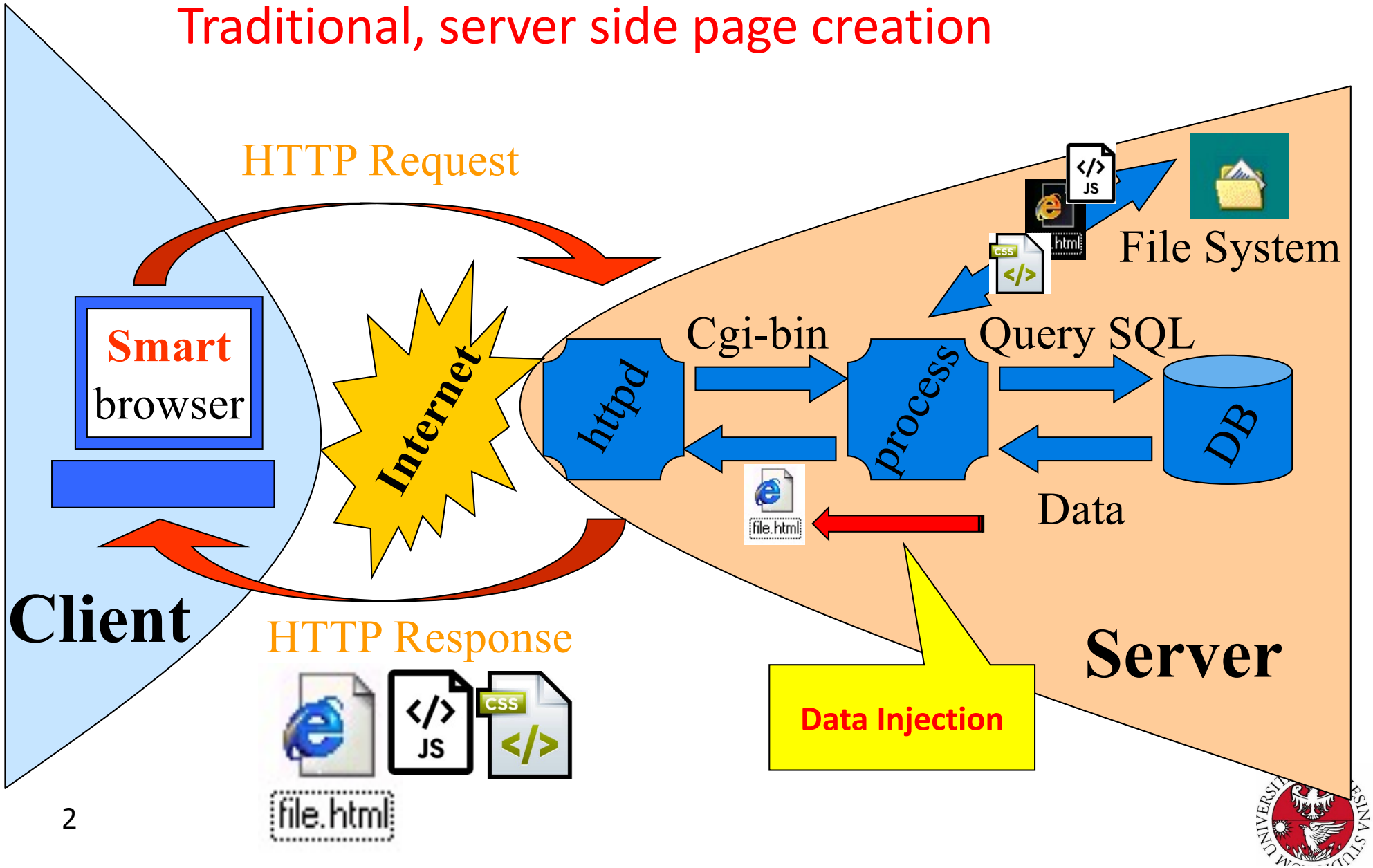
