Web Development

COMP 431 / COMP 531

Lecture 9: Modern JavaScript

Instructor: Mack Joyner

Department of Computer Science, Rice University

mjoyner@rice.edu

http://www.clear.rice.edu/comp431

Recap

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

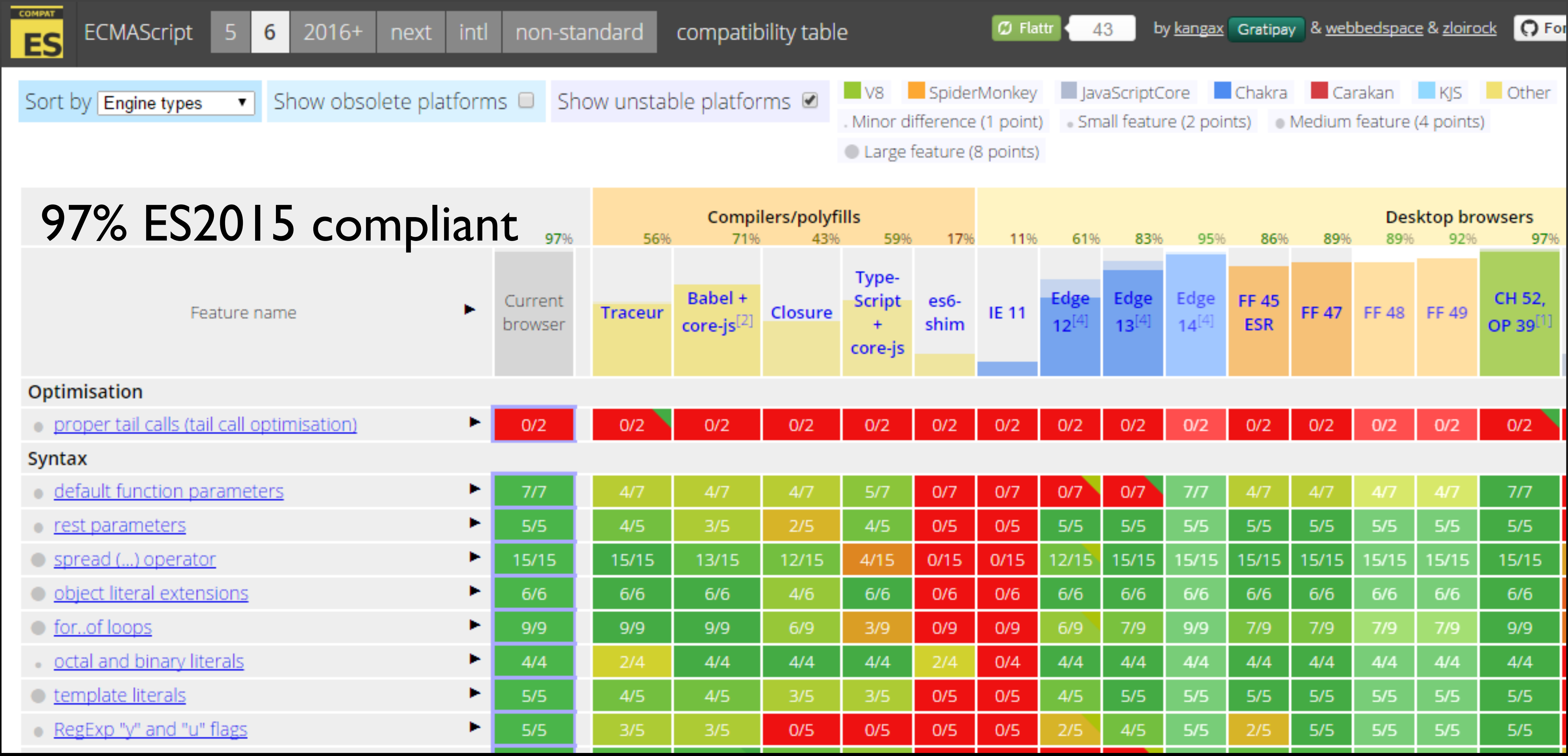
Homework Assignment 3
(JS Game)
Due Thursday 9/28

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)

…

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



ECMAScript 5 6 2016+ next intl non-standard compatibility table

Flattr 43 by kangax Gratipay & webbedspace & zloirock

Sort by Engine types ▾ | Show obsolete platforms ☐ | Show unstable platforms ☑

V8 | SpiderMonkey | JavaScriptCore | Chakra | Carakan | KJS | Other

. Minor difference (1 point) · Small feature (2 points) • Medium feature (4 points) ● Large feature (8 points)

97% ES2015 compliant

		Compilers/polyfills					Desktop browsers								
	97%	56%	71%	43%	59%	17%	11%	61%	83%	95%	86%	89%	89%	92%	97%
Feature name ▶	Current browser	Traceur	Babel + core-js[2]	Closure	Type-Script + core-js	es6-shim	IE 11	Edge 12[4]	Edge 13[4]	Edge 14[4]	FF 45 ESR	FF 47	FF 48	FF 49	CH 52, OP 39[1]
Optimisation															
proper tail calls (tail call optimisation) ▶	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax															
default function parameters ▶	7/7	4/7	4/7	4/7	5/7	0/7	0/7	0/7	0/7	7/7	4/7	4/7	4/7	4/7	7/7
rest parameters ▶	5/5	4/5	3/5	2/5	4/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
spread (...) operator ▶	15/15	15/15	13/15	12/15	4/15	0/15	0/15	12/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
object literal extensions ▶	6/6	6/6	6/6	4/6	6/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
for..of loops ▶	9/9	9/9	9/9	6/9	3/9	0/9	0/9	6/9	7/9	9/9	7/9	7/9	7/9	7/9	9/9
octal and binary literals ▶	4/4	2/4	4/4	4/4	4/4	2/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
template literals ▶	5/5	4/5	4/5	3/5	3/5	0/5	0/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
RegExp "y" and "u" flags ▶	5/5	3/5	3/5	0/5	0/5	0/5	0/5	2/5	4/5	5/5	2/5	5/5	5/5	5/5	5/5

