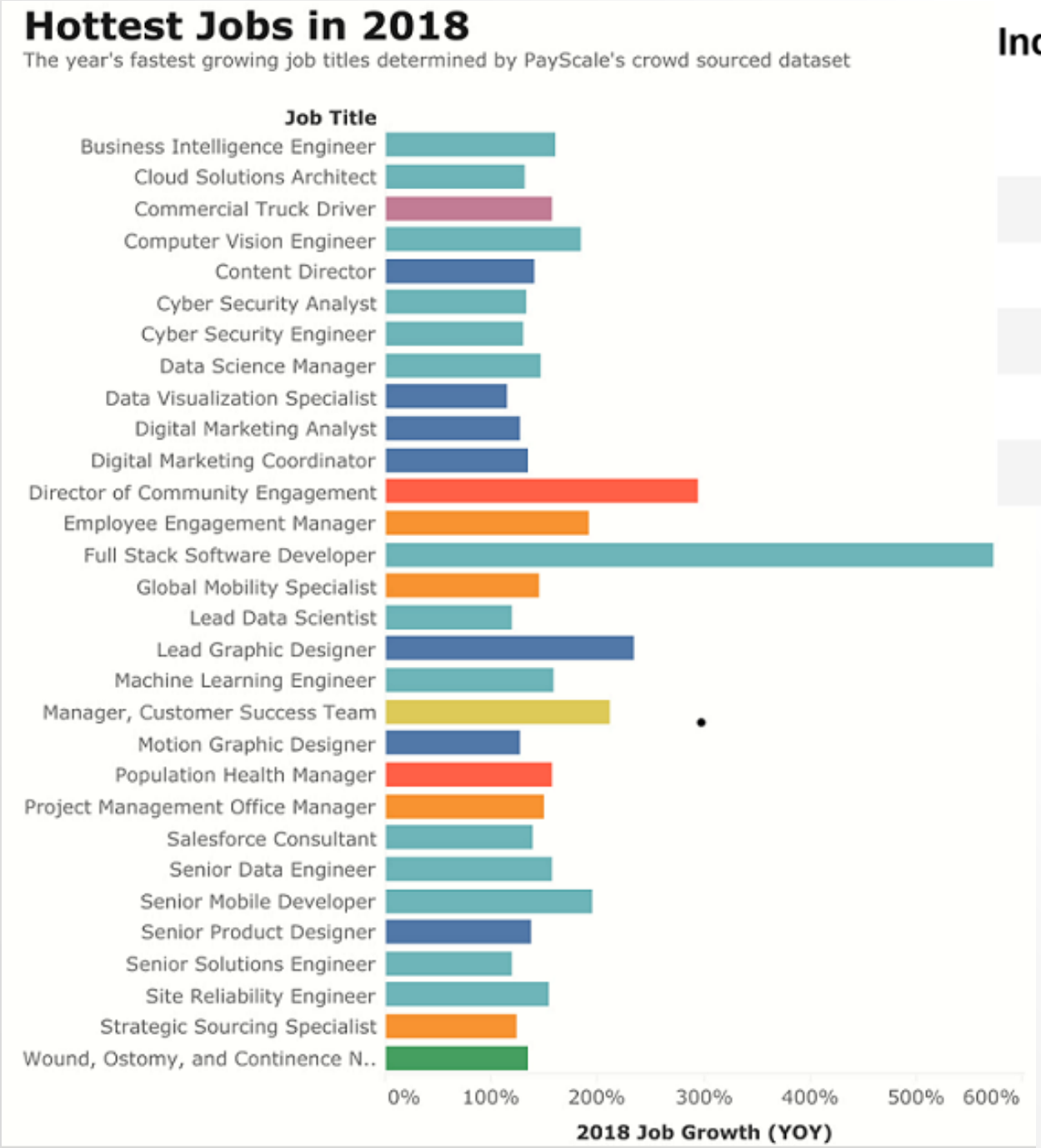
Welcome to JS/React Training!

PHASE I: ES6

Dr. Peiwei Mi
pmi@ustsv.org
408-892-6243

TRAINING OVERVIEW

- Right Timing and Proven Record
- Instructor
- Content and Schedule
- Class Materials



Indeed's Best Jobs of 2020

Rank	Job title	Job title's # of postings per 1 million total jobs, 2019	Average base salary	% growth in # of postings, 2016-2019
1	Software Architect	1,424	$119,715	18.64%
2	Full Stack Developer	893	$94,164	161.98%
3	Real Estate Agent	675	$90,439	157.08%
4	Dentist	674	$184,586	31.69%
5	Development Operations Engineer	635	$108,761	69.72%

Proven Record:

Almost 100% graduates got job from 2017 to 2023

EXPERIENCED INSTRUCTOR

- PhD, CS from USC
- 20+ years software experiences in Silicon Valley
- 8+ years in teaching this training
- Over 1000+ graduates

CONTENT

Pre-requisite
- HTML
- CSS
- Basic CS Knowledge

- JavaScript
  - Core language
  - ES6

- Back End
  - Node.js
  - Express.js
  - REST API
  - Database / MongoDB

- Front End
  - React.js
  - Redux.js
- Web App Architecture and Best Practice

- Projects
  - User List
  - Army Registry
  - Realtime Chat

- Job Search Training
  - Interview Process
  - Resume
  - Questions
  - Mock
  - Ongoing Support

SCHEDULE

- Phase I – JavaScript and ES6
  - Lecture: M/W/F 3:30pm to 5:30pm Pacific
  - Duration: 7 classes
- Phase II – Backend
  - Lecture: M/W/F 3:30pm to 5:30pm Pacific
  - Duration: 3 classes
- Phase III – Frontend, React & Redux
  - Lecture: M/W/F 3:30pm to 5:30pm Pacific
  - Duration: 8 classes
- Phase IV – Projects and Job Search Training
  - Lecture: Tue/Thu, 3:30pm to 5:30pm
  - Duration: about 6 classes depends on progress

CLASS MATERIALS

- in Google Docs (will be shared at beginning of each phase)

- Lecture Presentations
- Sample Code
- Homework
- eBooks on each subject
- Project descriptions
- Homework Submission:
  - pmi@ustsv.org

My Drive > JSTraining > JavaScript

Name	Owner	Last modified
Books	me	Oct 22, 2018 me
HackerRank	me	Oct 22, 2018 me
TA	Hao Ai	Feb 2, 2019 Hao Ai
U1-basic	me	Oct 22, 2018 me
U2-scope-and-function	me	Oct 22, 2018 me
U3-function	me	Oct 22, 2018 me
U4-oop	me	Oct 22, 2018 me
U5-regex	me	Oct 22, 2018 me
U6-async-and-exception	me	Oct 22, 2018 me
U7-async-and-module	me	Oct 22, 2018 me

JAVASCRIPT BASICS

UNIT I - BASICS

- What is JavaScript
- History and Versions
  - ES5, ES2015/ES6, TypeScript, JSX
- Development Tools and Installation
  - VS Code and NodeJS
- First Application (console.log)
- Lexical Structure
- Variables and Data Types
- Operators and Expressions
- Statements

WHAT IS JAVASCRIPT?

- JavaScript is an interpreted (or scripting) programming language
  - primarily used for building web applications
  - "just-in-time compiling"
- Used to be considered 'Client Side' technology
  - Embedded in your HTML page
  - Interpreted by the Web browser
- Now, it is also widely used on 'Server Side'
  - Via NodeJS
  - With a large ecosystem

Our focus in next 3 weeks is 'Server-Side JavaScript Programming"

HISTORY

- Beginnings at Netscape
  - originally developed by Netscape in 1995, used to be called LiveScript
- Adoption by Microsoft
  - Microsoft Windows script technologies including VBScript and Jscript were released in 1996.
- Trademark – by Oracle
- ECMAScript (ES) is the name of the standard of JavaScript from 1996
  - By ECMA International
  - ES5 in 2009 and ES6 (ES2015) in 2015
  - ES2016, ES2017, ES2018, ES2019, ES2020, ES2021, ES2022, ES2023 (ES14, 5/10), …

HOW TO RUN JAVASCRIPT?

- JavaScript can run on any browser, any host, and any OS
  - Client side in Browsers
  - Server side with nodeJS (Google V8)
- During this training, we use Visual Studio Code
  - Edit source code files
  - Execute and debug with nodeJS

TOOL INSTALLATION

- Install Visual Studio Code (VSC)
- Install nodeJS
- Create a new project (folder) and open it in VSC
- Setup to run nodeJS in ./.vscode/launch.json

```
{"version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Run JS",
            "program": "${file}"
        }
    ]}
```

FIRST APPLICATION

1-helloworld.js – ES5

2-helloworld.js – ES6

```
console.log("Hello World!");
```

1. Execute it in a Browser
2. Execute it in nodeJS – interactive
3. Execute it in nodeJS – inside a source code file
4. Execute it in VSC
5. Execute it in VSC Debugger
   - Run a program
   - Stop it at any time
   - Run it one line at a time
   - See current values of all variables

LANGUAGE STRUCTURE

The set of elementary rules that specify how programs are written in that language

- Syntax is similar to that of Java

- Lexical / Grammar Structure
- Variables and Constants
- Data Types
- Expression
- Statement

LEXICAL / GRAMMAR STRUCTURE

- Character Set
  - Unicode
  - Case sensitive
  - Ignores whitespace, line breaker, and other characters
- Comments
  - //
  - /* …. */
- Literals / Values
  - a data value that appears directly in a program
- Identifiers / Variables
  - Just a name for a variable
  - Rules for legal identifiers
- Reserved Words
  - A small set
  - System-defined objects
- (Optional) Semicolons
  - can be omitted if the statements are written on separate lines.
  - Experienced developers typically don’t use semicolons!

