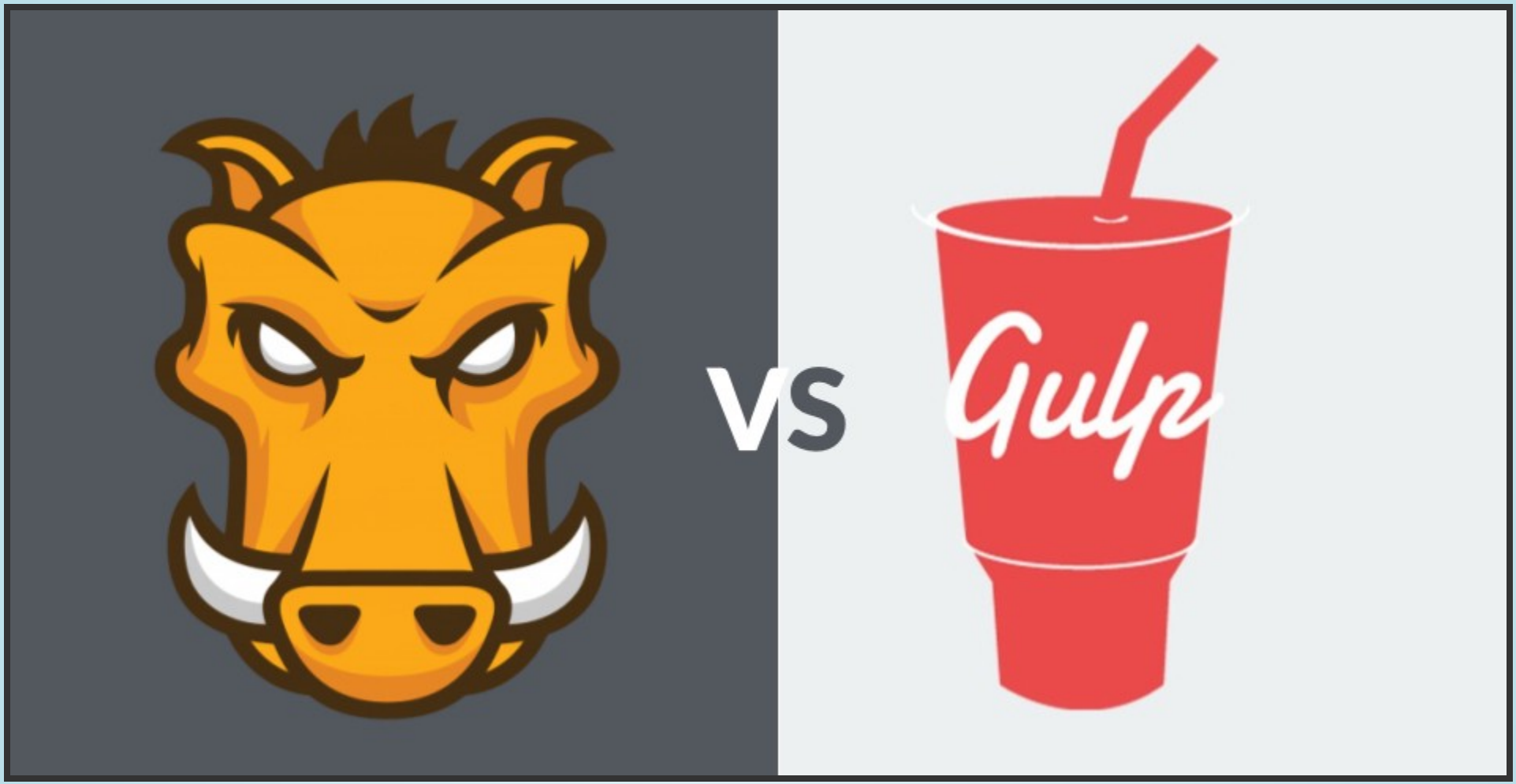
WELCOME TO DAY 0 OF REACTJS BOOTCAMP

AGENDA

- Grunt and Gulp
- Webpack
- Babel
- Eslint
- Arrows
- let
- Destructuring
- Default, Rest, Spread
- Classes
- Module Loader


