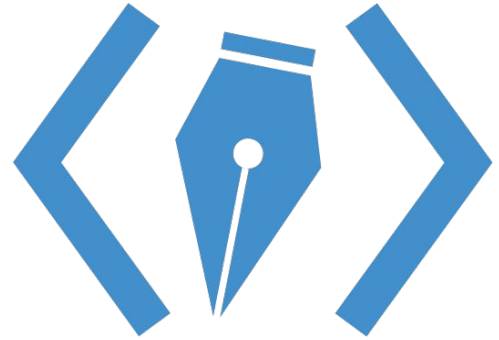


Week 11

Loops, Arrays and Libraries



Announcements

Homework 9 extended to tomorrow at 7pm

Homework 10 (last hw!) will be released tonight & no more labs!

Due next Tuesday so you'll have all of Thanksgiving break to work on project!

No class/lab/OH next week for Thanksgiving break! 🦃

Final Project!

Project due **Tuesday, Dec 1st**

Your project can be featured on our [showcase](#) website!

Final Project Submission: wdd.io/go/project-submit

Come to office hours via Piazza or wdd.io/go/OH

Give us anonymous feedback at wdd.io/go/feedback

RIT

PennState

SCAD

USC

NEW YORK UNIVERSITY

Student Office Hours

Here are some awesome students and full time designers who would love to help you out with your portfolio. Previously hosted on [Students Who Design](#).

[VOLUNTEER](#)

Hasque May

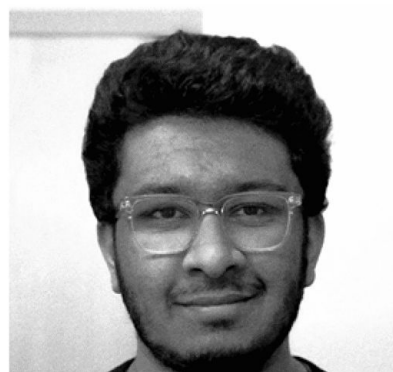
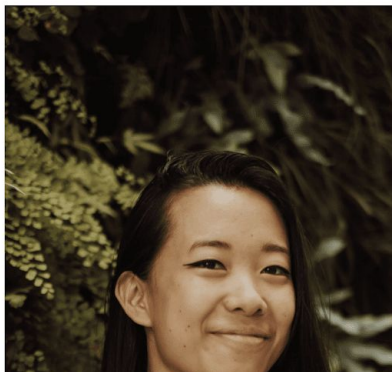
I'm not finished with my portfolio but could I get some feedback?

Cal

Ethelia Lung

Of course! What specific questions do you have?

W



Your online product design school.

