TERRALOGIC

Frontend Training

July, 2020

CONTENTS

- Code Standard
- Frontend Skills
- CSS - SASS
- ES6 - ES7

Code Standard

- Why do we need a standard ?
  - Teamwork
- Standard for Code Editor ? Visual Studio Code, IntelliJ IDEA, Sublime, Atom …
  - indent with tab, space
- Standard for Project Structure?
  - Folder name: lowercase
  - React component Folder: capitalize

Code Standard

- Code Standard
  - HTML: base on bootstrap lib
    - class name:
    - DOM name: text
    - React component name: <ComponentText />
  - Javascript
  - ES6 (Javascript): eslint - https://eslint.org/docs/rules
    - ex: variable: camel case
  - Type script: tslint - https://palantir.github.io/tslint/rules
  - SCSS: stylelint - https://stylelint.io/user-guide/rules/list

Frontend Skills

- HTML - Css:
  - Bootstrap:
    - https://getbootstrap.com/docs/4.5/getting-started/introduction
  - Selector:
    - https://www.w3schools.com/cssref/css_selectors.asp
  - Animation:
    - https://www.w3schools.com/css/css3_animations.asp

Frontend Skills

- Javascript:
  - Core:
    - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects
      - Array, Object, Date, JSON...
      - Promise ...
  - API:
    - Ajax: https://www.w3schools.com/js/js_ajax_intro.asp
    - ES6:
      - request npm: https://www.npmjs.com/package/request
      - axios npm: https://www.npmjs.com/package/axios

Frontend Skills

- Javascript:
  - API
    - Postman (tool):
      - https://www.postman.com
  - RegExp:
    - https://www.w3schools.com/jsref/jsref_obj_regexp.asp
  - Lodash:
    - https://lodash.com/docs/4.17.15
      - (vs underscore.js)

CSS - SASS

- Sass is an extension of CSS that adds power and elegance to the basic language.
- Types:
  - SCSS
    - https://sass-lang.com/documentation/style-rules
  - LESS
    - http://lesscss.org/usage/
  - PostCSS
    - https://postcss.org

SASS - Variables

```
// Initialize a global variable at root level.
$variable: 'initial value';
// Create a mixin that overrides that global variable.
@mixin global-variable-overriding {
    $variable: 'mixin value' !global;
}

.local-scope::before {
    $variable: 'local value';
    @include global-variable-overriding;
    content: $variable;
}
```

SASS - Multiple Variables Or Maps

```
/// Z-indexes map, gathering all Z layers of the application
/// @access private
/// @type Map
/// @prop {String} key - Layer's name
/// @prop {Number} value - Z value mapped to the key
$z-indexes: (
        'modal': 5000,
         'dropdown': 4000,
        'default': 1,
        'below': -1,
);
```

```
/// Get a z-index value from a layer name
/// @access public
/// @param {String} $layer - Layer's name
/// @return {Number}
/// @require $z-indexes
@function z($layer) {
        @return map-get($z-indexes, $layer);
}
```

SASS - Mixin

- Mixins are one of the most used features from the whole Sass language.
- They are the key to reusability and DRY components

```
/// Helper to clear inner floats
/// @author Nicolas Gallagher
@mixin clearfix {
    &::after {
        content: '';
        display: table;
        clear: both;
    }
}
```

```
/// Helper to size an element
/// @author Hugo Giraudel
/// @param {Length} $width
/// @param {Length} $height

@mixin size($width, $height:
$width) {
    width: $width;
    height: $height;
}
```

SASS - Conditional Statements

```
@if $support-legacy {
      // …
} @else {
      // …
}
```

```
@if not index($list, $item) {
      // …
}
@if $value == 42 {
  // …
}
```

SASS - Loops: For & Each

- **For:**

```
@for $i from 1 through 10 {
    .foo:nth-of-type(#{$i}) {
        border-color: hsl($i * 36, 50%, 50%);
    }
}
```

- **Each:**

```
@each $key, $value in $map {
    .section-#{$key} {
        background-color: $value;
    }
}
```

SASS - Warnings And Errors

@debug
@warn
@error

```
@function mq-px2em($px, $base-font-size: $mq-base-font-size) {
  @if unitless($px) {
    @warn 'Assuming #{$px} to be in pixels, attempting to convert it into
pixels.';
    @return mq-px2em($px + 0px);
  } @else if unit($px) == em {
    @return $px;
  }

  @return ($px / $base-font-size) * 1em;
}
```

What Is ES6 / ES7?

