AJAX

You'll learn to love it!

What is it?

- Ajax stands for **A**synchronous **Ja**vascript and **X**ML.

- It lets you send and receive information from the server without reloading the page, like an invisible front-end ninja.

What is it?

- It’s not limited to XML! You can also send and receive other data types like JSON, HTML, and files.

- It’s an important tool for communication between the front end and the back end.

When would you use it?

Anytime you want to send information between the client and the server without leaving the current page.

When would you use it?

For example:

- displaying new messages in a chat (Facebook)
- auto-complete in a search field (Google)
- refining search results (Airbnb)

The XMLHttpRequest object

Ajax relies on a JavaScript class called `XMLHttpRequest`.

To make an Ajax request, you have to create an instance of the `XMLHttpRequest`:

```
var httpRequest = new XMLHttpRequest();
httpRequest.open("GET", "https://ironhack-characters.herokuapp.com/characters");
```

How to use it

These are the basic steps of an Ajax request:

1. Function is called

```
document.querySelector('.js-characters').onclick = fetchCharacters
```

2. XMLHttpRequest object is created

```
function fetchCharacters () {
  var httpRequest = new XMLHttpRequest();
}
```

How to use it (continued)

3. XMLHttpRequest object makes asynchronous request to web server

```
(function makeRequest() {
  httpRequest.onreadystatechange = handleResponse;
  httpRequest.open('GET', 'https://ironhack-characters.herokuapp.com/characters');
  httpRequest.send();
})();
```

How to use it (continued)

4. When the callback function is called with `readyState` 4, the response has been received.

```
function handleResponse(response) {
  if (httpRequest.readyState === 4) {
    if (httpRequest.status === 200) {
      showCharacters(JSON.parse(httpRequest.responseText))
    } else {
      alert("There was an error");
    }
  }
}
```

IRON HACK

How to use it (continued)

4. The response is available as `responseText`. We will talk about `JSON.parse()` a bit later.

```
function handleResponse(response) {
  if (httpRequest.readyState === 4) {
    if (httpRequest.status === 200) {
      showCharacters(JSON.parse(httpRequest.responseText))
    } else {
      alert("There was an error");
    }
  }
}
```

IRON HACK

How to use it (continued)

5. The callback function does something with the response data.

```
function showCharacters (characters) {
  characters.forEach(function appendLi (chr) {
    // create new DOM node and inner HTML for each character
    // appended each one to the character list
  });
}
```

Full example

Ajax with jQuery

Ajax with jQuery

We’re going to use jQuery to make our ajax requests.
These three methods are your new best friends:

```
$.get()
$.post()
$.ajax()
```

Ajax GET request in jQuery

```
$.ajax({
  url: 'https://ironhack-characters.herokuapp.com/characters',
  data: '',
  success: handleCharacters, // do something with the response
  error: function() {
     console.log('error!')
  },
  dataType: 'json'
});
```

What if you want to POST?

```
$.ajax({
  type: "POST",
  url: "https://ironhack-characters.herokuapp.com/characters",
  data: newCharacter, // the object where we saved our data
  success: onSaveSuccess, // success handler
  error: function() {
      console.log('error!')
  },
  dataType: "json"
});
```

Alternatives to $.ajax

- jQuery makes it *even easier*!
- **$.get** for GET requests
- **$.post** for POST requests

$.get(url [, data] [, success] [,dataType)

The very simplest way to use this method is:

```
$.get('https://ironhack-characters.herokuapp.com/characters')
```

But, we can pass more arguments, and give our AJAX request more information

Sending Data

```
var character = {
  name: "Han Solo",
  occupation: "Smuggler"
}

$.get("https://ironhack-characters.herokuapp.com/characters", character);
```

Success Callback

```
var character = {
  name: "Han Solo",
  occupation: "Smuggler"
}

function handleCharacters(characters){
  console.debug("Found characters: " + characters)
}

$.get("https://ironhack-characters.herokuapp.com/characters", character, handleCharacters);
```

