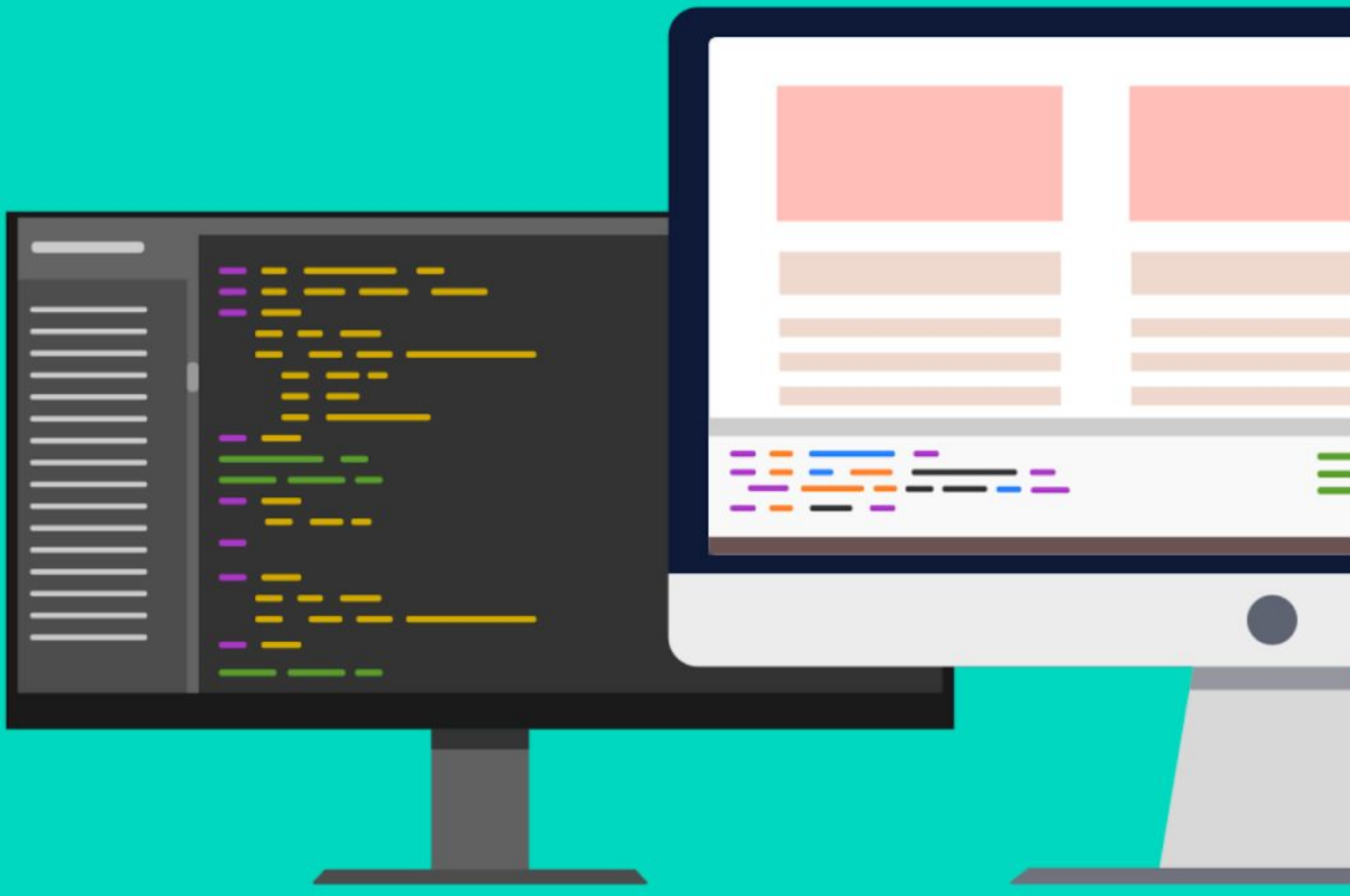
JAVASCRIPT HOWTOS



{ Junior Developer Central }

Complete list of all the JavaScript HowTos as found on the Junior Developer Central YouTube Channel

Complete list of JavaScript HowTos

Junior Developer Central
https://www.youtube.com/juniordevelopercentral
https://www.juniordevelopercentral.com/
https://twitter.com/codebubb

Introduction

Welcome to this JavaScript HowTo eBook!