Apply for our free 3 week product design course starting Spring 2020 →

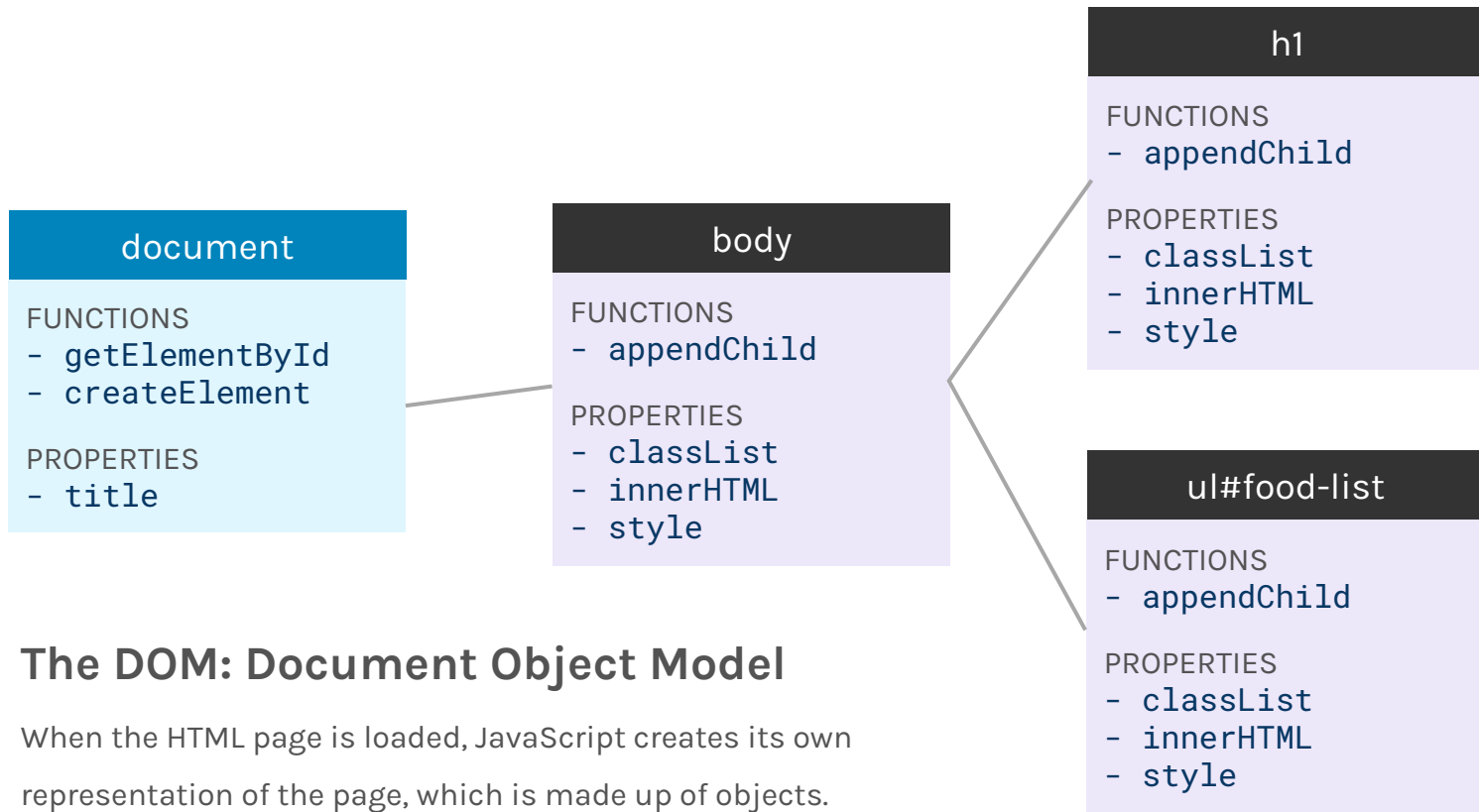
APPLY

WHAT YOU'LL LEARN



Quick Review: Event Handlers

```
let food =  
    document.getElementById( "food-list" )
```



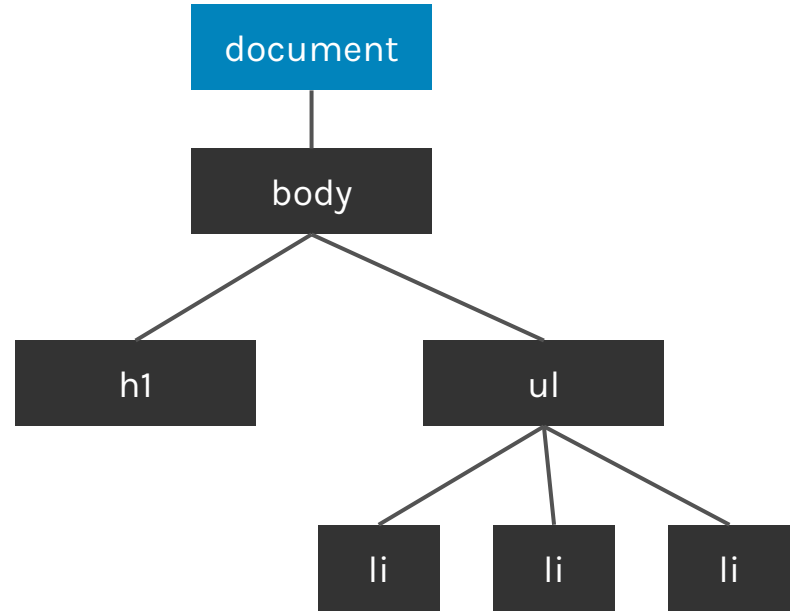
The DOM: Document Object Model

When the HTML page is loaded, JavaScript creates its own representation of the page, which is made up of objects.

Each HTML tag is an object.

`document.getElementById("food-list")`

What we're saying is that from our **document object model**, find the element that has the id **#food-list**.




```
let buttonElement =  
document.getElementById("popup-button");  
  
function showGreeting() {  
    alert("howdy!");  
}  
buttonElement.onclick = showGreeting;
```

Every DOM element
object has an
onclick function.