VS
Gulp

GRUNT

- focus on configuration
- does common tasks very well and very easily configured when going down a happy path
- picks up and drops files from src and dest options so each task opens file readers/writers

```
grunt.initConfig({
clean: {
  src: ['build/app.js', 'build/vendor.js']
},

copy: {
  files: [{
    src: 'build/app.js',
    dest: 'build/dist/app.js'
  }]
}

concat: {
  'build/app.js': ['build/vendors.js', 'build/app.js']
}

//     other task configurations
```

GULP

- focus on code
- leverages streams for piping inbetween tasks
- doesn't enforce much of anything. just use code to wire up tasks and pipe information

```
//import the necessary gulp plugins
var gulp = require('gulp');
var sass = require('gulp-sass');
var minifyCss = require('gulp-minify-css');
var rename = require('gulp-rename');

//declare the task
gulp.task('sass', function(done) {
  gulp.src('./scss/ionic.app.scss')
    .pipe(sass())
    .pipe(gulp.dest('./www/css/'))
    .pipe(minifyCss({
      keepSpecialComments: 0
    }))
    .pipe(rename({ extname: '.min.css' }))
    .pipe(gulp.dest('./www/css/'))
    .on('end', done);
```

SCRIPT LOADING

- Allows for modular applications
- Allow us to pull in dependencies when we need them
- Can bundle scripts on a per page basis
- AMD Script loading with require -> originally browser implementation of CommonJS Transport
- CommonJS define and export modules as well used in Node

```
var component = require('../component/component'); //amd and commonjs sy
```

BROWSERIFY

batteries not included

- Built to ship Node modules to browsers
- Big plugin environment to add things like watch, factor-bundles, deAMDify etc
- Manages JS only
- Uses transforms to modify code
- provides pre and post bundle callbacks
- Minimal config

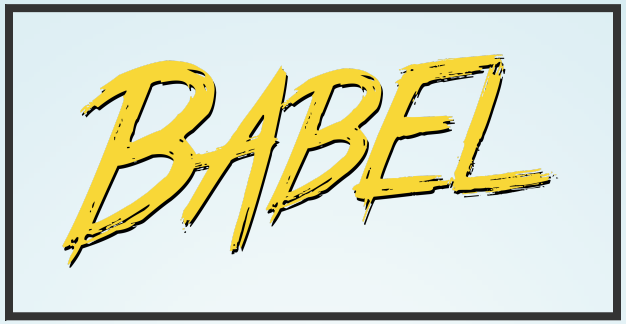
```
var outputs = [ // <- Add new bundle names to this list
  'common',
  'contact',
  'help',
  'enrollment',
  'forgot-credentials',
  'index',
  'initialLogin',
  'login',
  'plan-selection',
  'user-registration',
  'producer-services',
  'reset-password'
];


function generateOutputs(options) {
```

WEBPACK

batteries included

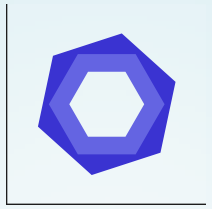
- Our solution for this bootcamp
- Built to be a browser solution with nodejs support
- Bundles all your assets and has loaders to make that easier - great for modularity
- Supports all module formats out of the box
- Complex setup with loaders and etc
- Nice hotloading functionality with its built in dev server

```
var pkg  = require('../package.json'),
    path = require('path');

var DEBUG = process.env.NODE_ENV === 'development';
var TEST = process.env.NODE_ENV === 'test';

module.exports = {
  context: path.join(__dirname, '../public'),
  cache: DEBUG,
  debug: DEBUG,
  watch: DEBUG,
  devtool: DEBUG || TEST ? '#inline-source-map' : false,
  target: 'web',
  entry: './scripts/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(pkg.config.buildDir)
```



- Formerly 6to5 but now handles more than es2015
- Transpiles esnext code into something all browsers can use
- Can transform jsx
- Very up to date and community driven
- Used as a pre-build step when writing esnext in the browser environments

ESLINT



- Extendable code linting and style checking
- Every facet is pluggable
- Built on espree parser
- Lints using AST to evaluate patterns unlike some other linters
- Many great community plugins for frameworks like react

```
{
  "rules": {
      "eqeqeq": 0,
      "curly": 2,
      "quotes": [2, "double"]
  }
}
```

ES2015

Innovation debt is the cost that companies incur when they don't invest in their developers.

