



## **IN5320 - Development in Platform Ecosystems**

Lecture 3: *json, ajax, APIs*

3rd of September 2018

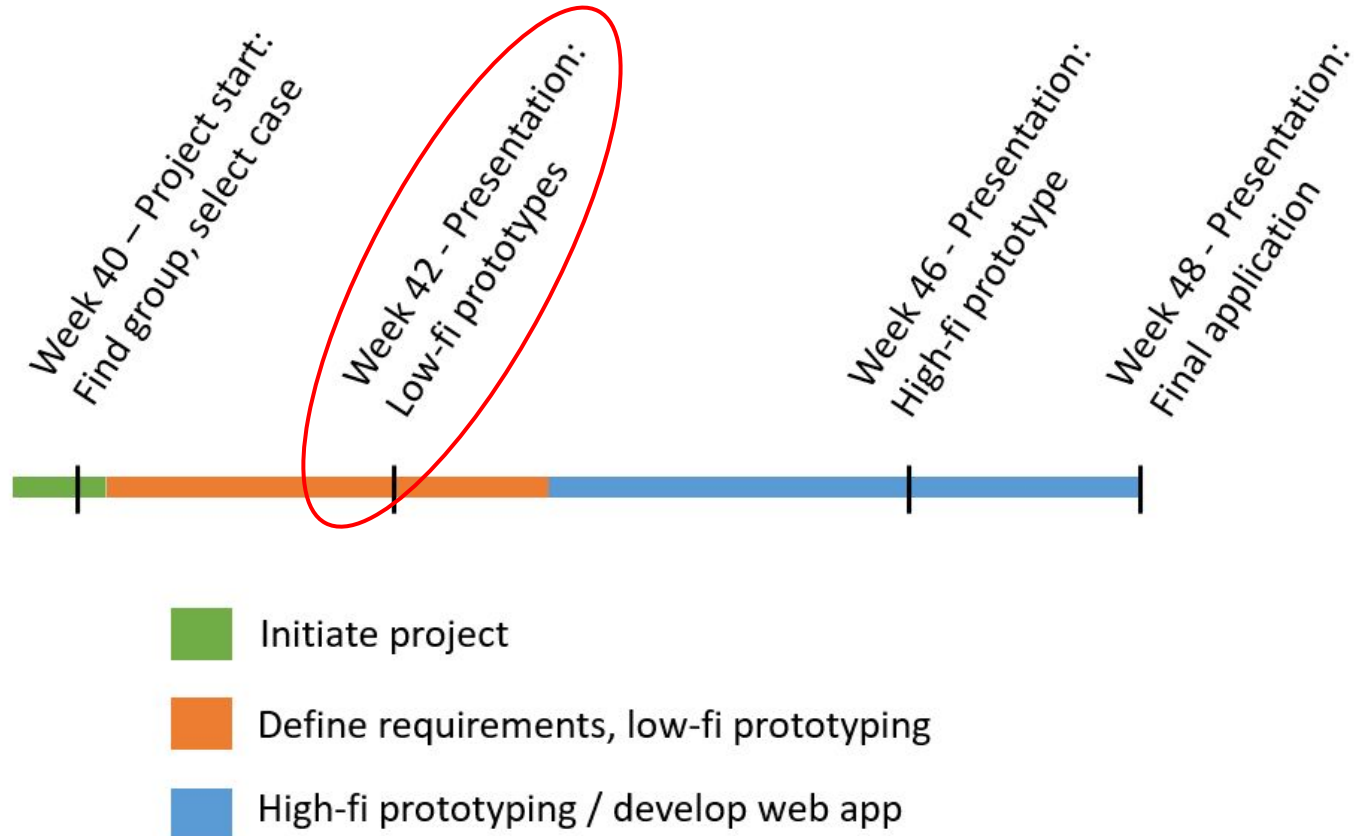
Department of Informatics, University of Oslo

Magnus Li - [magl@ifi.uio.no](mailto:magl@ifi.uio.no)

# Today's lecture

1. Objects and Json
2. Ajax and APIs
3. Deferred and Promise

# First presentation moved to week 42!



# Objects and JSON

# JavaScript objects

- JavaScript allows us to create objects.
- Objects in JavaScript is just a collection of key - value pairs /named values

```
var room = {  
  name:"Ada",  
  number:3407,  
  floor:3,  
  type:"Datastue"  
};
```

```
//Access variable  
room.name;  
  
//Change variable  
room.name = "Lisp";
```

