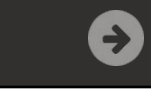
CHAPTER 9

APIS

WHAT IS AN API?

API stands for Application Programming Interface.



User interfaces allow humans to interact with programs.

APIs let programs (and scripts) talk to each other.

You do not need to know how a program does something…

So long as you know how to ask it to do something and process the response.

An API tells you how to ask a program to do something…

As well as the format you can expect the response to be in.

HTML5 JAVASCRIPT APIS

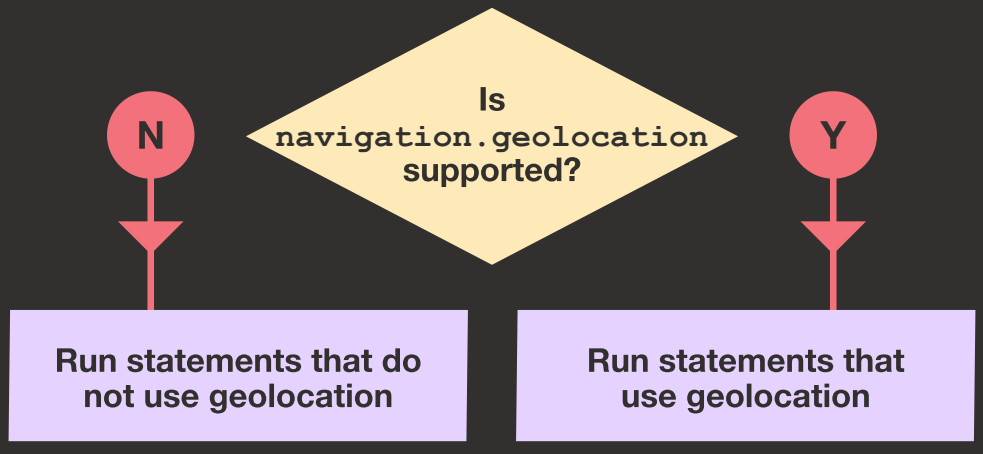
As devices have evolved, browsers can access new:

Hardware features such as GPS, accelerometers and cameras

Software functionality such as storage objects, file API, canvas

Feature detection lets you check whether or not the browser supports a feature.

You can choose to only run the code if the feature is supported.



Modernizr is a popular script that allows you to check if a feature is supported.

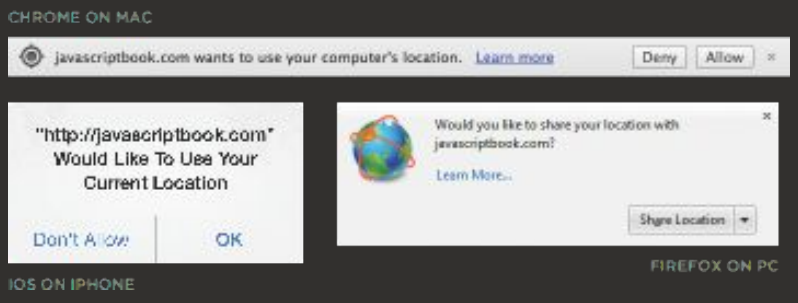
```
if (Modernizr.geolocation) {
  // Code to show location
} else {
  // Not supported / turned off
  // Or user rejected
}
```

GEOLOCATION API

Your visitor's location can be requested using the **geolocation API**.

Users are asked if they want to share their location, and it is only shared if they consent.