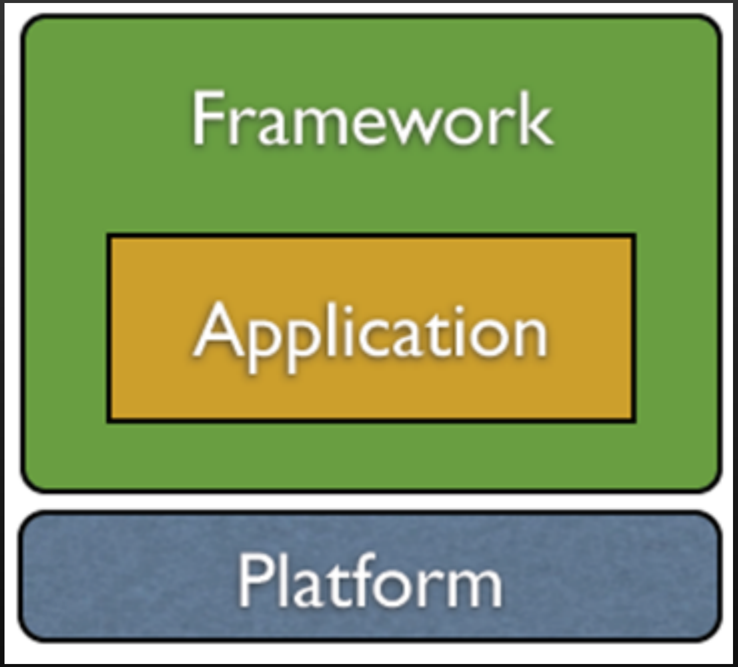
Transpilation

- 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

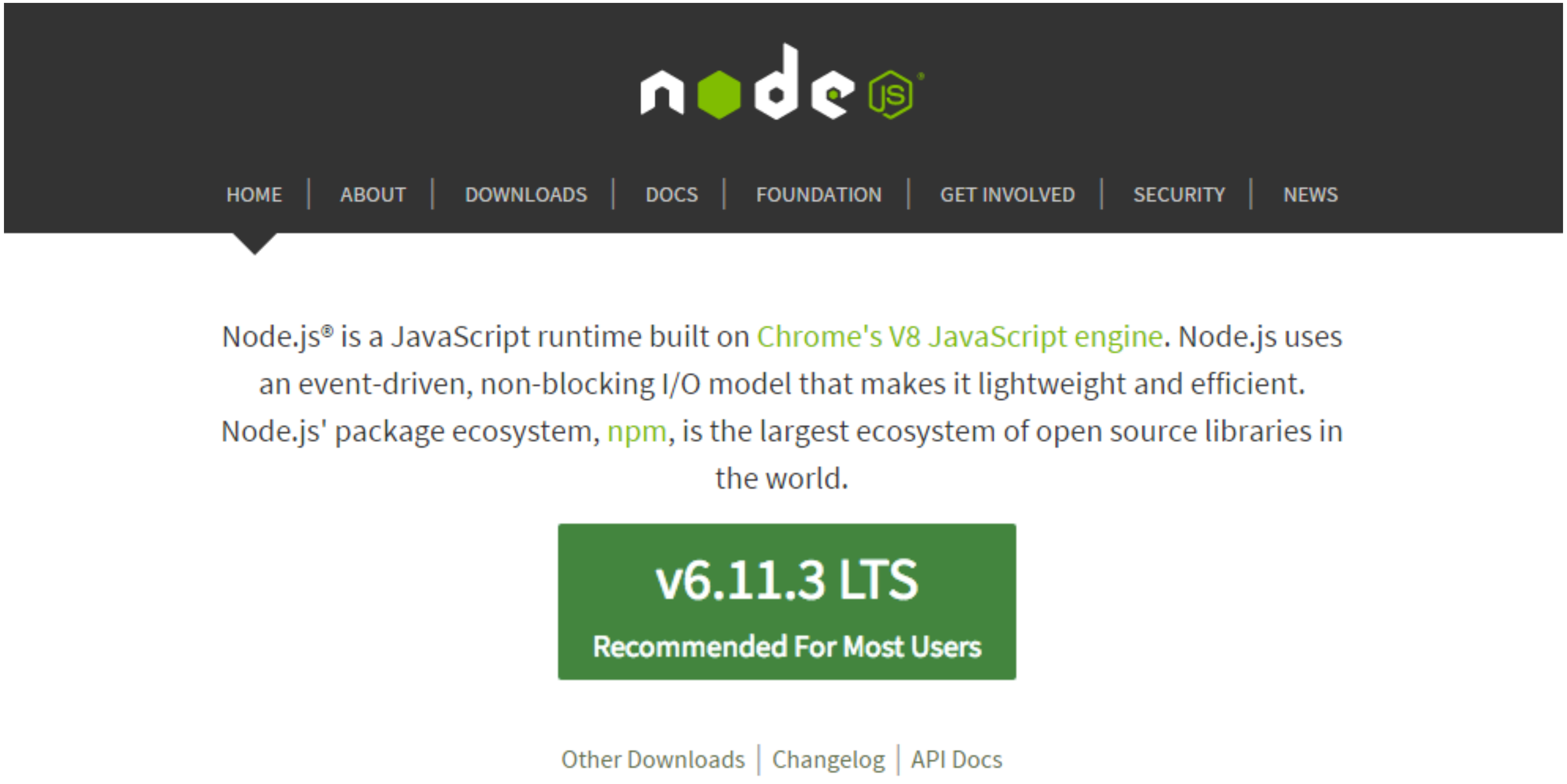
Node JS

- 2009 – Invented by Ryan Dahl at Joyent *(virtualization+cloud computing)*
- 2011 – npm created by Isaac Schlueter
- 2014 – Timothy 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)