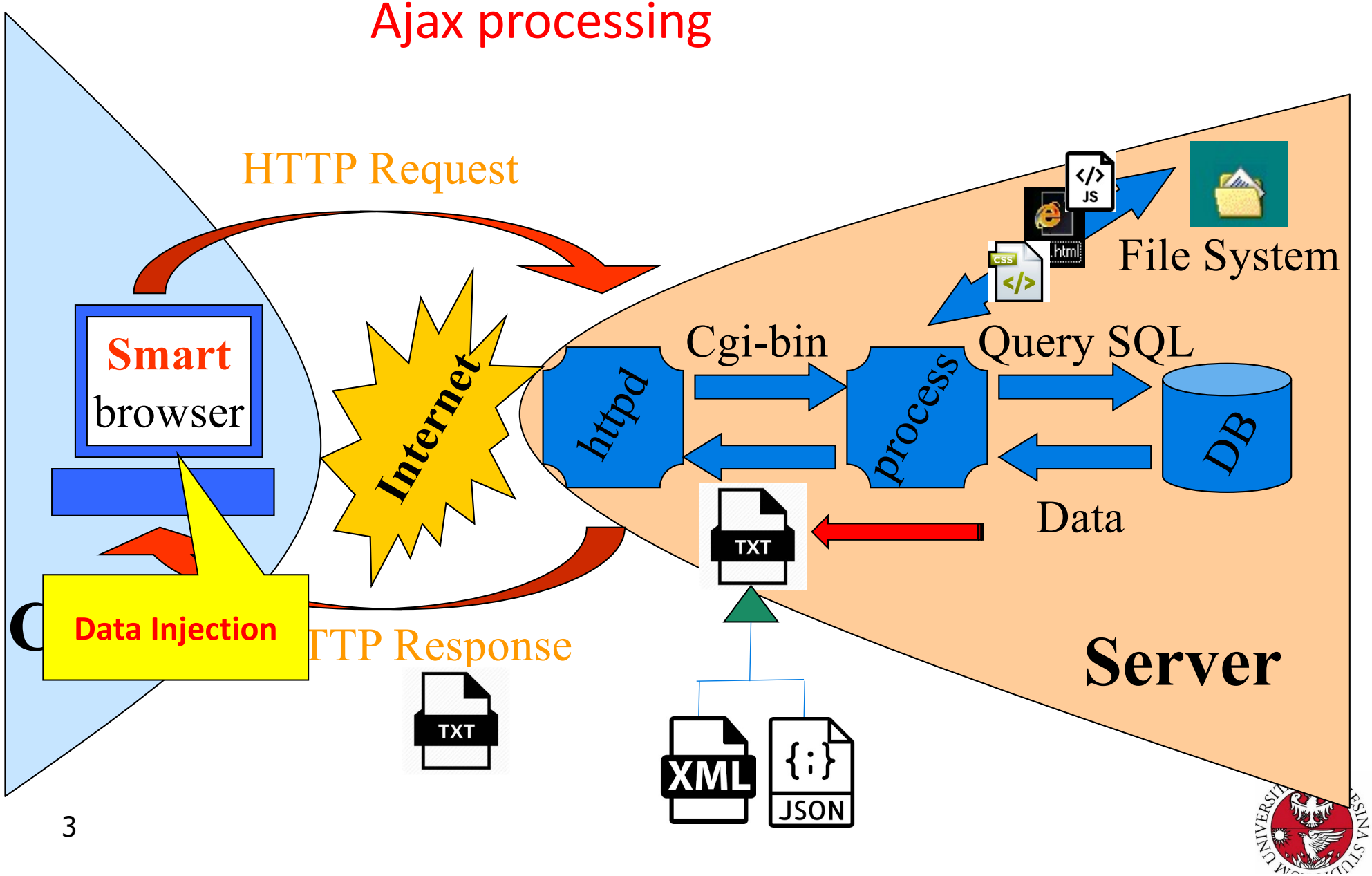
Data transfer: Json

