loadJSON()

Loads a JSON file to create an `Object`.

JavaScript Object Notation (JSON) is a standard format for sending data between applications. The format is based on JavaScript objects which have keys and values. JSON files store data in an object with strings as keys. Values can be strings, numbers, Booleans, arrays, `null`, or other objects.

The first parameter, `path`, is always a string with the path to the file. Paths to local files should be relative, as in `loadJSON('/assets/data.json')`. URLs such as `'https://example.com/data.json'` may be blocked due to browser security.

The second parameter, `successCallback`, is optional. If a function is passed, as in `loadJSON('/assets/data.json', handleData)`, then the `handleData()` function will be called once the data loads. The object created from the JSON data will be passed to `handleData()` as its only argument.

The third parameter, `failureCallback`, is also optional. If a function is passed, as in `loadJSON('/assets/data.json', handleData, handleFailure)`, then the `handleFailure()` function will be called if an error occurs while loading. The `Error` object will be passed to `handleFailure()` as its only argument.

Note: Data can take time to load. Calling `loadJSON()` within preload() ensures data loads before it's used in setup() or draw().

Examples

```
let myData;

// Load the JSON and create an object.
function preload() {
  myData = loadJSON('/assets/data.json');
}

function setup() {
  createCanvas(100, 100);

  background(200);

  // Style the circle.
  fill(myData.color);
  noStroke();

  // Draw the circle.
  circle(myData.x, myData.y, myData.d);

  describe('A pink circle on a gray background.');
}
```

```
let myData;

// Load the JSON and create an object.
function preload() {
  myData = loadJSON('/assets/data.json');
}

function setup() {
  createCanvas(100, 100);

  background(200);

  // Create a p5.Color object and make it transparent.
  let c = color(myData.color);
  c.setAlpha(80);

  // Style the circles.
  fill(c);
  noStroke();

  // Iterate over the myData.bubbles array.
  for (let b of myData.bubbles) {
    // Draw a circle for each bubble.
    circle(b.x, b.y, b.d);
  }

  describe('Several pink bubbles floating in a blue sky.');
}
```

```
let myData;

// Load the GeoJSON and create an object.
function preload() {
  myData =
loadJSON('https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_day.geojson');
}

function setup() {
  createCanvas(100, 100);

  background(200);

  // Get data about the most recent earthquake.
  let quake = myData.features[0].properties;

  // Draw a circle based on the earthquake's magnitude.
  circle(50, 50, quake.mag * 10);

  // Style the text.
  textAlign(LEFT, CENTER);
  textFont('Courier New');
  textSize(11);

  // Display the earthquake's location.
  text(quake.place, 5, 80, 100);

  describe(`A white circle on a gray background. The text "${quake.place}" is written beneath the circle.`);
}
```

```
let bigQuake;

// Load the GeoJSON and preprocess it.
function preload() {
  loadJSON(

'https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_day.geojson',
    handleData
  );
}

function setup() {
  createCanvas(100, 100);

  background(200);

  // Draw a circle based on the earthquake's magnitude.
  circle(50, 50, bigQuake.mag * 10);

  // Style the text.
  textAlign(LEFT, CENTER);
  textFont('Courier New');
  textSize(11);

  // Display the earthquake's location.
  text(bigQuake.place, 5, 80, 100);

  describe(`A white circle on a gray background. The text "${bigQuake.place}" is written beneath the circle.`);
}

// Find the biggest recent earthquake.
```

```
let bigQuake;

// Load the GeoJSON and preprocess it.
function preload() {
  loadJSON(

'https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_day.geojson',
    handleData,
    handleError
  );
}

function setup() {
  createCanvas(100, 100);

  background(200);

  // Draw a circle based on the earthquake's magnitude.
  circle(50, 50, bigQuake.mag * 10);

  // Style the text.
  textAlign(LEFT, CENTER);
  textFont('Courier New');
  textSize(11);

  // Display the earthquake's location.
  text(bigQuake.place, 5, 80, 100);

  describe(`A white circle on a gray background. The text "${bigQuake.place}" is written beneath the circle.`);
```

Syntax

```
loadJSON(path, [successCallback], [errorCallback])
```

Parameters

path	String: path of the JSON file to be loaded.
successCallback	Function: function to call once the data is loaded. Will be passed the object.
errorCallback	Function: function to call if the data fails to load. Will be passed an `Error` event object.

Returns

Object: object containing the loaded data.

This page is generated from the comments in src/io/files.js. Please feel free to edit it and submit a pull request!

Related References

addChild
Adds a new child element and returns a reference to it.

getAttributeCount
Returns the number of attributes the element has.

getChild
Returns the first matching child element as a new p5.XML object.

getChildren
Returns an array with the element's child elements as new p5.XML objects.