This is the function that will run
when you click on the element,
creating an onClick event.

```
let buttonElement =  
document.getElementById("button");  
  
let element =  
document.getElementById("color");  
  
function changeColor(color) {  
    element.style.backgroundColor = color;  
}  
  
buttonElement.onclick = function() {  
    changeColor("red");  
}
```

We don't have to add classes to modify CSS.

Note that we assign **style.property** to a string value -> our CSS reads text and so we use a string value to change the CSS property.

```
let buttonElement =  
document.getElementById("button");  
  
let element =  
document.getElementById("color");  
  
function changeColor(color) {  
    element.style.backgroundColor = color;  
}  
  
buttonElement.onclick = function() {  
    changeColor("red");  
}
```

We don't have to add classes to modify CSS.

Note that we assign **style.property** to a string value -> our CSS reads text and so we use a string value to change the CSS property.

```
let buttonElement =  
document.getElementById("button");  
  
let element =  
document.getElementById("color");  
  
function changeColor(color) {  
    element.style.backgroundColor = color;  
}  
buttonElement.onclick = function() {  
    changeColor("red");  
}
```

We don't have to add classes to modify CSS.

Here, we use anonymous functions to nest our action in our event handler. This is because we wait to change our color until we click!

Arrays

In JavaScript, an `array` is a list of values.

We use them to represent multiple objects in one placeholder.

```
let cart = ["protein", "caffeine", "bananas"]
```

```
let cart = ["protein", "caffeine", "bananas"]
```

Note: we declare our variables with a
let statement and a name as usual!


```
let cart = ["protein", "caffeine", "bananas"]
```

We declare an array by putting its value
between **square brackets []**.

```
let cart = ["protein", "caffeine", "bananas"]
```

We also separate each **item** in the array
with a comma.

```
let cart = ["protein", "caffeine", "bananas"]
```

“protein” is our first item in the array.

```
let cart = ["protein", "caffeine", "bananas"]
```

“caffeine” is our second item in the array.

```
let cart = ["protein", "caffeine", "bananas"]
```

“bananas” is our third item in our array. Since it is our last item, we do not use a comma afterwards.

```
let cart = ["protein", "caffeine", "bananas"]  
console.log(cart.length) // 3
```

`.length` is a built-in JavaScript attribute
for all arrays that returns the length of
the array.

Array indexing

A lot of programming languages have **zero-indexed arrays**, so to get the first item from an array, we use 0.

It's weird, but it's the way we count as programmers. **We start from [0]**.

```
let cart = ["apples", "oranges", "bananas"];

console.log(cart[0]); #"apples"
console.log(cart[1]); #"oranges"
console.log(cart[2]); #"bananas"
```


Where are arrays useful?


```
let food =  
    document.getElementById("food-list")
```

Up until now, we've only been using **document.getElementById** to get an element from our document one at a time.

We've used IDs. But what about **classes**? What if I told you, you can get multiple elements at a time?

it's plural!



```
let cart =  
  document.getElementsByClassName("food-item")
```

The above returns an array of elements that
have the class **food-item**!

```
let food =  
document.getElementById("food-list")
```

```
<html>  
<body>  
  <h1>Shopping List</h1>  
  <ul id="food-list" class="blue-box">  
    <li class="food-item">Apples</li>  
    <li class="food-item">Bananas</li>  
    <li class="food-item">Oranges</li>  
  </ul>  
</body>  
</html>
```

```
let food =  
document.getElementsByClassName("food-item")
```

```
<html>  
<body>  
  <h1>Shopping List</h1>  
  <ul id="food-list" class="blue-box">  
    <li class="food-item">Apples</li>  
    <li class="food-item">Bananas</li>  
    <li class="food-item">Oranges</li>  
  </ul>  
</body>  
</html>
```

```
<html>
<body>
  <h1>Shopping List</h1>
  <ul id="food-list" class="blue-box">
    <li class="food-item">Apples</li>
    <li class="food-item">Bananas</li>
    <li class="food-item">Oranges</li>
  </ul>
</body>
</html>
```

food =

```
[<li class="food-item">Apples</li>,
  <li class="food-item">Bananas</li>,
  <li class="food-item">Oranges</li>]
```