Traditional, server side page creation



Ajax processing



Two main forms of data transfer

XML

```xml
<employees>
 <employee>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
 </employee>
 <employee>
  <firstName>Anna</firstName>
  <lastName>Smith</lastName>
 </employee>
 <employee>
  <firstName>Peter</firstName>
  <lastName>Jones</lastName>
 </employee>
</employees>
```

JSON

```json
{"employees":[
  { "firstName":"John",
    "lastName":"Doe" },
  { "firstName":"Anna",
    "lastName":"Smith" },
  { "firstName":"Peter",
    "lastName":"Jones" }
]}
```

XML vs JSON

Both JSON and XML:

- are "self describing" (human readable)
- are hierarchical (values within values)
- can be parsed and used by lots of programming languages
- can be fetched with an XMLHttpRequest

For AJAX applications, JSON is faster and easier than XML:

XML

Fetch an XML document
Use the XML DOM to loop through the document
Extract values and store in variables

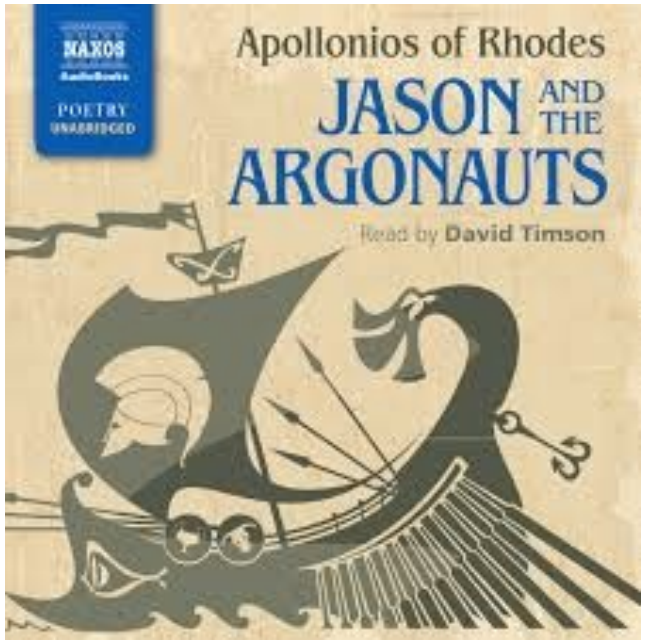
JSON

Fetch a JSON string
JSON.Parse the JSON string

JSON – JavaScript Object Notation

JSON is a language-independent data format.

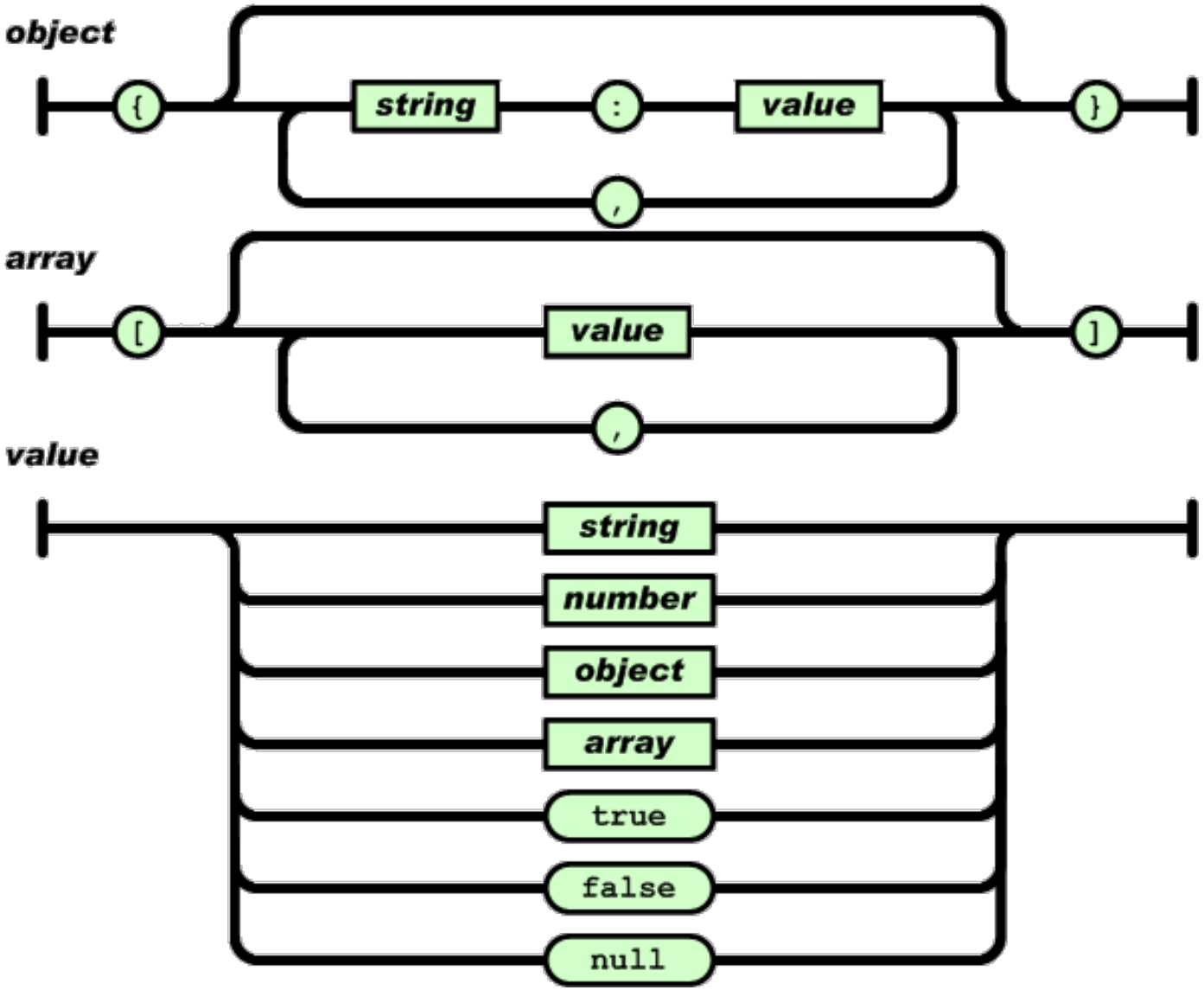
```
{       "name": "Mario",
        "surname": "Rossi",
        "active": true,
        "favoriteNumber": 42,
        "birthday": {
                "day": 1,
                "month": 1,
                "year": 2000
        },
        "languages": [ "it", "en" ]
}
```