PERMISSION REQUESTS



```
navigator.geolocation.getCurrentPosition(success, fail);
```

The statement begins with the object that represents the browser:

```
navigator.geolocation.getCurrentPosition(success, fail);
```

A child of the `navigator` object then allows you to ask for the location:

```
navigator.geolocation.getCurrentPosition(success, fail);
```

Next, a method of the `geolocation` object requests the position:

```
navigator.geolocation.getCurrentPosition(success, fail);
```

The `getCurrentPosition()` method has two parameters:

```
navigator.geolocation.getCurrentPosition(success, fail);
```

The first is the name of a function to call if the location of the user has been retrieved successfully:


```
navigator.geolocation.getCurrentPosition(success, fail);
```


The second is the name of a function to call if the location of the user has **not** been retrieved successfully:

```
navigator.geolocation.getCurrentPosition(success, fail);
```

If a request is successful, an object called `position` is returned.
It has a child object called `coords`, which allows you to access properties for longitude and latitude.

```
position.coords.longitude
position.coords.latitude
```

If the request is **not** successful then a different message can be shown to the user instead.

This shows how APIs are often implemented using objects that have:

Methods to do things

Properties to tell you the characteristics of the object

So learning to use an API is often a matter of learning the objects it uses.

Then what methods / properties you can use.

What format the reply will come in.

WEB STORAGE API

Web storage (or HTML5 storage) lets you store data in the browser.

The data it stores can only be accessed by the domain that set the data.

There are two types of storage: **local** and **session** storage.

They are implemented using the `localStorage` and `sessionStorage` objects.

Both local and session storage have the same methods to get and set data, but each lasts a different amount of time.

	LOCAL	SESSION
Stored when tab closed	Y	N
All open windows / tabs can access the data	Y	N

Accessing the storage API:

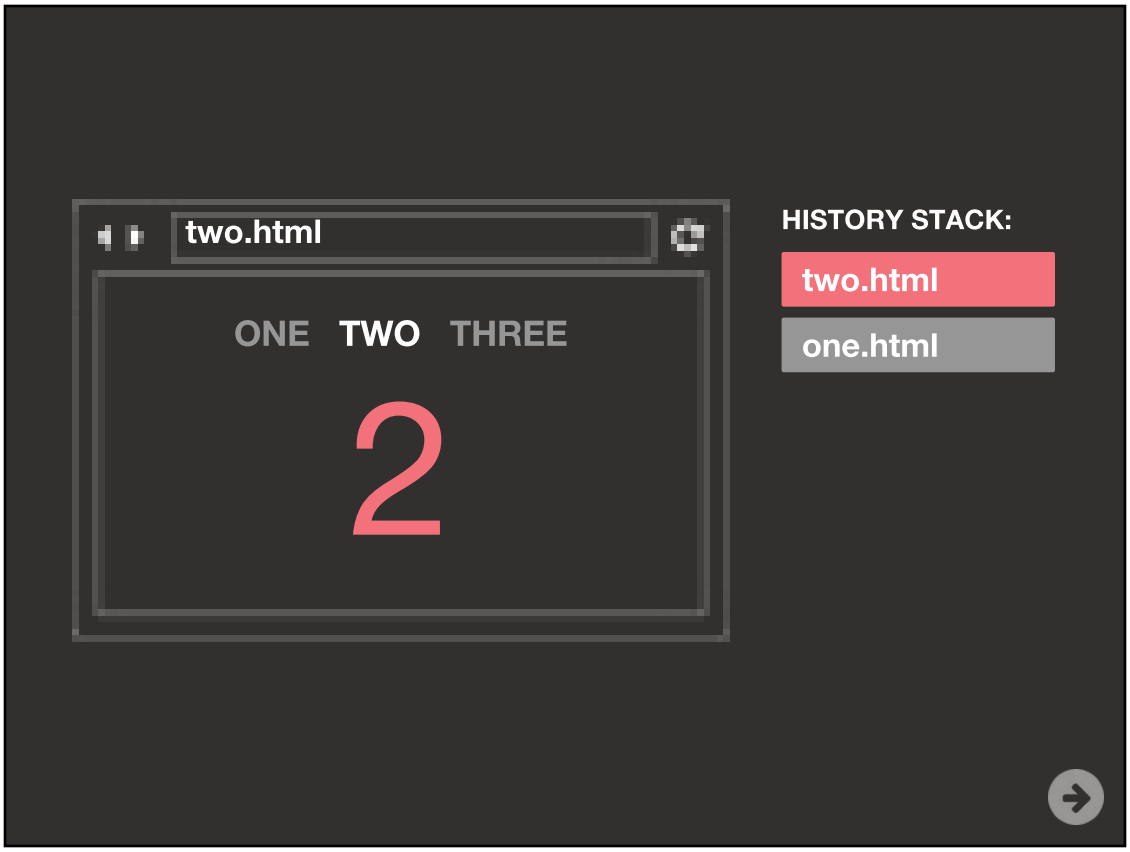
```
localStorage.setItem(key, value)
sessionStorage.setItem(key, value)

localStorage.getItem(key)
sessionStorage.getItem(key)
```