I put together over 40 different tutorials on the Junior Developer Central YouTube channel a while back covering some of the most commonly searched for tasks to be done with JavaScript so I thought i'd bring them all together here for you to keep as a reference.

Some of the 'HowTos' are very simple only requiring a line or two of code to achieve whereas some are more complex requiring you to write functions, network requests, and make use of 3rd party APIs.

You can use this eBook however you like - it might be useful to scan through once quickly to see what is in here and look through some of the points that you might find useful. Otherwise, it might serve as a handy reference if you need a bit more help on a particular topic when a Google search just doesn't cut it.

I hope you find this guide useful and if you have any feedback, comments, or spot any errors, reach out on Twitter (@codebubb) and let me know!

Enjoy!
James.

How to redirect a page with JavaScript

All you need to navigate to a new page with JavaScript is to assign a new value to the href property, on the location object which is found on the global Window object of the browser:

```
window.location.href =
'https://www.juniordevelopercentral.com';
```

You could wire this up to the click of a button or some other event like this:

```
document.querySelector('.someclass')
 .addEventListener('click', () => {
  window.location.href =
   'https://www.juniordevelopercentral.com';
  });
```

Watch the tutorial for more details.

How to refresh/reload a page with JavaScript

In order to force a reload of the current page with JavaScript you can use:

```
window.location.reload();
```

You can also bypass the browser's cache by passing a value of true to the reload function.

```
window.location.reload(true);
```

There is the option to set a timeout to trigger the reload after a certain time. For example to reload after 5 seconds:

```
setTimeout(() => {
  window.location.reload(true);
}, 5000);
```

Watch the tutorial for more details.

How to get and set cookies with JavaScript

You can access the cookies for a particular page by referencing the **cookie** property of the **document** object. i.e.

```
document.cookie
```

To set a new cookie you simply assign a new string to **document.cookie:**

```
document.cookie = 'name=james';
```

It might look like you're going to remove all the other cookies by doing this but the above will just set a new cookie with the name of '*name'* and a value of '*james'.*

If you want to get a particular cookie value you need to manipulate the cookie string, possibly by splitting it into an array.

```
document.cookie.split(';')
```

You will still need to do a bit of work to get the value of a particular cookie (like splitting each string in the array further by it's **=** symbol then search for the name of the cookie). Alternatively, you can reduce the split cookie array to a single object.

```
document.cookie
  .split(';')
  .map(cookie => cookie.split('='))
  .reduce((accumulator, [key, value]) => ({
   ...accumulator, [key.trim()]: decodeURIComponent(value)
 }), {});
```

[Watch the tutorial for more details.]

How to change button text with JavaScript

To change the text of a button with JavaScript you simply need to target the particular button element and update it's **textContent** (or **innerText)** property.

```
document.querySelector('#myButton')
  .textContent = 'New Text';
// or... .innerText = 'Different Text';
```

Watch the tutorial for more details.

How to change background color with JavaScript

Probably the simplest way to do change the background color of an element is to update it's inline style.

```
document.querySelector('#myElement')
  .style.backgroundColor = '#eeffaa';
```

This might not be ideal for some situations so another approach would be to use a class and add that to the element when you want to change it's background color. For example, set up a class that has a **background-color** set to something.

```
.green {
  background-color: green;
}
```

And then add it as a class to the element you want to change.

```
document.querySelector('#myElement')
  .classList.add('green');
```

Watch the tutorial for more details.

