RICE®

Web Development

COMP 431 / COMP 531

Modern JavaScript

Scott E Pollack, PhD

September 15, 2016

Recap

- HTML and HTML5, Storage, Canvas
- JavaScript and Scope
- Forms
- CSS
- Events
- jQuery, AJAX, and fetch

Modern JS (today)
MVC
React & Redux
Testing (today too!)

Homework Assignment 3
(Draft Frontend)
Due Tuesday 9/20

Recent Evolution of JavaScript

- June 1997 – ECMAScript as ECMA-262 specification
- Dec 1999 – ECMAScript 3 = *JavaScript*
  regular expressions, try/catch, function scope, etc...
- Dec 2009 – ECMAScript 5 = *strict mode*
- June 2015 – ECMAScript 6 = *Harmony* (aka ES2015)
  classes, modules, generators, arrow functions, collections, promises, reflection, block scope let & const, destructuring, template literals, extended parameter handling, proxying
- June 2016 – ECMAScript 7 (aka ES2016)
  improved rest & destructuring, [].includes, decorators, 2**3, async/await, single instruction multiple data (SIMD)

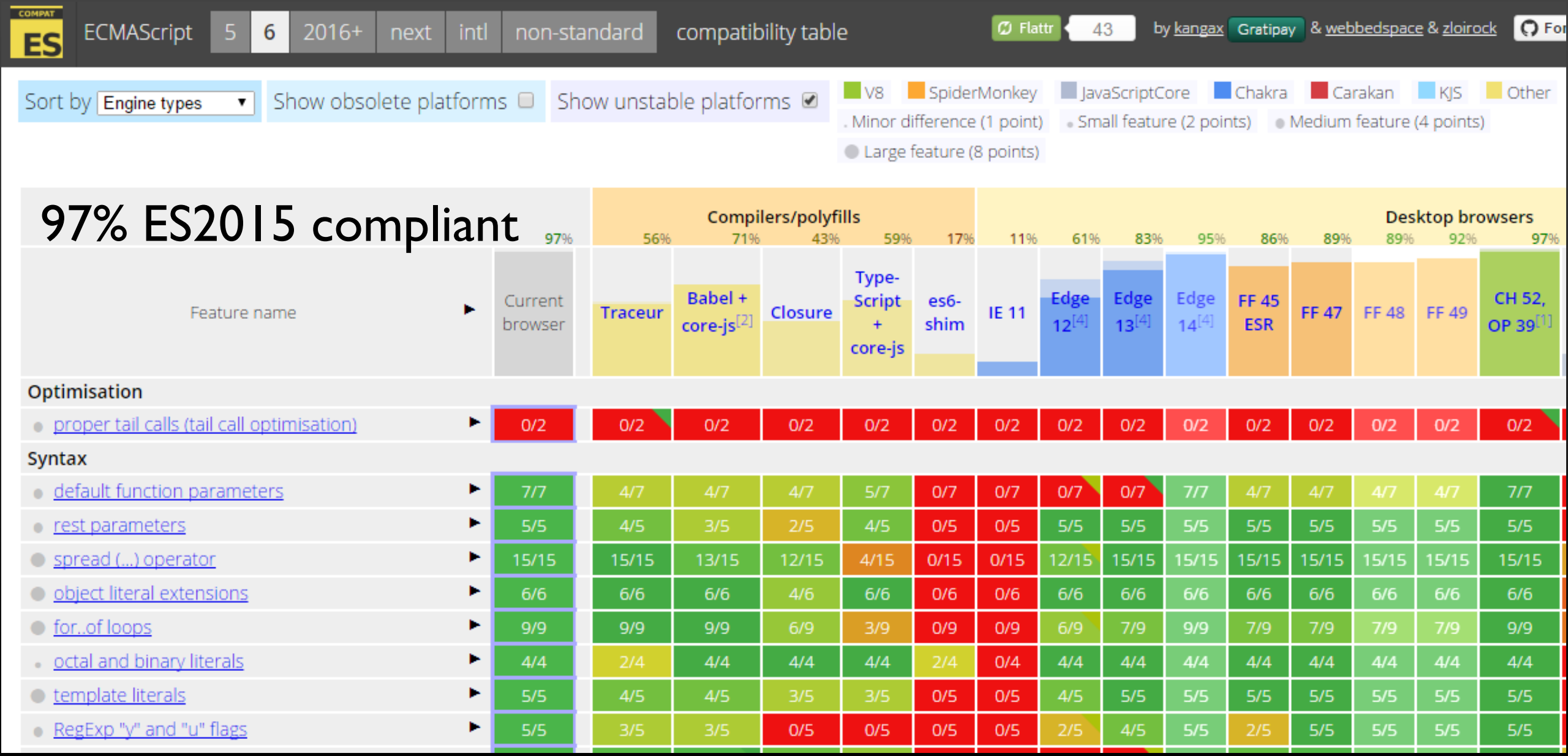
…

Polyfill

- Backport new technologies onto old platforms