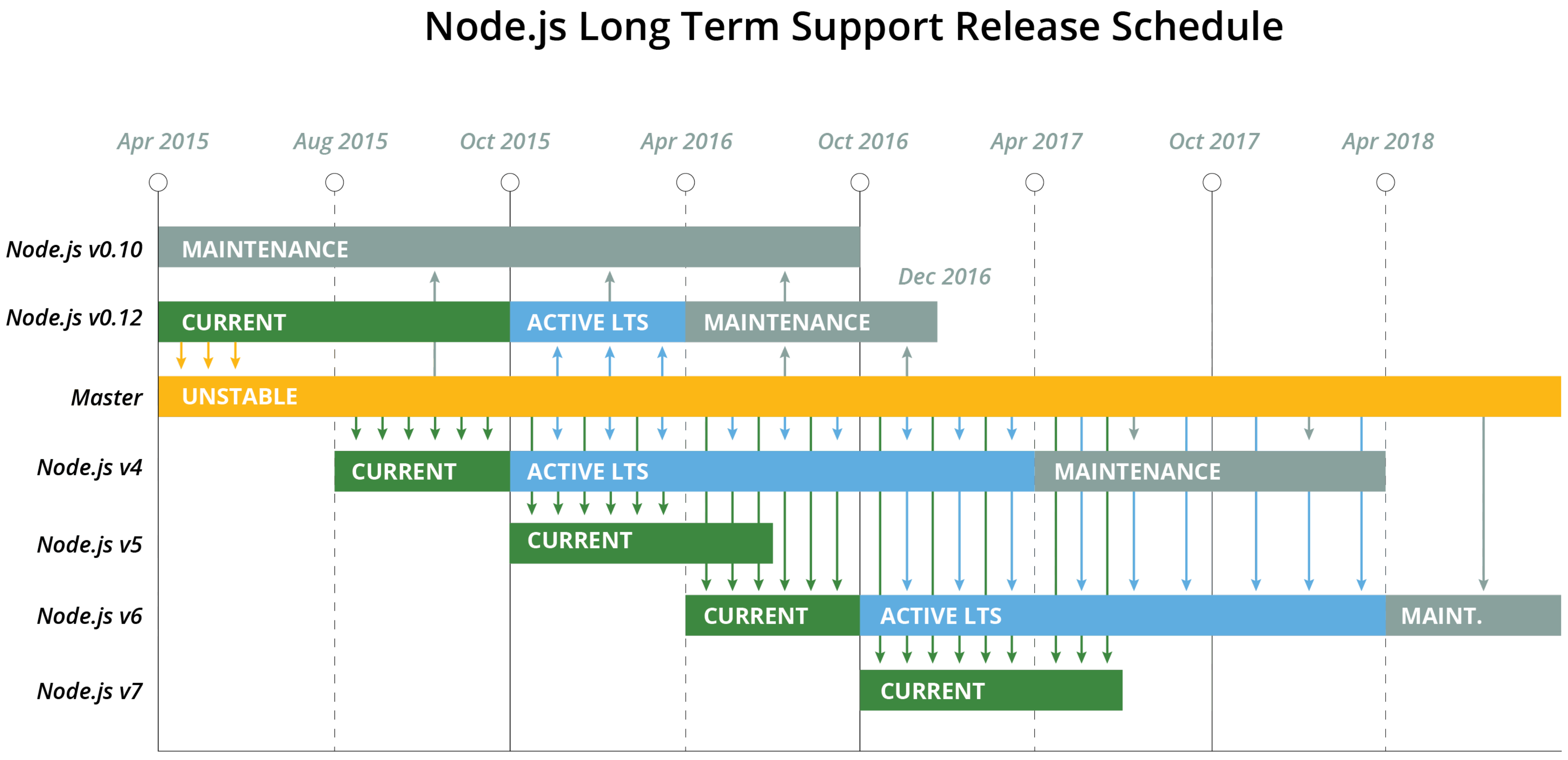
Install node.js https://nodejs.org



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

v6.11.3 LTS
Recommended For Most Users
Other Downloads | Changelog | API Docs

Node evolution



Getting started

Download

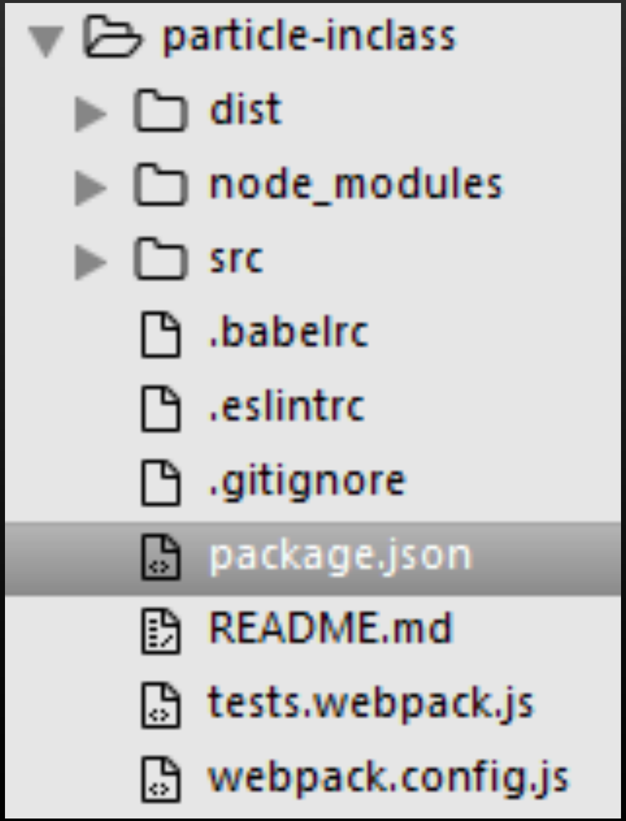
- https://www.clear.rice.edu/comp431/sample/particle-inclass.zip

```
unzip particle-inclass.zip
cd particle-inclass
npm install --verbose
npm run dev
```