Q Try "Barcelona"

Become a host

Saved

Trips

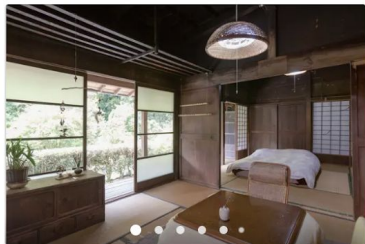
Messages

Credit

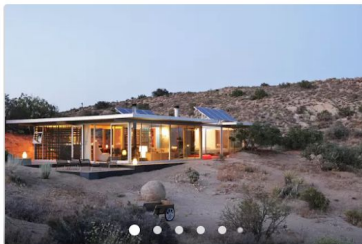
Help



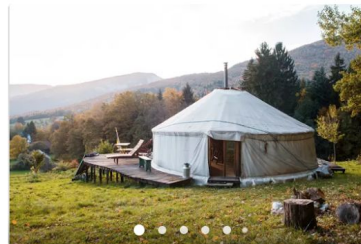
Homes around the world



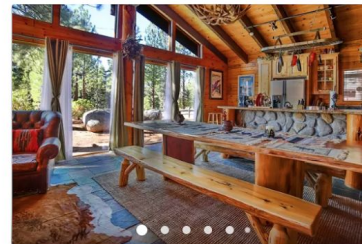
ENTIRE HOUSE · FUJIEDA
Yui Valley-Traditional house (easy to Tokyo/Kyoto)
\$99 per night · Free cancellation
★★★★★ 266 · Superhost



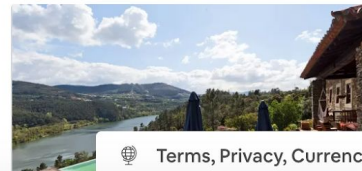
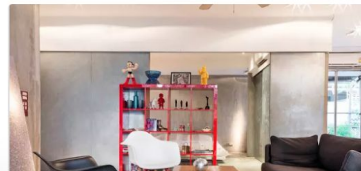
ENTIRE HOUSE · PIONEERTOWN
Off-grid itHouse
\$400 per night · Free cancellation
★★★★★ 212



YURT · BELLECOMBE-EN-BAUGES
A quiet yurt in Savoie - Bauges
\$81 per night · Free cancellation
★★★★★ 283



ENTIRE CHALET · SOUTH LAKE TAHOE
The Lake Tahoe Chalet
\$125 per night · Free cancellation
★★★★★ 177 · Superhost



[Terms, Privacy, Currency & More](#)

What elements might use classes? What elements might use IDs?

Web Design DeCal Fall 2020

Loopy Loops!

Kinds of Loops

- While loops
- Do-while loops
- For-each loops
- **For loops**



A **for loop** allows you to repeat some code until
a certain **condition is met**,
or for a certain **number of times**.

What is a for loop?

- Does a thing for you a certain number times without you having to write out every single iteration
- Cleaner, more maintainable code

```
for (let i = 1; i < 10; i++) {  
    console.log("counting to 9: ", + i);  
}
```

for loop breakdown: start

- Create a variable named `i`
 - Stands for “index”
- Set the variable value to 1
 - Starting value could be any number

```
for (let i = 1; i < 10; i++) {  
    console.log("counting to 9: ", + i);  
}
```

for loop breakdown: end

- Keep doing the action(s) inside the for loop while `i` is less than 10

```
for (let i = 1; i < 10; i++) {  
  console.log("counting to 9: ", + i);  
}
```

for loop breakdown: repeat

- Increment `i` by 1 every time the loop repeats execution
- Also written as:
 - `i += 1`
 - `i = i + 1`

```
for (let i = 1; i < 10; i++) {  
    console.log("counting to 9: ", + i);  
}
```

Why use a for loop?

```
console.log("counting to 9: ", + 1);  
console.log("counting to 9: ", + 2);  
console.log("counting to 9: ", + 3);  
console.log("counting to 9: ", + 4);  
console.log("counting to 9: ", + 5);  
console.log("counting to 9: ", + 6);  
console.log("counting to 9: ", + 7);  
console.log("counting to 9: ", + 8);  
console.log("counting to 9: ", + 9);
```



```
for (let i = 1; i < 10; i++) {  
    console.log("counting to 9: ", + i);  
}
```

What will this print?

```
for (let i = 0; i < 10; i++) {  
    console.log(i);  
}
```

What will this print?

```
for (let i = 0; i <= 10; i++) {  
    console.log(i);  
}
```


What will this print?

```
let cart = ["protein", "caffeine", "bananas"];  
for (let i = 0; i < cart.length; i++) {  
    console.log(cart[i]);  
}
```

JavaScript Libraries

JavaScript, like HTML or CSS, has a set of **syntax rules** that define a logical, executable environment.

