JavaScript cheatsheet – v4.4.3 – https://github.com/Serrin/Celestra/

Web Storage api and JSON

IE8+

localStorage:
```
localStorage.length;
localStorage.key(index);
localStorage.getItem(key);
localStorage.setItem(key, data);
localStorage.removeItem(key);
localStorage.clear();
```

sessionStorage:
```
sessionStorage.length;
sessionStorage.key(index);
sessionStorage.getItem(key);
sessionStorage.setItem(key, data);
sessionStorage.removeItem(key);
sessionStorage.clear();
```

hasItem:
```
localStorage.getItem(key) !== null
sessionStorage.getItem(key) !== null
```

setJSON:
```
localStorage.setItem(key,
JSON.stringify(object));
sessionStorage.setItem(key,
JSON.stringify(object));
```

getJSON:
```
JSON.parse(localStorage.getItem(key));
JSON.parse(sessionStorage.getItem(key));
```

element.dataset & data-* attributes

- IE11 compatible
- element data-* attributes
- no methods and events

camelcase:
```
element.data-name
  -> element.dataset.name
element.data-first-second
  -> element.dataset.firstSecond
```

set:
```
element.dataset.name = "value";
element.dataset["name"] = "value";
element.setAttribute("data-name",
"value");
element["data-name"] = "value";
```

get:
```
element.dataset.name;
element.dataset["name"];
element.getAttribute("data-name");
element["data-name"];
```

remove:
```
element.removeAttribute("data-
name");
```

check:
```
element.hasAttribute("data-name");
```

TypedArray

IE10+11 compatible

```
new <TypedArray>(); ES2017
new <TypedArray>(length);
new <TypedArray>(typedArray);
new <TypedArray>(object);
new <TypedArray>(buffer[,byteOffset[,len]]);
```

Int8Array();
-128 to 127, 1 byte, int8_t, Shortint
Uint8Array();
0 to 255, 1 byte, uint8_t, Byte
Uint8ClampedArray(); - not in IE10-11
0 to 255, 1 byte, uint8_t, Byte
Int16Array();
-32768 to 32767, 2 byte, int16_t, Smallint
Uint16Array();
0 to 65535, 2 byte, uint16_t, Word
Int32Array();
-2147483648 to 2147483647, 4 byte, int32_t
Uint32Array();
0 to 4294967295, 4 byte, uint32_t, Longword
BigInt64Array(); - not in IE10-11
-2**63 to 2**63-1, 8 byte, int64_t, Int64
BigUint64Array(); - not in IE10-11
0 to 2**64-1, 8 byte, uint64_t, QWord
Float32Array();
1.2x10-38 to 3.4x1038, 4 byte, float, Real
Float64Array();
5.0x10-324 to 1.8x10308, 8 byte, Double

DOM events

```
target.addEventListener(<type>,<listener>[,useCapture]); or target.addEventListener(<type>,<listener>[,options]);
target.removeEventListener(<type>,<listener>[,useCapture]); or target.removeEventListener(<type>,<listener>[,options]);
target.dispatchEvent(<event>);
target.type(); or target["type"]();
```

element.classList

IE10+IE11 don't have support for classList on SVG or MathML elements.

element.classList.add(String[,String]);
IE10+11: yes (except the multiple arguments)

element.classList.remove(String[,String]);
IE10+11: yes (except the multiple arguments)
- Removing a class that does not exist, does NOT throw an error.

element.classList.contains(String);
IE10+11: yes

element.classList.toggle(String[,force]);
IE10+11: yes (except the second argument)
- When only one argument is present: Toggle class value; if class exists then remove it and return false, if not, then add it and return true.
- When a second argument is present: If the second argument evaluates to true, add specified class value, and if it evaluates to false, remove it.

element.classList.item(Number);
IE10+11: yes

element.classList.length;
IE10+11: yes

element.classList.replace(oldClass, newClass);
IE10+11: No and the method isn't compatible with the Safari and mobile browsers too.

Remove all classes:
element.className = "";

JSON

IE8+

Valid Data Types
- string
- number
- object (containing valid JSON values)
- array
- boolean
- date
- null

Invalid Data Types
- function
- Symbol
- NaN, Infinity, undefined – *will be "null"*
- an object with method(s) (functions)
- Map, Set, WeakMap, WeakSet – *fix: convert to array*
- BigInt - *fixed in Celestra - BigInt.prototype.toJSON();*

JSON.stringify(value[,replacer[,space]]);
Convert a JavaScript object to a JSON string.

```
JSON.stringify( { a: 1, b: "2", c: true } );
// -> "{\"a\":1,\"b\":\"2\",\"c\":true}"

JSON.stringify( [1, 2, 3, 4, 5] );
// -> "[1,2,3,4,5]"
```