```
/*we define the isThereSessionStorage variable
  which will store either true or false
*/

var isThereSessionStorage = (function() {
  try {
    return typeof window.sessionStorage !== 'undefined';
  } catch (e) {
    return false;
  }
})();

if(!isThereSessionStorage)
// our polyfill code goes here....
```

http://kangax.github.io/compat-table/es6/

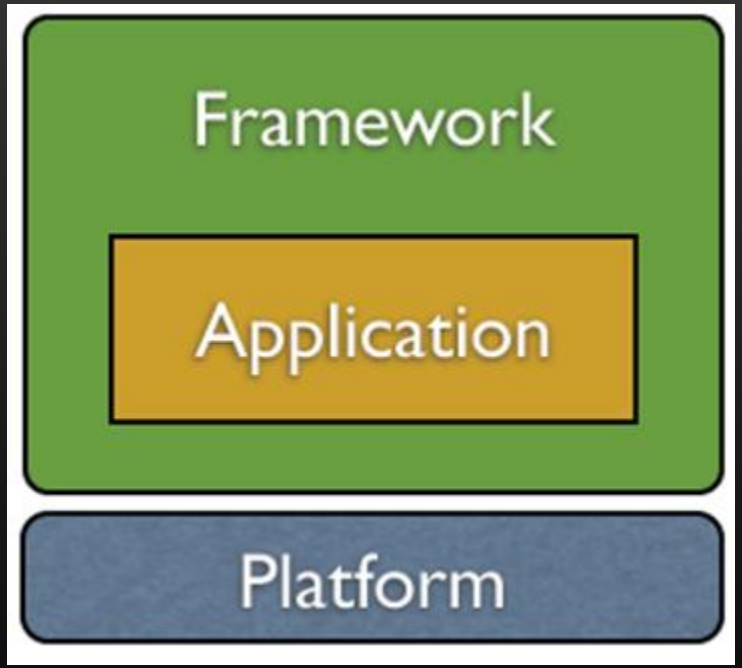


Transpilation

- Source-to-source compilation
- Compile next generation JavaScript to today's JavaScript
- Heavily used prior to 2016 because most browsers did not natively support ES2015 features
- Still used today, chiefly for "import" but also other next generation features such as improved destructuring and decorators
- Even though your browser likely supports ES2015, try out transpilation to see what it looks like: https://babeljs.io/repl

Node JS

- 2009 – Invented by Ryan Dahl at Joyent *(virtualization+cloud computing)*
- 2011 – npm created by Isaac Schlueter
- 2014 – Timohy Fontaine is new lead
- June 2015 – Node.js Foundation