How to hide a button with JavaScript

There are a few ways to do this.

```
document.querySelector('#myElement')
  .style.visibility = 'hidden';
```

^^ This hides the element but it still takes up space on the page.

```
document.querySelector('#myElement')
  .style.display = 'none';
```

^^ This removes the element from the page, it doesn't take up space any more (the element is still part of the DOM).

Watch the tutorial for more details.

How to disable a button with JavaScript

Simply set the **disabled** attribute to have a value of **disabled** and your button will become inactive.

```
document.querySelector('#myButton')
  .setAttribute('disabled', 'disabled');
```

Watch the tutorial for more details.

How to get the current URL with JavaScript

There is a property inside the **location** object which will give you the full URL of the page you are currently browsing.

```
window.location.href
```

In addition, you can pass the string that's contained in the **href** property to construct a new **URL** object.

```
new URL(window.location.href)
```

This will give you access to properties like the **hostname, path, origin** etc which makes it easier to extract parts of the URL that you may need.

Watch the tutorial for more details.

How to open a new tab with JavaScript

There is an **open** function you can call on the window object in the browser.

```
window.open()
```

This should open a brand new blank tab for you but the function's behaviour is pretty much determined by the browser you're using. For example, you might get a new window if your browser has tabs turned off or something weird like that.

You can also open the tab with a specific page by passing in a URL.

```
window.open('https://www.juniordevelopercentral.com/')
```

Of course you can always add the **target="_blank"** attribute to your anchor tags for them to load their content in a new tab without the need for JavaScript.

Watch the tutorial for more details.

How to get a value from a radio button with JavaScript

Say you have a radio button or checkbox element and you want to determine whether it is checked or not, you can determine its state by accessing the **checked** property.

```
// <input type="radio" id="radioBtn">
document.querySelector('#radioBtn').checked;
```

If you access the element with a querySelector call, you just need to check whether the **checked** property is truthy to determine whether the input control has been selected or not.

Watch the tutorial for more details.

How to print with JavaScript

Just call the **print** function available on the window object of the browser to open the print dialog to send the current page to a printer.

```
window.print();
```

You can adjust exactly what gets sent to the print output by determining CSS rules that format the web content for printing.

Watch the tutorial for more details.

How to add or remove a class with JavaScript

Each HTML element that you select in JavaScript has a **classList** property which contains an object with a few handy functions.

To add a new class:

```
document.querySelector('#myElement')
  .classList.add('newCSSClass');
```

To remove an existing class applied to an element:

```
document.querySelector('#myElement')
  .classList.remove('newCSSClass');
```

To add a new class if it isn't already applied to the element or to remove it if it already is (i.e. toggle the class):

```
document.querySelector('#myElement')
  .classList.toggle('newCSSClass');
```

Watch the tutorial for more details.

How to append HTML with JavaScript

Probably the best approach here is to use one of several functions to insert new HTML DOM elements into your page. The most flexible is the **insertAdjacentHTML** function.

```
document.querySelector('#myElement')
  .insertAdjacentHTML('beforebegin', '<h1>New HTML
Content</h1>');
```

The function enables you to pass in HTML as a string as the second argument and the first argument allows you to specify where to put it. It can either be:

- **beforebegin:** right before the element you've selected
- **afterbegin:** inside the element you've selected, before it's first child HTML node
- **beforeend:** inside the element after the last child node
- **afterend:** after the element itself

Watch the tutorial for more details.

How to add style with JavaScript

So you can add style to elements on the page in a few different ways with JavaScript.

One way is to set an inline style by accessing the **style** property on an element.

```
document.querySelector('#myElement')
  .style.color = '#fff';
```

You'll need to be careful with **kebab-cased** properties as these will need to be camelCased e.g. **background-color** becomes **backgroundColor**.

The other way is to add a new class to the element that has the styles that you want to add to the element.

```
// .mystyle { color: '#fff'; }
document.querySelector('#myElement')
  .classList.add('mystyle');
```

You could even do something crazy like create your own style tag and insert some new CSS rules into it.

```
const style = document.createElement('style');
style.innerText = `.mystyle { color: '#fff'; }`
document.head.appendChild(style);
```

Watch the tutorial for more details.

