```
mkdir my-addon
cd my-addon
```

`binding.gyp`

```
{
  "targets": [
    {
      "target_name": "my-addon",
      "sources": [ "my-addon.cc" ]
    }
  ]
}
```

C++ code

```
#include <node.h>

namespace demo {

  using v8::FunctionCallbackInfo;
  using v8::Isolate;
  using v8::Local;
  using v8::Object;
  using v8::String;
  using v8::Value;

  void Method(const FunctionCallbackInfo<Value>& args) {
    Isolate* isolate = args.GetIsolate();
    Local<String> world = String::NewFromUtf8(isolate, "world");
    args.GetReturnValue().Set(world);
  }
  void Initialize(Local<Object> exports) {
    NODE_SET_METHOD(exports, "hello", Method);
  }

  NODE_MODULE(NODE_GYP_MODULE_NAME, Initialize)

}
```

Build

```
node-gyp configure build
```

js file

```
const myAddon = require('./build/Release/my-addon.node');
```

addition code

```
#include <node.h>

namespace demo {

  using v8::FunctionCallbackInfo;
  using v8::Isolate;
  using v8::Local;
  using v8::Object;
  using v8::Number;
  using v8::Value;

  // This is a simple C++ function that adds two numbers
  void Add(const FunctionCallbackInfo<Value>& args) {
    Isolate* isolate = args.GetIsolate();

    // Check the number of arguments passed to the function
```

```
 if (args.Length() < 2) {
  isolate->ThrowException(
     v8::Exception::TypeError(
        v8::String::NewFromUtf8(isolate, "Wrong number of arguments")
     )
  );
  return;
 }

 // Check the argument types
 if (!args[0]->IsNumber() || !args[1]->IsNumber()) {
  isolate->ThrowException(
     v8::Exception::TypeError(
        v8::String::NewFromUtf8(isolate, "Wrong arguments type")
     )
  );
  return;
 }

 // Perform addition
 double sum = args[0]->NumberValue(isolate) + args[1]->NumberValue(isolate);

 // Return the result as a Number object
 Local<Number> num = Number::New(isolate, sum);
 args.GetReturnValue().Set(num);
}

void Initialize(Local<Object> exports) {
 NODE_SET_METHOD(exports, "add", Add);
}
```

```
  NODE_MODULE(NODE_GYP_MODULE_NAME, Initialize)

}
```

.js file

```
const myAddon = require('./build/Release/my-addon.node');
console.log(myAddon.add(2, 3)); // Prints 5
```

opencv code

```
#include <opencv2/opencv.hpp>
using namespace cv;

void GetImageDimensions(const FunctionCallbackInfo<Value>& args) {
  Isolate* isolate = args.GetIsolate();

  // Check the number of arguments passed to the function
  if (args.Length() < 1) {
    isolate->ThrowException(
        v8::Exception::TypeError(
            v8::String::NewFromUtf8(isolate, "Wrong number of arguments")
        )
    );
    return;
  }

  // Check the argument type
  if (!args[0]->IsString()) {
    isolate->ThrowException(
        v8::Exception::TypeError(
            v8::String::NewFromUtf8(isolate, "Wrong argument type")
        )
    );
    return;
  }

  // Read the image file using OpenCV
  std::string filename(*v8::String::Utf8Value(args[0]->ToString()));
  Mat img = imread(filename, IMREAD_UNCHANGED);

  // Return the image dimensions as an object
  Local<Object> result = Object::New(isolate);
  result->Set(String::NewFromUtf8(isolate, "width"), Number::New(isolate, img.cols));
  result->Set(String::NewFromUtf8(isolate, "height"), Number::New(isolate, img.rows));
  args.GetReturnValue().Set(result);
}

void Initialize(Local<Object> exports) {
  NODE_SET_METHOD(exports, "getImageDimensions", GetImageDimensions);
}
NODE_MODULE(NODE_GYP_MODULE_NAME, Initialize)
```

Build command

```
node-gyp configure build
```

js file

```
const myAddon = require('./build/Release/my-addon.node');
const dimensions = myAddon.getImageDimensions('path/to/image.jpg');
console.log(dimensions); // Prints { width: 640, height: 480 }
```

binding file opencv c++

```
{
  "targets": [
    {
      "target_name": "my-addon",
      "sources": ["my-addon.cc"],
      "include_dirs": [
        "<!(node -p \"require('nan')\")",
        "C:/path/to/opencv/build/include"
      ],
      "libraries": [
        "-lopencv_core412",
        "-lopencv_highgui412",
        "-lopencv_imgcodecs412"
      ],
      "link_settings": {
        "libraries": [
          "-L\"C:/path/to/opencv/build/x64/vc16/lib\"",
          "-L\"C:/path/to/opencv/build/x64/vc16/bin\"",
        ]
      }
    }
  ]
}
```

Binding file with opencv dll

```
{
  "targets": [
    {
      "target_name": "my-addon",
      "sources": ["my-addon.cc"],
      "include_dirs": [
        "<!(node -p \"require('nan')\")",
        "C:/path/to/opencv/build/include"
      ],
      "libraries": [
        "-lopencv_core412",
        "-lopencv_highgui412",
        "-lopencv_imgcodecs412"
      ],
      "link_settings": {
        "libraries": [
          "-L\"C:/path/to/opencv/build/x64/vc16/lib\"",
          "-L\"C:/path/to/opencv/build/x64/vc16/bin\"",
        ],
        "ldflags": [
          "/MANIFEST:NO"
        ],
        "defines": [
          "OPENCV_DLL"
        ],
        "msvs_settings": {
          "VCCLCompilerTool": {
            "AdditionalOptions": [
              "/wd4996"
            ]
          }
        }
      }
    }
  ]
```

```
}
```

Copy dll in package.json file

```
"scripts": {
  "postinstall": "node-gyp configure build && xcopy /Y /S C:/path/to/opencv/build/x64/vc16/bin/*.dll build\\Release"
}
```