implementations often package everything into one or two files
- JavaScript Module Systems Showdown: CommonJS vs AMD vs ES2015

TRANSPILERS

Compiling some other language to JS

- Dart: OO language, drives Google Adwords
- ClojureScript: functional language, dynamic typing
- Scala.js: OO and functional
- CoffeeScript: "improved" JS, Ruby-like
- TypeScript: JS superset, static typing
- ... lots of others

Speaker notes

- TS is not about types, it is about tooling

TRANSPILERS

Compiling JS to JS

- Babel
  - Using tomorrow's JavaScript today
  - Transform syntax
  - Polyfill features that are missing in your target environment
  - Used by React and others
- Traceur
  - same as Babel but inactive(?)

NPM

- Node Package Manager
- Open source packages in CommonJS format with metadata in JSON file
- World's largest collection: ~700.000 packages
- Example

LOTS OF TOOLS

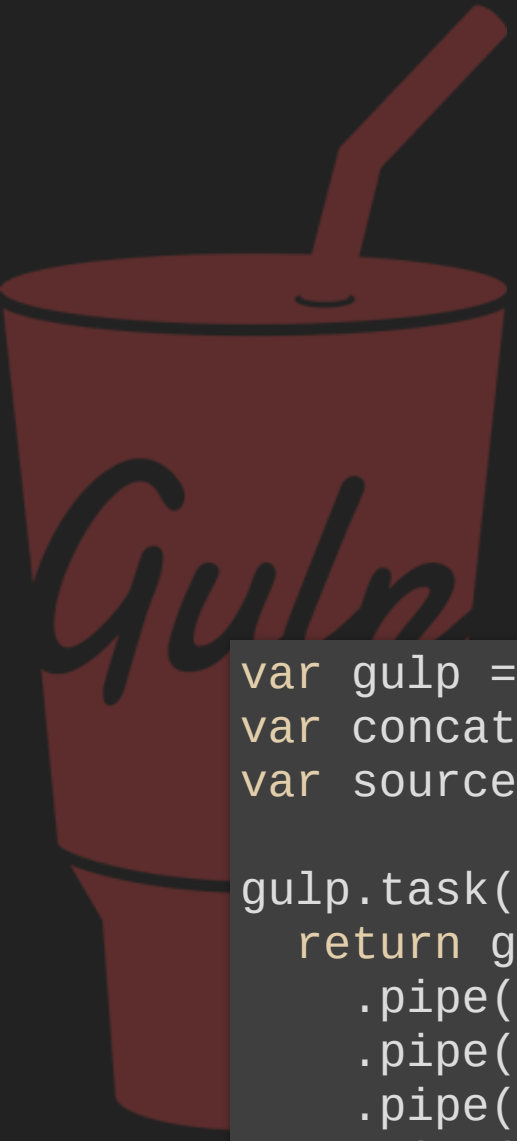
- Lint
- Transpile
- Uglify
- Minify
- Optimize
- Pack
- Test
- ...

BUILD TOOLS

- Grunt
  - ~6,500 plugins
- Gulp
  - JS build files, ~3,900 plugins, ...
- npm
  - install, build
- (Webpack)

GRUNT EXAMPLE

```
// Project configuration.
grunt.initConfig({
  pkg: grunt.file.readJSON('package.json'),
  uglify: {
    options: {
      banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %
    },
    build: {
      src: 'src/<%= pkg.name %>.js',
      dest: 'build/<%= pkg.name %>.min.js'
    }
  }
});
// Load the plugin that provides the "uglify" task.
grunt.loadNpmTasks('grunt-contrib-uglify');
```



GULP EXAMPLE

```
var gulp = require('gulp');
var concat = require('gulp-concat');
var sourcemaps = require('gulp-sourcemaps');

gulp.task('js', function() {
  return gulp.src('client/javascript/*.js')
    .pipe(sourcemaps.init())
    .pipe(concat('app.min.js'))
    .pipe(sourcemaps.write())
    .pipe(gulp.dest('build/js'))
});

gulp.task('css', function(){
  // ...
});
```

