

Javascript

JSON

Introdução

- ▷ *JavaScript Object Notation* (ou simplesmente *JSON*) é um formato de arquivo e para troca de dados, podendo representar valores e objetos.
 - Este formato foi originalmente criado para *Javascript*, porém foi adotado por diversas linguagens em tarefas de serialização de objetos.
 - Quando utilizado em arquivos, é comum o uso da extensão *.json*
 - É frequentemente utilizado para comunicação com *API's* (*Application Programming Interface* ou interface de programação de aplicações).
 - Outro formato comum em *API's* é o formato *XML*.

Introdução

- ▷ O formato *JSON* é semelhante ao que utilizamos para criar objetos em *Javascript*, com algumas diferenças:
 - Não se pode utilizar aspas simples e nem *backslash* (```) para *strings*.
 - Somente aspas duplas são aceitas.
 - Os nomes das propriedades dos objetos precisam estar entre aspas duplas.
 - Somente propriedades que armazenam dados são representadas.
 - Os métodos dos objetos, por exemplo, ficam de fora do *JSON*
 - Propriedades com valor *undefined* não são serializadas.

Introdução

- ▷ Os dados são serializados em uma string no formato *UTF-8*.
- ▷ Os tipos de dados que o *JSON* utiliza são:
 - *Number*
 - *NAN*
 - *String*
 - *Boolean*
 - *Array*
 - *Object*
 - *null*

Introdução

- ▷ O *MIME type* para textos *JSON* é "application/json".
 - *MIME type* é um identificador de formato de arquivos ou dados transmitidos pela internet.

Diferenças Entre *JSON* e objetos *Javascript*

```
{  
  id: 2,  
  matricula: 234,  
  nome: 'Bruno',  
  email: 'bruno@email.com'  
}
```

```
{  
  "id":2,  
  "matricula":234,  
  "nome":"Bruno",  
  "email":"bruno@email.com"  
}
```

Diferenças Entre *JSON* e objetos *Javascript*

```
{  
  id: 2,  
  matricula: 234,  
  nome: 'Bruno',  
  email: 'bruno@email.com'  
}
```

```
'{'  
  "id":2,  
  "matricula":234,  
  "nome":"Bruno",  
  "email":"bruno@email.com"  
}'
```

Isso é uma string, apesar de não ter sido representada nos slides com aspas (').

Diferenças Entre *JSON* e objetos *Javascript*

```
{  
  id: 2,  
  matricula: 234,  
  nome: 'Bruno',  
  email: 'bruno@email.com'  
}
```

```
{  
  "id":2,  
  "matricula":234,  
  "nome":"Bruno",  
  "email":"bruno@email.com"
```

Espaços e quebra de linha são aceitos, porém descartados. Geralmente utilizados apenas quando se deseja mostrar o conteúdo para um ser humano. Para transmissões ou para guardar informações, frequentemente os espaços e a quebra de linha são suprimidos.

Conversões

JSON.stringify e JSON.parse

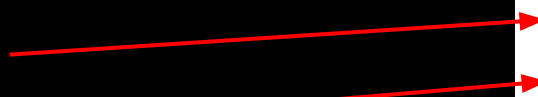
Conversões

- ▷ O *Javascript* possui os seguintes métodos para conversões entre objetos e JSON:
 - *JSON.stringify*
 - Converte objeto para *JSON*
 - *JSON.parse*
 - Converte *JSON* para objeto

JSON.stringify

- ▷ O método *JSON.stringify* converte um objeto Javascript para *JSON*.
 - Também é possível converter tipos simples, como *String* e *Number* e *arrays*.

```
const obj = {  
  id: 2,  
  matricula: 234,  
  nome: 'Bruno'  
};  
  
const objStr = JSON.stringify(obj);  
console.log(obj);  
console.log(objStr);
```



► {id: 2, matricula: 234, nome: 'Bruno'}
{"id":2,"matricula":234,"nome":"Bruno"}

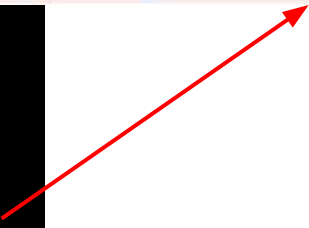
JSON.stringify

- ▷ É possível passar por parâmetro até três argumentos para a função *JSON.stringify*:
 - O primeiro parâmetro é o objeto a ser serializado.
 - O segundo parâmetro é uma das seguintes opções:
 - Um *array* de propriedades a serem serializadas
 - Uma função que retorna o que será serializado para cada propriedade.
 - O terceiro parâmetro possibilita passar o número de espaços ou um caractere para formatação do *JSON* resultante.
 - Este parâmetro é utilizado apenas quando se deseja apresentar o *JSON* para um humano.