- ECMAScript 6 is also known as ES6 and ECMAScript 2015. Some people call it JavaScript 6
- ES7 - ECMAScript 7 (ES2016)
- Features :
  - Default Parameters
  - Template Literals
  - Multi-line Strings
  - Destructuring Assignment
  - Enhanced Object Literals
  - Arrow Functions
  - Promises
  - Block-Scoped Constructs Let and Const
  - Classes
  - Modules
  - Array.prototype.includes
  - Exponentiation Operator

ES6 - Arrow function (Lambda)

Functions are also called as Lambda functions.

```
([param1, parma2,…param n] )=>statement;
```

Example – Expression

```
const foo = (x)=> 10+x
```

ES6 - Arrow function (Lambda)

Statement

```
hello = () => {
 return "Hello World!";
} // ES6

hello = function () {
   return "Hello World!";
} // JS
```

Lexical this

```
this.nums.forEach((v) => {
   if (v % 5 === 0)
       this.fives.push(v)
}) // ES6
this.nums.forEach(function
(v) {
   if (v % 5 === 0)
       this.fives.push(v);
}, this); // JS
```

ES6 - Class (Definition)

OOP-style

```
class Shape {
    constructor (id, x, y)
{
        this.id = id
        this.move(x, y)
    }
    move (x, y) {
        this.x = x
        this.y = y
    }
}
```

Boilerplate-free

```
var Shape = function (id, x, y) {
    this.id = id;
    this.move(x, y);
};
Shape.prototype.move = function
(x, y) {
    this.x = x;
    this.y = y;
};
```

ES6 - Class (Inheritance)

```
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        this.radius = radius
    }
}
```

ES6 - Class (Getter/Setter)

```
class Rectangle {
    constructor (width, height) {
        this._width  = width
        this._height = height
    }
    set width  (width)  { this._width = width                 }
    get width  ()       { return this._width                  }
    set height (height) { this._height = height               }
    get height ()       { return this._height                 }
    get area   ()       { return this._width * this._height }
}
var r = new Rectangle(50, 20)
r.area === 1000
```

ES6 - Shorthand

Shorthand Property and Method name

```
function formatMessage (name, id, avatar) {
    return {
      name,
      id,
      avatar,
      timestamp: Date.now(),
      save () {
        //save message
      }
    }
}
```

ES6 - Shorthand (1)

Destructuring Assignment

```
function getASTNode({op, lhs, rhs, hhghg}) {
    return {
        op,
        lhs,
        rhs,
    }
}
var { op, lhs, rhs } = getASTNode()
// op = 1; lhs = 2; rhs = 3
```

ES6 - let and const

const is a signal that the identifier won't be reassigned

```
const PI = 3.141593
```

Note: Some case can change the value of const

ES6 - let and const

let is a signal that the variable may be reassigned, such as a counter in a loop, or a value swap in an algorithm. It also signals that the variable will be used only in the block it's defined in, which is not always the entire containing function.

```
let callbacks = []
for (let i = 0; i <= 2; i++) {
    callbacks[i] = function () { return i * 2 }
}
```

ES6 - Destructuring

Object And Array Matching, Default Values

```
var obj = { a: 1 }

var list = [ 1 ]

var { a, b = {}, c = [] } = obj

var [ x, y = 2 ] = list
```

ES6 - Destructuring (1)

Object Matching - Deep Matching

```
function getASTNode() {
    return {
        op:1,
        lhs:{ op:2 },
        rhs:3,
    }
}
var { op: a, lhs: { op: b } = {}, rhs: c } =
getASTNode()
```

ES6 - Destructuring (2)

Parameter Context Matching

```
function f ([ name, val ]) {
   console.log(name, val)
}
function g ({ name: n = 12, val: v }) {
   console.log(n, v)
}
function h ({ name, val }) {
   console.log(name, val)
}
f([ "bar", 42 ])
g({ name: "foo", val:  7 })
h({ name: "bar", val: 42 })
```

ES6 - Extended Parameter Handling

Rest Parameter

```
function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, "hello", true, 7) === 9 //true
```

ES6 - Extended Parameter Handling (1)

Spread Operator

```
var params = [ "hello", true, 7 ]
var other = [ 1, 2, ...params ] // [ 1, 2, "hello",
true, 7 ]

function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, ...params) === 9

var str = "foo"
var chars = [ ...str ] // [ "f", "o", "o" ]
```

ES6 - Template Literals

String Interpolation

```
var customer = { name: "Foo" }
var card = { amount: 7, product: "Bar",
unitprice: 42 }

var message = `Hello ${customer.name},
want to buy ${card.amount} ${card.product} for
a total of ${card.amount * card.unitprice}
bucks?`
```

ES6 - Promise

Syntax

```
var promise = new Promise(function(resolve , reject) {
    // do a thing, possibly async , then..
    if(true) // condition
        resolve("stuff worked");
    else
        reject(Error("It broke"));
});
return promise; // Give this to someone
```

ES6 - Promise Combination (All)

```
let fetchPromised = (name, timeout) => {
    return new Promise((resolve, reject) => {
        setTimeout(function() {
            console.warn(name);
            resolve(name);
        }, timeout);
    })
}
Promise.all([fetchPromised("1", 500),fetchPromised("2",
500),fetchPromised("3", 500)]).then((data) => {
        let [ foo, bar, baz ] = data
        console.log(`success: foo=${foo} bar=${bar} baz=${baz}`)
    }, (err) => {
        console.log(`error: ${err}`)
})
```