Specifying Data Type

```
var character = {
  name: "Han Solo",
  occupation: "Smuggler"
}

function handleCharacters(characters){
  console.debug("REQUEST DONE ", characters)
}

$.get("https://ironhack-characters.herokuapp.com/characters", character, handleCharacters,
"json");
```

Another way of handling success

Define a callback function to process the response.

```
function handleCharacters (characters) {
  console.log('REQUEST DONE', characters);
}
```

Ensure that the callback function executes when the request is successfully completed.

```
var request = $.get('https://ironhack-characters.herokuapp.com/characters');
request.done(handleCharacters);
```

Success handler

Even better, let's update the HTML using jQuery

```
function handleCharacters (characters) {
    characters.forEach(function appendLi (chr) {
      // shortened html for this example
      // see full solution at the end of the slides
      var html = '<li>'+ chr.name + '</li>';
      $('.js-character-list').append(html);
    });
  }
```

Error handler

Define a callback function that will only be executed if the API request fails.

```
function handleError (error) {
  console.error('OH NO!!', error.responseJSON);
}
```

The .fail() method calls the error function for HTML status codes 4XX or 5XX or if the request takes too long.

```
request.fail(handleError);
```

Same thing, this time with $.ajax()

```
$.ajax({
  url: 'https://ironhack-characters.herokuapp.com/characters',
  data: '',
  success: handleCharacters,
  error: handleError,
  dataType: 'json'
});
```

$.post(url [, data] [, success] [, dataType])

What about making a POST request? No problem, we can do that with **$.post()**.

```
var newCharacter = {
    name: 'Chewbacca',
    occupation: 'Muscle',
    weapon: 'Bowcaster'
};
var request = $.post('https://ironhack-characters.herokuapp.com/characters',
newCharacter);
```

$.post() callback functions

Don’t forget to set the success and error handlers!

```
function onSaveSuccess (response) {
  console.debug('Saved!', response);
}
function onSaveFailure (error) {
  console.error(error.responseJSON);
}
request.done(onSaveSuccess);
request.fail(onSaveFailure);
```

Same thing, this time with $.ajax()

```
$.ajax({
  type: "POST",
  url: "https://ironhack-characters.herokuapp.com/characters",
  data: newCharacter,
  success: onSaveSuccess,
  error: onSaveFailure,
  dataType: "json"
});
```

JSON

What is it?

JSON is short for **J**ava**S**cript **O**bject **N**otation, and it is a way to store information in an organized, easy-to-access manner. It lets you send objects and arrays with ajax.

Check out this Pokemon API to see what JSON looks like: **http://pokeapi.co**

How to use it - with pure JS

When you want to send data that is not a String, for example an object, you can use **JSON.stringify().**

When you receive a response in JSON format, you can parse it with **JSON.parse().**

```
var characters = JSON.parse(httpRequest.responseText);
showCharacters(characters);
```

IRON HACK

How to use it - with jQuery

Load JSON from a server with an ajax request by using $.get(url)

Parse the response with $.parseJSON(json)

```
$.get('https://ironhack-characters.herokuapp.com/characters', function(data) {
  var characters = $.parseJSON(data);
  handleCharacters(characters);
}
```

In Rails

If you request JSON from your rails app, you need to edit your controller to provide a response in JSON.

```
def create
    @user = User.new(user_params)
    respond_to do |format|
      if @user.save
        format.html { redirect_to root_path, notice: "Thanks for registering!" }
        format.json { render json: @user, status: :created}
      else
        format.html { redirect_to root_path, notice: "Please try again." }
        format.json { render json: @user.errors, status: :unprocessable_entity}
      end
    end
  end
```

Exercise: Ajax and the DOM

- Create an HTML page to display the characters from the Ironhack-characters API
  - https://ironhack-characters.herokuapp.com/characters
- Create a form to add new characters to the API
- Update the HTML whenever a new character is added
- Use Ajax for everything!

Hint to get you started

Solution