# JavaScript objects

- We can at any time add new variables to our object.

```
var room = {  
  name:"Ada",  
  number:3407,  
  floor:3,  
  type:"Datastue"  
};
```

```
//Add new variable  
room.size = 35;
```

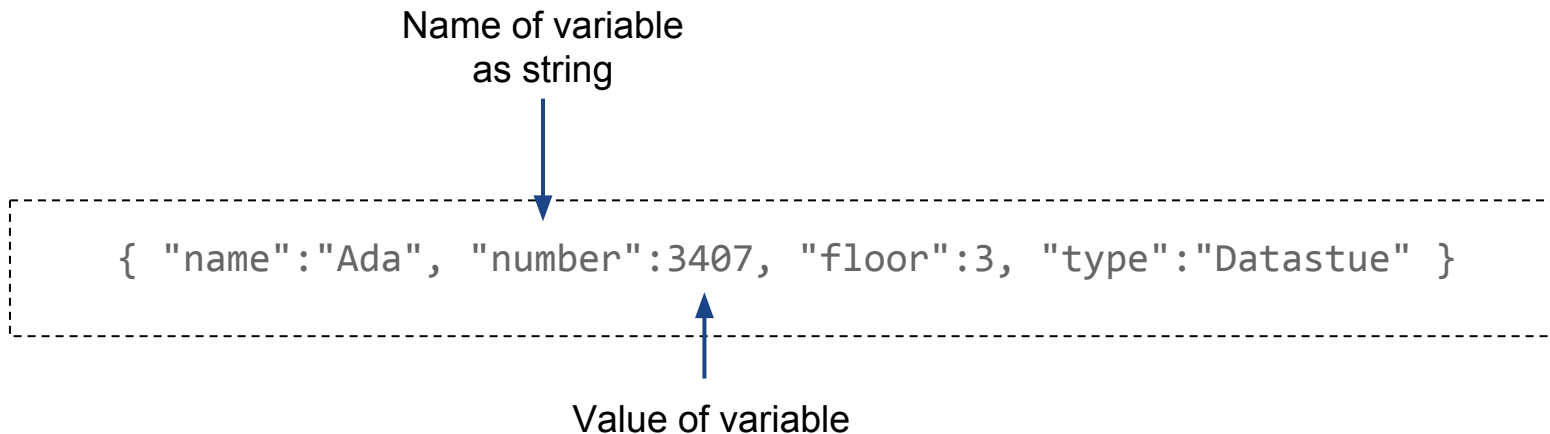
# JavaScript object methods

- Objects can also contain functions

```
var room = {  
  name: "Ada",  
  number: 3407,  
  floor: 3,  
  type: "Datastue",  
  getDescription: function() {  
    return this.name + " is a " + this.type + " located on floor " + this.floor;  
  },  
};
```

# JavaScript Object Notation (JSON)

- JSON is a syntax for storing and exchanging data.
- In text-format using the JavaScript object notation standard.





# JSON nested objects

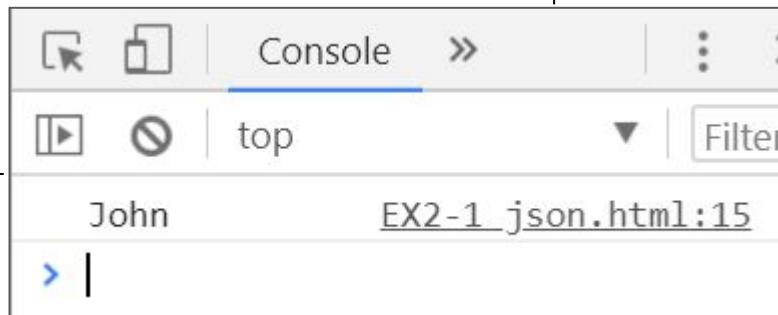
- JSON objects can contain arrays and new objects
- In the example below, we have an object “rooms” with an array of three objects representing different rooms.

```
{ "rooms": [  
  { "name": "John", "number": "3407", "type": "Datastue" },  
  { "name": "Awk", "number": "3118", "type": "Møterom" },  
  { "name": "Assembler", "number": "3417", "type": "Terminalstue" }  
]}
```

# JSON + JavaScript

- JSON is convenient since the format is immediately compatible with JavaScript.
- In the example below, we store the JSON in a variable.
- We can access the variables of the objects as a normal JavaScript object.

```
var ifi = {"rooms":[  
  { "name":"John", "number":"3407", "type":"Datastue" },  
  { "name":"Awk", "number":"3118", "type":"Møterom" },  
  { "name":"Assembler", "number":"3417", "type":"Terminalstue" }  
]};  
console.log(ifi.rooms[0].name);
```



