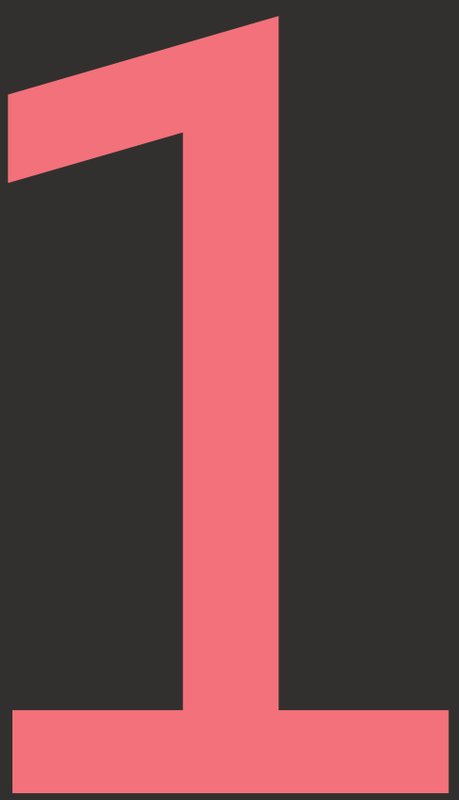
Web Applications Development

# AJAX & JSON

# WHAT IS AJAX?

# Ajax lets you...



# 1

Request data
from a server

1

2

Request data
from a server

1

Request data from a server

2

Load it without refreshing the entire page

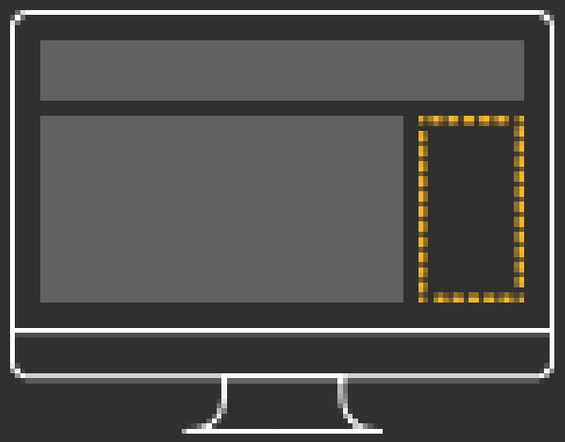
It uses an asynchronous processing model.

(Users can do other things while the data is loading.)

1

# THE REQUEST

The browser requests information from the server



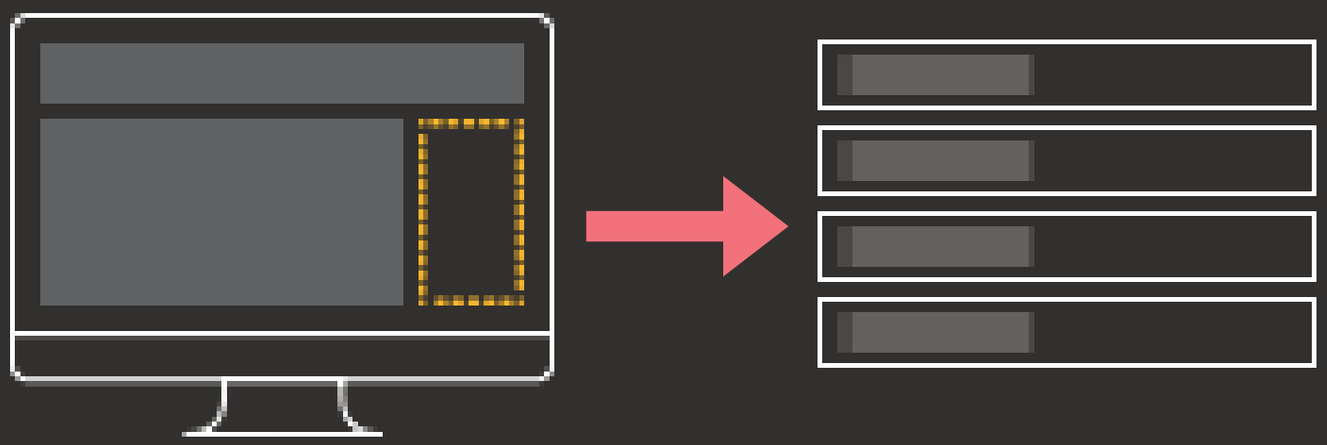
1

THE REQUEST

The browser requests information from the server

ON THE SERVER

The server responds with data (usually HTML, XML, or JSON)



1

THE REQUEST

The browser requests information from the server

ON THE SERVER

The server responds with data (usually HTML, XML, or JSON)

2

THE RESPONSE

The browser processes the content and adds it to the page

# REQUEST

Web browsers use the `XMLHttpRequest` object to implement Ajax functionality.