- Peter Bell

- Westin Wrzesinski

ARROWS =>

- Inspired by CoffeeScript
- Bind to outer this
- Not newable
- No arguments psuedo array
- Always Anon
- upgrade to es5 bind for callbacks essentially

CODE

```
//Arrows
var evens = numbers.map(num => num % 2 === 0);
nums.map((x) => x * 2);
//or as a statement body
var specialNums = numbers.map(num => {
  return doSomething(num);
});
// Lexical this
var person = {
  _name: "Westin",
  _friends: ["Not Justin", "Doug", "Brendan", "Igor"],
  printFriends() {
    this._friends.forEach(f =>
      console.log(`${this._name} knows ${f}`));
  }
}
```

LET

Allows for block scoping

```
function() {
  if(x) {
    var foo = 3;
  }
  var baz = 1;
  //foo and baz in same scope due to hoisting
}
```

```
function() {
  if(x) {
    let foo = 3; //only inside the conditional
  }
  var baz = 1;
  //foo and baz NOT in same scope as foo is no longer hoisted
}
```

DESTRUCTURING OBJECT

```
var people = [
  {
    name: 'Westin',
    age: 25
  }
];
people.forEach(function({name, age}) //shorthand if key = value
{
  console.log(name + ":" + age)
});
```

DESTRUCTURING ARRAY

- Fails quietly to undefined

```
var [month, date, year] = [3, 14, 1977];
//swapping
x = 3;
y = 4;
[x, y] = [y, x];
//ignore an index
var [a, ,b] = [1,2,3];
var doWork = function() {
    return [1, 3, 2];
};
let [, x, y, z] = doWork();
```

DEFAULT, REST, SPREAD

DEFAULT PARAMS

```
function f(x, y=12, z=y) {
  // y is 12 if not passed (or passed as undefined)
  return x + y;
}
f(3) == 15
```

REST

- rest parameters are only the ones that haven't been given a separate name, while the arguments object contains all arguments passed to the function
- the arguments object is not a real array, while rest parameters are Array instances, meaning methods like sort, map, forEach or pop can be applied on it directly
- true array unlike the argument psuedo array

```
function multiply(multiplier, ...theArgs) {
  return theArgs.map(function (element) {
    return multiplier * element;
  });
}
```

SPREAD

Expand array params like Func.apply

```
function sum(x,y,z) {
  return x + y + z;
}
total(1, 2, 3);
//before
total.apply(null, [1,2,3]);
//now
total(...[1,2,3]);
function hello({name, age}) {
  console.log(name, age);
}
foo({name: 'Westin', age: 25});
```

CLASSES

- just some syntactic sugar for prototype
- we will have supers and constructors

```
class TodoModel {
    constructor(storage) {
        this.storage = storage;
    }
    create(title) {
        title = title || '';
        var newItem = {
            title: title.trim(),
            completed: false
        };
        return this.storage.save(newItem);
    }
}
export default TodoModel;
```

```
class EnhancedTodoModel extends TodoModel {
    constructor(storage) {
        this.storage = storage;
    }
    save(item) {
        alert('Saving a new task');
        super.save(item);
    }
}
```

MODULES

```
import name from "module-name";
import { member } from "module-name";
import {member as alias } from "module-name";
import { member1 , member2 } from "module-name";
import { member1 , member2 as alias2 , [...] } from "module-name";
import name , { member [ , [...] ] } from "module-name";
import "module-name" as name;
//export syntax
Example 1:
export name1, name2, ..., nameN;
Example 2:
export *;
Example 3:
export default function() {...}
```

CAN HAVE BOTH NAMED AND DEFAULT EXPORTS

Default is really just another named export Default are favored however

```
//------ underscore.js ------
export default function (obj) {
    ...
};
export function each(obj, iterator, context) {
    ...
}
export { each as forEach };
//------ main.js ------
import _, { each } from 'underscore';
```