# JSON parse and stringify

- Often, JSON is stored as a string in a local text file, or transferred in pure text from the server.
- We can then use `JSON.parse()` to convert it to a JavaScript Object
- Similarly, we can convert a JavaScript object to a JSON string with the `JSON.stringify()` method.

```
var dataAsString = '{ "name":"Ada", "number":3407, "floor":3, "type":"Datastue" }';  
  
var dataAsJSObject = JSON.parse(dataAsString);  
  
var stringAgain = JSON.stringify(dataAsJSObject);
```

# JSON parse and stringify



**Our app in the browser**

Request some data



**Some web-server**

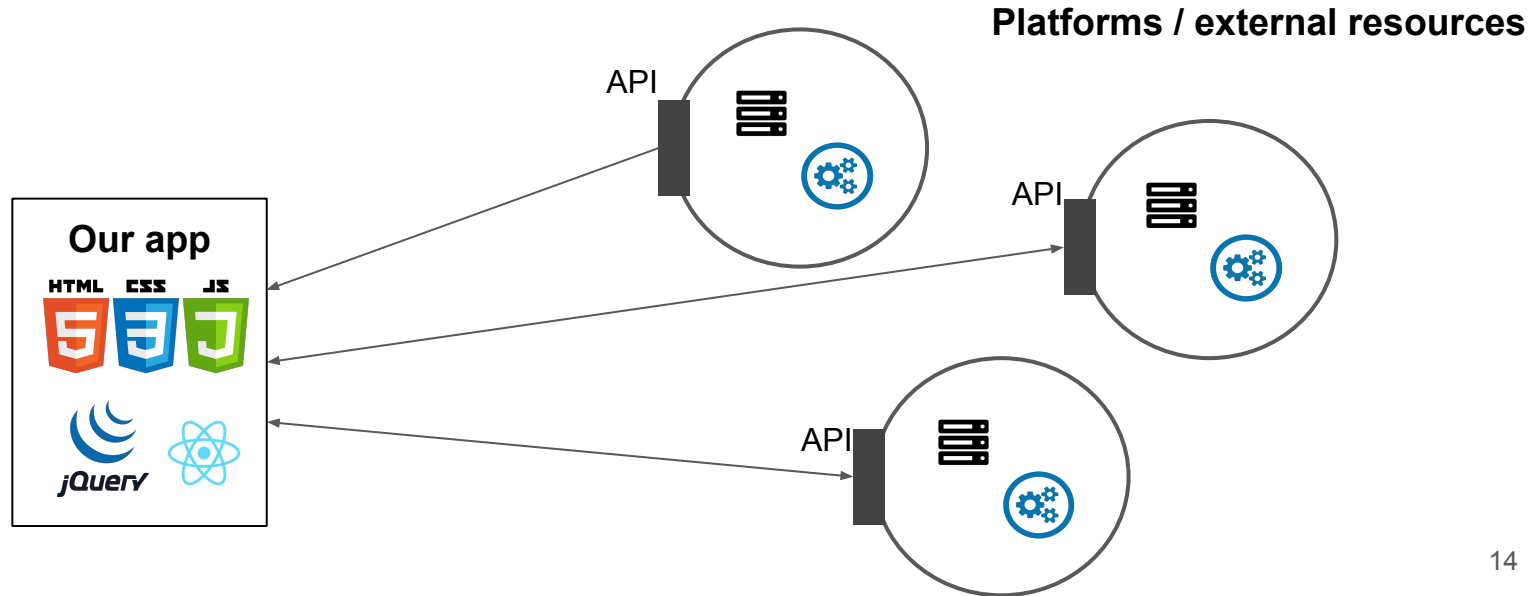
JSON.parse()

