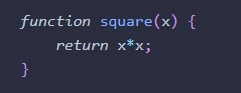
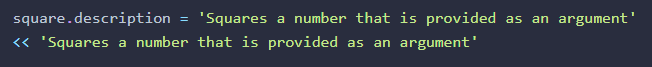
**Further Functions**

* The fact that functions are first-class objects means they can have properties and methods themselves.
* Example property **length**

**Text

Description automatically generated**

**Custom Properties**

* There is nothing to stop you adding your own properties to functions in the same way that you can add properties to any object in JavaScript. For example, you could add a description property to a function that describes what it does:

**Safe Use of Strict Mode**

* This ensures that only your code inside the IIFE is forced to use strict mode.

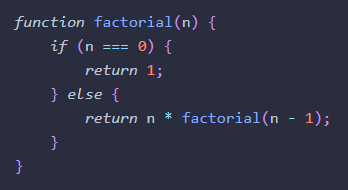
Text

Description automatically generated

**Init-Time Branching**

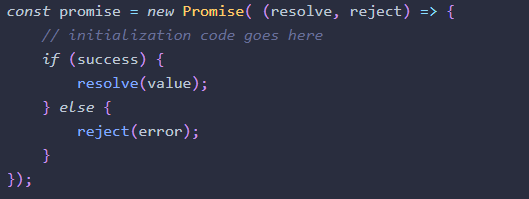
* This enables the functions to work more effectively in the browser and avoid checking for features every time they’re invoked.

**Recursive Functions**

* A recursive function is one that invokes itself until a certain condition is met. It’s a useful tool to use when iterative processes are involved. A common example is a function that calculates the [factorial](http://en.wikipedia.org/wiki/Factorial) of a number:

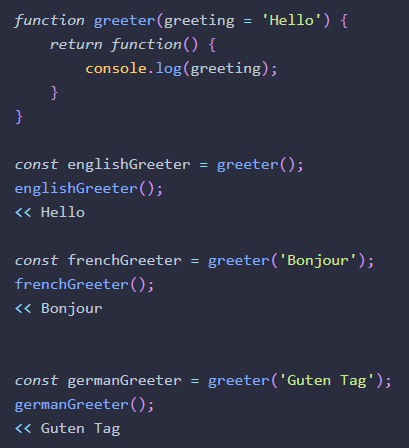
**Promises**

* A promise represents the future result of an asynchronous operation. Promises don't do anything that can't already be achieved using callbacks, but they help simplify the process, and avoid the convoluted code that can result from using multiple callbacks.



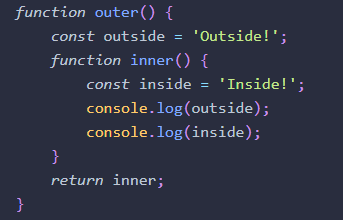
**Functions That Return Functions**

* We've just seen that functions can accept another function as an argument (a callback), but they can also return a function.



**Closure**

* A **closure** is formed when the inner function is returned by the outer function, maintaining access to any variables declared inside the enclosing function.



Text

Description automatically generated

**Pure Functions**

* A key aspect of functional programming is its use of pure functions. A pure function is a function that adheres to the following rules:

**1)** The return value of a pure function should only depend on the values provided as arguments. It doesn't rely on values from somewhere else in the program.

**2)** There are no side-effects. A pure function doesn't change any values or data elsewhere in the program. It only makes non-destructive data transformations and returns new values, rather than altering any of the underlying data.

**3)** Referential transparency. Given the same arguments, a pure function will always return the same result.

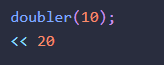
* By only performing a single task, pure functions are more flexible, they also help to make your code **more modular**

**Higher-Order Functions**

* Higher-order functions are functions that accept another function as an argument, or return another function as a result, or both.
* **Closures are used extensively in higher-order functions** as they allow us to create a generic function that can be used to then return more specific functions based on its arguments.

Text

Description automatically generated



Graphical user interface, text, application

Description automatically generated

**CSS3 Transforms and Transitions**

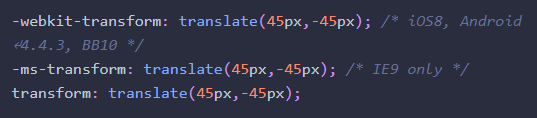
**Transforms**

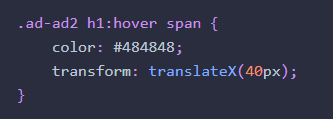
* The CSS3 transform property lets you lets you translate, rotate, scale, and/or skew any element on the page.

**Translation**

* Translation functions allow you to move elements left, right, up, or down. These functions are similar to the behavior of position: relative; when declaring top and left, moving elements up and down or left and right along the x and y axes.

**Note: Transforms and Older Browsers**





Transforms don’t work on inline elements. But that’s easy enough to fix. We’ll just add **display: inline-block**; to our span:

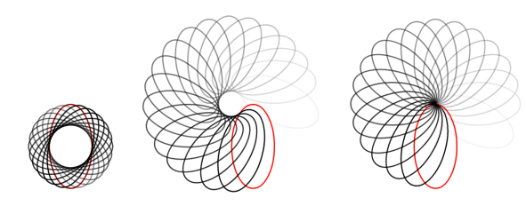
Text

Description automatically generated

**Changing the Origin of the Transform**

As we hinted at earlier, you can control the origin from which your transforms are applied. This is done using the **transform-origin** property

 If you gave your circle or ellipse a **transform-origin of 10% 10%** or **top center**, you would notice the rotation

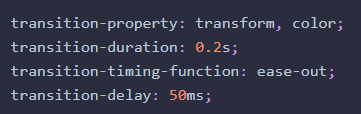


**The transition-timing-function Property**

The most common timing functions include the key terms **ease**, **linear**, **ease-in**, **ease-out**, or **ease-in-out.**

**Multiple Transitions**

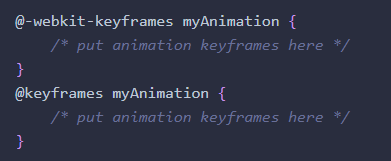
The **transition** properties allow for multiple transitions in one call. For example, if we want to change the color at the same time as changing the rotation and size, we can.



**Keyframes**

To create an animation, use the **@keyframes** rule for IE10+ and FF16+. Include **@-webkit-keyframes** for all WebKit implementations followed by a name of your choosing, which will serve as the identifier for the animation. Then, you can specify your keyframes.

For an animation called **myAnimation**, the **@keyframes** rule would look like this:



Text

Description automatically generated