#	Section	Description	Example

1	Naming Standards	Variables must be in Camel case naming standard.
		
		Constant names must be in all upper case, and words must be separated by an underscore (_).
			messageCodes: string = "";
			
			static readonly RECORDS_PER_PAGE: number = 10;

2		The class name must be in the Pascal case naming standard.	export class AdminLoginPageComponent

3		The interface name must be prefixed with the letter “I“ and follow the Pascal case naming standard.	export interface AddressCityRecordData {
			}

4	Style Rules	Forbid JSDOC comment, which duplicates typescript functionality.	/**
			  * Function Triggers when state value is changed
			  * @param {string} stateValue
			  * @returns void
			  */
			 onStateChange(stateValue: string): void {

5		All keys in an Object must be surrounded with double quotes.
		
		"quote-props": ["error", "as-needed"]
			{
			  "PARM": "/parm/view",
			  "BSTL": "/bstl/status"
			}

6		All import statements must be alphabetized and grouped	import {
			  IScreenInfo,
			  ISearchable,
			  ScreenFormComponent
			} from "@ccsesplus/uibase";
			import { MessageConstants } from "@shared/constants/error-constants";

7	Formatting Rules	Disabled requires parentheses for the arrow function and allows the use of parentheses around arrow function parameters as preferred.
		
		"arrow-parens": ["off", "always"]
			(status: statuss) => {
			  if (status[0].code.equals(this.messageConstants.IMSG0093)) {
			    this.marriageDivorce.createMarriageList([]);
			  }
			 }

8		Max line length 80 chars and Ignore long import statements, URLs, Comments, Strings, and Template Literals in the line length calculation.	

9		All files must end with a new line char. "eol-last": "error"	
10		Enforce the use of double quotes if the string contains an escape sequence.	selector: "app-add-process-workflow", templateUrl: "./add-process-workflow.component.html",
11		All statements must be finished with a semi-colon. "@typescript-eslint/semi": ["error", "always"]	const addOutcomeInDataList = this.rootFormGroup.value.processWorkflow;

12		There must be no trailing comma.
		
		"trailing-comma": “off”
			this.gridColumns = [
			{
			header: "Last Name",
			field: "lastName",
			maxlength: this.fieldSizeConstants.LASTNAME,
			(this trailing comma will be automatically removed by printer formatter)
			}

13		There must be no trailing spaces at the end of lines. "no-trailing-spaces": "error"	this.gridColumns = [{ header: "Last Name", field: "lastName", maxlength: this.fieldSizeConstants.LASTNAME },
14		There must be no irregular whitespace between IN statements. "no-irregular-whitespace": "error"	
15		To display a warning whenever using a deprecated API in the application. deprecation: "warn"	
16		There must be a single class in a file. "max-classes-per-file": "error"	export class ViewProcessWorkflowComponent extends ScreenFormComponent implements ISearchable { }
17		Disable alphabetical ordering of members in object literal syntax. "object-literal-sort-keys": "off"	

18		Forbid importing unnecessary library. "no-restricted-imports": ["error", "rxjs/Rx"]	
19		Forbid Console statement. "no-console": "error"	
20		Forbid debugger statement. "no-debugger": "error"	
21		Check duplication parameter for method body vs method signature. "no-redeclare": "error"	
22		The empty block is needed for life cycle events (ngOnInit) and the constructor. This is required to ensure the initialization of the component and guarantees that the component has already been created. "no-empty": "error"	

23		This rule flags any case statement that doesn't end with a break, return, or throw statement unless another case statement follows it.
		"no-fallthrough": "error"
			switch (event.eventData.name) {
			    case "Search":
			this.queryBuilderPanel.toggle(event.originalEvent);
			      break;
			    case "Notification":
			this.loadSystemAlert(true);
			      break;
			  }

24		Variables must be declared before using it.	addressNormalized: boolean = false;
			addressKeyFilter: string;

25		Enforces the use of curly braces for single and multi-line control flow statements.
		
		curly: ["error", "multi-line"]
			if (this.isValid) {
			
			this.buttonList[CcsesplusConstants.NUMBER_0].disabled = true;
			    if (this.processGrid.filterFormGroup) {
			      this.processGrid.reset();
			    }
			    this.getProcessDetailsGridData();
			  } else {
			    event.preventDefault();
			  }

26	TypeScript Coding Standards	Enforce strict type exceptions to common code for any explicit use of any type.
		
		"@typescript-eslint/no-explicit-any": [
		    "warn",
		    {
		      ignoreRestArgs: true
		    }
		  ]	

27		Default all members are public. We have to define private and protected.
		
		"@typescript-eslint/explicit-member-accessibility": [
		    "error",
		    {
		      accessibility: "no-public"
		    }
		  ]
			constructor(
			private renderer: Renderer2,
			 themingService: ThemingService
			)
			{
			super();
			}