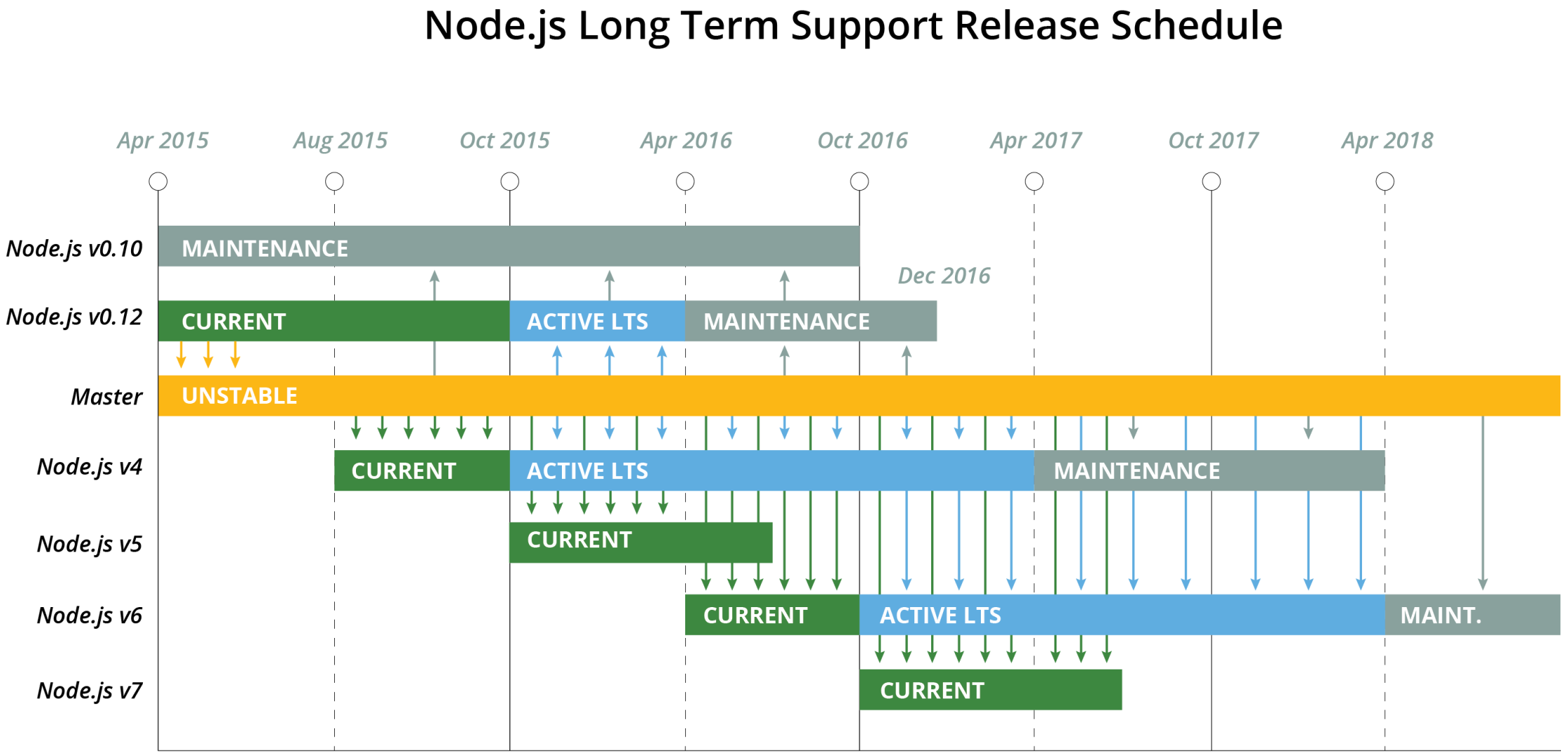


- Operating system agnostic
- Built on Google's V8 JavaScript engine
- asynchronous, event driven, single thread
- Non-blocking and Event driven I/O
- Data Intensive Real-Time (DIRT)
- Node is a **platform** (not a framework)

