

最終回

JavaScriptから始める プログラミング2016

京都大学工学部情報学科

計算機科学コース3回

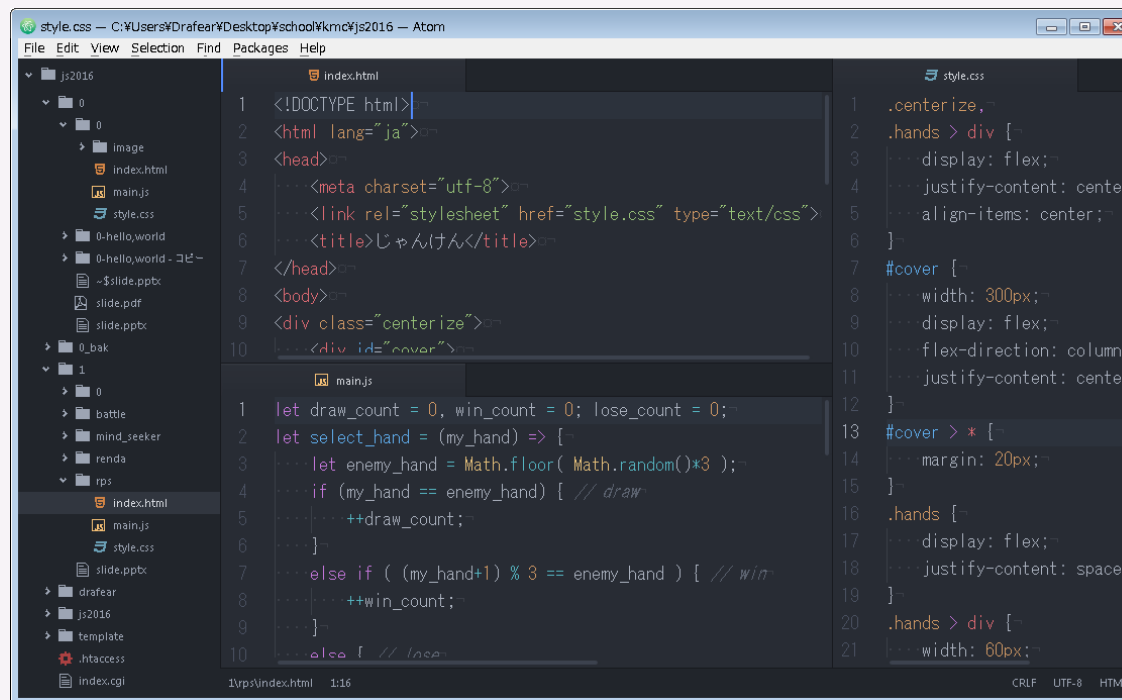
KMC2回 drafear



@drafear

この講座で使用するアイテム

- Google Chrome 
 - <https://chrome.google.com>
- Atom 
 - <https://atom.io/>



この講座で使用するアイテム

- Node.js (v6.2.2)
 - <https://nodejs.org/>



本日の内容

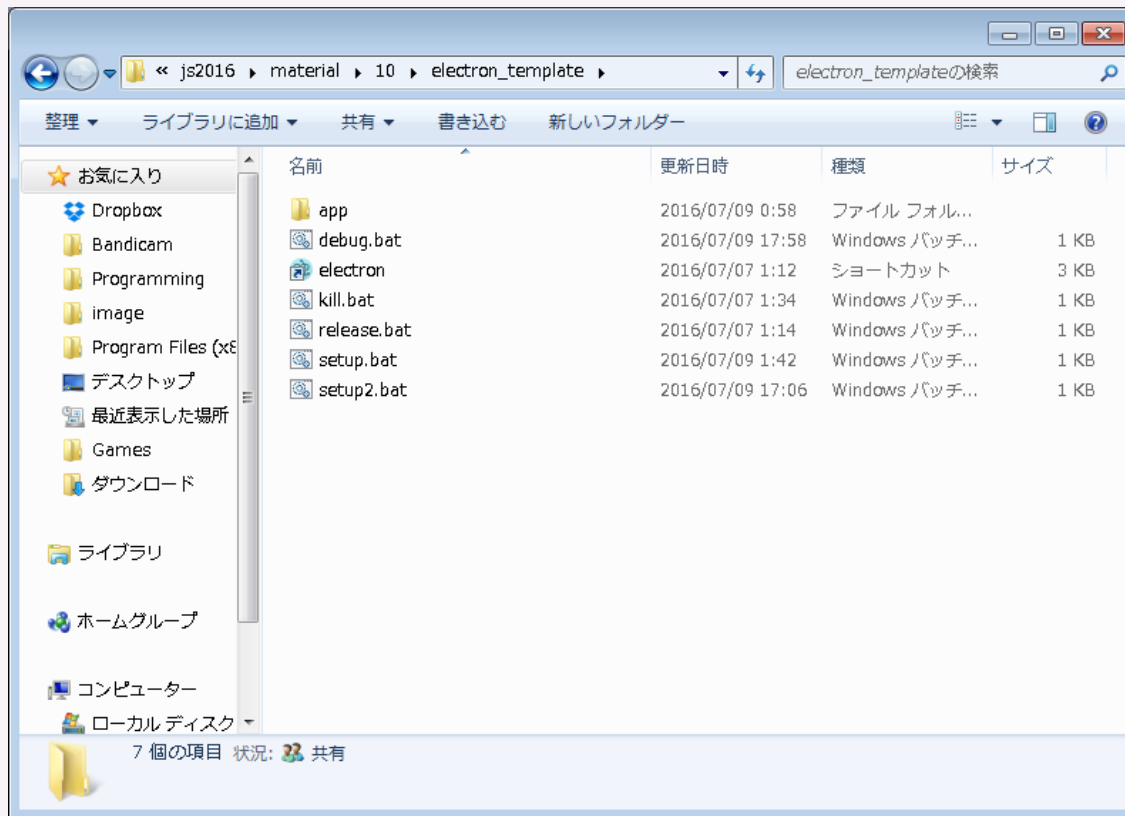