How to close a browser tab with JavaScript

So you can close a browser tab or window with the **close** function available on the window object.

```
window.close();
// Scripts may close only the windows that were opened by
it.
```

But depending on your browser you might run into problems such as the error above. This makes sense as you wouldn't want tabs randomly closing without the user's approval. This should be OK if you've just opened a tab with the **window.open()** function for example.

Watch the tutorial for more details.

How to get a user's IP address with JavaScript

Basically you can't with JavaScript. At least there's nothing directly in the language or Web APIs that will allow you to get the user's remote IP address. What you can do however, is send a network request to an API service which will return the user's IP address. For example, the **ipify.org** service.

```
fetch('https://api.ipify.org/?format=json')
  .then(response => response.json())
  .then(data => console.log(data.ip)); // 123.123.123.123
```

Obviously this is subject to the API's terms and conditions etc. so for a production use you'll want to use a more robust service (rather than a free one) if you're relying on the user's IP address for some purpose.

Watch the tutorial for more details.

How to get URL parameters with JavaScript

So let's imagine you're working with an app that has various different parameters added to the URL when loading data for example. You might want to retrieve the various parameters to use in your JavaScript code.

The **window.location.search** property gives you everything in the search request but it's a long string so you might want to use a **URLSearchParams** object to parse it.

```
// https://someurl.com/?search=abcd&user=1234
const p = new URLSearchParams(window.location.search);

p.get('search'); // abcd
```

You can then call the **get** function to retrieve any specific parameter you need.

Alternatively, you can reduce all of the keys in the **URLSearchParms** object to a single object.

```
// https://someurl.com/?search=abcd&user=1234
const p = new URLSearchParams(window.location.search);
const searchParams = Array.from(p.keys())
  .reduce((acc, key) => ({ ...acc, [key]: p.get(key) }),
{});

// { search: 'abcd', user: '1234' }
```

Watch the tutorial for more details.

How to get an input value with JavaScript

You can get the value of any input with JavaScript (regardless of its actual type) by directly accessing it's value property.

```
// <input id="myInput" type="text">

document.querySelector('#myInput').value;
```

Watch the tutorial for more details.

How to get a select value with JavaScript

When you're dealing with select boxes you get access to the value that has been selected by the user simply by accessing the element's **value** property.

```
/*
  <select id="languages">
    <option value="javascript">JavaScript></option>
    <option value="php">PHP></option>
    ...
  </select>
*/

document.querySelector('#languages').value;
```

Watch the tutorial for more details.

How to make a POST request with JavaScript

Probably the simplest way to make a POST request with JavaScript is using the Fetch API. This should be supported in most modern browsers. Normally, you would just pass a URL to the **fetch** function and it would perform a GET request. To change this to a POST request, simply pass in an object specifying the method you want to use.

```
fetch('https://myurl.com/', { method: 'POST' })
 .then(result => result.json())
 .then(console.log);
```

You can use the returned promise from the call to fetch to extract an object from the data returned and make use of it if the endpoint your calling returns JSON data.

Watch the tutorial for more details.

How to get the current date with JavaScript

If you want to get the current date in a format a bit like DD-MM-YYYY or something similar you'll need to do a bit of work with JavaScript. First, you'll need to create a new date object and then call various functions in order to get the components of the date you need.

```
const today = new Date();
const yyyy = today.getFullYear();
const mm = today.getMonth() + 1;
const dd = today.getDate();

console.log(`${dd}-${mm}-${yyyy}`); // 9/9/2020
```

You might also want to pad the day and month values so they are nicely formatted.

```
const today = new Date();
const yyyy = today.getFullYear();
const mm = String(today.getMonth() + 1).padStart(2, '0');
const dd = String(today.getDate()).padStart(2, '0');

console.log(`${dd}-${mm}-${yyyy}`); // 09/09/2020
```

Watch the tutorial for more details.