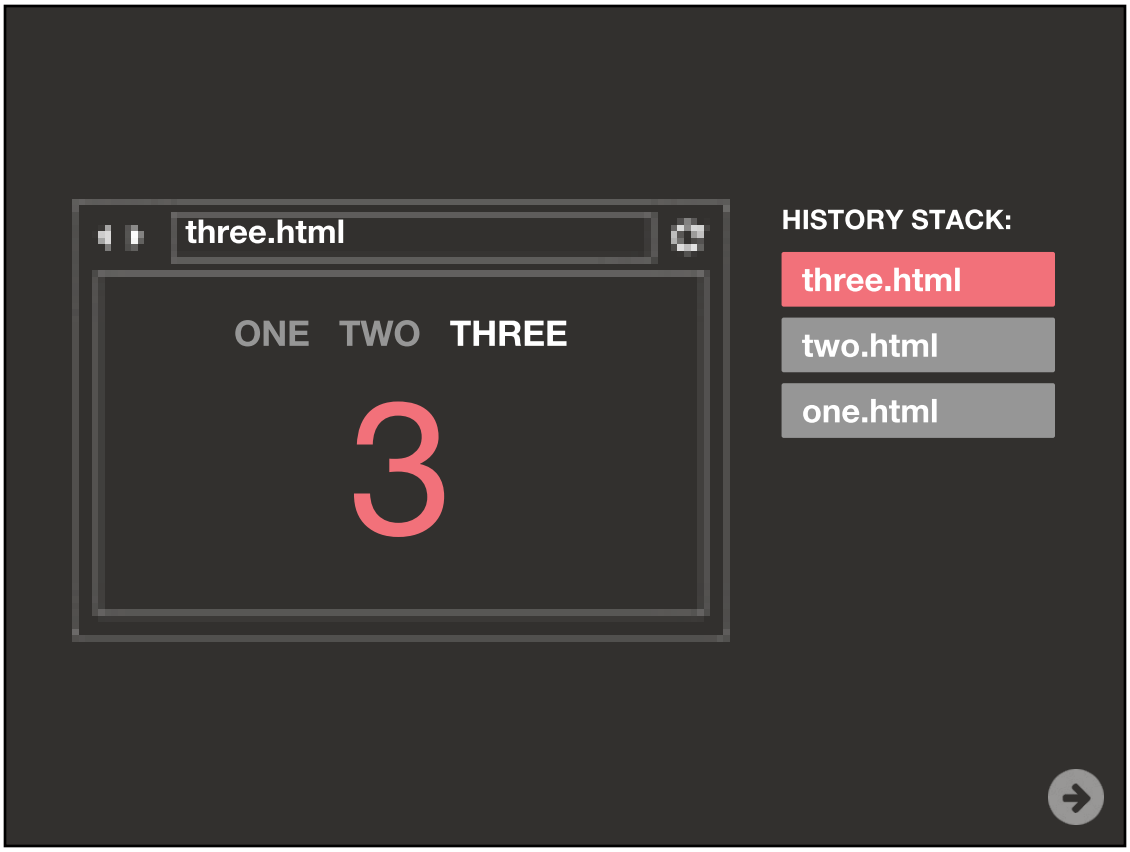
HISTORY API

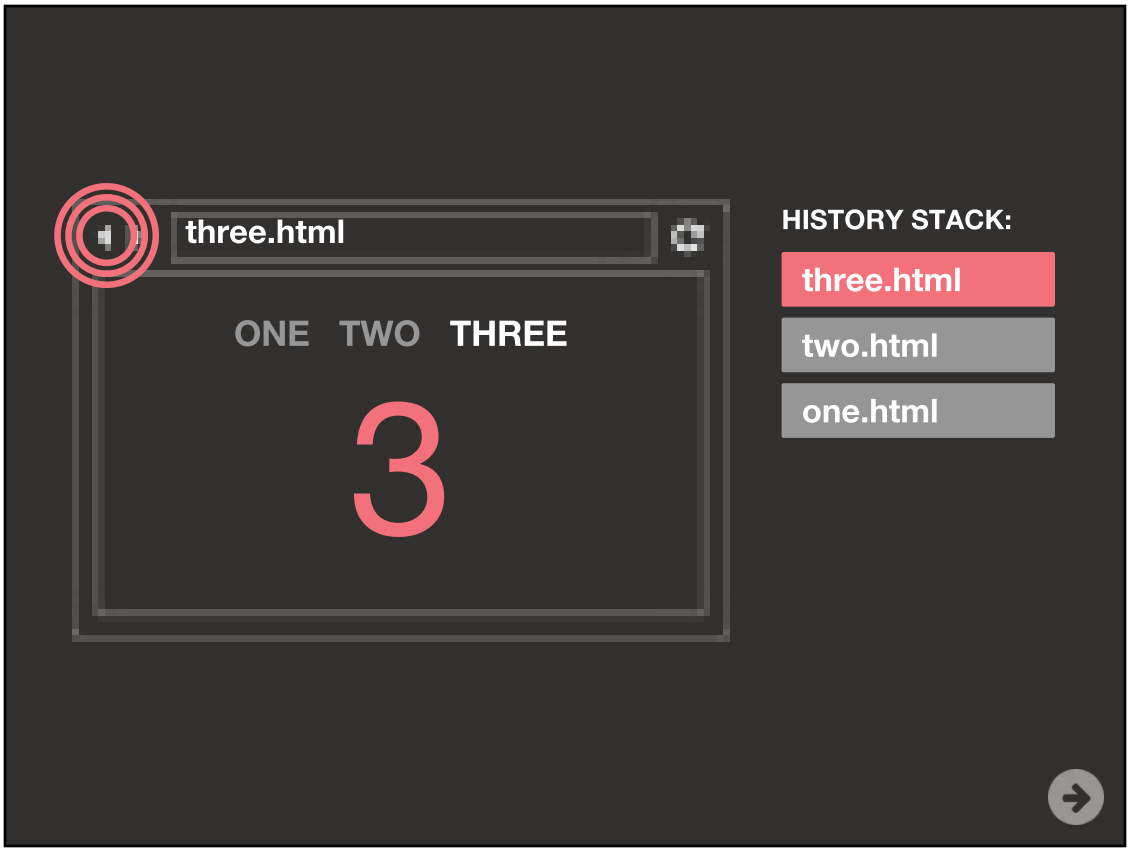
The history API stores which pages a user has visited. It is implemented using the `history` object.

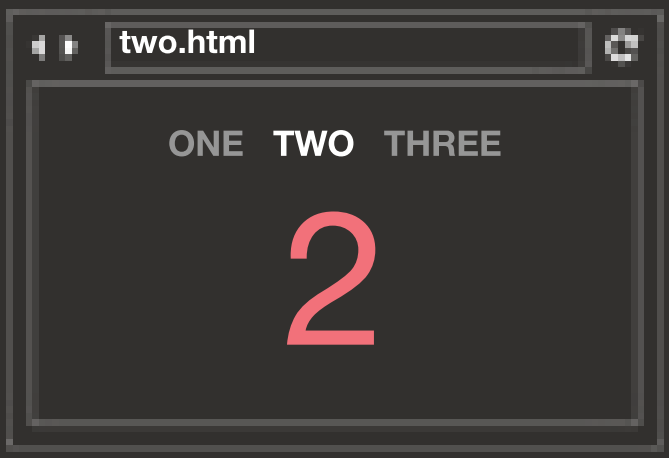
The history API allows Ajax applications to update the URL without loading an entirely new page.