Datatypes:

- int, float
- Boolean
- String
- Arrays []
- Associative Arrays {}
- null

JSON


object
{
string
:
value
}
,
array
[
value
]
,
value
string
number
object
array
true
false
null

Parsing JSON in JavaScript

```
var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';
var obj = JSON.parse(text);
obj.birth = new Date(obj.birth);

document.getElementById("demo").innerHTML = obj.name + ", " +
obj.birth;
```

Argo (Parsing JSON in Java)

```
{
   "name": "Black Lace",
   "sales": 110921,
   "totalRoyalties": 10223.82,
   "singles": [
      "Superman", "Agadoo"
   ]
}
```

```
String secondSingle = new JdomParser().parse(jsonText)
    .getStringValue("singles", 1);
```

http://argo.sourceforge.net/index.html

AJAJ

```
var my_JSON_object;
var url=" https://mdn.github.io/learning-
area/javascript/oojs/json/superheroes.json"
var xhttp = new XMLHttpRequest();
xhttp.open("GET", url, true);
xhttp.responseType = "json";
xhttp.onreadystatechange = function () {
  var done = 4, ok = 200;
  if (this.readyState === done && this.status === ok)
  {
    my_JSON_object = this.response;
  }
};
xhttp.send();}
```

An example

https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json


```
{
  "squadName" : "Super Hero Squad",
  "homeTown" : "Metro City",
  "formed" : 2016,
  "secretBase" : "Super tower",
  "active" : true,
  "members" : [
    {
      "name" : "Molecule Man",
      "age" : 29,