Libraries are built on top of a language, and act as tools that we can use right out of the box.





[particles.js](https://danielshaver.github.io/particles.js/)

Waypoints

[About](#)[Guides](#)[Shortcuts](#)[API](#)

Waypoints is the easiest way to trigger a function when you scroll to an element.

```
var waypoint = new Waypoint({
  element: document.getElementById('waypoint'),
  handler: function(direction) {
    console.log('Scrolled to waypoint!')
  }
})
```

Builds are available for multiple DOM libraries.

jQuery
1.8+

Zepto
1.1+

No Framework
IE 9+

npm install waypoints

- or -

bower install waypoints

- or -



GitHub



Download



Get Help

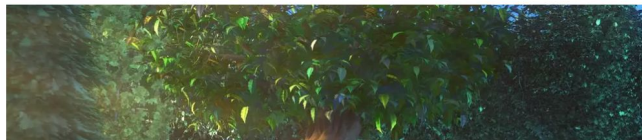
Waypoints



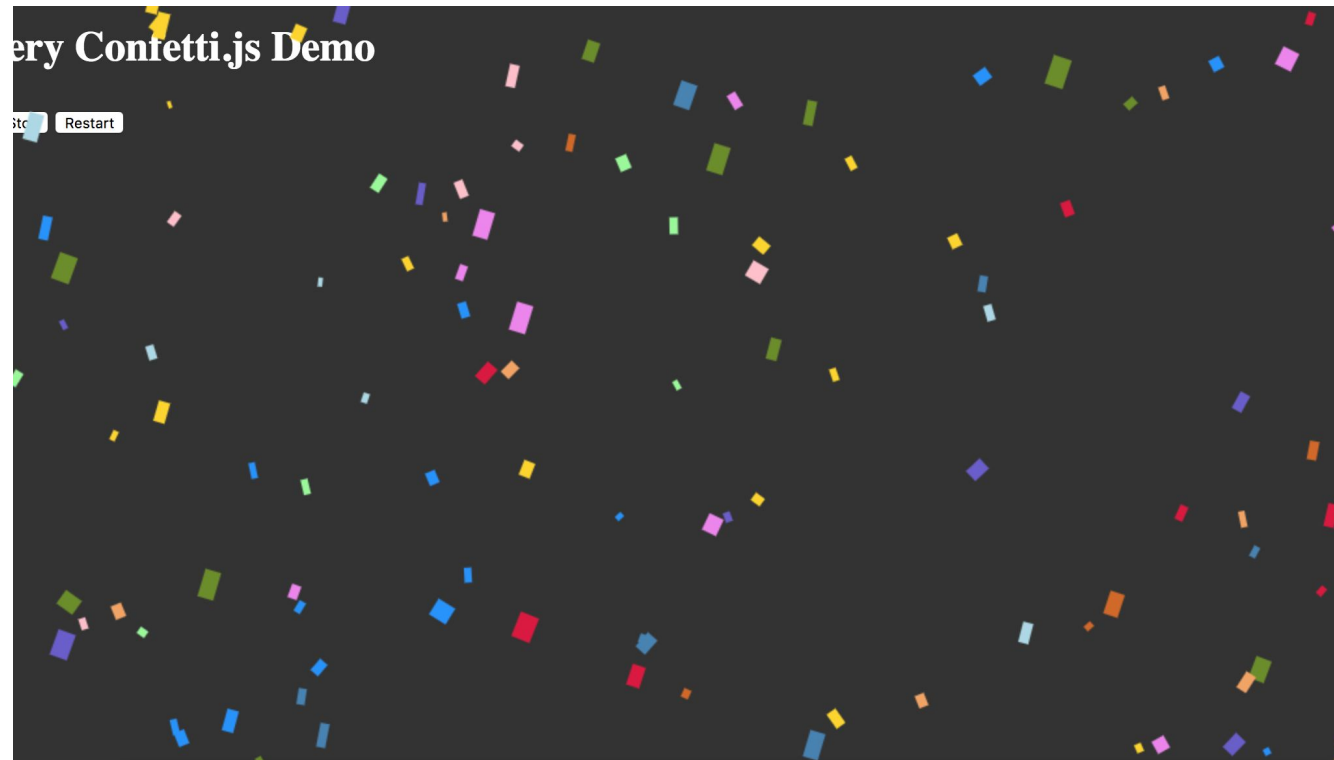
Normal ▾ Sailec Light ▾ **B** *I* U ☰ ☷ ☹ ☶ 🔗 🖼️ 💾 f_x `</>` \mathcal{I}_x