one.html
ONE TWO THREE
1
HISTORY STACK:
one.html


two.html
ONE TWO THREE
2
HISTORY STACK:
two.html
one.html


three.html
ONE TWO THREE
3
HISTORY STACK:
three.html
two.html
one.html


three.html
ONE TWO THREE
3
HISTORY STACK:
three.html
two.html
one.html



HISTORY STACK:

- two.html
- three.html
- one.html

Adding to the `history` object:

```
history.pushState(state, title, url)
```

Pages stored in the `history` object:

```
history.back()
history.forward()
history.go()
```

The `history` object's `popstate` event fires when the contents of the `history` object changes.

SCRIPTS WITH APIS

The jQuery Foundation maintains its own set of jQuery plugins called **jQuery UI**.

To make use of the jQuery UI set of plugins, you must first understand:

How to structure your HTML

The method to call to make it work

The HTML for the jQuery UI accordion is structured like this:

```
<div id="prizes">
  <h3>1st Prize</h3>
  <div>First prize in the…</div>
  <h3>2nd Prize</h3>
  <div>Second prize is the…</div>
  <h3>3rd Prize</h3>
  <div>Third prize is the…</div>
</div>
```

The script for the jQuery UI accordion is used like this:

```
<script src="jquery.js"></script>
<script src="jquery-ui.js"></script>

<script>
  $(function() {
    $('prizes').accordion();
  }
</script>
```

AngularJS is a framework that makes it easier to create web apps.

In particular, it helps to create apps that read, write, update, and delete data in a database on the server.

AngularJS is based on a software development approach called **Model View Controller** or **MVC**.

MODEL VIEW CONTROLLER

1: Commands are sent up the chain to update the model
2: The data is synchronized to keep the view updated
3: Change notifications are sent back to the browser via the controller

Here is an example of AngularJS:

```
<form>
  <input ng-model="name" type="text" />
  <textarea ng-model="message">
  </textarea>
</form>

<div class="postcard">
  <div>{{ name }}</div>
  <p>{{ message }}</p>
</div>
```

PLATFORM APIS

Large websites like Facebook, Google, and Twitter let you access and update data that is stored on their platform via APIs.

Google Maps allows you to add maps to your pages.

High-traffic sites that use Google Maps need an API key, as there is a limit to the number of maps they will serve to a domain per day.

The HTML has an element that will be replaced by the map, and a link to a script:

```
<div id="map"></div>

<script src="js/google-map.js">
</script>
```

When the page is ready, the script is called:

```
window.onload = loadScript;
```

It creates the `<script>` element to load the API from Google's servers:

```
function loadScript() {
  var script;
  script = document.createElement('script');
  script.src = 'http://maps.googleapis.com/
  maps/api/js?sensor=false&callback=init';
  document.body.appendChild(script);
}
```

The `init()` function creates a `mapOptions` object to store map setup data:

```
function init() {
  var mapOptions = {
    center: new google.maps.LatLng(40.782710,-73.965310),
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    scaleControl: true,
    zoom: 13
  };
  var venueMap;
  var el = document.getElementById('map');
  venueMap = new google.maps.Map(el, mapOptions);
}
```

You can add to the `mapOptions` object to make the map more tailored to your needs.

```
styles: [
  {
    stylers: [
      { hue: "#00ff6f" },
      { saturation: -50 }
    ]
  }, {
    featureType: "road",
    elementType: "geometry",
    stylers: [
      { lightness: 100 },
      { visibility: "simplified" }
    ]
  }
]
```