      "secretIdentity" : "Dan Jukes",
      "powers" : [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name" : "Madame Uppercut",
      "age" : 39,
      "secretIdentity" : "Jane Wilson",
      "powers" : [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name" : "Eternal Flame",
      "age" : 1000000,
      "secretIdentity" : "Unknown",
      "powers" : [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

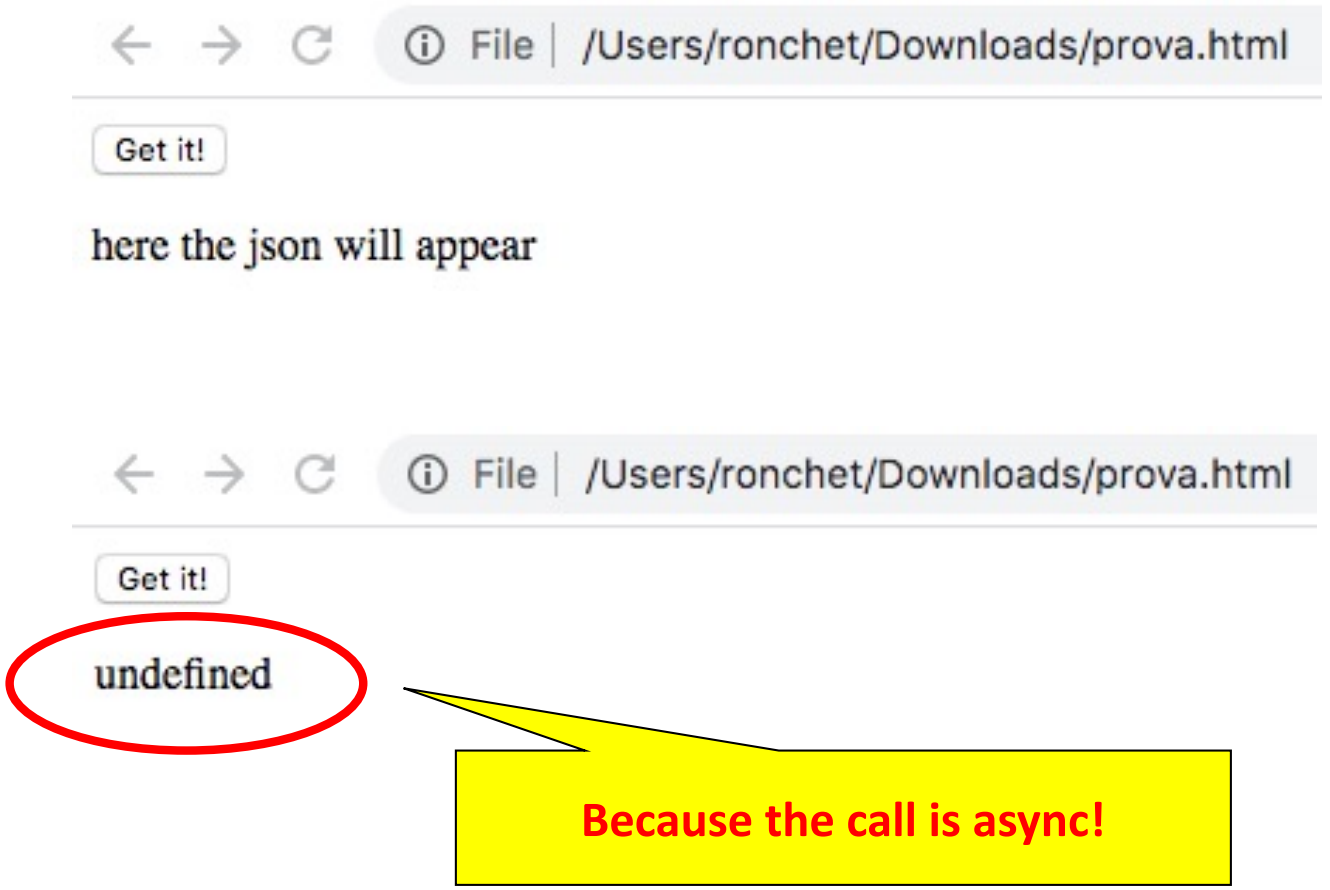
The traps of asynchronous computing 1A

```
<script>
  function getJson() {
    var my_JSON_object;
    var url="https://mdn.github.io/learning-
       area/javascript/oojs/json/superheroes.json"
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", url, true);
    xhttp.responseType = "json";
    xhttp.onreadystatechange = function () {
       var done = 4, ok = 200;
       if (this.readyState === done && this.status === ok){
          my_JSON_object = this.response;
       }
    };
    xhttp.send();
    return my_JSON_object;
  }
</script>
```


The traps of asynchronous computing 1B

```
<!DOCTYPE html>
<head>
    <title>AJAJ Demo</title>
    <script>…</script>
</head>
  <body>
    <form>
     <input type="BUTTON" onClick=
       'document.getElementById("myPar").innerHTML=getJson();'>
    </form>
    <p id="myPar">here the json will appear</p>
  </body>
</html>
```

The traps of asynchronous computing 1 out



The traps of asynchronous computing 2A