JSON in string format

# WHAT TO SEND, AND HOW?

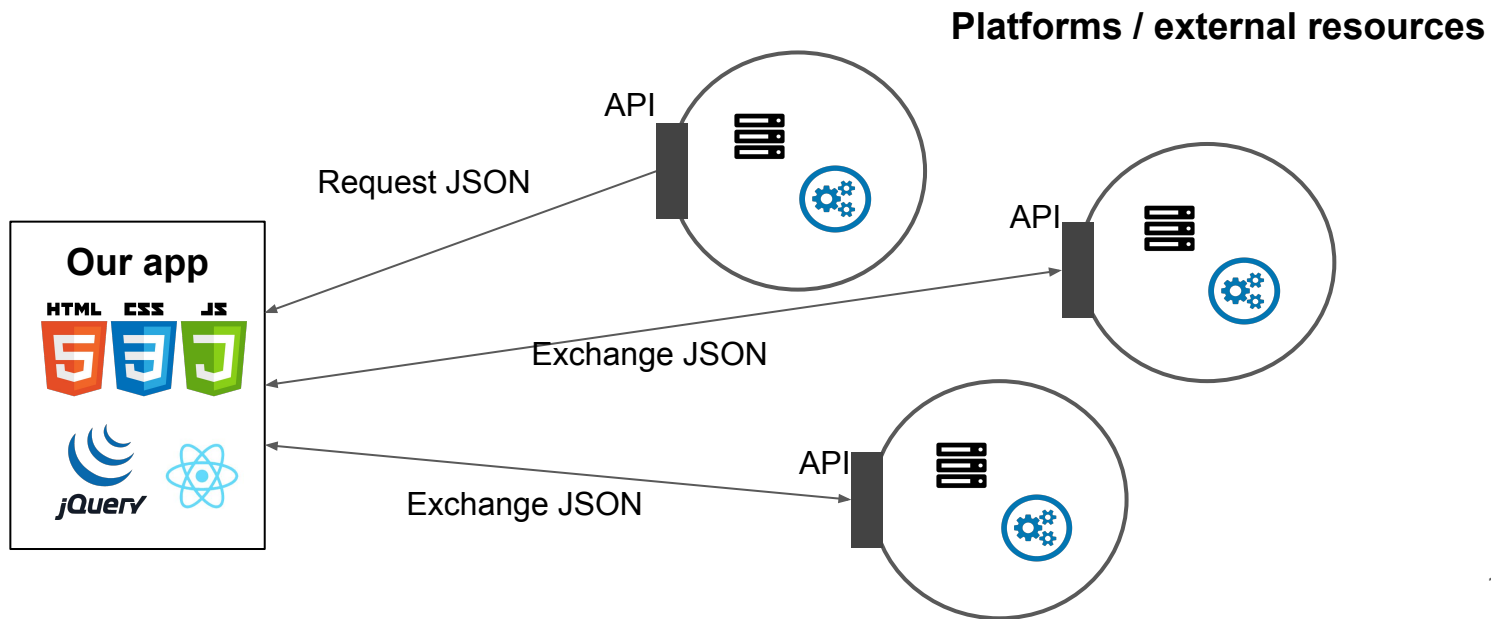
# Development in Platform Ecosystems

- This course focus on developing applications within platform ecosystems.
- We communicate with other resources within these platforms using APIs
- These APIs can provide us with data, or we can send data to them to interact with the platforms core resources, or other components.