28		Member reordering in a class.
		
		"@typescript-eslint/member-ordering": [
		    "error",
		    {
		      default: {
		        memberTypes: [
		          "static-field",
		          "instance-field",
		          "static-method",
		          "instance-method"
		        ]
		      }
		    }
		  ]	

29		Disallows reassigning parameters. "no-parameter-reassignment": true, "typedef": [true,"call-signature","parameter","member-variable-declaration"]	

30		0,1,2 like numbers are not allowed must be in constants, or valid variable names must be given. "no-magic-numbers": "error"	

31		TypeDefinition is a must for Parameters and member variable declaration.
		
		"@typescript-eslint/typedef": [
		"error",
		{
		parameter: true, memberVariableDeclaration: true
		}
		]
			taskGroupBy: string = ProcConstants.taskGroupBy;
			processSubscription: Subscription;

32		Disallows explicit type annotations for variables, parameters, and class properties when the TypeScript compiler can easily infer their types.
		
		"@typescript-eslint/no-inferrable-types": [
		"error",
		{
		ignoreParameters: true,
		ignoreProperties: true
		}
		]
			additionalAddress: string = "";

33		Enforce strict type. "@typescript-eslint/no-non-null-assertion": "error"	
34		Disable require statements and downloads. Only have to use it to import. "@typescript-eslint/no-var-requires": "error"	
35		As shown in the example, we will prefix the “on” word for the events, so disabling this rule will allow using the “on” word. "@angular-eslint/no-output-on-prefix": "off"	onRowDblClick, onLinkclick, etc.
36		This is required for Library Project "@angular-eslint/use-lifecycle-interface": "error"	

37		@pipe must implement PipeTransform interface. "@angular-eslint/use-pipe-transform-interface": "error"	@Pipe({ name: "titleCase" }) export class TitleCasePipe implements PipeTransform {}
38		For the library component, it must be “lib” and for the application component, it must be “app”. "@angular-eslint/component-class-suffix": "error"	<lib-input formControlName="outcomeDisplayNumb" Class="p-col-2" label="Outcome Display Order" maxlength="4" [keyFilter]="regex.numeric" ></lib-input> <app-document-header [rowData]="processData"></app-document-header>
39	Module Standards	Project Structure Organize the application in three logical groups i) Core ii) Shared iii) Screens Internal structure will be based on the functionality	
40		lazy loading: Load the modules on demand (lazy loading) after user navigation; never directly import lazy-loaded folders.	[lazy]="true"
41		Use the LIFT application design principle. i) Locate: Make locating code intuitive, simple, and fast ii) Identify: Name the files so that they contain the file; use a descriptive and meaningful name. iii) Flat: Maintain a flat folder structure as much as possible iv) T-DRY: Don't repeat the functionality (Try to be DRY)	

42		Create a NgModule for all distinct features in an application; place the module in the same-named folder as the feature area; name the feature (screen) module file reflecting the name of the feature (screen) area and folder.	“app/screens/admn” folder defines AdminLoginModule in admin-login.module.ts file
43	File Header Comments	File Header Format /* * Copyright (c) 2023 - 2028, all rights reserved. * Office of Child Support * Dept. of Social Services * State of Connecticut * * File: <<filename>> * Project: <<projectname>> * File Created: <<date>> * Author: <<author>> * Description: <<comments>> * * History * ------- * Last Modified: <<date updated>> * Modified By: <<author>> * Update Comments: <<update comments>> * ----- */	/** * Copyright (c) 2023 - 2028, all rights reserved. * Office of Child Support * Dept. of Social Services * State of Connecticut * * File: "address-information.component.ts" * Project: "CCSESPLUS" * File Created: "" * Author: "Protech Solutions Inc." * Description: "" * * History * ------- * Last Modified: "" * Modified By: "Protech Solutions Inc." * Update Comments: "" * ----- */

44		Separate each section with a blank line.	}
			/**
			  * Get Sticky Data
			  * @returns void
			  */
			 getStickyData(): void {
			   // check if local sticky is defined
			   if (this.procService.stickyData) {
			
			this.setFormValue(this.procService.stickyData);
			     this.search(event);
			     this.procService.stickyData = null;
			   }
			 }

45	Import Statements Standards	Group all imports in three groups i) Core imports ii) Third-Party/open-source imports iii) Application imports	
46		Do not use wild card (*) imports	
47		Separate each group with a blank line	
48		List import alphabetized by module name	
49	Component Definition Standards	Define a single component, service etc. (Single Responsibility Principle) in a separate file	export class LoginComponent (File: login.component.ts)
50		Limit file length up to 400 lines (excluding comments and change history)	
51		Use consistent names for all components named after what they represent.	
52		Use upper camel case for class names (first character of name is upper case).	export class AddDocumentsAssociatedComponent
53		Match the name of the symbol to the name of the file.	
54		Append the symbol name with the conventional suffix (such as Component, Directive, Module, Pipe, or Service) for a thing of that type.	

