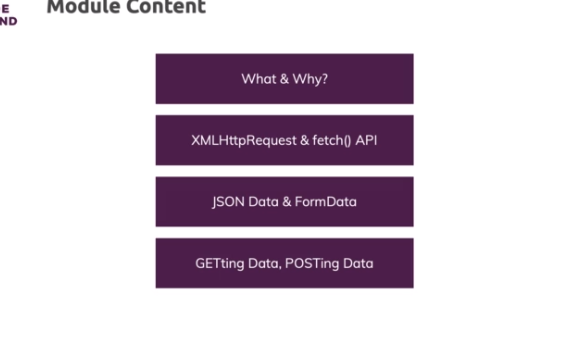
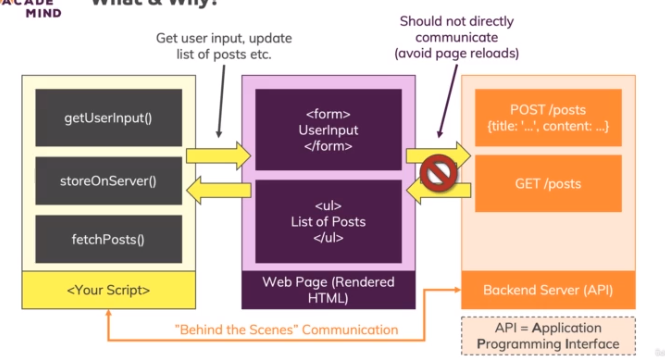
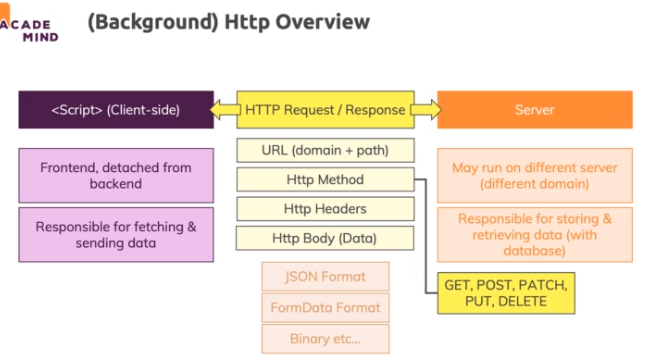
**Working with Http Requests**



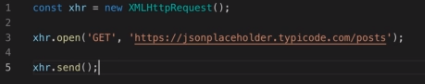
**What & Why**

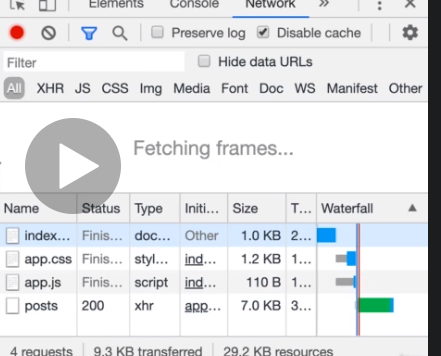


**More Background about Http**

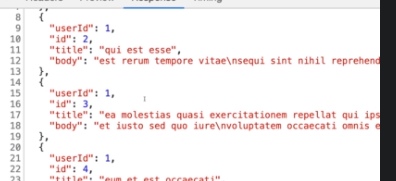


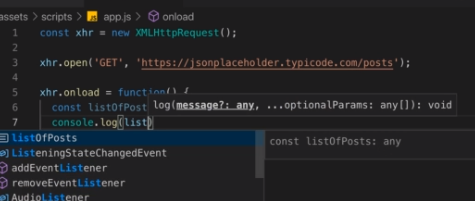
**Sending a GET Request**





**JSON Data & Parsing Data**





**JSON Data Deep Dive**

Typically, data is transferred as "**JSON**" data between your client-side code and your backend ("the server").

JSON stands for **J**ava**S**cript **O**bject **N**otation and it looks like this:

    1. {

    2.     "name": "Max",

    3.     "age": 30,

    4.     "hobbies": [

    5.         { "id": "h1", "title": "Sports" },

    6.         { "id": "h2", "title": "Cooking" }

    7.     ],

    8.     "isInstructor": true

    9. }

JSON data supports **objects** ({}), **arrays** ([]), **strings** (MUST use double-quotes!), **numbers** (NO quotes) and **booleans** (also NO quotes).

All object keys (e.g. "name") **HAVE to be wrapped by double quotes**. **No quotes** or **single quotes** are **NOT ALLOWED**!