ES6 - Async/Await

```
async function connect() {
    try {
        const connection = await
asyncDBconnect('http://localhost:1234'),
            session = await asyncGetSession(connection),
            user = await asyncGetUser(session),
            log = await asyncLogAccess(user);
        return log;
    } catch (e) {
        console.log('error', err);
        return null;
    }
}
// run connect (self-executing async function)
(async () => { await connect(); })();
```

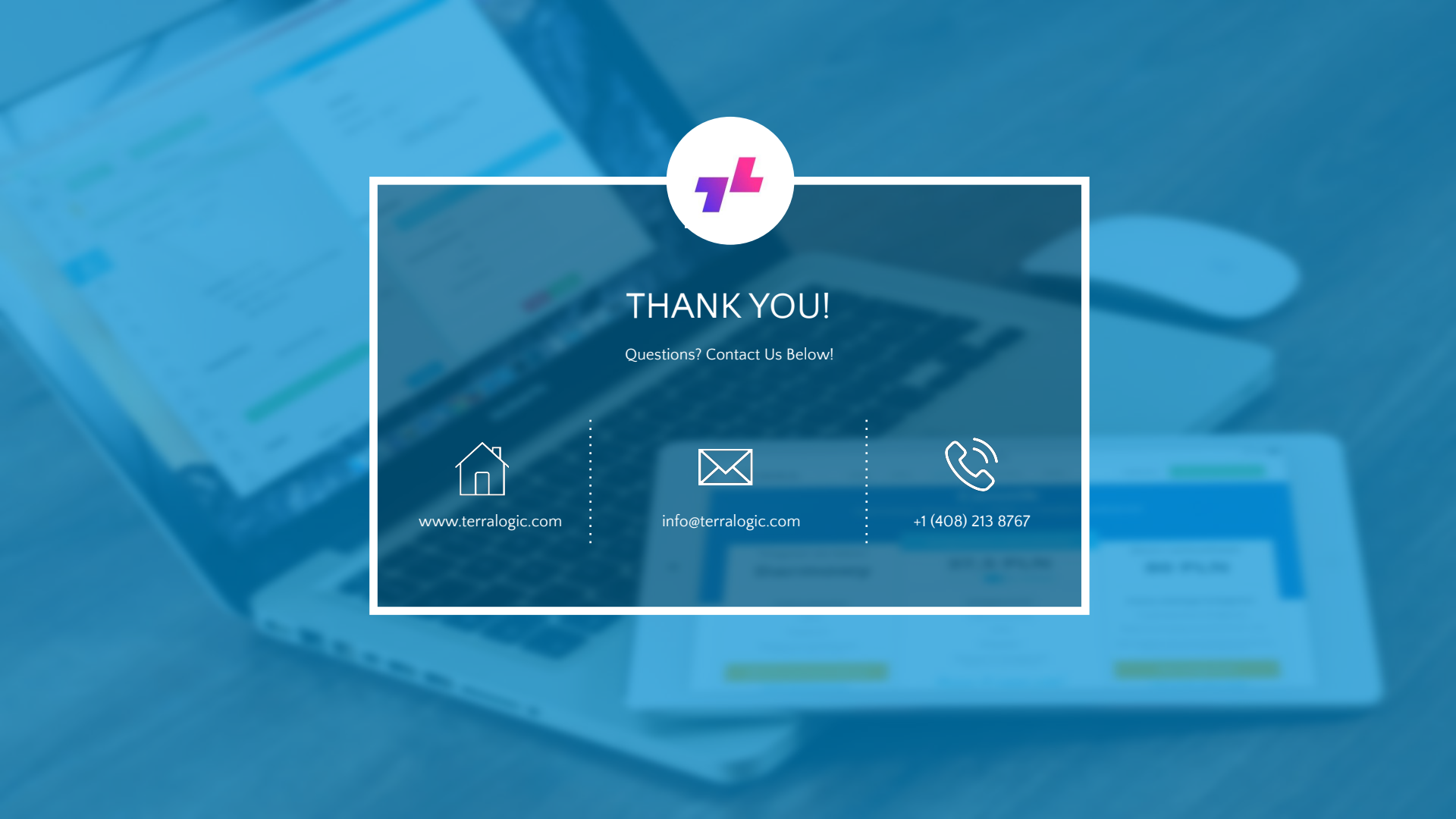
React Function & Class Components

- Class Component:

```
class Welcome extends
React.Component {
  render() {
    return
<h1>H{this.text}</h1>;
  }
}
```

- Function Component:

```
function App() {
  return (
    <h1>H{this.text}</h1>
  );
}
```


THANK YOU!
Questions? Contact Us Below!
www.terralogic.com
info@terralogic.com
+1 (408) 213 8767