Install node.js

https://nodejs.org

Node evolution

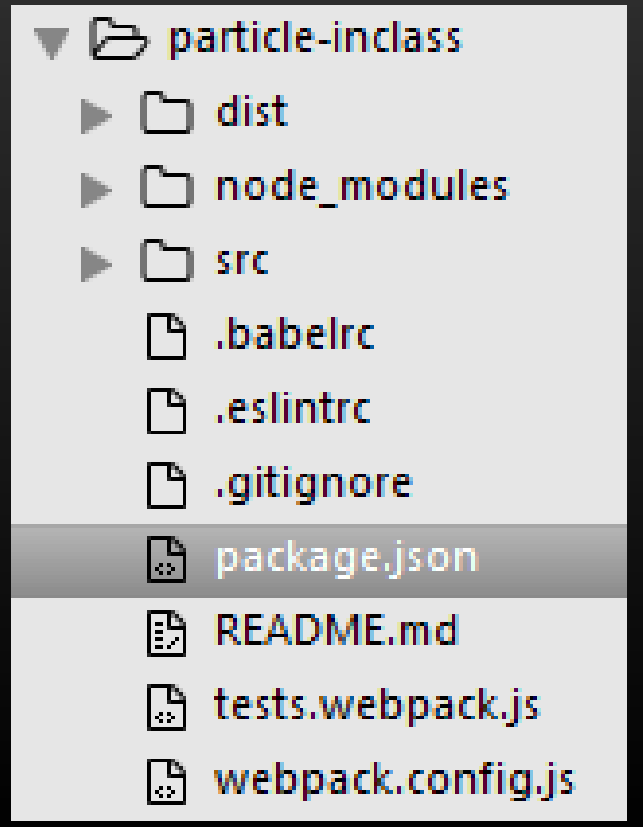


Getting started

Download

- https://www.clear.rice.edu/comp431/sample/particle-inclass.zip

```
unzip particle-inclass.zip
cd particle-inclass
npm install
npm start
```



package.json

package.json

```
{
  "name": "starter",
  "version": "1.0.0",
  "description": "COMP 431/531 starter",
  "main": "./src/index.js",
  "scripts": {
    "clean": "rimraf dist/bundle.js*",
    "lint": "eslint src --ext .js --ext .jsx --cache",
    "watch": "webpack --watch",
    "build": "webpack",
    "deploy": "webpack && surge -p dist",
    "prestart": "npm run build",
    "start": "webpack-dev-server --content-base dist",
    "test": "npm run mocha",
    "mocha": "mocha --compilers js:babel-core/register --recursive src/**/*.spec.js
    "karma": "karma start || exit 0",
    "e2e": "mocha --opts e2e/mocha.opts --reporter spec e2e/*.spec.js",
    "e2e-watch": "mocha -w --opts e2e/mocha.opts --reporter spec",
    "e2e-xunit": "mocha --opts e2e/mocha.opts --reporter xunit e2e/*.spec.js > e2e/
    "e2e-serve": "cd dist && (python -m SimpleHTTPServer 8080 || python -m http.ser
  },
  "author": "Scott Pollack",
  "engines": {
    "node": ">=6",
    "npm": ">=3"
  },
  "license": "MIT",
  "devDependencies": {
    "babel-core": "^6.8.0",
    "babel-istanbul": "^0.8.0"
```

JavaScript modules

Modules provide us encapsulation. When *imported* (or *required*) a file is wrapped in an IIFE and provided to the caller as an object with "handles" to the default and optional exported members (functions, variables)

index.js

```
1  import particle from './particle'
2  import { update } from './particle'
3
4  window.onload = () => {
5      const canvas = document.getElementById( app )
6      const c = canvas.getContext("2d")
7
8      const frameUpdate = (cb) => {
9          const rAF = (time) => {
10             requestAnimationFrame(rAF)
11             const diff = ~~(time - (rAF.lastTime || 0)) // ~~ is like floor
12             cb(diff)
13             rAF.lastTime = time
14         }
15         rAF() // go!
16     }
17
18     const log = (msg) => {
19         if (!msg) { log.x = 30; log.y = canvas.height }
```

JavaScript modules

Modules provide us encapsulation. When *imported* (or *required*) a file is wrapped in an IIFE and provided to the caller as an object with "handles" to the default and optional exported members (functions, variables)

index.js

```
1  import particle from './particle'
2  import { update } from './particle'
3
4  window.onload = () => {
5      const canvas = document.getElementById( app )
6      const c = canvas.
7
8      const frameUpdate
9          const rAF = (
10             requestAni
11             const dif
12             cb(diff)
13             rAF.lastT
14         }
15         rAF() // go!
16     }
17
18     const log = (msg)
19         if (!msg) { log.x = 30; log.y = canvas.height }
```

particle.js

```
13
14 const update = ({acceleration, velocity, position, mass}
15     return { mass, acceleration, velocity, position }
16 }
17
18 export default particle
19
20 export { update }
21
```