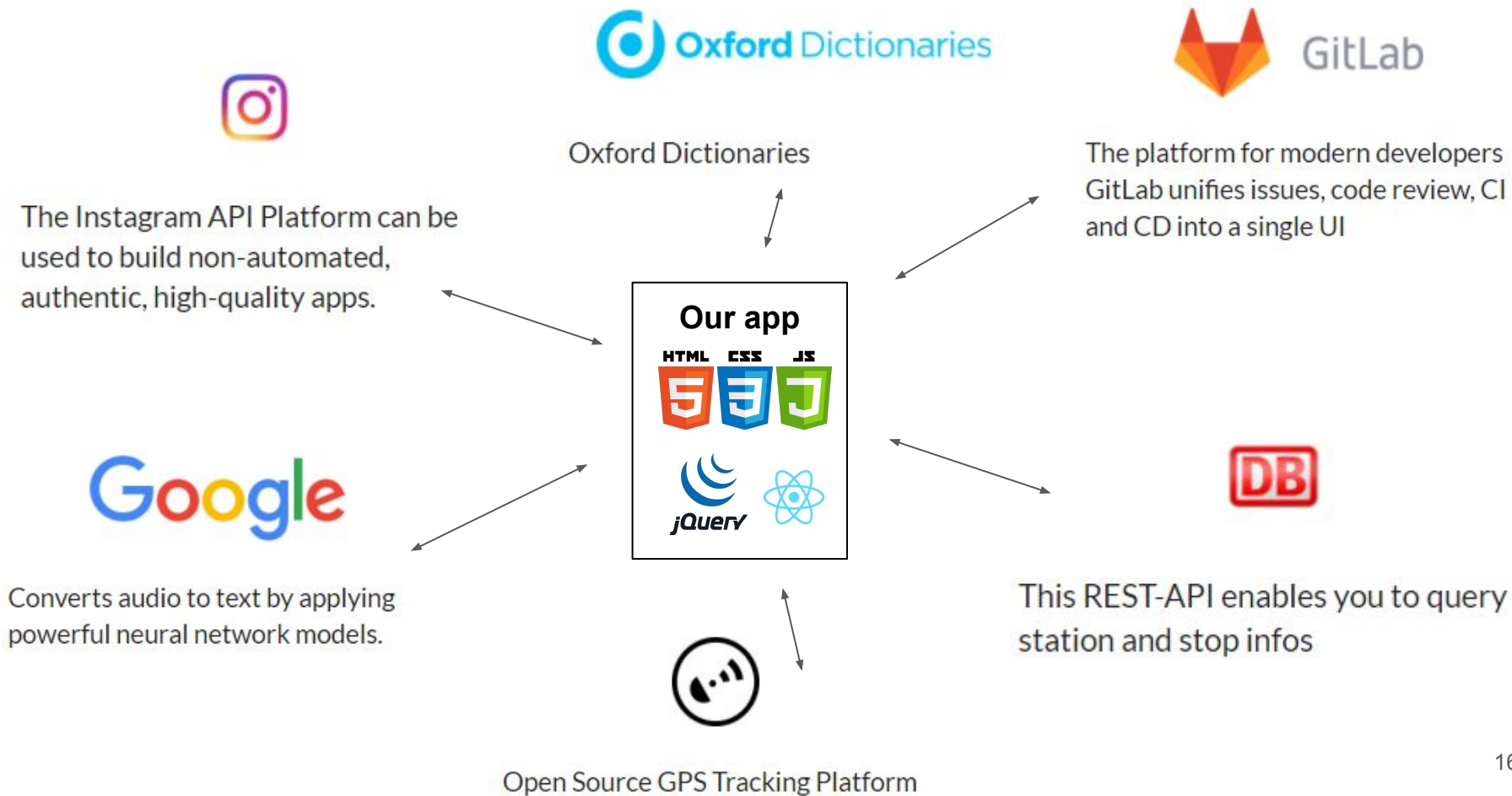


# Development in Platform Ecosystems

- This exchange of information is often reliant on JSON.



# APIs





AJAX

- AJAX = Asynchronous JavaScript And XML
- Asynchronous in that requests can run in parallel with the main thread.
  - Transfer of data can happen without affecting other dynamic components of the web-application.
- Allows transfer of data in formats such as XML, JSON or plain text.
- In essence, ajax allows you to:
  - Read data from a web-server after the web-page has loaded.
  - Update a web page without reloading the page.
  - Send data to a server in the background (without reloading the page).

- AJAX uses the browser built-in XMLHttpRequest object to request data from a server.
- By sending a request to a server, this is processed and data is returned.



**Our app in the browser**



**Some web-server**

**1.**

- Some event, for example button click
- Send HttpRequest



**2.**

- Process HttpRequest.
- Create response, which is sent back to client.



**3.**

- Process returned data with JavaScript
- Update page content

- jQuery has some neat functionality to make AJAX-calls easy


```
$.ajax({  
  dataType: "json",  
  url: url,  
  success: success  
});
```

- ← Type of data, for example json
- ← Url to the location of the data
- ← What to do when data is successfully retrieved.

e.g  
<https://api.chucknorris.io/>

- In this example, we use the API of chucknorris.io to get a random joke.
- Our call returns a json object containing several elements.