```
<script>
  function getJson() {
    var my_JSON_object;
    var url="https://mdn.github.io/learning-
       area/javascript/oojs/json/superheroes.json"
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", url, false);
    xhttp.responseType = "json";
    xhttp.onreadystatechange = function () {
       var done = 4, ok = 200;
       if (this.readyState === done && this.status === ok){
          my_JSON_object = this.response;
       }
    };
    xhttp.send();
    return my_JSON_object;
  }
</script>
```

Let us make it sync

The traps of asynchronous computing 2 out a

The traps of asynchronous computing 2 out b

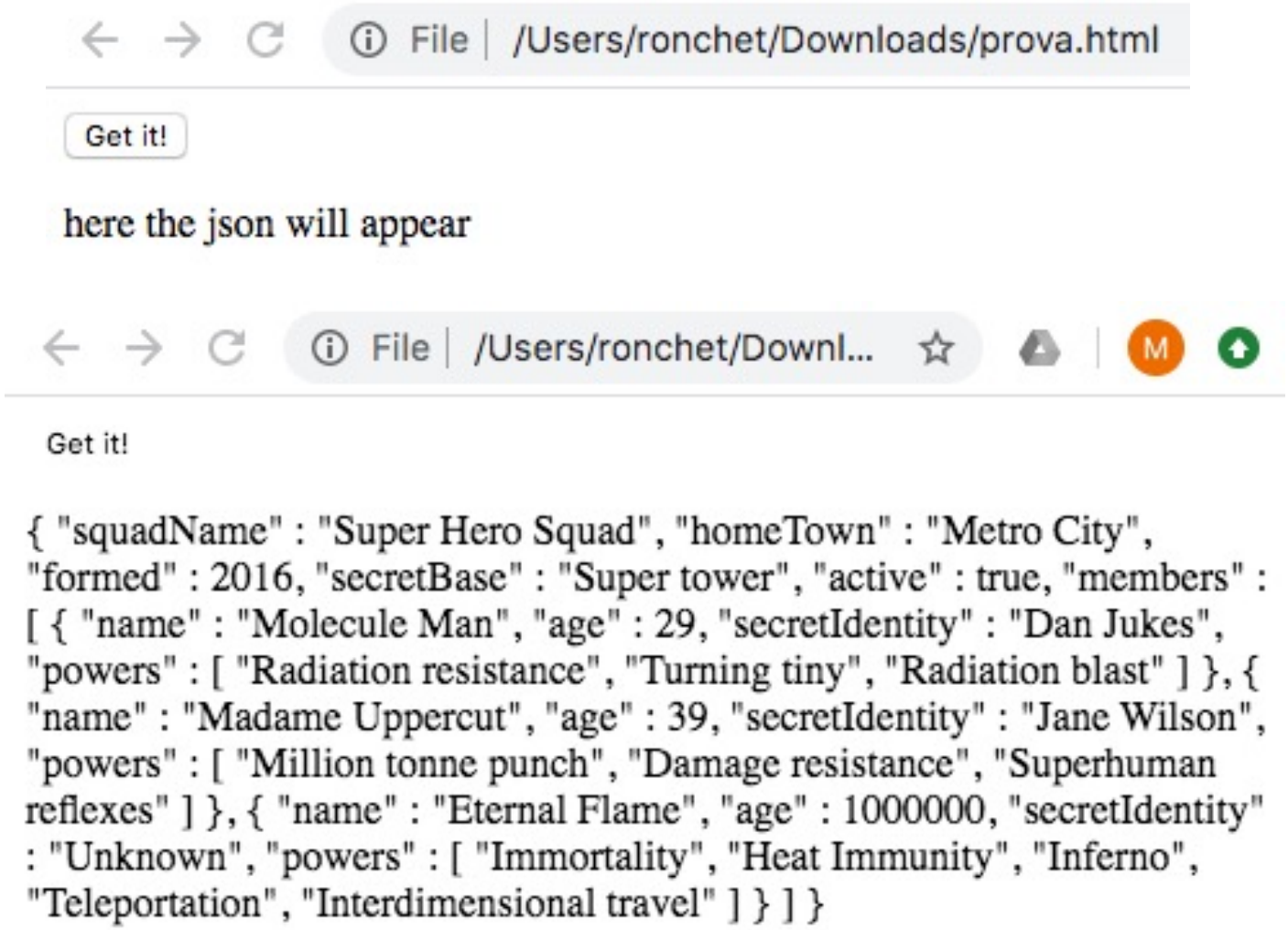