3-variables.js

VARIABLE DECLARATION

- Variables should be declared before their use
  - Use it without declaration is allowed
- With one of “var”, “let”, or “const” keywords
- No ‘type’ is provided at declaration

```
        i = 1;          // use it without declaring it
var     i;              // declaration
var     i = 1;          // initialization
```

DATA TYPES

- Primary
  - Number
  - String
  - Boolean
  - null
  - undefined
  - Symbol

- Object / Complex
  - Object
  - Date
  - RegExp
  - Map and WeakMap
  - Set and WeakSet
  - Promise, Error, …

- Array
- Function

We'll introduce primary ones now

4-simpledatatypes.js

DATA TYPE - NUMBER

- No distinction between integer and floating-point
- Represented internally as floating-point values
- Can be displayed in different formats (integer, decimal, …)

```
let     count = 10;             // integer literal; count is still a double
const   blue = 0x0000ff;        // hexadecimal (hex ff = decimal 255)
const   umask = 0o0022;         // octal (octal 22 = decimal 18)
const   roomTemp = 21.5;        // decimal
const   c = 3.0e6;              // exponential (3.0 × 10^6 = 3,000,000)
const   e = -1.6e-13;           // exponential (-1.6 × 10^-19 = 0000000000000016)
const   inf = Infinity;
const   ninf = -Infinity;
const   nan = NaN;              // "not a number"
```

4-simpledatatypes.js

DATA TYPE - STRING

- A string is an ordered sequence of Unicode (16-bit) characters
  - The length of a string is a number attached to each
  - Zero-based indexing
  - Empty string (“”)
- No type for a single element of a string (Char)

```
const   dialog = 'Sam looked up, and said "hello, old friend!", as Max walked in.';
const   imperative = "Don't do that!";

const   dialog1 = "He looked up and said \"don't do that!\" to Max.";

const   dialog2 = 'He looked up and said "don\'t do that!" to Max.';
const   s = "In JavaScript, use \\ as an escape character in strings.";
```

4-simpledatatypes.js

DATA TYPE - BOOLEAN

- Booleans are value types that have only two possible values:
  - true
  - false
  - (they are reserved words and constants)
- Their main usage is Boolean Expression (later)
- They can also be 0/null/undefinded/… (false)
  - and any other value(true)

```
const its_true = true;
const its_false = false;

const its_true = 12;
const its_false = 0;
```

4-simpledatatypes.js

DATA TYPE - NULL AND UNDEFINED

- Both haves one possible value
  - null
  - undefined
- Both represent something that doesn't exist
- Difference
  - null is usually available to programmers
  - undefined should be reserved for JavaScript itself, to indicate that something hasn't been given a value yet.

console.log() returned undefined before

4-simpledatatypes.js

DATA TYPE - ARRAY

- An ordered collection of values / elements
  - Each element has a numeric position in the array,
  - Zero-base index: first element in the array is element 0
- What's special:
  - Size is not fixed; add or remove elements at any time
  - Not homogeneous: individual element can be of any type

```
const   breakfast = ["coffee", "croissant"];
const   hodgepodge = [100, "paint", [200, "brush"], false];     // no type
const   emptyarray = [];                                        // empty array
const   emptyarray = new Array();                               // empty array

typeof (breakfast)                                              // ?
```

4-simpledatatypes.js

DATA TYPE – BASIC ARRAY METHODS

- push() and pop()
- splice()
- join()
- reverse()
- sort()
- concat()
- slice()
- unshift() and shift()
- toString()
- foreach()

Crucial to your success in JS programming.

There will be a dedicated time to learn array methods later

4-simpledatatypes.js

DATA TYPE - DATE

- Date represents dates and times

```
const   now = new Date();
        now;                    // example: Thu Aug 20 2019 18:31:26 GMT-0700 (Pacific Daylight Time)

const   halloweenParty = new Date(2019, 9, 31, 19, 0); // 19:00 = 7:00 pm

        halloweenParty.getFullYear();           // 2019
        halloweenParty.getMonth();              // 9
        halloweenParty.getDate();               // 31
        halloweenParty.getDay();                // 1 (Mon; 0=Sun, 1=Mon,...)
        halloweenParty.getHours();              // 19
        halloweenParty.getMinutes();            // 0
        halloweenParty.getSeconds();            // 0
        halloweenParty.getMilliseconds();       // 0
```