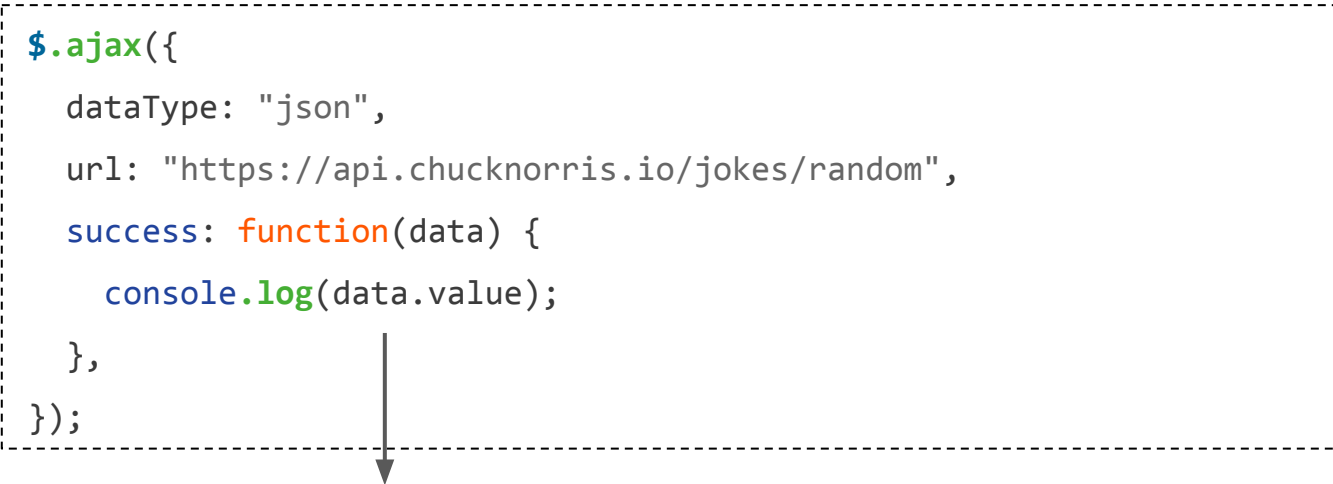
```
$.ajax({  
  dataType: "json",  
  url: "https://api.chucknorris.io/jokes/random",  
  success: function(data) {  
    console.log(data);  
  },  
});
```



```
category: null  
icon_url: "https://assets.chucknorris.host/img/avatar/chuck-norris.png"  
id: "zGleww66Q5aE54X2iuRW6w"  
url: "https://api.chucknorris.io/jokes/zGleww66Q5aE54X2iuRW6w"  
value: "Unlike Mr T, Chuck Norris does not pity the fool. Chuck Norris"
```

- data.value gives us the random chuck norris quote

```
$.ajax({  
  dataType: "json",  
  url: "https://api.chucknorris.io/jokes/random",  
  success: function(data) {  
    console.log(data.value);  
  },  
});
```



Aliens DO indeed exist. They just know better than [EX2-1 ajax.html:14](#)  
to visit a planet that Chuck Norris is on.

---

- We can easily present this data in the HTML-document.

```
<body>  
  <p id="norris_joke"></p>  
</body>
```

```
$.ajax({  
  dataType: "json",  
  url: "https://api.chucknorris.io/jokes/random",  
  success: function(data) {  
    $("#norris_joke").text(data.value);  
  },  
});
```

```
<p id="norris_joke"></p>
<button id="get_joke">Get new joke</button>
```

```
$("#get_joke").click(function() {
    $.ajax({
        dataType: "json",
        url: "https://api.chucknorris.io/jokes/random",
        success: function(data) {
            console.log(data.value);
            $("#norris_joke").text(data.value);
        },
    });
});
```



Chuck Norris can compile syntax errors.

Get new joke

# AJAX + jQuery

- jQuery provide an even shorter syntax for retrieving json.

Long

```
$.ajax({
  dataType: "json",
  url: "https://api.chucknorris.io/jokes/random",
  success: function(data) {
    $("#norris_joke").text(data.value);
  },
});
```

Short

```
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {
  $("#norris_joke").text(data.value);
});
```

# AJAX + jQuery

- jQuery provide an even shorter syntax for retrieving json.

URL to data



Success-function



```
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {  
    $("#norris_joke").text(data.value);  
});
```

## Any API

Documentation and Test Consoles for Over 500 Public APIs

Powered by [LucyBot](#) and [APIs Guru](#)

ALL

ANALYTICS

BACKEND

CLOUD

COLLABORATION

CUSTOMER RELATION

DEVELOPER TOOLS

ECOMMERCE

EDUCATION

EMAIL

ENTERPRISE

ENTERTAINMENT

FINANCIAL

HOSTING

IOT

LOCATION

MACHINE LEARNING

MARKETING

Oxford Dictionaries



Oxford Dictionaries

NBA Stats



The destination for current and historic NBA statistics.

Spotify



Our Web API lets your applications fetch data from the Spotify music catalog and manage user's playlists and saved music.

Amadeus Travel Innovation Sandb...



Amadeus Travel Innovation Sandbox

traccar



Open Source GPS Tracking Platform

Books



The Books API provides information about book reviews and The New York Times bestsellers lists.