Webpack / Browserify

index.html

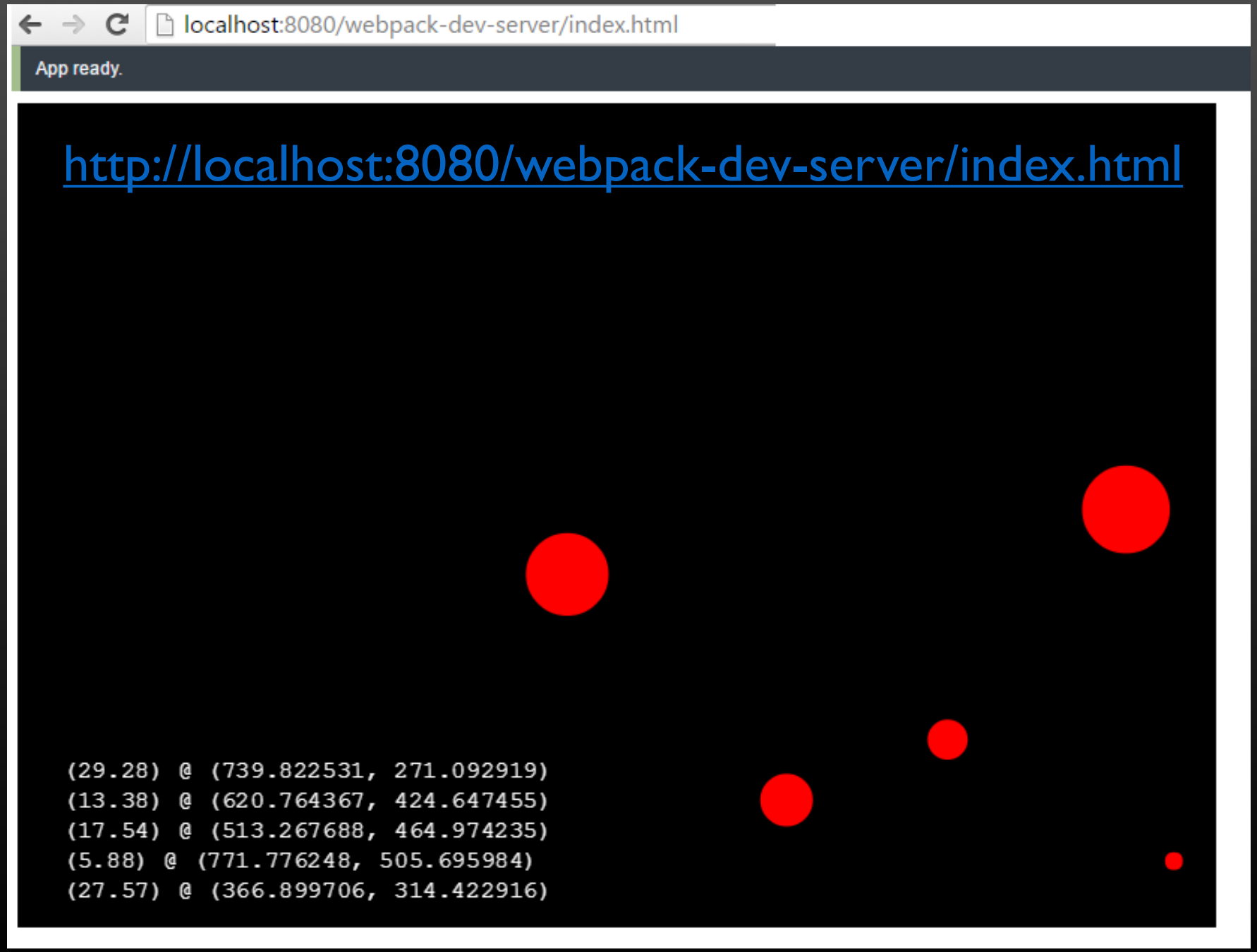
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Physics</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>
    <canvas id="app" width="800" height="550"></canvas>
    <script src="bundle.js"></script>
</body>
</html>

```

Transpiling and packing: import to require to …

bundle.js

```
/******/
/******/    // Load entry module and return exports
/******/    return __webpack_require__(0);
/******/ })
/************************************************************************
/******/ ([
/* 0 */
/***/ function(module, exports, __webpack_require__) {