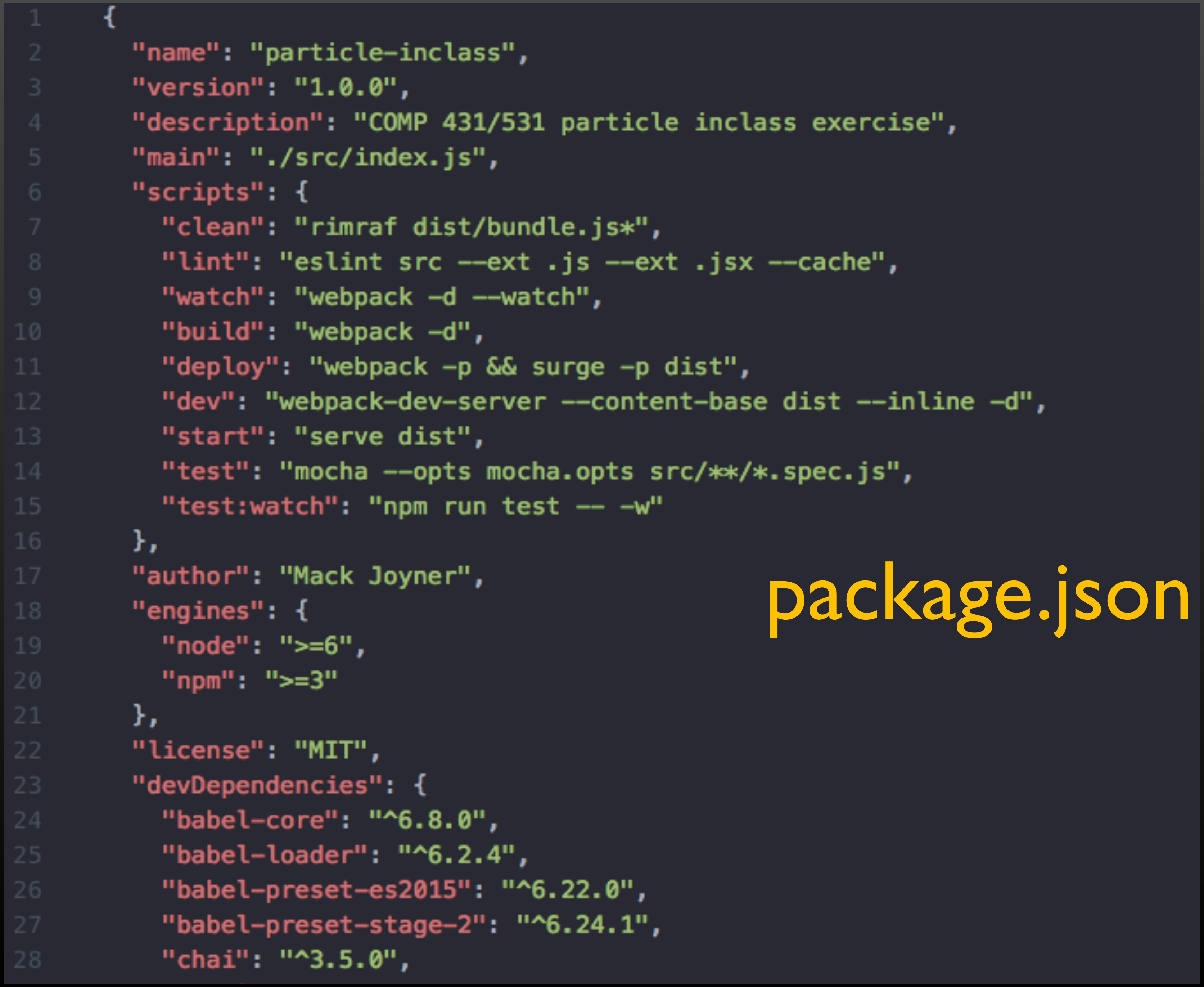
http://localhost:8080

… or

http://127.0.0.1:8080



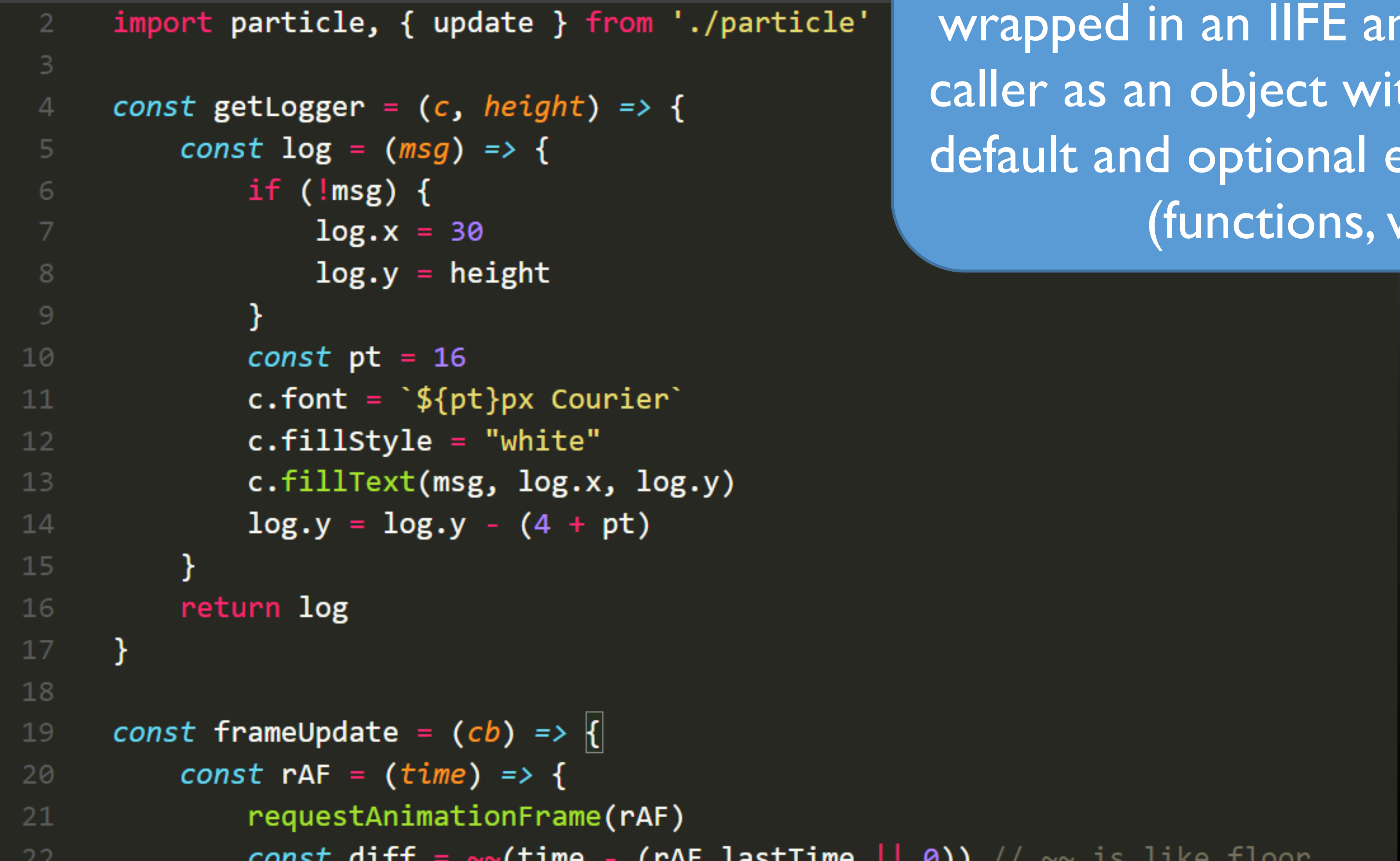
```
particle-inclass
    dist
    node_modules
    src
    .babelrc
    .eslintrc
    .gitignore
    package.json
    README.md
    tests.webpack.js
    webpack.config.js
```



```json
{
  "name": "particle-inclass",
  "version": "1.0.0",
  "description": "COMP 431/531 particle inclass exercise",
  "main": "./src/index.js",
  "scripts": {
    "clean": "rimraf dist/bundle.js*",
    "lint": "eslint src --ext .js --ext .jsx --cache",
    "watch": "webpack -d --watch",
    "build": "webpack -d",
    "deploy": "webpack -p && surge -p dist",
    "dev": "webpack-dev-server --content-base dist --inline -d",
    "start": "serve dist",
    "test": "mocha --opts mocha.opts src/**/*.spec.js",
    "test:watch": "npm run test -- -w"
  },
  "author": "Mack Joyner",
  "engines": {
    "node": ">=6",
    "npm": ">=3"
  },
  "license": "MIT",
  "devDependencies": {
    "babel-core": "^6.8.0",
    "babel-loader": "^6.2.4",
    "babel-preset-es2015": "^6.22.0",
    "babel-preset-stage-2": "^6.24.1",
    "chai": "^3.5.0",
```

package.json

JavaScript modules

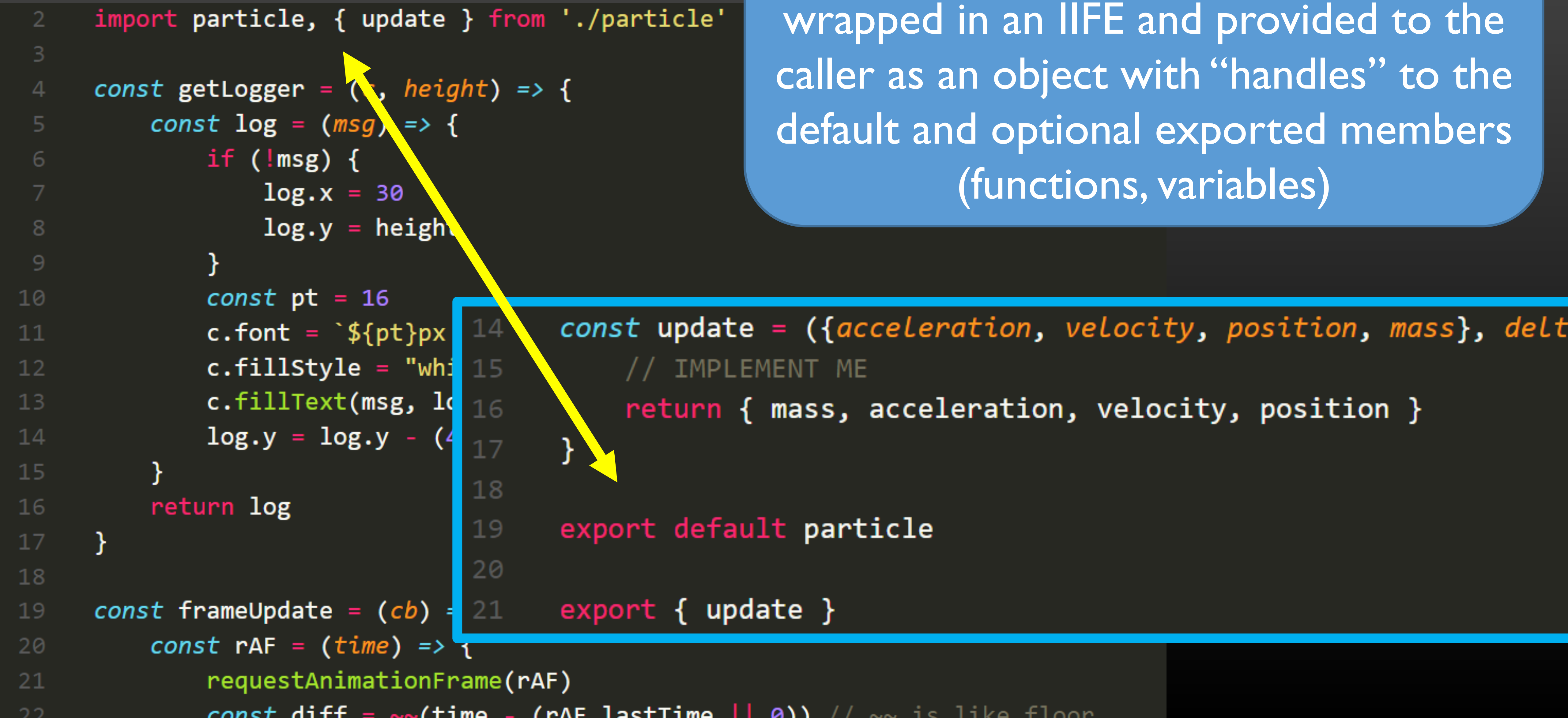


```
import particle, { update } from './particle'

const getLogger = (c, height) => {
    const log = (msg) => {
        if (!msg) {
            log.x = 30
            log.y = height
        }
        const pt = 16
        c.font = `${pt}px Courier`
        c.fillStyle = "white"
        c.fillText(msg, log.x, log.y)
        log.y = log.y - (4 + pt)
    }
    return log
}

const frameUpdate = (cb) => {
    const rAF = (time) => {
        requestAnimationFrame(rAF)
        const diff = ~~(time - (rAF.lastTime || 0)) // ~~ is like floor
```

