AngularJS Coding Standards

CREDENCYS
Delivering WOW

Following are the guideline which we have to take in our code:

- Code MUST have a semicolon at the end of statement.
- Code MUST not have unused/unnecessary variables.
- Code MUST not have trailing whitespace.
- Line length MUST be limited to 80 characters.
- Use single quotes, unless you are writing JSON.

  For example

  Right:

```
var foo = 'bar';
```

  Wrong:

```
var foo = "bar";
```

- Control statements MUST have braces on the same line.

  For example

  Right:

```
          if (true) {
            console.log('winning');
          }
```

  Wrong:

```
if (true)
{
          console.log('winning');
      }
```

- Declare one variable per var statement, it makes it easier to reorder the lines.

  For example

  Right:

```
var keys = ['foo', 'bar'];
var values = [23, 42];
var obj = {};
```

Wrong:

```
var keys = ['foo','bar'];
      values = [23,42];
      object = {};
      key;
```

- Variables and properties should use lower camel case capitalization. They should also be descriptive. Single character variables and uncommon abbreviations should generally be avoided.

  For Example

  Right:

```
var adminUser = 'admin';
```

Wrong:

```
var admin_user = 'admin';
```

- Constants should be declared as regular variables or static class properties, using all uppercase letters.

  For Example

  Right:

```
var SECOND = 1 * 1000;
```

Wrong:

```
const SECOND = 1 * 1000;
```

- Use trailing commas and put short declarations on a single line. Only quote keys when your interpreter complains:

  For Example

  Right:

```
var a = ['hello', 'world'];
var b = {
             good: 'code',
            isGenerally: 'pretty',
};
```

Wrong:

```
var a = [
    'hello', 'world'
];
var b = {"good": 'code',
         "isGenerally": 'pretty'
       };
```

- Use triple equality operator in conditions.

  For Example
  Right:

```
var a = 0;
if (a !== '') {
     console.log('winning');
}
```

Wrong:

```
var a = 0;
if (a != '') {
     console.log('winning');
}
```

- If a condition is very lengthy then assign it to a variable or function.

  For Example
  Right:

```
var isValidPassword = password.length >= 4 &&
/^(?=.*\d).{4,}$/.test(password);
if (isValidPassword) {
    console.log('winning');
}
```

Wrong:

```
if (password.length >= 4 && /^(?=.*\d).{4,}$/.test(password)) {
    console.log('losing');
}
```

- Function length should be small. Always return function value as early as possible.

For Example

Right:

```
function isPercentage(val) {
 if (val < 0) {
   return false;
 }
 if (val > 100) {
   return false;
 }
return true;
}
```

Wrong:

```
function isPercentage(val) {
  if (val >= 0) {
      if (val < 100) {
      return true;
      } else {
      return false;
      }
  } else {
      return false;
  }
}
```

Angular 1.X

- Directory structure must be by feature.
- Use suffix for filename for controller, services, directive etc. Ex.
  - home.controller.js
  - header.directive.js
  - User.service.js
- Must use dependency and injection both in controllers, services, directives, modules etc.
- There must not be any html in your controller.
- Never access DOM element with jquery. Always use angular.element().
- All API calls must be written in services.
- Do not use watch as much as possible.

Angular 2+

- Always install packages via package manager with save attribute.
- Use angular animations only for any kind of animation.
- Use const bindings when declaring references.
- Prefer small functions as a primary means of abstraction.
    - Simplifies understanding program operations and execution along with programmer intent.
    - Maximizes reuse of code through granularity of abstractions.

```
// avoid
const newMember = new User();
newMember.setName(fullName);
newMember.setEmail(email);

const newTeam = TeamsManager.get(team);
newTeam.invite(newMember);


// avoid
function invite(team, { fullName, email }) {
  const names = fullName.split(' ');

  const newMember = {
    no: team.length
    firstName: names[0],
    lastName: names[1],
    joinedOn: new Date(),
  };

  const newTeam = team.concat([newMember]);

  return newTeam;
}


// good
function createUser(fullName, email) {
  const names = fullName.split(' ');

  return {
    firstName: names[0],
    lastName: names[1],
    joinedOn: new Date(),
  };
}

function addToTeam(team, candidate) {
  const newMember = Object.assign(
    {},
    { no: team.length },
    candidate
  );

  return team.concat([newMember]);
}

function invite(team, { name, email }) {
  const newUser = createUser(name, email);
  const newTeam = addToTeam(team, newUser);

  return newTeam;
}
```

- Use upper CamelCase for classes.

```
// avoid
class button {}


// good
class Button {}
```