Here, an instance of the object is stored in a variable called `xhr`:

```
var xhr = new XMLHttpRequest;
```

The `.open()` method **prepares** the request:

```
var xhr = new XMLHttpRequest;
xhr.open('GET', 'test.json', true);
```

The first argument can be either HTTP `GET` or `POST`:

```
var xhr = new XMLHttpRequest;
xhr.open('GET', 'test.json', true);
```

The second argument specifies the file to be loaded:

```
var xhr = new XMLHttpRequest;
xhr.open('GET', 'test.json', true);
```

The third argument states whether the request is asynchronous or not:

```
var xhr = new XMLHttpRequest;
xhr.open('GET', 'test.json', true);
```

An additional line is then written to send the request:

```
var xhr = new XMLHttpRequest;
xhr.open('GET', 'test.json', true);
xhr.send('search=arduino');
```

# RESPONSE

When the server has responded, the `onload` event calls an anonymous function:

```
xhr.onload = function() {
   // process response
};
```

A property of the object called `status` is then used to make sure the data loaded okay:

```
xhr.onload = function() {
   if (xhr.status === 200) {
      // process response
   }
};
```

IE9 was the first version of IE to support this way of dealing with Ajax responses.

# DATA FORMATS:

# HTML

HTML is the simplest way to get data into a page:

```
<div class="event">
    <img src=“img/map-ny.png"
         alt="New York, NY” />
    <p><b>New York, NY</b>
    <br>May 30</p>
</div>
```

It is available in the `responseText` property of the object:

```
$el.innerHTML = xhr.responseText;
```

The browser renders this HTML like any other HTML - no extra work required.

# DATA FORMATS:

# XML

XML looks like HTML but the tags contain different words:

```
<event>
    <location>New York, NY</location>
    <date>May 15</date>
    <map>img/map-ny.png</map>
</event>
```

It is available in the `responseXML` property of the object:

```
var events = xhr.responseXML;
```

You need to write JavaScript to convert the XML data into HTML so it can be displayed.

# DATA FORMATS: JSON

## JavaScript Object Notation

JSON looks like object literal syntax but it is just data, not an object:

```
{
  “location”: “New York, NY”,
  “date”: “May 30”,
  “map”: “img/map-ny.png”
}
```

It is available in the `responseText` property of the object:

```
var events = xhr.responseText;
```

You need to write JavaScript to convert the JSON into HTML so it can be displayed.

JSON data is made up of keys and values:

```
{
  “location”: “New York, NY”,
  “date”: “May 30”,
  “map”: “img/map-ny.png”
}
```

JSON data is made up of keys and values:

```
{
    “location”: “New York, NY”,
    “date”: “May 30”,
    “map”: “img/map-ny.png”
}
```

The value can be a string, number, Boolean, array, object or null.

You can nest objects.

```
{
  “events”: [
    {
      “location”: “Austin, TX”,
      “date”: “May 15”,
      “map”: “img/map-tx.png”
    },
    {
      “location”: “New York, NY”,
      “date”: “May 30”,
      “map”: “img/map-ny.png”
    }
  ]
}
```

```
{
  "events": [
    {
      "location": "Austin, TX",
      "date": "May 15",
      "map": "img/map-tx.png"
    },
    {
      "location": "New York, NY",
      "date": "May 30",
      "map": "img/map-ny.png"
    }
  ]
}
```

```
{
  "events": [
    {
      "location": "Austin, TX",
      "date": "May 15",
      "map": "img/map-tx.png"
    },
    {
      "location": "New York, NY",
      "date": "May 30",
      "map": "img/map-ny.png"
    }
  ]
}
```

```
{
  "events": [
    {
      "location": "Austin, TX",
      "date": "May 15",
      "map": "img/map-tx.png"
    },
    {
      "location": "New York, NY",
      "date": "May 30",
      "map": "img/map-ny.png"
    }
  ]
}
```

JavaScript has a `JSON` object with two important methods:

1: Convert a JavaScript object to a string:

```
JSON.stringify();
```

2: Convert a string to a JavaScript object:

```
JSON.parse();
```

# JSONP

Ajax only works with data from the same domain. To get around this, you can use JSONP (JSON with Padding).

First, a function is included in the HTML page to process the JSON data and display it on the page:

```
<script>
   function showEvents(data) {
      // code to process & display data
   }
</script>
```

Next, a `<script>` element calls the JSON data from a remote server:

```
<script>
  function showEvents(data) {
    // code to process & display data
  }
</script>

<script
  src="http://example.org/jsonp">
</script>
```

The script then calls the function that was in the browser and passes the data to it as an argument:

```
showEvents({
  “events”: [
    {
      “location”: “New York, NY”,
      “date”: “May 30”,
      “map”: “img/map-ny.png”
    }...
  ]
});
```