```
2 ▸Uncaught DOMException: Failed to set        prova.html:10
the 'responseType' property on 'XMLHttpRequest': The
response type cannot be changed for synchronous
requests made from a document.
    at getJson (file:///Users/ronchet/Downloads/prova.h
tml:10:28)
    at HTMLInputElement.onclick (file:///Users/ronchet/
Downloads/prova.html:27:112)
```

The traps of asynchronous computing 3A

```
<script>
  function getJson() {
    var my_JSON_object;
    var url="https://mdn.github.io/learning-
       area/javascript/oojs/json/superheroes.json"
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", url, false);
    //xhttp.responseType = "json";
    xhttp.onreadystatechange = function () {
       var done = 4, ok = 200;
       if (this.readyState === done && this.status === ok){
          my_JSON_object = this.response;
       }
    };
    xhttp.send();
    return my_JSON_object;
  }
</script>
```

Let's comment this line

The traps of asynchronous computing 3 out



The traps of asynchronous computing 4A

```
<script>
  function getJson() {
    var my_JSON_object;
    var url="https://mdn.github.io/learning-
       area/javascript/oojs/json/superheroes.json"
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", url, true);
    xhttp.responseType = "json";
    xhttp.onreadystatechange = function () {
       var done = 4, ok = 200;
       if (this.readyState === done && this.status === ok){
          my_JSON_object = this.response;
          document.getElementById("myPar".innerHTML=
             my_JSON_object.squadName;
       }
    };
    xhttp.send();
    return my_JSON_object;
  }
</script>
```

Let's make it async again

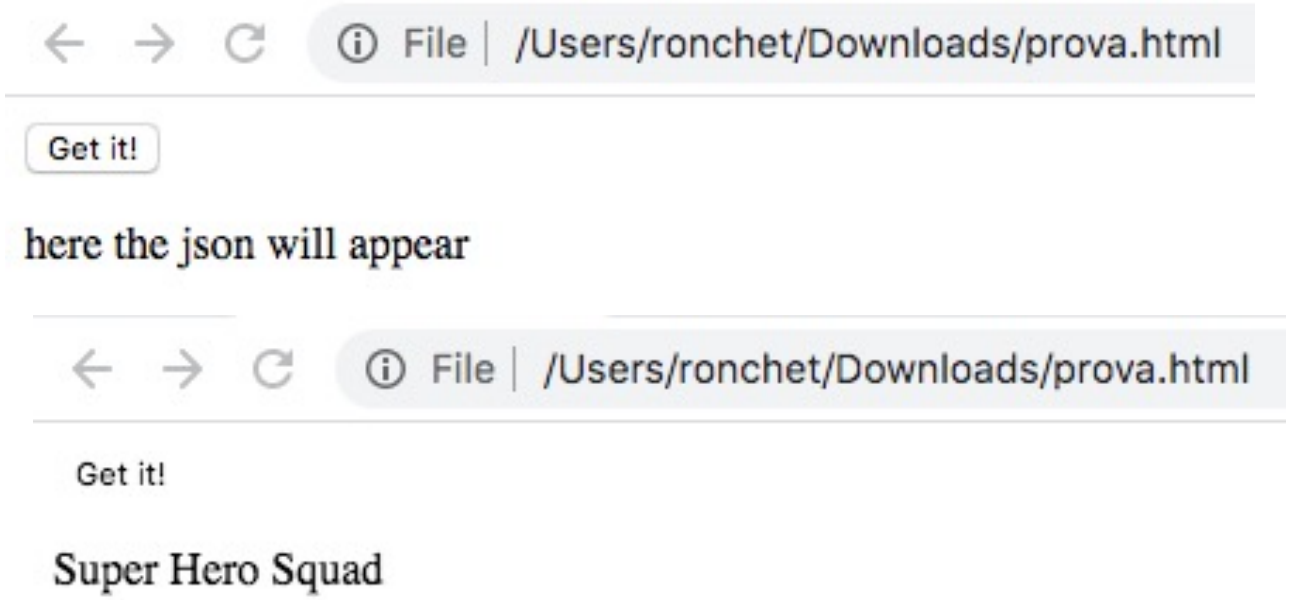
This is the right way of doing things!

The traps of asynchronous computing 4B

