# Reading and Iterating over JSON files

We open a JSON file with multiple records and using the ForEach array method iterate over all the records displaying the fields we want to the browser console.

# What is an array?

```
const integers = [98,56,345,87,5,76,91,123,43];

const instruments = ['piano','flute','guitar','ukulele’];

const octaves = [
  {id: 10, instrument: 'Tenor Banjo', keyOctave: 'G2'},
  {id: 20, instrument: 'Guitar', keyOctave: 'E2'},
  {id: 30, instrument: 'Bass Guitar', keyOctave: 'E1'},
  {id: 40, instrument: 'Mandolin', keyOctave: 'G2'},
  {id: 50, instrument: 'Ukulele', keyOctave: 'G4'}
];
```

# What is JSON?

- JSON stands for JavaScript Object Notation.

- It is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. JSON is based on a subset of the JavaScript programming language, but is language-independent, which means that it can be used with many programming languages, not just JavaScript.

```
{
  "name": "John",
  "age": 30,
  "isStudent": true,
  "courses": ["Math", "Science", "English"],
  "address": {
    "street": "123 Main St",
    "city": "New York",
    "state": "NY",
    "zip": "10001"
  }
}
```

# key-value pairs 1

```
{
  "name": "John",
  "age": 30,
  "isStudent": true
}
```

- JSON data is represented as key-value pairs, where the key is a string and the value can be a string, number, boolean, array, or another JSON object.
- JSON is commonly used for exchanging data between a client and a server, or between different applications. It has become the standard format for web APIs because it is easy to read and parse, and can be used with many different programming languages.

# key-value pairs 2

```
const octaves = [
  {id: 10, instrument: 'Tenor Banjo', keyOctave: 'G2'},
  {id: 20, instrument: 'Guitar', keyOctave: 'E2'},
  {id: 30, instrument: 'Bass Guitar', keyOctave: 'E1'},
  {id: 40, instrument: 'Mandolin', keyOctave: 'G2'},
  {id: 50, instrument: 'Ukulele', keyOctave: 'G4'}
];

octaves.forEach(function(item) {
  console.log(`instrument of id ${item.id} is a ${item.instrument}.`);
});
```

# JSON file can store an array of records. OR an array of JSON objects.

**As a disk file;**

```
[{
  "name": "John",
  "age": 30,
  "isStudent": true
},{
  "name": "Sue",
  "age": 27,
  "isStudent": true
},{
  "name": "Bob",
  "age": 32,
  "isStudent": false
}]
```

**As a javascript array;**

```
[{
  name: 'John',
  age: 30,
  isStudent: true
},{
  name: 'Sue',
  age: 27,
  isStudent: true
},{
  name: 'Bob',
  age: 32,
  isStudent: false
}]
```

# DEMO: Iterating through an array

```
const people = [
  {name: 'John', age: 3, isStudent: true},
  {name: 'Sue', age: 27, isStudent: true},
  {name: 'Bob', age: 18, isStudent: false}
];

people.forEach( person => {
  console.log(`Welcome ${person.name}.`)
})
```

# Reading a file with Fetch() API

In JavaScript ES6, you can use the built-in **fetch** API to read a JSON file from disk.

```
fetch('path/to/file.json')
  .then(response => response.json())
  .then(data => {
    // Do something with the JSON data
  })
  .catch(error => {
    // Handle any errors that occur while fetching the file
  });
```

# Reading a file with Fetch() API - security

- This method is considered secure because it uses the fetch API, which is built into modern browsers and Node.js, to retrieve the file. The fetch API is designed to work with network requests and has built-in security features to prevent cross-site scripting (XSS) and other attacks.

- When reading a JSON file from disk, it's important to make sure that the file is located in a secure directory and that access to the file is restricted to authorized users. You should also validate the JSON data to ensure that it is in the expected format and does not contain any malicious code.

# DEMO: Reading a JSON file from disk

```
{"name": "Tenor Banjo", "key-octave": "G2"}
```

```
fetch('./instruments.json')
  .then(response => response.json())
  .then(data => {
    // Do something with the JSON data
    for(let i = 0; i < data.length; i++){
      console.log(data[i].name);
    }
  })
  .catch(error => {
    // Handle any errors that occur while fetching the file
    console.error(error);
  });
```

# DEMO: building functions with iterators

```
people.forEach( person => {
  console.log(`${person.name} is a ${maturity(person.age)}.`)
})

function maturity(age) {
  if(age <= 12) return 'child'
  else if(age > 12 && age <= 19) return 'teenager'
  else if(age > 19 && age <= 65) return 'adult'
  else return 'retiree'
}
```