How to set focus with JavaScript

When working with form elements, you can direct the user's focus to a particular input field by calling the focus function.

```
// <input id="myInput" type="text">
document.querySelector('#myInput').focus();
```

Watch the tutorial for more details.

How to copy to the clipboard with JavaScript

If you want to copy to the clipboard the text from an input element with JavaScript you can call the **select** function on the HTML element and then execute the copy command on the document.

```
//  <input id="myInput" type="text">

document.querySelector('#myInput').select();
document.execCommand('copy');
```

If the browser you're working with supports it, you could also use the clipboard API found on the navigator object.

```
window.navigator.clipbaord.writeText('Some text');
```

The user might also have to provide permission for you to directly write text to the clipboard.

Watch the tutorial for more details.

How to validate an email address with JavaScript

The best way to validate an email is to use a Regular Expression to check whether your string contains an email.

```
const simpleEmailRegex = /\S+@\S+\.\S+/;

simpleEmailRegex.test('dfsfd'); // false
simpleEmailRegex.test('email@example.com'); // true
```

The regex can be as simple or as complex to suit your needs. The above example will do a simple check but you can use a more complicated pattern to do more precise / extensive checks.

```
const complexEmailRegex =
/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".
+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])
|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
```

Watch the tutorial for more details.

How to detect key presses with JavaScript

You can set up event listeners for various different elements and listen for **keyup** or **keydown** events. Alternatively, to catch any key presses on a particular page, create an event listener on the window object.

```
window.addEventListener('keydown', (event) => {
  console.log(event.key);
});
```

The key property of the event object in the callback will contain the string representation of the key pressed and there are additional properties on the event object. For example, you can determine whether the shift key was pressed by examining the **shiftKey** property.

Watch the tutorial for more details.

How to hover with JavaScript

So if you want to change an element's look you can set up a **:hover** selector which modifies its appearance. If however you want to do some specific task in JavaScript, you can create events to be fired when the user moves their mouse over and off the element.

```
myElement.addEventListener('mouseenter', (e) => {
  console.log('Mouse enter');
});

myElement.addEventListener('mouseleave', (e) => {
  console.log('Mouse leave');
});
```

With the event listener's setup, you can call any JavaScript code you like.

Watch the tutorial for more details.

How to move an object with arrow keys with JavaScript

Bit of a strange one but if you have a need to manipulate an HTML element's position on the page, you can use JavaScript to alter it's position with the arrow keys on your keyboard.

```
// <div id="block" style="position: absolute; top: 0; left: 0; width: 100px; height: 100px; background-color: red;"></div>

const modifier = 5;
window.addEventListener('keydown', (event) => {
  const { style } = block;
  switch(event.key) {
    case 'ArrowUp':
 style.top = `${parseInt(style.top) - modifier}px`; break;
    case 'ArrowDown':
 style.top = `${parseInt(style.top) + modifier}px`; break;
    case 'ArrowLeft':
 style.left = `${parseInt(style.left) - modifier}px`;
 break;
    case 'ArrowRight':
 style.left = `${parseInt(style.left) + modifier}px`;
break;
  }
});
```

Watch the tutorial for more details.

How to alert (Yes/No OK/Cancel) with JavaScript

Rather than use the standard **alert** function, use the **confirm** function instead.

```
const response = confirm('Are you sure?');
```

The value returned from the confirm function will be true or false depending on whether the user presses the OK or cancel button.

Watch the tutorial for more details.

How to encrypt data in the browser

JavaScript and Web APIs in the majority of browsers don't have any common encryption functions built in. You can load in 3rd party scripts for common hashing algorithms.

Like md5:

```
// <script
src="https://cdnjs.cloudflare.com/ajax/libs/blueimp-md5/2.
12.0/js/md5.min.js"></script>

const hash = md5('password');
```

Or sha256:

```
// <script
src="https://cdnjs.cloudflare.com/ajax/libs/js-sha256/0.9.
0/sha256.min.js"></script>

const hash = sha256('password');
```

Watch the tutorial for more details.