4-simpledatatypes.js

DATA TYPE CONVERSION

```
Number("33.3");                    // number 33.3
Number("abc");                     // NaN

const a = parseInt("16 volts", 10); // the " volts" is ignored, 16 is parsed in base 10
const b = parseInt("3a", 16);       // parse hexadecimal 3a; result is 58
const c = parseFloat("15.5 kph"); // the " kph" is ignored; parseFloat always assumes base 10

const n = 33.5;                    // 33.5 - a number
const s = n.toString();            // all JS types have .toString method
[1, true, "hello"].toString();     // "1,true,hello"

const n = 0;                       // "falsy" value
const b1 = !!n;                    // false
const b2 = Boolean(n);             // false
```

5-dynamic-typing.js

DYNAMIC TYPING

- Variables are untyped by itself
- A value (constant) of any type can be assigned to a variable,
  - and can later get a value of a different type to the same variable

A variable's type is that of its current/last value

MAIN OPERATORS

6-operators.js

- Numeric operators
  - + , -, *, /, %
- Increment operators
  - ++, --
- Assignment operators
  - = , +=, -=, *=, /=, %=
- String operator
  - + (concatenation)
    - “a” + “b”
    - “3” + 5
- Comparison operators
  - == (same value),
  - === (same value and same type)
    - ‘3’ == 3 //True
    - ‘3’ === 3 //False
  - !=, !==
  - >, <, >=, <=
- Boolean and Logic operators
  - && (logical “and”) || (logical “or”) ! (logical “not”)

7-string-template.js

STRING TEMPLATES USING ${ }

- Mixing of constant and variable concatenation is annoy

```
'Hi, ' + name + ', did you know that 5*3= ' + total + '?'
```

- To allow the ${expression} to be evaluated they must be wrapped with backticks.

```
let name = 'John';
let val = `Hi ${name}, did you know that 5*3= ${5*3}?`

console.log(val)

// "Hi John, did you know that 5*3=15?"
```

EXPRESSION

- A units of code that can be evaluated and resolve to a value
  - 4 + 5 => 9
- Types:
  - Arithmetic expression => a number, with arithmetic operators (+, -, *, /, …)
  - String expression => a string, with '+'
  - Logical (Boolean) expression => true or false, with logical operators (| |, &&,..)
  - Primary expression => constant
  - Function definition expression => function definition
  - Invocation expression => function execution
  - Array and object initializers expression
  - Left-hand-side expression
  - Property access expression
  - Object creation expression

STATEMENT

- Assignment
  - <Left-hand Side> <Assignment Operator> <Expression> (a = b + c)
  - <Assignment Operator> : = , +=, -=, *=, /=, %=, …
- Conditionals (e.g. if, switch)
  - Make the JavaScript interpreter execute or skip other statements depending on the value of an expression
- Loops (e.g. while, for)
  - Execute other statements repetitively
- Jumps (e.g. break, return, throw)
  - Cause the interpreter to jump to another part of the program

IF STATEMENTS

8-statement-if.js

```
if (condition)
{
        ...code...
}
```

```
if (condition)
{
        ...code if true...
}
else
{
        ...code if false...
}
```

```
if (condition1)
{
        ...code if 1 true...
}
else if (condition2)
{
        ...code if 2 true...
}
else
{
```

SWITCH STATEMENT

9-statement-switch.js

```
switch (variable) {
  case 1:
    // do something
    break;
  case 'a':
    // do something else
    break;
  case 3.14:
    // another code
    break;
  default:
    // something completely different
}
```

10-statement-loop.js

LOOP STATEMENTS

```
for ( var=startvalue; var<=endvalue; var=var+increment)
{
        ...code to be executed...
}
```

```
while ( condition_is_true )
{
        ...code to be executed...
}
```

```
do {
        ...code to be executed...
} while ( condition_is_true)
```

JUMP STATEMENTS

11-statement-jump.js

- **break** - Breaks out of loop early.
- **continue** - Skip to the next step in the loop.
- **return** - Exits the current function (regardless of control flow).
- **throw** - Indicates an *exception* that must be caught by an exception handler

```
while ( ... ) // or for
{
        code
        break
        code
}
```

```
while ( ... ) // or for
{
        code
        continue
        code

}
```

HOMEWORK

- #1 – JavaScript tutorial at w3schools.com, also for html, css
- #2 – a lot of small exercises for learning JS grammar
  - Not required
  - Keep doing it until you are comfortable with JS grammar
- #3 – JS expression
  - The key is to know why since the answer is available by running it
  - Aaron may ask you to give your 'why'
- #4 – A little JS gambling game
  - Not required, but to understand your current JS skill
  - Can be either easiest way or complicated way
- #5 – Concepts you learned today and should memorize for interview