Timers, Animations

ITP 301
Spring 2021

Timeout

Executes a callback function after specified delay in milliseconds.

Syntax:

`setTimeout(`*`callback`*`, `*`delay`*`)`

`setTimeout()` returns an ID that can be used to cancel the function before expiration:

`clearTimeout(`*`timeout_id`*`)`

```
setTimeout( function(){
  console.log('1 second passed.');
}, 1000 );

.....................................................

function callbackFunction(){
  console.log('1 second passed.')
}

setTimeout( callbackFunction, 1000 );

.....................................................

var timeoutId = setTimeout( callbackFunction, 1000 );

clearTimeout(timeoutId);
```

Interval

Repeatedly executes a callback function every specified interval in milliseconds.

Syntax:

```
setInterval(callback, delay)
```

`setInterval()` returns an ID that can be used to cancel the function at any time:

```
clearInterval(interval_id)
```

```
setInterval( function(){
  console.log('Prints every 1 second.');
}, 1000 );
```

```
function callbackFunction(){
  console.log('Prints every 1 second.');
}

setInterval( callbackFunction, 1000 );
```

```
var intervalId = setInterval( callbackFunction, 1000 );

clearInterval(intervalId);
```

Animations

While timers can be used to perform animations, this is **not** the recommended method.

`requestAnimationFrame()` performs animations whenever the browser is ready to repaint (refresh) the DOM.

Advantages:

- Optimized by browsers for different displays.
- Smoother animations.
- Does not animate inactive windows or tabs.

```
function animate(){
  /*****
  * Perform Animations Here
  *****/

  requestAnimationFrame(animate);
}

requestAnimationFrame(animate);
```