NPM EXAMPLE

package.json:

```
{
  "name": "react-todo",
  "version": "0.1.0",
  "dependencies": {
    "react": "^16.4.1",
    "react-dom": "^16.4.1",
    "react-scripts": "1.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom"
  }
}
```

UI FRAMEWORKS AND LIBRARIES

JQUERY

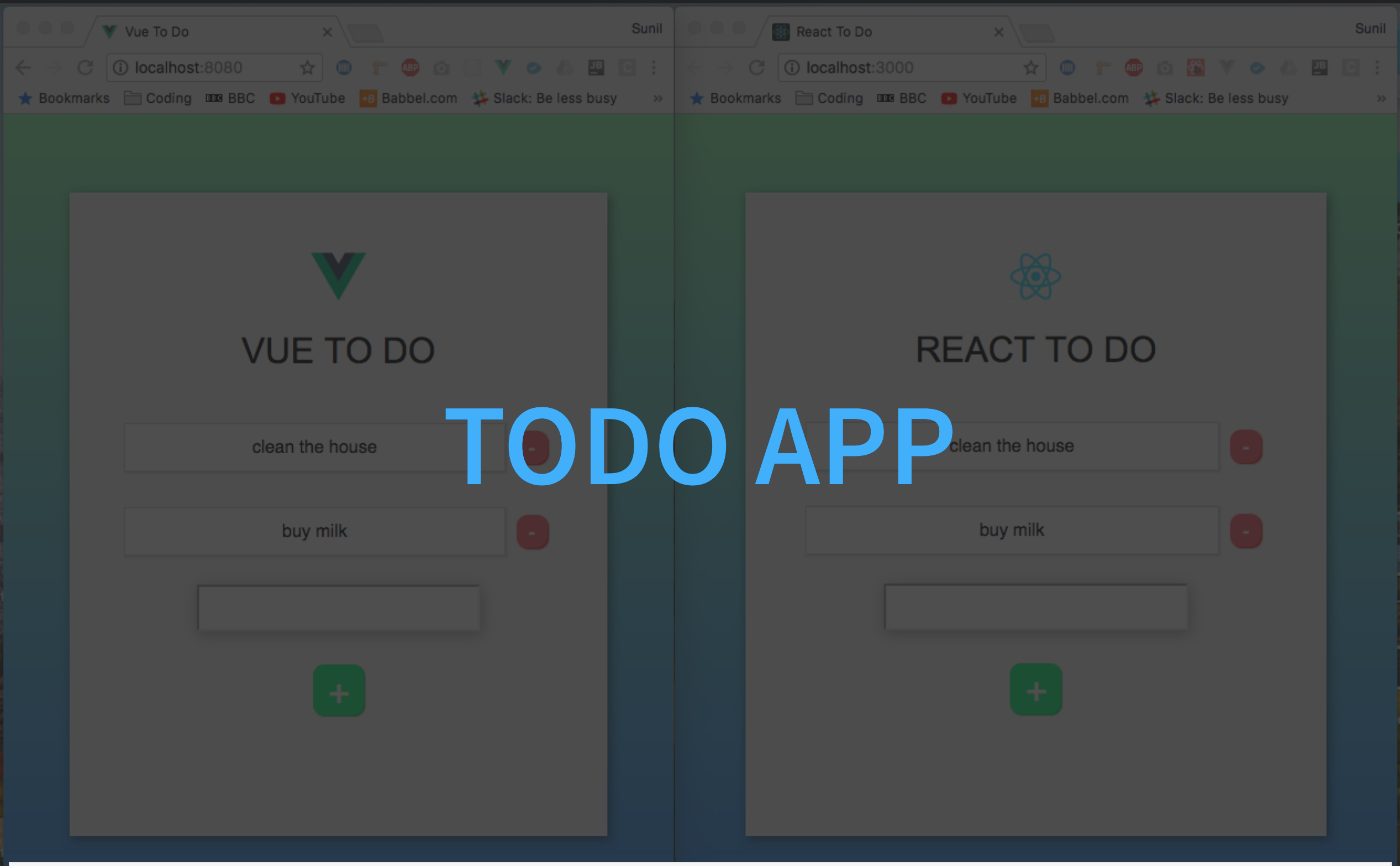
- By far the most widely used JS library
- DOM traversal and manipulation
- Can lead to badly structured code
- Has to some extent become unnecessary because of DOM improvements