55		Name the file with the conventional suffix (such as .component.ts, .directive.ts, .module.ts, .pipe.ts, or .service.ts) for a file of that type.	proc.module.ts proc-routing.module.ts proc.service.ts proc-add-document-in-data.model.ts proc-constants.ts
56	Constant and Variable Definition	Declare variables as const if the value will not change during the application lifetime	const message = "Hello World!";

57		Name the variables in lower camel case (the first character of the name is lowercase)	processCodeInputData: ProcessDetailsInData = new ProcessDetailsInData();
			filterItems: { [key: string]: string } = {};

58		Initialize all variables with the appropriate value	modifiedDataIndex: number[] = [];
			nextTaskList: Observable<OutData<ProcTaskListRecordData[]>>;
			nextTaskDataList: ProcTaskListRecordData[] = [];

59		Declare each variable on its own line	
60	Functions format	Declare methods in the following order 1. Constructors 2. Methods invoked by constructors 3. Static methods 4. Interface method implementations	
61		Perform a single task in each function	
62		Recommend to keep the function length to 50 lines (one-page rule)	
63		Refactor lengthy functions	
64		Function Header Format /** *Creates an instance of LoginComponent. * @author <<author>> * @date <<date created>> * @param <<ParameterType>> parameterName * @returns <<returnType>> * @memberof <<Component>> */	/** * Function Triggers when state value is changed * @author "PROTECH" * @date 12/27/2024 * @param {string} stateValue * @returns void */

65	Form Data Validation	Validate all user inputs, and form data for accuracy, completeness, and integrity	
66		Ensure data matches its intended usage	all digits for integer
67		Load all relevant reference values during the application/module loading event.	
68		When an expression is more than 80 characters/ does not fit on a single line, break it according to the general principles 1. Break after a comma 2. Break before an operator	Auto Formatter Applied
69		Use Configurable parameters instead of hard coding constant values	Screenconstants
70		Use two spaces for indentation; do not use tabs	Auto Format
71	HTML	Document Type: Declare the correct document type as the first line	<!DOCTYPE html>
72	HTML	Encoding: Use UTF-8 encoding so that the page will be viewed correctly	<meta charset='utf-8'>.
73	HTML	Title: Use a meaningful title for each document	<title>CCSES+</title>
74	HTML	Markup tags: Use lowercase markup tags. Always use ALT tags for ADA compliance.	<div id='participant'> <img src="case/partc.jpg" alt="Jane Doe" /> <p>Jane Doe</p> </div>
75	HTML	Meta tags: Use descriptive, useful meta tags for CCSES+ public-facing portals.	<meta name="description" content="Connecticut Child Support Enforcement System"> <meta name="keywords" content="Connecticut, welfare, child support, ccses" >
76		Special Characters 1. Do not use tab characters for indentation 2. Eliminate all trailing white spaces	
77	HTML File Naming Convention	3. Use meaningful lower-case names for all HTML files. 4. Start the file name with a letter 5. Do not use spaces or special characters in the filename 6. Use alphanumeric for the file name (a-z,0-9, -, _) 7. Use .html extensions for HTML files 8. Keep file names short	
78		Single Line: Add a comment text and opening and closing angle brackets on a single line	<!-- This is a single-line comment. -->

79		Multi-Line: Start and end line angle brackets must be on separate lines.	<!-- This is a multi-line comment. .... -->
80		Separate HTML content from CSS (presentation) and scripts	
81		Use double quotation marks for attribute values	<link rel="stylesheet" href="case_initiation.css">
82	CSS File	Organization: Organize all global styles in a separate folder	
83		Create one file for each global component and organize style files in subfolders.	
84	CSS File Header	Add a comment on the top of each file describing the purpose of the file, author, organization, module, and date. Separate each section with a blank line.	/* * – Document: CCSES+ Theme * – Version: 0.0.1 * – Organization: Connecticut – Division of Welfare and Supportive Services * – Date: 04/03/2019 * – Author: “PROTECH” */

85	CSS Syntax and Formatting	Use 2 spaces for indentation; do not use tabs	.alert-info
			{
			background-color: blue !important;
			color: $black-color;
			font-weight: bold;
			font-size: medium;
			height: 2pc;
			vertical-align: middle;
			}

86		Maintain an 80-character line width.	
87		Keep each selector, property, attribute, etc., on a new line.	
88		Use Allman brace style.	
89		Separate each style by one (or more – where applicable) blank lines.	