JSON.stringify

- ▷ Não é possível exportar quando existe referência circular:

```
const aluno1 = {  
  matricula: 234,  
  nome: 'Bruno'  
};  
const turma = {  
  periodo: 1,  
  alunos: [aluno1]  
}  
aluno1.turma = turma;  
console.log(JSON.stringify(turma));
```

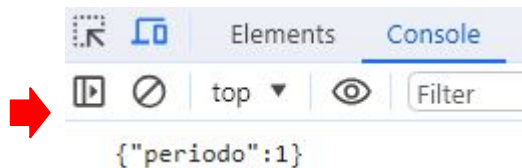


```
✖ ▶ Uncaught TypeError: Converting circular structure to JSON  
--> starting at object with constructor 'Object'  
|   property 'alunos' -> object with constructor 'Array'  
|       index 0 -> object with constructor 'Object'  
--- property 'turma' closes the circle  
at JSON.stringify (<anonymous>)  
at Ex01.js:43:18
```

JSON.stringify

- ▷ É possível definir quais propriedades serão exportadas
 - Basta passar um *array* contendo os nomes das propriedades a serem exportadas no **segundo parâmetro** da função *JSON.stringify*:

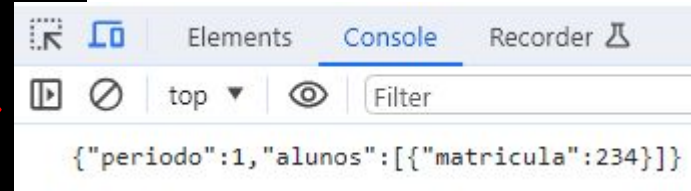
```
console.log(JSON.stringify(turma, ['periodo']));
```



JSON.stringify

- ▶ Veja este segundo exemplo:

```
const aluno1 = {  
  matricula: 234,  
  nome: 'Bruno'  
};  
const turma = {  
  periodo: 1,  
  alunos: [aluno1]  
}  
aluno1.turma = turma;  
console.log(JSON.stringify(turma,  
  ['periodo', 'alunos', 'matricula']));
```

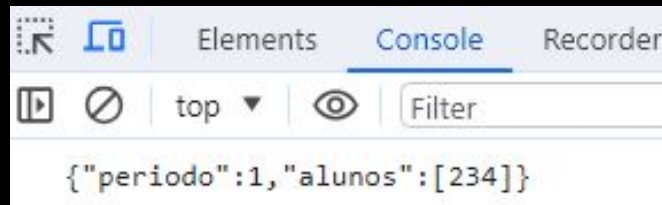


O nome do aluno e a turma do mesmo não foram exportados por não constarem no vetor de propriedades a serem exportadas.

JSON.stringify

- ▶ Também é possível definir uma função que retorna o valor da propriedade para cada objeto.

```
console.log(JSON.stringify(turma, (chave, valor) => {  
  if (chave === 'turna') {  
    return undefined;  
  } else if (chave === 'alunos') {  
    return valor.map(x => x.matricula);  
  } else {  
    return valor;  
  }  
})));
```



JSON.stringify

- ▶ Também é possível definir uma função que retorna o valor da propriedade para cada objeto.

```
console.log(JSON.stringify(turma, (chave, valor) => {  
    if (chave === 'turna') {  
        return undefined;  
    } else if (chave === 'alu  
        return valor.map(x =>  
    } else {  
        return valor;  
    }  
})));
```

Propriedades com valor *undefined* não são exportadas, logo a propriedade turma não será exportada e, com isso, não teremos referência circular.

