School of Infocomm

C219 Front-end Web Development

Lesson 8

Animation using JavaScript Libraries

Recap

Lessons 1 - 7

JavaScript Animation Libraries

JavaScript animation libraries are a critical component of any web developer's toolkit. CSS can be used to add simple animations, but for more complex and advanced effects, JavaScript is the superior tool.

There are various categories of JavaScript animation libraries available – from audio animation to SVG animation to 3D animation.

Popular JS Animation Libraries

Anime.js

- Anime.js is a JavaScript animation library with a simple, yet powerful API. It works with CSS properties, SVG, DOM attributes and JavaScript Objects.

fullPage.js

- fullPage.js is a JavaScript library that creates fullscreen scrolling websites, also known as single page websites.

Vivus

- Vivus is a lightweight JavaScript class that allows you to animate SVGs, giving them the appearance of being drawn.

Anime.js

Targets and Properties

First of all, include Anime.js in your HTML document before the </body> tag:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/animejs/3.2.1/anime.min.js"></script>
```

You can then target elements using any CSS selector (except for pseudo selectors) and then manipulate it using any CSS property. Note that the syntax is slightly different for some properties.

```
anime({
  targets: ".demo1",
  translateX: 250,
  rotate: 90,
  opacity: [0, 1], //start and end values
  delay: 500
});
```

Parameters

You can define animation parameters like delay, duration, easing, loop and autoplay, and also specific parameters to each property of the animation using an object as the values.

```
anime({
  targets: '.demo1',
  translateX: {value: 300, duration: 100},
  rotate: {value: 360, duration: 1800, easing: 'easeInOutSine'},
  scale: {value: 2, duration: 1600, easing: 'easeInOutQuart'},
  delay: 250,
  direction: 'alternate',
  loop: true
});
```

Staggering

Staggering allows you to animate multiple elements that have the same class with follow through and overlapping action. `300` is the delay for each animation in ms. `{start: 1000}` specifies the start delay of the animation.

```
anime({
  targets: '.demo3',
  translateY: -300,
  rotate: 180,
  delay: anime.stagger(300, {start: 1000}),
  direction: 'alternate'
});
```

Timeline

Timelines let you synchronise multiple animations together. By default each animation added to the timeline starts after the previous animation ends but you can set offsets to control this.

```
// Create a timeline with default parameters
var tl = anime.timeline({
  easing: 'easeOutExpo',
  duration: 1000,
  delay: 500
});

// Add children
tl
  .add({ targets: '.circle', translateY: 250, scale: 0.5, duration: 500 })
  .add({ targets: '.square', translateX: 250, scale: 0.5 }, 200);
```

Controls

Anime.js allows for animation controls like play, pause and restart.

```
var animation = anime({
  targets: '.demo5',
  translateX: 300,
  scale: 2,
  easing: 'easeInOutExpo',
  direction: 'alternate',
  duration: 2000,
  loop: true,
  autoplay: false
});

document.querySelector('.play').onclick = animation.play;
document.querySelector('.pause').onclick = animation.pause;
document.querySelector('.restart').onclick = animation.restart;
```

Numbers

Any HTML element containing a numerical value can be animated using Anime.js. You simply need to assign the start and end values in an array, using innerHTML. The result would be an animated number counter.

```
anime({
  targets: '.demo6',
  innerHTML: [0, 1500], //start and end values
  easing: 'linear',
  round: 1, //rounds off to zero decimal places
  duration: 2000
});
```

Anime.js Versus CSS

Anime.js	CSS
Code is easy to understand	Code is not as easy to understand
Concise code	Lengthy code
Default easing is `easeOutElastic`	Default easing is `ease`
Stops animation at the end by default	Requires a `forwards` value
Allows for advanced features like staggering, number counters, functions and controls	No advanced features

Exercise 1

Create a web page about yourself and animate all HTML elements using Anime.js.

Requirements:

- Your name, photo, job title, age and description. You may include any other relevant information.
- You may use your web page done in Lesson 1 and enhance it.
- You must apply staggering, controls and number counter animations to your web page.

fullPage.js

fullPage.js

Usage

To get started, you need to include the CSS and JS files for fullPage.js.

CSS (before </head>):

```
<link rel="stylesheet" href="https://unpkg.com/fullpage.js/dist/fullpage.min.css">
```

JS (before </body>):

```
<script src="https://unpkg.com/fullpage.js/dist/fullpage.min.js"></script>
```

HTML Structure

Each section should be defined with an element containing the `section` class. The active section by default will be the first section.

Sections should be placed inside a container called `<div id="fullpage">` .

```
<div id="fullpage">
  <div class="section">Section 1</div>
  <div class="section">Section 2</div>
  <div class="section">Section 3</div>
</div>
```

Initialisation

You can use Vanilla JS or jQuery to initialise fullPage.js and set the options. In this example, we will use jQuery (remember to include jQuery in your document).

```
$(document).ready(function () {
  $('#fullpage').fullpage({ //initialize
    //set options
    sectionsColor: ['#03A9F4', '#2ECC71', '#00BCD4'],
    navigation: true,
    navigationPosition: 'right'
});
```

You can view all available options here.

Sliders

In order to create a landscape slider within a `section`, each slide must be defined with an element containing the `slide` class.

```
<div class="section">
  <div class="slide">Slide 1</div>
  <div class="slide">Slide 2</div>
  <div class="slide">Slide 3</div>
</div>
```

Linking to Sections

To create custom links to sections, you must first define the anchor names using the anchors property and then link to them using the <a> tag.

JS:

```
anchors: ['section1', 'section2', 'section3'] //name the anchors for each section
```

HTML:

```
<a href="#section3">Go to Section 3</a> //link to section 3
<a href="#section3/1">Go to Section 3 Slide 2</a> //link to the second slide in section 3
```

Callback Functions

Using the `afterLoad` property, you can trigger animations to start when a section has been loaded.

The animation in this example is done using Anime.js.

Anime.js

```
var spinTitle = anime({
  targets: ".s2",
  rotate: "360deg",
  autoplay: false,
  duration: 1000
});
```

fullPage.js

```
afterLoad: function (origin, destination, direction) {
  if (destination.index == 1) { //section 2
    spinTitle.play(); //anime.js play method
  }
}
```

You can view all callback functions here.

Exercise 2

Create a single-page website using fullPage.js.

Requirements:

- The web page must have three sections
- Each section must have a title in the middle of the page. You can name them Section 1, 2 and 3 accordingly.
- The third section should contain three slides with any content

L08 Assignment

Design a single-page website about yourself using Anime.js and fullPage.js.

Requirements:

- Create a web page with two sections
- Insert your contents from Exercise 1 into the first section
- The second section should contain your contact information, social media icons and other relevant information. All elements must be animated and only played after the section has loaded.

Refer to https://dribbble.com/search/portfolio for inspiration

Deliverables

Individual Submission:

- Exercise 1
- Exercise 2
- L08 Assignment

Others:

- Reflection Journal

Submit all deliverables by 2359 today