JSON.parse(text[,reviver]);
Parses a JSON string and returns a JavaScript object.

```
JSON.parse(JSON.stringify( {a: 1, b: "2", c: true} ));
// -> Object { a: 1, b: "2", c: true }

JSON.parse(JSON.stringify( [1, 2, 3, 4, 5] ));
// -> Array(5) [ 1, 2, 3, 4, 5 ]
```

DOMParser

```
IE9: XML support
IE10+IE11: XML, SVG and HTML support

var parser = new DOMParser();
var doc = parser.parseFromString("sourceStr", "application/xml");
Returns a Document, but not a SVGDocument nor a HTMLDocument.

var parser = new DOMParser();
var doc = parser.parseFromString(sourceStr, "image/svg+xml");
Returns a SVGDocument, which also is a Document.

var parser = new DOMParser();
var doc = parser.parseFromString(sourceStr, "text/html");
Returns a HTML document.
```

DOMParser sample function

```
function parseHTML (str) {
  return Array.from(
    (new DOMParser())
      .parseFromString(str, "text/html")
      .childNodes[0]
      .childNodes[1]
      .childNodes
  );
}

parseHTML(
  "<div>1<p>2</p><p>3</p></div>"
  + "<div>4<p>5</p><p>6</p></div>"
  + "<div>7</div>"
  + "<p>8</p>"
);
// -> Array(4) [ div, div, div, p ]
// Tested in IE11, Edge, Firefox and Chrome.
```

BigInt (Int64)

```
let theBiggestInt = 9007199254740991n;
let alsoHuge = BigInt(9007199254740991);
let hugeString = BigInt("9007199254740991");
let hugeHex = BigInt("0x1fffffffffffff");
let hugeBin =
BigInt("0b11111111111111111111111111111111111111111111111
1111111");
// all -> 9007199254740991n

typeof 1n;          // -> "bigint"
typeof BigInt("1"); // -> "bigint"

Operators: +, *, -, **, %, Bitwise operators (e.g.: >>>)

!0n      // -> true
!1n      // -> false

4n / 2n  // -> 2n
5n / 2n  // -> 2n, not 2.5n -> rounded

1n < 2   // -> true
2n > 1   // -> true
2n > 2   // -> false
2n >= 2  // -> true

let mixed = [4n, 6, -12n, 10, 4, 0, 0n];
mixed.sort();// -> [-12n, 0, 0n, 4n, 4, 6, 10]

Methods
BigInt.asIntN();
  - BigInt value to a signed integer
BigInt.asUintN();
  - BigInt value to an unsigned integer
BigInt.prototype.toLocaleString();
BigInt.prototype.toString();
BigInt.prototype.valueOf();
```

Fetch	Fetch POST
Firefox 39, Chrome 42, Edge14, Opera29 , Safari 10.1, Samsung I. 4.0 *// Example GET method implementation with TEXT:* fetch("https://api.coindesk.com/v1/bpi/currentprice.json") .then(response => response.text()) .then(data => console.log(data)) .catch(error => console.log(error)); *// Example GET method implementation with JSON:* fetch("https://api.coindesk.com/v1/bpi/currentprice.json") .then(response => response.json()) .then(data => console.log(data.bpi.USD.rate)) .catch(error => console.log(error)); *// Example GET method implementation with TEXT and JSON:* fetch("https://api.coindesk.com/v1/bpi/currentprice.json") .then(response => response.text()) .then(text => console.log(JSON.parse(text).bpi.USD.rate+"\n"+text)) .catch(error => console.log(error)); *// Example POST method implementation with upload JSON data:* const data = { username: "example" }; fetch("https://example.com/profile", { method: "POST", // or "PUT" headers: { "Content-Type": "application/json", }, body: JSON.stringify(data), }) .then(response => response.json()) .then(data => { console.log("Success:", data); }) .catch((error) => { console.error("Error:", error); });	*// Example POST method implementation:* *// Default options are marked with ** async function postData(url = "url", data = {}) { const response = await fetch(url, { method: "POST", *// *GET, POST, PUT, DELETE, etc.* mode: "cors", *// no-cors, *cors, same-origin* cache: "no-cache", *// *default, no-cache, reload, force-cache, only-if-cached* credentials: "same-origin", *// include, *same-origin, omit* headers: {"Content-Type": "application/json"}, *// "Content-Type": "application/x-www-form-urlencoded"* redirect: "follow", *// manual, *follow, error* referrerPolicy: "no-referrer", *// no-referrer, *no-referrer-when-downgrade, origin, origin-when-cross-origin, same-origin, strict-origin, strict-origin-when-cross-origin, unsafe-url* body: JSON.stringify(data) *// body data type must match "Content-Type" header* }); return response.json(); *// parses JSON response into native JavaScript objects* } postData("https://example.com/answer", { answer: 42 }) .then(data => { console.log(data); }); *// JSON data parsed by `data.json()` call*