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)

JavaScript modules

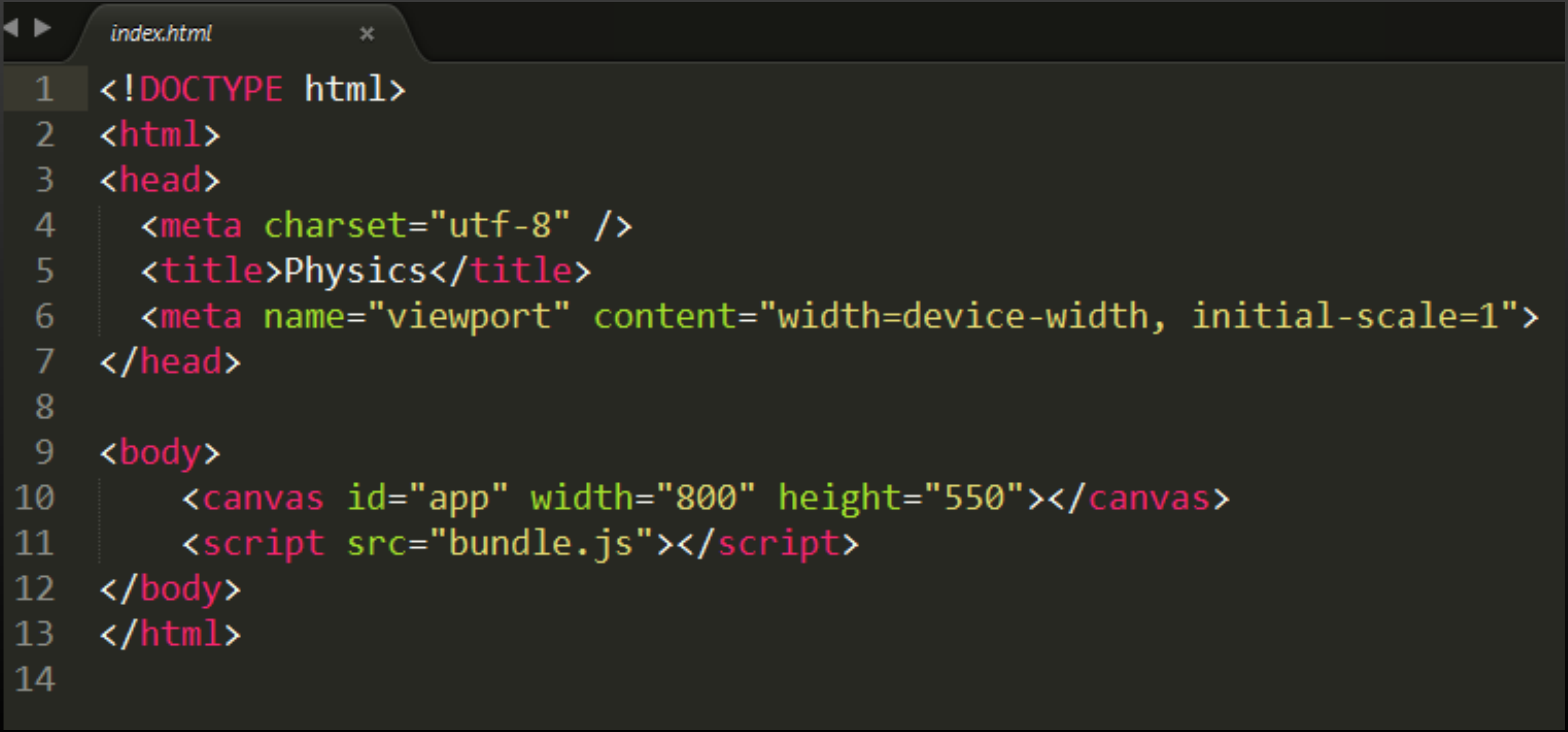
Modules provide us encapsulation. When *imported* (or *required*) a file is wrapped in an IIFE and provided to the caller as an object with “handles” to the default and optional exported members (functions, variables)



```
import particle, { update } from './particle'

const getLogger = (c, height) => {
    const log = (msg) => {
        if (!msg) {
            log.x = 30
            log.y = heigh
        }
        const pt = 16
        c.font = `${pt}px
        c.fillStyle = "whi
        c.fillText(msg, lo
        log.y = log.y - (4
    }
    return log
}

const frameUpdate = (cb) =
    const rAF = (time) => {
        requestAnimationFrame(rAF)
        const diff = ~~(time - (rAF.lastTime || 0)) // ~~ is like floor
```