    'use strict';

    var _slicedToArray = function () { function sliceIterator(arr, i) {

    var _particle = __webpack_require__(1);

    var _particle2 = _interopRequireDefault(_particle);

    function _interopRequireDefault(obj) { return obj && obj.__esModule

    window.onload = function () {
        var canvas = document.getElementById('app');
        var c = canvas.getContext("2d");
```

Particles


localhost:8080/webpack-dev-server/index.html
App ready.
http://localhost:8080/webpack-dev-server/index.html
(29.28) @ (739.822531, 271.092919)
(13.38) @ (620.764367, 424.647455)
(17.54) @ (513.267688, 464.974235)
(5.88) @ (771.776248, 505.695984)
(27.57) @ (366.899706, 314.422916)

Testing

1. Unit tests prove that your code actually works
2. You get a low-level regression-test suite
3. You can improve the design without breaking it
4. It's more fun to code with them than without
5. They demonstrate concrete progress
6. Unit tests are a form of sample code
7. It forces you to plan before you code
8. It reduces the cost of bugs
9. It's even better than code inspections
10. It virtually eliminates coder's block
11. Unit tests make better designs
12. It's faster than writing code without tests

CODING HORROR

programming and human factors

20 Jul 2006

I Pity The Fool Who Doesn't Write Unit Tests

J. Timothy King has a nice piece on the twelve benefits of writing unit tests first.

You'll get no argument from me on the overall importance of unit tests. I've increasingly come to believe that **unit tests are so important that they should be a first-class language construct**.

http://blog.codinghorror.com/i-pity-the-fool-who-doesnt-write-unit-tests/

http://www.jtse.com/blog/2006/07/11/twelve-benefits-of-writing-unit-tests-first

Testing

particle.spec.js

```
1  import { expect } from 'chai'
2  import particle from './particle'
3  import { update } from './particle'
4
5  describe('Particle Functionality', () => {
6
7      it('should have default values', () => {
8          const p = particle()
9          expect(p).to.be.ok
10         expect(p.missingAttribute).to.not.be.ok
11         // check position, velocity, acceleration, mass
12     })
13
14     it('should update the position by the velocity', () => {
15         const p = particle({ position: [1, 1], velocity: [0.5, -0.5] })
16         const { position } = update(p, 1.0)
17         expect(position).to.equal([1.5, 0.5])
18     })
19
20     it('should update the position by the velocity and time delta', () => {
21         const p = particle({ position: [1, 1], velocity: [0.5, -0.5] })
22         const { position } = update(p, 2.0) // dt is different here
23         expect(position).to.equal([2.0, 0.0])
24     })
```

Hosting Assignment 3 Draft Frontend

Homework Assignment 3
(Draft Frontend)
Due Tuesday 9/20

In addition to submitting your repo for grading!

- Host your web app on surge! (it's free)
- Here's an example package:
  - https://www.clear.rice.edu/comp431/sample/surge-example.zip
  - Hosted: http://chivalrous-credit.surge.sh/

```
# get surge installed
npm install
# host your files locally
npm start
# deploy to surge
npm run deploy
```

```
{
  "name": "starter",
  "version": "1.0.0",
  "description": "COMP 431/531 starter",
  "main": "./src/index.js",
  "scripts": {
    "deploy": "surge -p dist",
    "start": "cd dist && (python -m SimpleHTTPServer 8080
  },
  "author": "Scott Pollack",
  "license": "MIT",
  "devDependencies": {
    "surge": "^0.18.0"
  },
  "dependencies": {
```

http://localhost:8080

In-Class Exercise: Test Driven Development with Mocha and Modern JS

/inclass-8/...

```
inclass-8
|-- dist
|   |-- bundle.js
|   |-- bundle.js.map
|   `-- index.html
|-- package.json
|-- README.md
|-- src
|   |-- index.js
|   |-- particle.js
|   `-- particle.spec.js
|-- tests.webpack.js
`-- webpack.config.js

2 directories, 10 files
```

- Start with
  - https://www.clear.rice.edu/comp431/sample/particle-inclass.zip
- Implement the tests in particle.spec.js with the correct logic
- Execute the test suite with `npm test`
- NPM can watch your source and spec files and rerun your tests everytime you save: `npm run test:watch`

- *AFTER* the tests have all been written *THEN* implement the logic in particle.js
- `npm run watch` will rebundle your src as you develop, so you can watch the improvements made to the app in realtime.
- Execute `npm start` in a separate terminal.