Quill Rich Text Editor

Quill is a free, [open source](#) WYSIWYG editor built for the modern web. With its [modular architecture](#) and expressive [API](#), it is completely customizable to fit any need.





Quill



Confetti

for when you need to recreate your berkeley letter of acceptance to feel a sense of gratification again

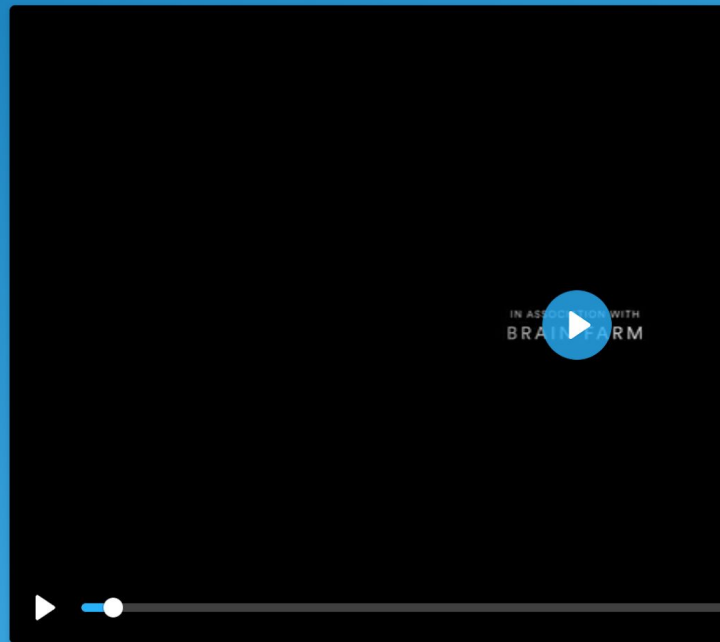
Plyr

A simple, accessible and customisable media player for  Video,  Audio,  YouTube and  Vimeo

Premium video monetization from 

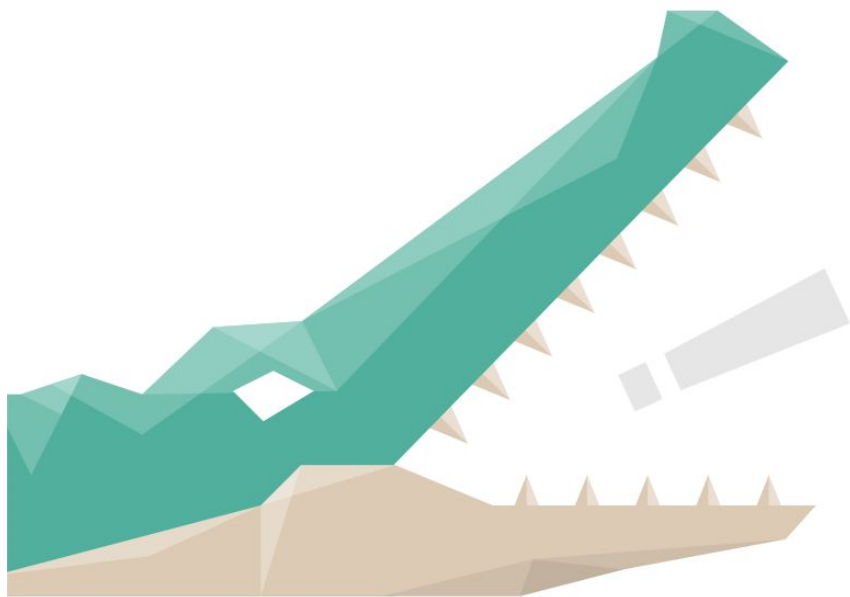
 Download on GitHub

11,057



 View From A Blue Moon © Brainfa

Plyr



[SnapSVG](#)