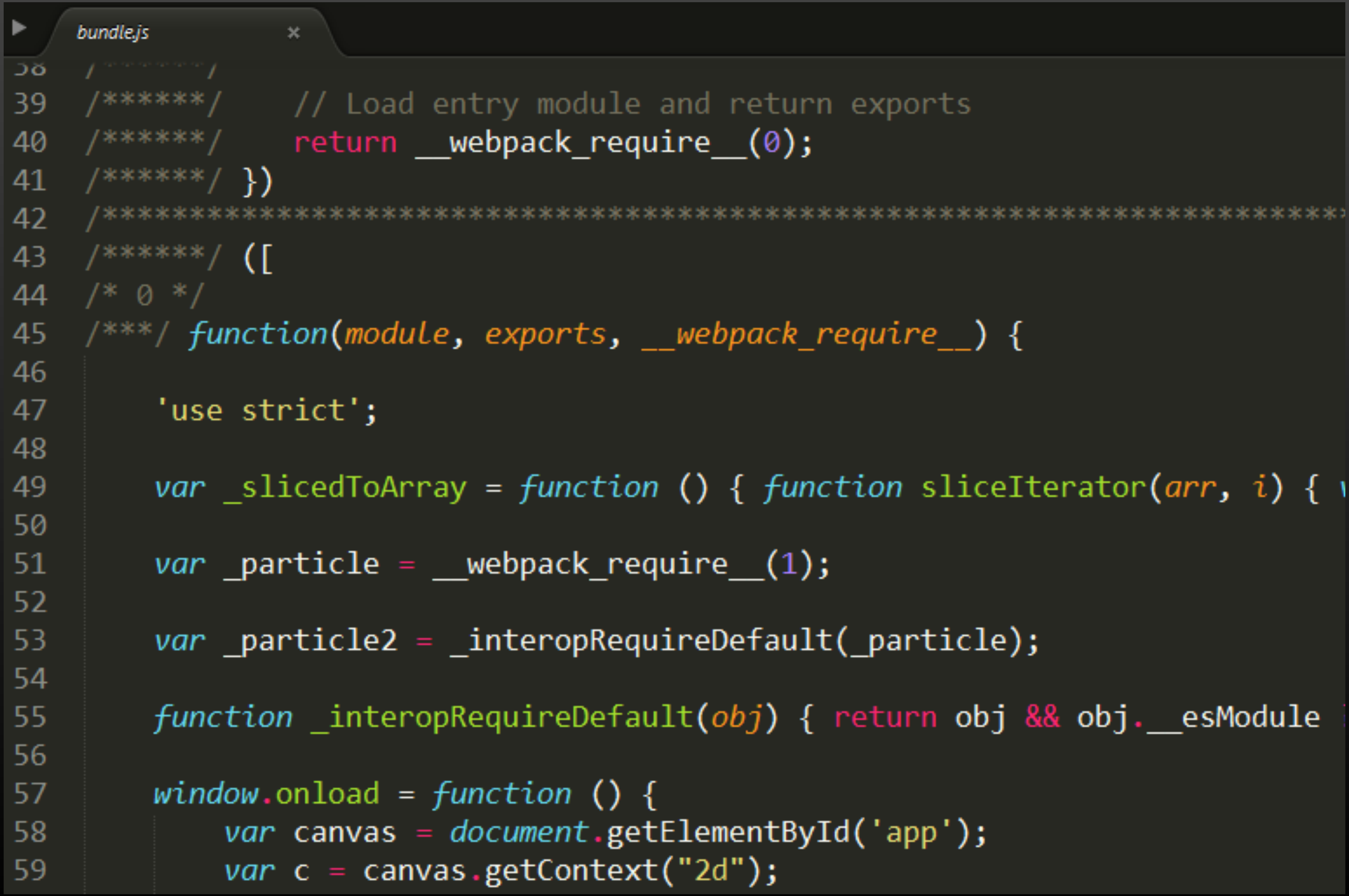
```
const update = ({acceleration, velocity, position, mass}, delt
    // IMPLEMENT ME
    return { mass, acceleration, velocity, position }
}

export default particle

export { update }
```

Webpack



```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>

```

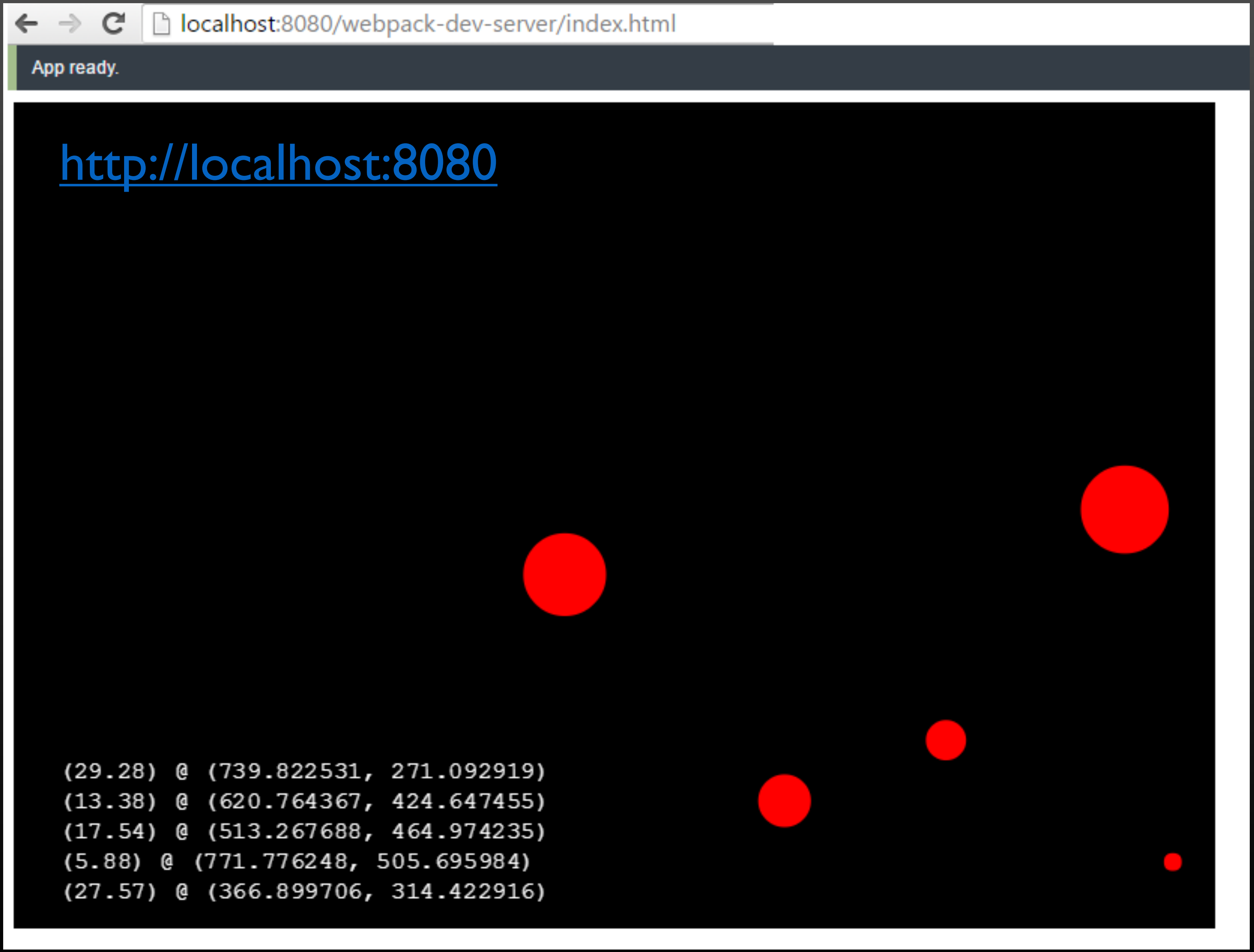
Transpiling and packing: import to require to …