JSON.stringify

- ▶ Também é possível definir uma função que retorna o valor da propriedade para cada objeto.

```
console.log(JSON.stringify(turma, (chave, valor) => {  
  if (chave === 'turna') {  
    return undefined;  
  } else if (chave === 'alunos') {  
    return valor.map(x => x.matricula);  
  } else {  
    return valor;  
  }  
}));
```

Os alunos foram exportados como um vetor de matrículas.

JSON.stringify

- ▶ Também é possível definir uma função que retorna o valor da propriedade para cada objeto.

```
console.log(JSON.stringify(turma, (chave, valor) => {  
  if (chave === 'turna') {  
    return undefined;  
  } else if (chave === 'alunos') {  
    return valor.map(x => x.matricula);  
  } else {  
    return valor;  
  }  
}));
```

Todas as outras propriedades são exportadas com seus valores inalterados.

JSON.stringify

- ▷ É possível criar a função *toJSON* para definir como o objeto deve ser serializado.

```
const aluno1 = {  
  matricula: 234,  
  nome: 'Bruno',  
  toJSON() {  
    return { matricula:this.matricula, nome:this.nome };  
  }  
};  
  
const turma = {  
  periodo: 1,  
  alunos: [aluno1]  
}  
  
aluno1.turma = turma;  
console.log(JSON.stringify(turma));
```

JSON.parse

- Para decodificar uma string em *JSON*, pode ser utilizada a função *JSON.parse*.

```
const jsonStr = '{"periodo":1,"alunos":[{"matricula":234,"nome":"Bruno"}]';  
const aluno1 = JSON.parse(jsonStr);  
console.log(aluno1);
```



```
▼ {periodo: 1, alunos: Array(1)} ⓘ  
  ▼ alunos: Array(1)  
    ▶ 0: {matricula: 234, nome: 'Bruno'}  
      length: 1  
    ▶ [[Prototype]]: Array(0)  
  periodo: 1  
  ▶ [[Prototype]]: Object
```

JSON.parse

- ▷ A função *JSON.parse* recebe até dois parâmetros:
 - *str*
 - *String* a ser deserializada.
 - *reviver*
 - Função que será chamada para cada propriedade e que retorna o valor deserializado.

JSON.parse

- ▶ A função *reviver* foi fornecida abaixo para transformar um vetor de objetos em um vetor de *numbers* contendo as matrículas dos alunos.

```
const jsonStr = '{"periodo":1,"alunos":[{"matricula":234,"nome":"Bruno"}]}' ;
const aluno1 = JSON.parse(jsonStr, (chave, valor) => {
  if (chave === 'alunos') {
    return valor.map(x => x.matricula);
  }
  return valor;
});
console.log(aluno1);
```



```
▼ {periodo: 1, alunos: Array(1)} ⓘ
  ▶ alunos: [234]
    periodo: 1
    ▶ [[Prototype]]: Object
```

Requisições que retornam *JSON*

XMLHttpRequest com *responseType* “json”

Requisição

- ▷ No próximo *slide* é apresentado um exemplo que realiza uma requisição *HTTP* para um servidor que retorna um *JSON*.
- ▷ O retorno é automaticamente convertido para um objeto *Javascript* e alguns dados retornados são impressos no *console*.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```

Objeto que nos permite fazer uma requisição *HTTP*.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```

O primeiro parâmetro indica o verbo *HTTP* que será utilizado na requisição (o tipo da requisição). O segundo parâmetro é o endereço que receberá a requisição.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```


Com este tipo de resposta, a *string* recebida será automaticamente passada pelo *JSON.parse*, o que a deserializará automaticamente.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();
request.onload = function () {
    var objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```



Envia a requisição.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```

Quando recebido o retorno, a requisição chamará a função *onload*.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```

Obtém a resposta da requisição.

Requisição

```
const url = "https://camillo-classroom.github.io/des_web_1/alunos.json";

const request = new XMLHttpRequest();
request.open("GET", url);
request.responseType = "json";
request.send();

request.onload = function () {
    let objetos = request.response;

    console.log("Alunos recebidos:");
    objetos.forEach(x => {
        console.log(`${x.matricula} - ${x.nome}`);
    });
};
```

Imprime a matrícula e o nome de cada aluno recebido.

Fim de material

Dúvidas?