Rotten Tomatoes



Access Tomatometer, critic reviews and ratings through the Rotten Tomatoes APIs

Instagram



The Instagram API Platform can be used to build non-automated, authentic, high-quality apps.

# Interacting with the API

- Most APIs can be interacted with to provide us specific data
- We can do this by providing variables through the URL.
- In the example below, we use the OMDb API to search for a movie.
- We thus need to provide the title of the movie in the URL
- OMDb require a authorization key, so this is also provided in the URL

## OMDb API

The Open Movie Database

<http://www.omdbapi.com/?t=there+will+be+blood&apikey=574a2d9f>

Search string

Auth key

What do you want to do in the seminar groups?

<https://goo.gl/iLz6BK>

## Example: movie rating search

- Say we write the following HTML code, to enable users to enter the title of a movie, click a button, and the title and IMDB rating will appear in two paragraphs.

```
<body>
  <input id="keyword" type="text" placeholder="Enter movie title"></input>
  <button id="find_button">Find Ratings</button>
  <p id="movie_title"></p>
  <p id="imdb_rating"></p>
</body>
```

## Example: movie rating search

- We first need to add an event-listener on the button.
- Then retrieve the keyword entered by the user in the input-field.

```
$("#find_button").click(function() {  
    var keyword = $("#keyword").val(); //get search-string from input element  
});
```



## Example: movie rating search

- We then write the code to retrieve from the API using ajax.
- The keyword retrieved from the input-field is added to the URL-string in the ajax call.
- To test, we log the retrieved data to the console.

```
$("#find_button").click(function() {  
    var keyword = $("#keyword").val(); //get search-string from input element  
  
    $.getJSON("http://www.omdbapi.com/?t=" + keyword + "&apikey=574a2d9f", function(data) {  
        console.log(data);  
    });  
});
```

Keyword provided by the user

# Example: movie rating search

- Having located the Title in the json data object, we print it to the HTML document.

```
$("#find_button").click(function() {  
    var keyword = $("#keyword").val(); //get search-string from input element  
  
    $.getJSON("http://www.omdbapi.com/?t=" + keyword + "&apikey=574a2d9f", function(data) {  
        $("#movie_title").text(data.Title);  
    });  
});
```

# Example: movie rating search

- We then print the IMDB-rating which is located in the array “Ratings” in the json object.

```
$("#find_button").click(function() {  
    var keyword = $("#keyword").val(); //get search-string from input element  
  
    $.getJSON("http://www.omdbapi.com/?t=" + keyword + "&apikey=574a2d9f", function(data) {  
        console.log(data);  
        $("#movie_title").text(data.Title);  
        $("#imdb_rating").text(data.Ratings[0].Value);  
    });  
});
```

# Example: movie rating search

The Shawshank Redemption

9.3/10

# Deferred and Promise

# Asynchronous calls

- Calls to APIs and happens asynchronously. That is, another thread is created in parallel to the main thread to handle the call.
- This can create challenges when we want to synchronize our events.

Main thread

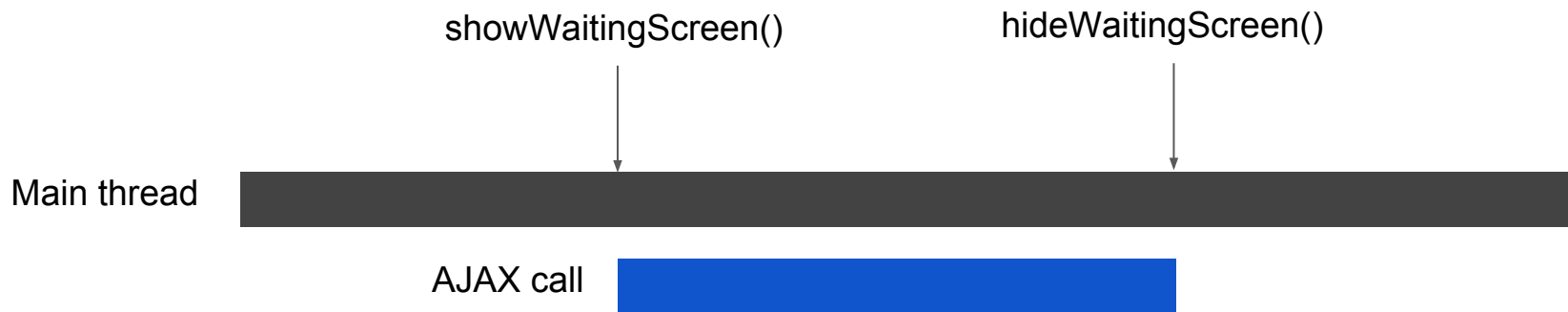


AJAX call



# Asynchronous calls

- Let's say we want to display a waiting screen while our application is requesting and retrieving data from a server.