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");
```

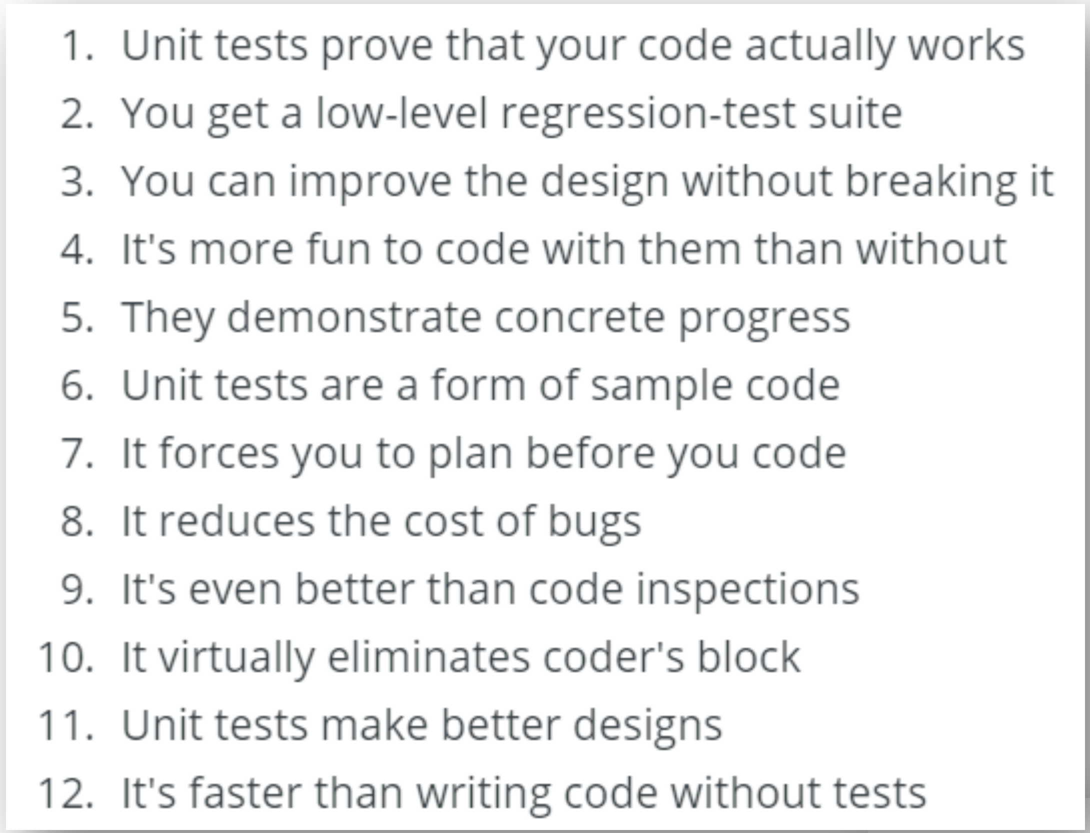
Particles



http://localhost:8080

```
(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



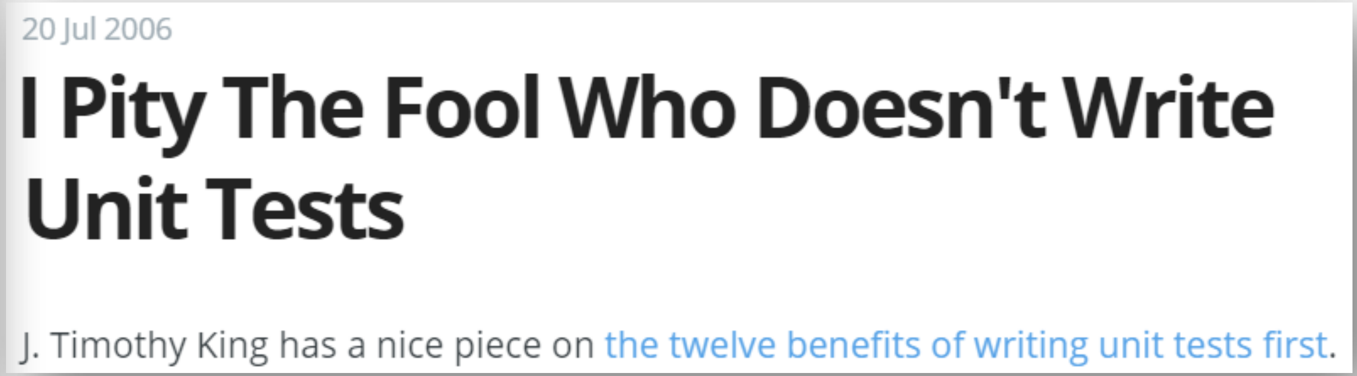
1. Unit tests prove that your code actually works
2. You get a low-level regression-test suite
3. You can improve the design without breaking it
4. It's more fun to code with them than without
5. They demonstrate concrete progress
6. Unit tests are a form of sample code
7. It forces you to plan before 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