Speaker notes

- As of June 2018, jQuery is used on 73% of the top 1 million websites, and by 22.4% of all websites (according to BuiltWith).

JQUERY EXAMPLE

```
$.ajax({
  url: "/api/getWeather",
  data: {
    zipcode: 97201
  },
  success: function( result ) {
    $("#weather-temp").html("<strong>" + result + "</strong> degrees");
  }
});
```



Speaker notes

- I created the exact same app in React and Vue. Here are the differences.

ANGULAR

- 1st generation: AngularJS
- 2nd generation: Angular 2, 4, 5, 6, 7
  - TypeScript
  - Component-based
  - Modularity
  - Complete application framework
  - Progressive Web Apps
  - CLI

REACT

- UI Library
- Component-based
- Application state synchronized with views
- Uses proprietary JSX format
- Applications may need additional libraries
- CLI

VUE.JS

- UI Library
- Component-based
- Application state synchronized with views
- Supports React's JSX format
- Applications may need additional libraries
- CLI

Speaker notes

- Vue was created by Evan You after working for Google using AngularJS in a number of projects. He later summed up his thought process: "I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight."[6] Vue was originally released in February 2014.

WEB COMPONENTS

- **Custom Elements**
  JS APIs that allow you to create custom elements.
- **Shadow DOM**
  JS APIs for attaching a "shadow" DOM tree to an element.
- **HTML Templates**
  HTML placeholder elements with markup that is not rendered.

All standard HTML and ES6

LIT-HTML

Next-generation HTML Templates in JavaScript

```
import {html, render} from 'lit-html';

// A lit-html template uses the `html` template tag:
let sayHello = (name) => html`<h1>Hello ${name}</h1>`;

// It's rendered with the `render()` function:
render(sayHello('World'), document.body);

// And render only updates the data that changed, without VDOM diffing!
render(sayHello('Everyone'), document.body);
```

Faster than React!

LIT-ELEMENT

```
<script type="module">
  import {LitElement, html} from '@polymer/lit-element';

  class MyElement extends LitElement {

    static get properties() {
      return { mood: {type: String} };
    }

    constructor() {
      super();
      this.mood = '';
    }

    render() {
```

JSPM

- Load npm packages with the native browser ES module loader
- Modules are all served as separate files over HTTP/2 with CDN edge caching.
- Far-future expires are provided for exact package versions for fast reloads.
- Packages are lazy-loaded and cached
- Version support

Speaker notes

- HTTP/2 may eliminate the need for minifying and packing
- Using ES6 modules implies lazy-loading

TODO APP COMPARISON

	Angular	React	Vue	Lit-Elm	ES6
Disk	376 M	186 M	168 M	299 M	560 k
LoC	807	218	245	291	236
LoC JS	248	93	186	105/196	105/196
Devel	3.61 M	1.58 M	1.65 M	114 k	102 k
Optimized	3.30 M	141 k	89 k	46 k	32 k

WRAPPING UP

- ECMAScript 2015 is a complete platform for client app development
- If you really like static types, TypeScript is a nice alternative
- Think twice about relying on a framework (and its dependencies)

LINKS

- A Brief History of JavaScript
- How it feels to learn JavaScript in 2016 :)
- ECMAScript 6 compatibility table
- A night experimenting with Lit-HTML…