```
<!DOCTYPE html>
<head>
    <title>AJAJ Demo</title>
    <script>…</script>
</head>
  <body>
    <form>
     <input type="BUTTON" onClick=
       'getJson();'>
    </form>
    <p id="myPar">here the json will appear</p>
  </body>
</html>
```

This is the right way of doing things!

The traps of asynchronous computing 4 out



The traps of asynchronous computing 5A

```
<script>
  function getJson() {
    var my_JSON_object;
    var url="https://mdn.github.io/learning-
       area/javascript/oojs/json/superheroes.json"
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", url, true);
    //xhttp.responseType = "json";
    xhttp.onreadystatechange = function () {
       var done = 4, ok = 200;
       if (this.readyState === done && this.status === ok){
          my_JSON_object = this.response;
          document.getElementById("myPar".innerHTML=
             my_JSON_object.squadName;

       }
    };
    xhttp.send();
    return my_JSON_object;
  }
</script>
```

Let's comment this line

The traps of asynchronous computing 5 out

The JavaScript JSON object

The **JSON** object contains methods for parsing JSON text, and converting values to JSON. It can't be called or constructed, has two method properties:

- JSON.parse(*text*[, *reviver*]) Parse the string *text* as JSON, optionally transform the produced value and its properties, and return the value. Any violations of the JSON syntax, including those pertaining to the differences between JavaScript and JSON, cause a SyntaxError to be thrown. The *reviver* option allows for interpreting what the *replacer* has used to stand in for other datatypes.

- JSON.stringify(*value*[, *replacer*[, *space*]]) Return a JSON string corresponding to the specified value, optionally including only certain properties or replacing property values in a user-defined manner. By default, all instances of undefined are replaced with null, and other unsupported native data types are censored.
The *replacer* option allows for specifying other behavior.

JSON.parse

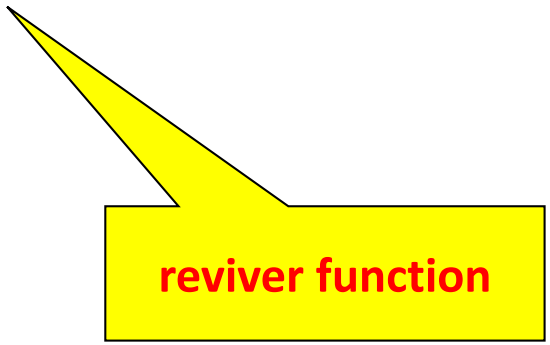
JSON string => JS data

```
const json = '{"result":true, "count":42}';
const obj = JSON.parse(json);

console.log(obj.count);   // expected output: 42

console.log(obj.result);  // expected output: true
```

JSON.parse with reviver function

JSON string => JS data

```
var text = '{ "name":"Dorothea Wierer",
              "birthDate":"1990-04-03",
              "city":"Brunico"}';

var obj = JSON.parse(text, function (key, value) {
  if (key == "birthDate") {
    return new Date(value);
  } else {
    return value;
  }
});
```



JSON.stringify

JS data => JSON string

JS_Object

JS_Array

```
console.log(JSON.stringify({ x: 5, y: 6 }));
// expected output: "{"x":5,"y":6}"

console.log(JSON.stringify([new Number(3), new String('false'),
new Boolean(false)]));
// expected output: "[3,"false",false]"

console.log(JSON.stringify({ x: [10, undefined, function(){},
Symbol('')] }));
// expected output: "{"x":[10,null,null,null]}"

console.log(JSON.stringify(new Date(2006, 0, 2, 15, 4, 5)));
// expected output: ""2006-01-02T15:04:05.000Z""
```

JS_special types

JS_date

The JSON.stringify() function will remove any functions from a JavaScript object, both the key and the value

Json Tutorial and reference

https://www.w3schools.com/js/js_json_intro.asp

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

Exercise

Modify assignment 3 so that when you put something in the cart, the number of items in the cart gets changed without reloading the page.