Copyright Jeff Atwood © 2015

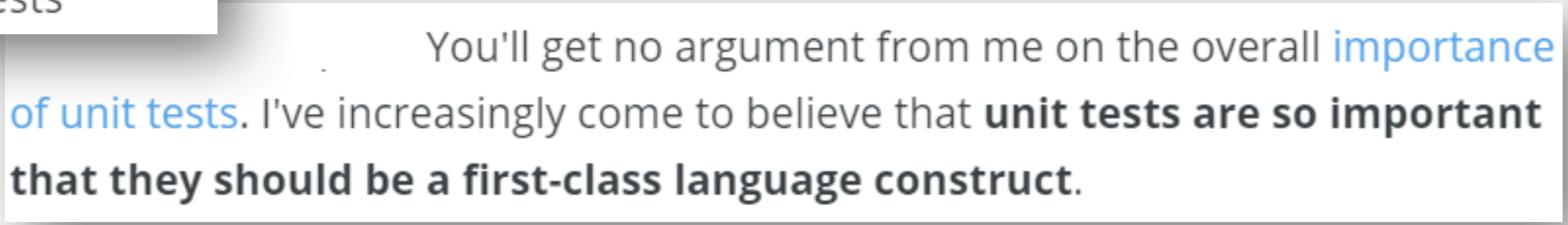
Logo image © 1993 Steven C. McConnell



20 Jul 2006

I Pity The Fool Who Doesn't Write Unit Tests

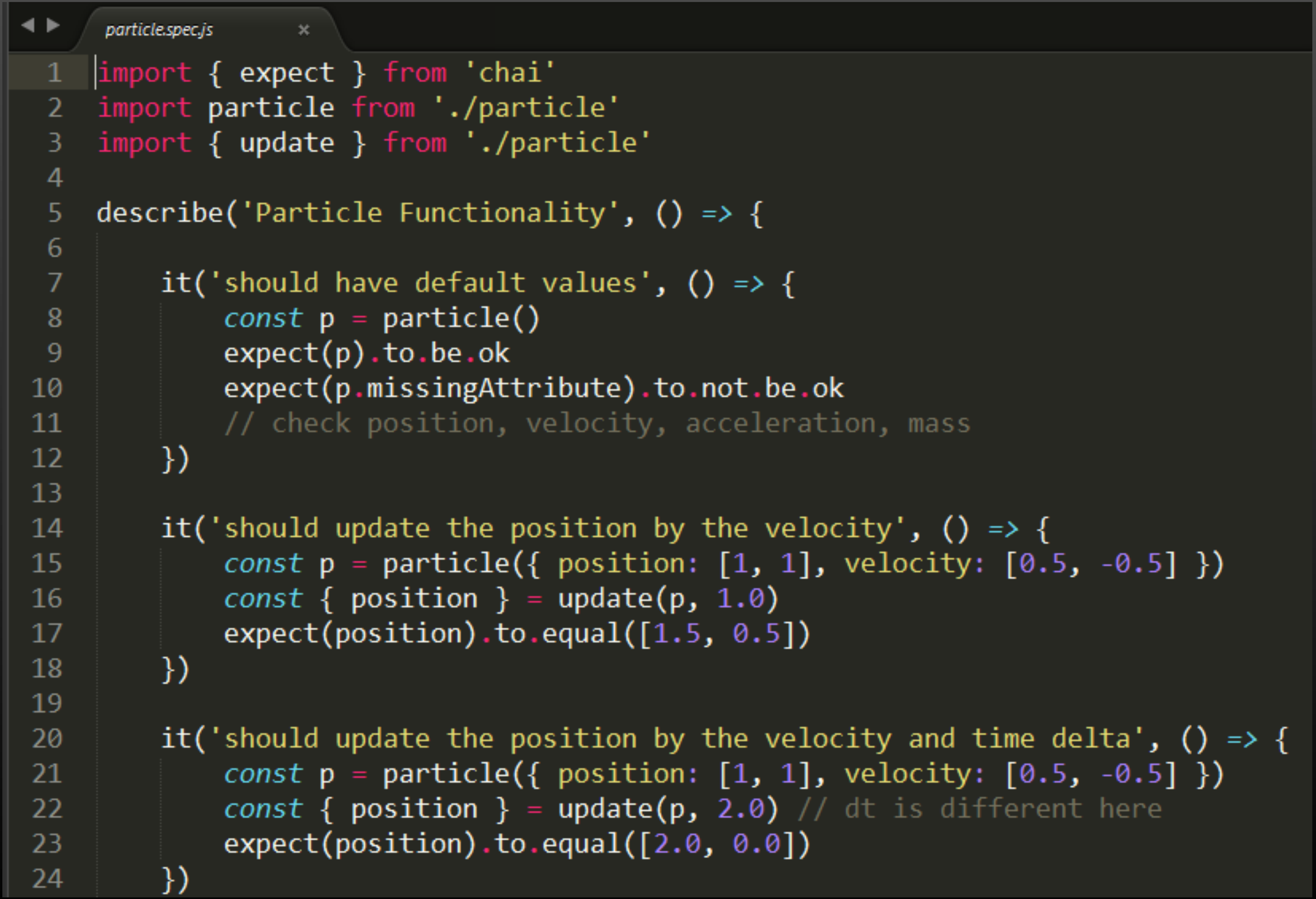
J. Timothy King has a nice piece on the twelve benefits of writing unit tests first.



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

```
import { expect } from 'chai'
import particle from './particle'
import { update } from './particle'

describe('Particle Functionality', () => {

    it('should have default values', () => {
        const p = particle()
        expect(p).to.be.ok
        expect(p.missingAttribute).to.not.be.ok
        // check position, velocity, acceleration, mass
    })

    it('should update the position by the velocity', () => {
        const p = particle({ position: [1, 1], velocity: [0.5, -0.5] })
        const { position } = update(p, 1.0)
        expect(position).to.equal([1.5, 0.5])
    })

    it('should update the position by the velocity and time delta', () => {
        const p = particle({ position: [1, 1], velocity: [0.5, -0.5] })
        const { position } = update(p, 2.0) // dt is different here
        expect(position).to.equal([2.0, 0.0])
    })
```

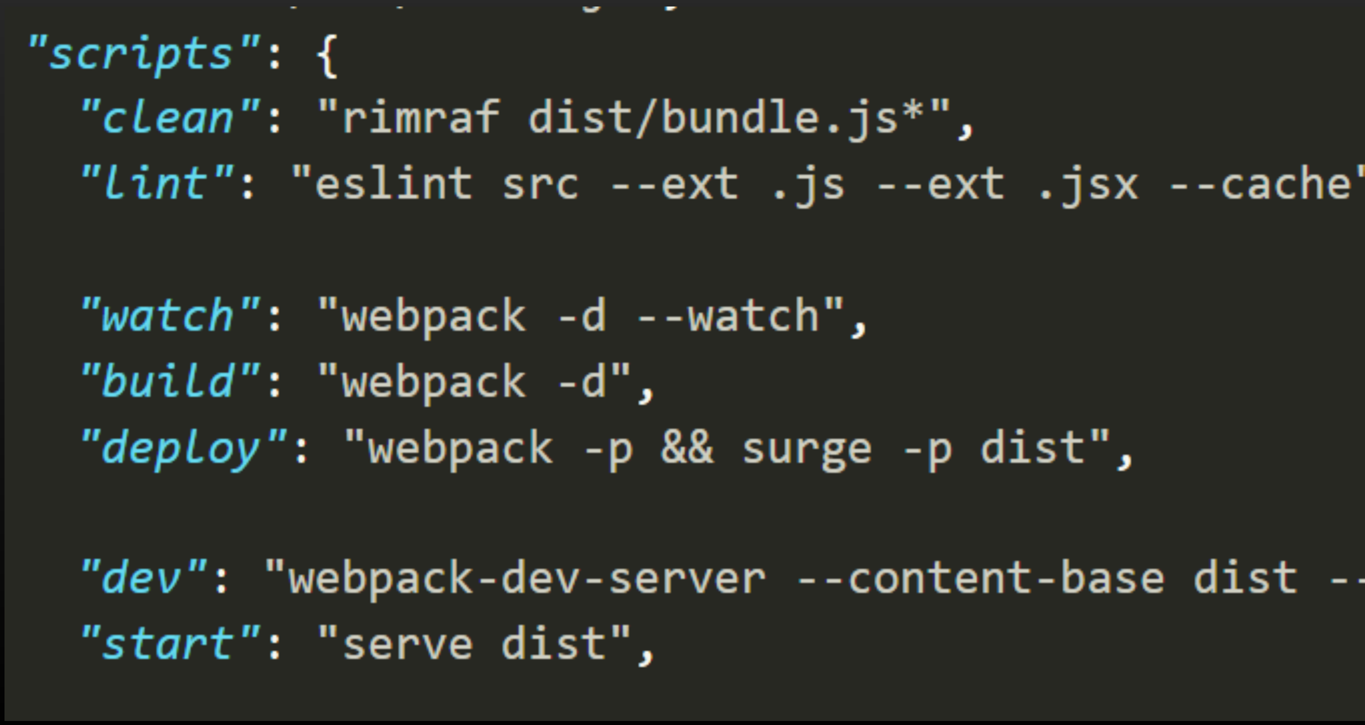
Hosting Assignment 3 JavaScript Game

Homework Assignment 3
(JS Game)
Due Thursday 9/28

In addition to submitting your repo for grading!

- Host your web app on **surge**! (it's free)

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



```
"scripts": {
  "clean": "rimraf dist/bundle.js*",
  "lint": "eslint src --ext .js --ext .jsx --cache'

  "watch": "webpack -d --watch",
  "build": "webpack -d",
  "deploy": "webpack -p && surge -p dist",

  "dev": "webpack-dev-server --content-base dist --
  "start": "serve dist",
```