[three.js](#)
(example: Codrops' "The Aviator")



[Download](#)

[Latest Build](#)

[Source Code](#)

[npm package](#)

[Docs](#)

[Demos](#)

[Documentation](#)

[Wiki](#)

[License](#)

[Changelog](#)

[Info](#)

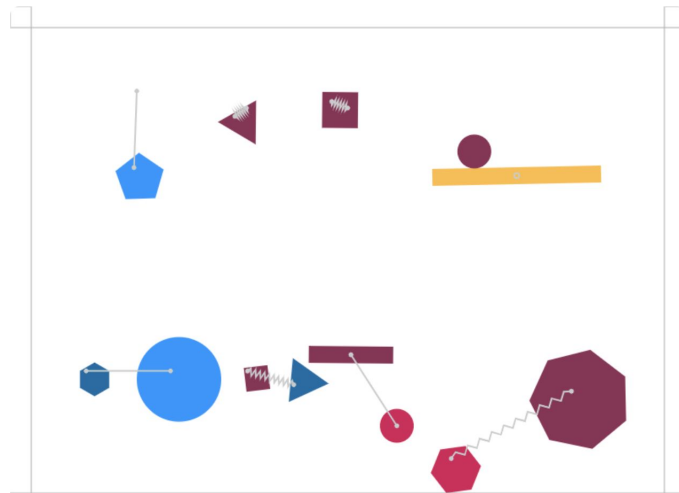
[Features](#)

[Install](#)

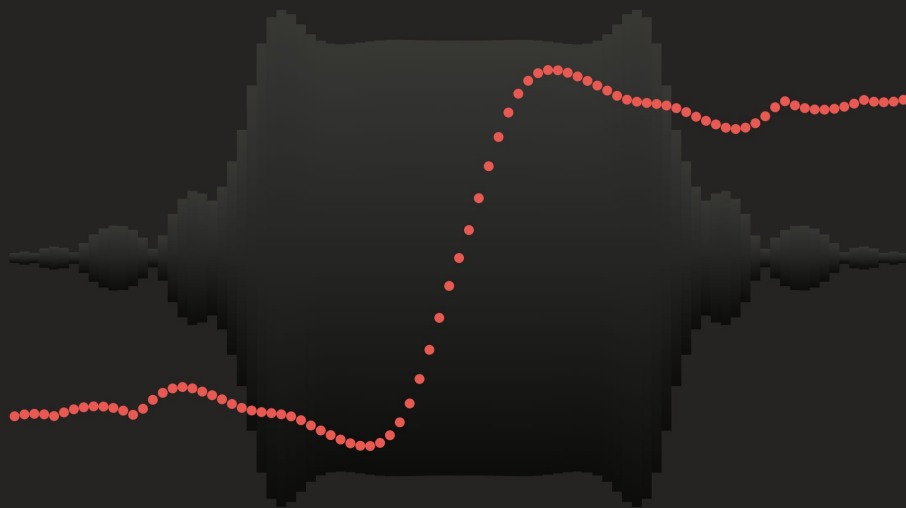
[Usage](#)

Matter.js is a 2D physics engine for the web

— [see all demos](#) →



[matter.js](#)



GitHub



CodePen



Docs



Twitter

Anime.js (/ˈæn.ə.meɪ/) is a lightweight JavaScript animation library with a simple, yet powerful API. It works with CSS properties, SVG, DOM attributes and JavaScript Objects.

[Getting started](#)

AnimeJS

Jump.js

A small, modern, dependency-free
smooth scrolling library by **callmecavs**.



Documentation on GitHub

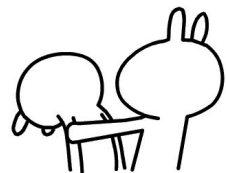


[Jump.js](#)

The most versatile animated typing utility on the internet

```
new TypeIt('#hero', {
  speed: 50,
  startDelay: 900
})
.type('The most versatile animated typing utility on the internet')
.pause(300)
.delete(2)
.pause(250)
.type('et')
.pause(750)
.options({speed: 100, deleteSpeed: 75})
.delete(8)
.pause(750)
.type('<em>planet.</em>')
.go();
```

Typeit



Demo!