# Asynchronous calls

Main thread



AJAX call



```
showWaitingScreen();  
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {  
    $("#norris_joke").text(data.value);  
});  
hideWaitingScreen();
```



# Asynchronous calls

Main thread



AJAX call



```
showWaitingScreen();  
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {  
    $("#norris_joke").text(data.value);  
    hideWaitingScreen();  
});
```

# Asynchronous calls

- With several API-calls these *nested* actions can become confusing. Often referred to as “callback hell”
- For example, what if we want to use the data retrieved from one call, to perform another call, and so on.

Main thread

AJAX call



AJAX call



AJAX call



# Asynchronous calls

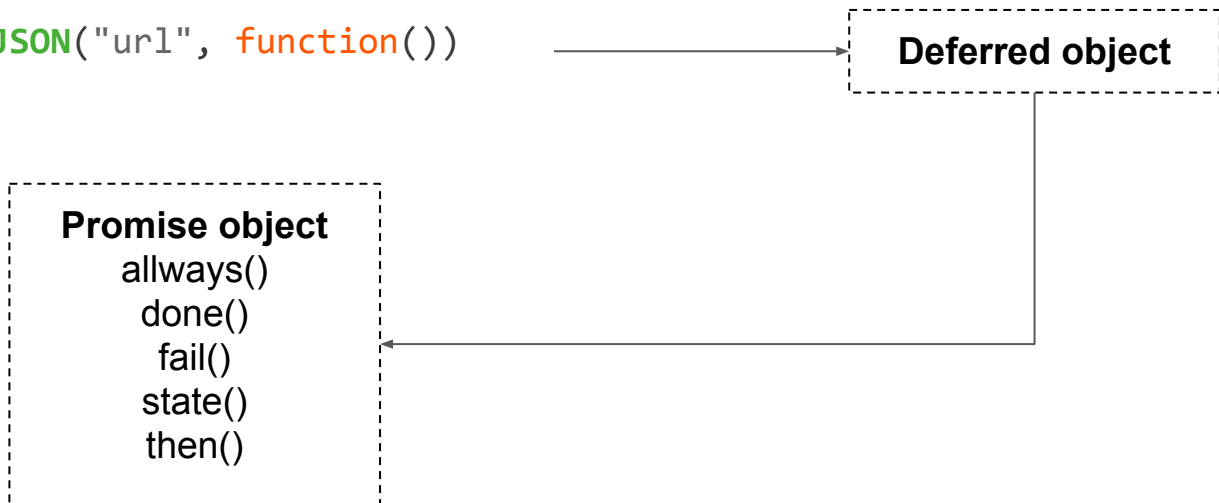
- These *nested* actions can become confusing with increasing functionality and other API-calls. Often referred to as “callback hell”

```
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {  
    $("#norris_joke").text(data.value);  
    $.getJSON("some other api", function(data) {  
        //do something with the data  
        $.getJSON("some other api", function(data) {  
            //do something with this data also  
        });  
    });  
});  
});
```

# Deferred and Promise

- To avoid this, we can use the jQuery Deferred/ Promise.
- Promises are now also available in native JavaScript, but we will here use jQuery.
- Below is a (very) simplified illustration

```
$.getJSON("url", function())
```



# Deferred and Promise

```
$.getJSON("url", function())
```

Deferred object

```
graph LR; A[Deferred object] --> B[Promise object];
```

Promise object

```
always()  
done()  
fail()  
state()  
then()
```

```
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {  
    $("#norris_joke").text(data.value);  
}).then() {  
    //do after resolve  
});
```

# Deferred and Promise

```
$.getJSON("https://api.chucknorris.io/jokes/random", function(data) {  
    $("#norris_joke").text(data.value);  
}).then() {  
    //new api-call  
}).then() {  
    //new api-all  
}).then(){  
    hideWaitingScreen();  
});
```

# Deferred and Promise

```
function getRandomNorrisJoke() {
    return $.getJSON("https://api.chucknorris.io/jokes/random");
}

function displayNewJoke() {
    showWaitingScreen();
    getRandomNorrisJoke().then(function(data) {
        $("#norris_joke").text(data.value);
        hideWaitingScreen();
    });
}

$("#get_joke").click(function() {
    displayNewJoke();
})
```