How to get a timestamp with JavaScript

A timestamp is the number of seconds since the Unix epoch (1st Jan 1970) and you get the number of **milliseconds** since then with the **Date.now()** function in JavaScript. So you might want to divide this by 1000 and round it to get the number of seconds which is more commonly used for timestamps.

```
Date.now(); // Milliseconds

Math.round(Date.now() / 1000); // Seconds
```

Watch the tutorial for more details.

How to check if an element is hidden with JavaScript

If you want to check if an element is actually hidden then you can't actually just access it's **style** property with JavaScript (as the properties on the object will be empty unless you have already assigned a value with JavaScript).

You can however use the **getComputedStyle** function available on the window object.

```
window.getComputedStyle(myElement).display === 'none';
window.getComputedStyle(myElement).visibility ===
'hidden';
```

You pass in a reference to the element you are checking and it's probably worth checking the value of both the display and visibility properties.

You could combine this into a function to check both at the same time.

```
const isElementHidden = elem =>
  window.getComputedStyle(elem).display === 'none' ||
  window.getComputedStyle(myElement).visibility ===
'hidden';
```

Watch the tutorial for more details.

How to create a button with JavaScript

The same way you would create any new elements really - either by assigning a new blob of HTML as a string or creating the individual elements from scratch.

```
const container = document.querySelector('#container');

container.innerHTML = '<button>New Button</button>';
// or
const newBtn = document.createElement('button');
newBtn.innerText = 'New Button';

container.appendChild(newBtn);
```

The two techniques do exactly the same thing however assigning a value to **innerHTML** will replace the entire contents of the container element. You also get more flexibility by creating the new element as you have a reference to it stored that you can use later on.

Watch the tutorial for more details.

How to create a filterable list with JavaScript

If you have a list of items you might want to filter them based on some user input. You can do this dynamically by using a regular expression for matching.

```
/*
 <ul id="list">
  <li>Eggs</li>
  <li>Bacon</li>
  <li>Sausages</li>
  ...
 </ul>
 <input id="filter" type="text">
*/

const filter = document.querySelector('#filter');
const list = document.querySelector('#list');

filter.addEventListener('keyup', () => {
  Array.from(filter.children.forEach(li => {
    const matchFound = new RegExp(filter.value, 'gi')
      .test(li.innerText);
      if (!matchFound) {
        li.classList.add('hidden');
      } else {
        li.classList.remove('hidden');
      }
    });
  });
```

You will need to also add a class of hidden in order to make the items disappear from the list or alternatively you could update the inline styles of the elements on the fly.

Watch the tutorial for more details.

How to load an image (lazy load) with JavaScript

So there are lots of libraries available for handling lazy loading of images but if you just want to show a temporary image (like a loader for example) whilst a particularly large image is being downloaded, you can simply create a second **img** element and load the big image to it. Once it's finished, you can swap out the temporary loading image.

```
const img = document.createElement('img');
img.src = 'spinner.gif'; // Set the temporary image

const largeImg = document.createElement('img');
largeImg.addEventListener('load', () => {
  console.log('Large image loaded');
  img.src = largeImg.src; // The large image has loaded so
swap out the temporary image
});

largeImg.src = 'bigimg.png'; // Starts process of loading
large image
```

Watch the tutorial for more details.

How to get screen / browser size with JavaScript

You can access either the inner dimensions of the browser window or the outer dimensions. The inner dimension will change if for example you have your dev tools open.

```
window.outerWidth;
window.outerHeight;
window.innerWidth;
window.innerHeight;
// Also...
window.screen;
```

There is also the **screen** object available on the window object which has loads of useful information like the available screen size, orientation etc. although it's probably best not to use this for mobile detection.

Watch the tutorial for more details.

