Object methods, “this”

Objects can be given properties ie:   
 let user = {  
 name: “John”,  
 age: 30 };

Can be given methods ie:  
 user.sayHi = function () {  
 alert(“Hello!”); };

For this:  
 let user = {  
 name: “John”,  
 age: 30,  
   
 sayHi() {  
 alert(this.name); } };  
 user.sayHi();

In the example above this refers to the “user” object. And this.name would be “John”  
 object can also be accessed by using: alert(user.name); both work

“this” is not bound. In the sayHI function we can use it like this:  
 let user = { name: “john” };  
 let admin = { name: “Admin” };

User.f = sayHi;  
 admin.f = sayHi;

User.f(); // prints out John  
 admin.f(); //prints out admin

This works abstractly to call the name of whatever the object calling it is.

Arrow functions don’t have a “this”, the arrow function has to be given a name, and called within a function if you want “this” to work in it.

CSS-animations  
 this would have been a lot more helpful when I was originally trying to figure out how to do animations.

To change something in css:  
 .animated {  
 transition-property: background-color;  
 transition-duration: 3s; }

A script with the animated property:

<button id=”color”>Click me</button>  
  
<script>  
 color.onclick = function() {  
 this.style.backgroundColor = “red”; };  
</script>

The 4 properties of transitions:  
 transition-property  
 transition-duration  
 transition-timing-function  
 transition-delay

Multiple properties can be transitioned at once ie:  
 this.style.fontSize = “36px”;  
 this.style.color= “red”;

The transition will happen at the same time.

Transition property:  
 chooses whats animated. Examples: left, margin-left, height, color, all.

Transition duration:  
 how long the transition will occur or take to occur

Transition delay:  
 delay how long it takes the animation to start occurring

Transition timing function:  
 transition: left 5s cubic-bezier(0,0,1,1);  
 moves in a straight line at a constant speed

Transition: left 5s cubic-bezier(0.0, 0.5, 0.5, 1.0);  
 this will go fast first, then slow down towards the end.

Make it bounce a little:  
 left: 100px;  
 transition: left 5s cubic-bezier(.5, -1, .5, 2);  
 goes left slowly, then flies right, slows, and moves left a little again.

<div id=”digit”>  
 <div id=”stripe”>0123456789</div>  
</div>

Transform: translate(-90%);  
transition: transform 9s steps(9, start);

This will cause numbers to count up from 0-9 in one space on the page. All numbers won’t be shown at once.

asdf

boat.onclick = function() {

//...

let times = 1;

function go() {

if (times % 2) {

// sail to the right

boat.classList.remove('back');

boat.style.marginLeft = 100 \* times + 200 + 'px';

} else {

// sail to the left

boat.classList.add('back');

boat.style.marginLeft = 100 \* times - 200 + 'px';

}

}

go();

boat.addEventListener('transitionend', function() {

times++;

go();

});

};

All this makes a boat go back and forth getting farther and farther from starting point.

@keyframes in css allows multiple simple animations

@keyframes go-left-right { /\* give it a name: "go-left-right" \*/

from { left: 0px; } /\* animate from left: 0px \*/

to { left: calc(100% - 50px); } /\* animate to left: 100%-50px \*/

}

Infinite loop of going right to left.

Animations go through 3 process every frame:  
 layout: re calculate geometry position  
 paint: re calculate how everything looks at their place  
 composite: render final result

These things can affect performance of computer and animation. Some tasks and properties are very taxing on the system.