Week 4 Lesson 7 Readings

***Chapter 11: Further Functions***

.length = a property that returns the number of parameters the function has.

const outside = 'In the global scope';

function fn() {

const inside = 'In the function scope';

}

outside

<< 'In the global scope'

inside

<< ReferenceError: inside is not defined

function outer() {

const outside = 'Outside!';

function inner() {

const inside = 'Inside!';

console.log(outside);

console.log(inside);

}

console.log(outside);

inner();

}

function outer() {

const outside = 'Outside!';

function inner() {

const inside = 'Inside!';

console.log(outside);

console.log(inside);

}

return inner;

}

function closure() {

const a = 1.8;

const b = 32;

return c => c \* a + b;

}

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.

In order to follow these rules, any pure function must have:

At least one argument; otherwise the return value must depend on something other than the arguments of the function, breaking the first rule

A return value; otherwise there’s no point in the function (unless it has changed something else in the program – in which case, it’s broken the 'no side-effects' rule).

***Chapter 8: Transforms and Transitions***

Transform = lets you translate, rotate, scale, and/or skew any element on the page.

--

position: relative; when declaring top and left, moving elements up and down or left and right along the x and y axes.

position: relative, which allows you to position an element either against its current position or against a parent or other ancestor, a translated element can only be moved relative to its current position.

--

The translate(x,y) function moves an element x from the left, and y from the top:

transform: translate(45px, -45px);

--

f you only want to move an element vertically or horizontally, you can use the translatex or translatey functions respectively.

To move 45px to the right along the x axis

transform: translateX(45px);

To move up along the y axis by 30px

transform: translateY(-30px);

--

The scale(x,y) function scales an element by the defined factors horizontally then vertically. If only one value is provided, it will be used for both the x and y values, growing or shrinking your element or pseudo-element while maintaining the original aspect ratio.

transform: scale(1.5, 0.25);

--

While CSS3 transforms are unsupported in IE before version 9, you can mimic these effects with other CSS properties, including filters. To mimic translation use position: relative;, and top and left values:

Copy

.translate {

position: relative;

top: 200px;

left: 200px;

}

--

You can also scale an element by altering its width and height, or changing the font-size. Remember, though, that while transformed elements still take up the space they did before being scaled, altering a width, height or font-size alters the space allocated for the element and can affect the layout.

You can use filters to rotate an element in older versions of Internet Explorer, but it’s ugly and performs poorly:

.rotate {

transform: rotate(15deg);

filter: progid:DXImageTransform.Microsoft.Matrix(

sizingMethod='auto expand', M11=0.9659258262890683,

M12=-0.25881904510252074, M21=0.25881904510252074,

M22=0.9659258262890683);

-ms-filter: "progid:DXImageTransform.Microsoft.Matrix(

M11=0.9659258262890683, M12=-0.25881904510252074,

M21=0.25881904510252074, M22=0.9659258262890683,

sizingMethod='auto expand')";

zoom: 1;

}