How to format currency with JavaScript

If you're using a modern browser you might want to consider using the number formatter objects available in the Internationalisation API.

```
const numberFormatter = new Intl.NumberFormat({
  style: 'currency',
  currency: 'USD',
});

numberFormatter.format(1000000); // $1,000,000.00
```

This way you can keep your values with a type of number but use the formatter when you want to display them in a more human readable format.

Watch the tutorial for more details.

How to make the browser full screen with JavaScript

With JavaScript you can select a particular element on the page and then switch the browser to full screen mode, only showing that element.

```
document.querySelector('#container')
  .requestFullScreen();
```

If you try this out, the downside is that you'll get the element on the page but the rest of the page will be black / white (depending on which browser you're using).

If you call the **requestFullScreen** function on the documentElement then this shouldn't happen.

```
document.documentElement
  .requestFullScreen();
```

You can also call the corresponding **document.exitFullScreen()** when you want to return to the normal browser size.

Watch the tutorial for more details.

How to detect the user's browser with JavaScript

It's a bad idea really but if you have a need to you can use feature detection or user agent sniffing.

The user agent method involves looking at the string provided on the **navigator** object and seeing if you can recognise the browser.

```
window.navigator.userAgent;
// "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.122 Safari/537.36"
```

You could therefore look for the string of **'Chrome'** to determine whether the user is using the Chrome browser but what if you were checking for '**Safari'**? This would give a false positive in this case.

If you are doing browser detection to customise the code that runs, try checking whether the feature you are trying to use exists on the browser to determine if you need to use a fallback.

```
let isFetchSupported = true;

try {
  fetch;
} catch (error) {
  isFetchSupported = false;
}
```

Watch the tutorial for more details.

How to split a string in JavaScript

Use the **split** function of course! But there are a few extra things you can do with it.

```
const string = 'The quick  brown fox, jumped over the lazy dog';
string.split('');
// Split every character
string.split(',');
// Split based on the delimiter provided
string.split(' ');
// Get every word (assuming only one whitespace per word)
string.split(/\s+/);
// Or match using a pattern, like matching more than one whitespace
string.split(' ', 3);
// Finally, limit the number of returned items
```

Watch the tutorial for more details.

How to get form data with JavaScript

To get the individual values from a form you can write several query selectors or just use a **querySelectorAll** or similar. Alternatively, you can access the **elements** property of a form.

```
/*
 <form id="myForm">
   <input id="myInput" type="text">
   <button>Submit</button>
*/

document.querySelector('#myInput').value; // An individual value
document.querySelector('#myForm').elements; // All of the form elements
```

The only problem with the ‘elements’ approach is that it will return any unnecessary form elements, like buttons, aswell.

Watch the tutorial for more details.

How to delete cookies with JavaScript

To remove a cookie from the browser, you simply need to assign it an expiration date in the past.

```
document.cookie = "myCookie=; expires=Thu, 01 Jan 1970 00:00:01 GMT;"
```

You still need the assignment operator when referencing the cookie but the value you give it can be blank as above.

Watch the tutorial for more details.

How to get a user's location with JavaScript (Gelocation)

If your browser supports it you can request the user's location from the Geolocation API.

```
window.navigator
  .geolocation.getCurrentPosition(console.log,console.log);
```

The **getCurrenPosition** function takes two arguments which are functions to be performed when there is a successful geolocation lookup or a failure.

The user usually has to give consent (by clicking an acceptance in a popup) and you'll get back an object in the success callback that has the user's longitude and latitude values.

These aren't much use on their own if you're looking to identify the user's location in a human readable format so you'll need to use a lookup service like opencagedata.com (account and API key required) to translate the longitude and latitude values.

```
const successfulLookup = position => {
  const { latitude, longitude } = position.coords;

fetch(`https://api.opencagedata.com/geocode/v1/json?q=${la
titude}+${longitude}&key=1234`)
    .then(response = response.json();
    .then(console.log);

Window.navigator
.geolocation.getCurrentPosition(successfulLookup,
  console.log);
```

Watch the tutorial for more details.