Actually, the whole JSON "object" is wrapped in quotes itself because **JSON data in the end is just a string** that contains data in the format shown above.

You can test it yourself - take the following non-JSON JavaScript object and apply JSON.stringify() on it. This will convert it to JSON data. If you do that in the dev tools console, you'll see that you in the end get a string which contains data formatted as shown above.

    1. const person = { // this is NOT JSON - it's a normal ("raw") JavaScript object!

    2.     name: 'Max',

    3.     age: 30,

    4.     hobbies: [

    5.         { id: 'h1', title: 'Sports' },

    6.         { id: 'h2', title: 'Cooking' }

    7.     ],

    8.     isInstructor: true

    9. };

    10.

    11. const jsonData = JSON.stringify(person); // convert raw JS data to JSON data string

    12. console.log(jsonData); // a string with machine-readable JSON data in it

    13. console.log(typeof jsonData); // string

We use JSON data because it's easy to parse for machines - and as an extra benefit it's also quite readable to us humans.

If you receive some JSON data and you want to convert it back into normal JS data, you can use JSON.parse():

    1. const parsedData = JSON.parse(jsonData); // yields a "raw" JS object/ array etc.

Also nice to know: You're **NOT LIMITED to objects** when converting data to JSON. You can also convert numbers, arrays, booleans or just strings - all data types JSON supports:

    1. const jsonNumber = JSON.stringify(2); // "2"

    2. const jsonText = JSON.stringify('Hi there! I use single quotes in raw JS'); // ""Hi there! ...""

    3. const jsonArray = JSON.stringify([1, 2, 3]); // "[1,2,3]"

    4. const jsonBoolean = JSON.stringify(true); // "true"

JSON Data Deep Dive

Typically, data is transferred as "**JSON**" data between your client-side code and your backend ("the server").

JSON stands for **J**ava**S**cript **O**bject **N**otation and it looks like this:

    1. {

    2.     "name": "Max",

    3.     "age": 30,

    4.     "hobbies": [

    5.         { "id": "h1", "title": "Sports" },

    6.         { "id": "h2", "title": "Cooking" }

    7.     ],

    8.     "isInstructor": true

    9. }

JSON data supports **objects** ({}), **arrays** ([]), **strings** (MUST use double-quotes!), **numbers** (NO quotes) and **booleans** (also NO quotes).

All object keys (e.g. "name") **HAVE to be wrapped by double quotes**. **No quotes** or **single quotes** are **NOT ALLOWED**!

Actually, the whole JSON "object" is wrapped in quotes itself because **JSON data in the end is just a string** that contains data in the format shown above.

You can test it yourself - take the following non-JSON JavaScript object and apply JSON.stringify() on it. This will convert it to JSON data. If you do that in the dev tools console, you'll see that you in the end get a string which contains data formatted as shown above.

    1. const person = { // this is NOT JSON - it's a normal ("raw") JavaScript object!

    2.     name: 'Max',

    3.     age: 30,

    4.     hobbies: [

    5.         { id: 'h1', title: 'Sports' },

    6.         { id: 'h2', title: 'Cooking' }

    7.     ],

    8.     isInstructor: true

    9. };

    10.

    11. const jsonData = JSON.stringify(person); // convert raw JS data to JSON data string

    12. console.log(jsonData); // a string with machine-readable JSON data in it

    13. console.log(typeof jsonData); // string

We use JSON data because it's easy to parse for machines - and as an extra benefit it's also quite readable to us humans.

If you receive some JSON data and you want to convert it back into normal JS data, you can use JSON.parse():

    1. const parsedData = JSON.parse(jsonData); // yields a "raw" JS object/ array etc.

Also nice to know: You're **NOT LIMITED to objects** when converting data to JSON. You can also convert numbers, arrays, booleans or just strings - all data types JSON supports:

    1. const jsonNumber = JSON.stringify(2); // "2"

    2. const jsonText = JSON.stringify('Hi there! I use single quotes in raw JS'); // ""Hi there! ...""

    3. const jsonArray = JSON.stringify([1, 2, 3]); // "[1,2,3]"

    4. const jsonBoolean = JSON.stringify(true); // "true"

**The "Fetch" Button Always Appends**

Just a quick note to avoid confusion: In the demo app, the "Fetch" button always appends data without clearing existing data first. That means that pressing the button multiple times will add more and more items.

You can of course adjust the code to clear the content first.