- Electronを使ってデスクトップアプリを作る
 - フチなしウィンドウの時計を作る
 - 外部サイトにアクセスする
 - localStorageを使ってみる
 - socket.ioサーバのGUIを作る
 - Canvasでお絵かきする



Electronを使ってみよう

毎度ながらのテンプレ

- 一식을落としてもらって `material/10/electron_template` が今回のテンプレート (別の場所にコピーして使って下さい)
 - <https://github.com/kmc-jp/js2016/>



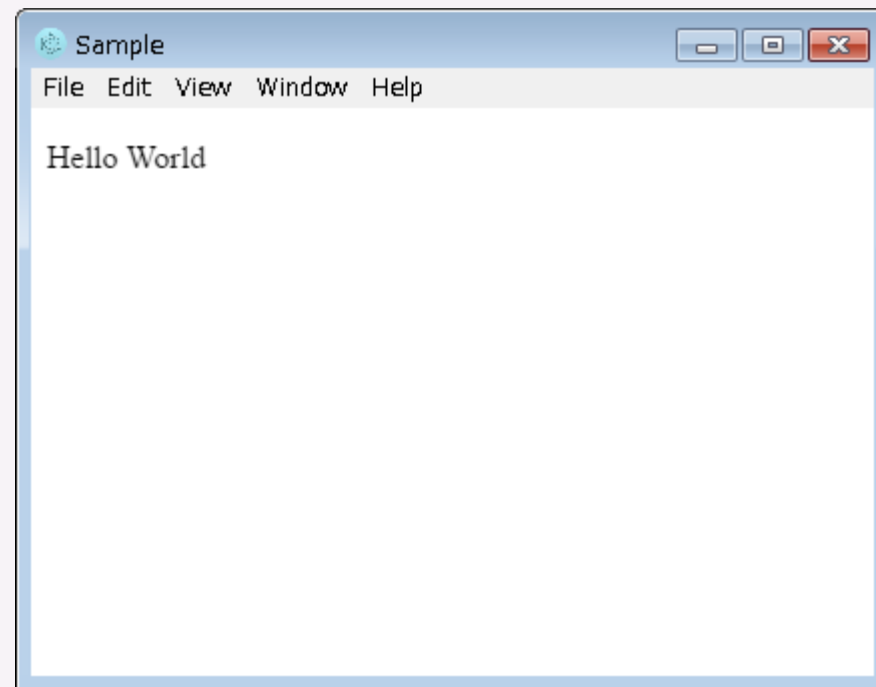
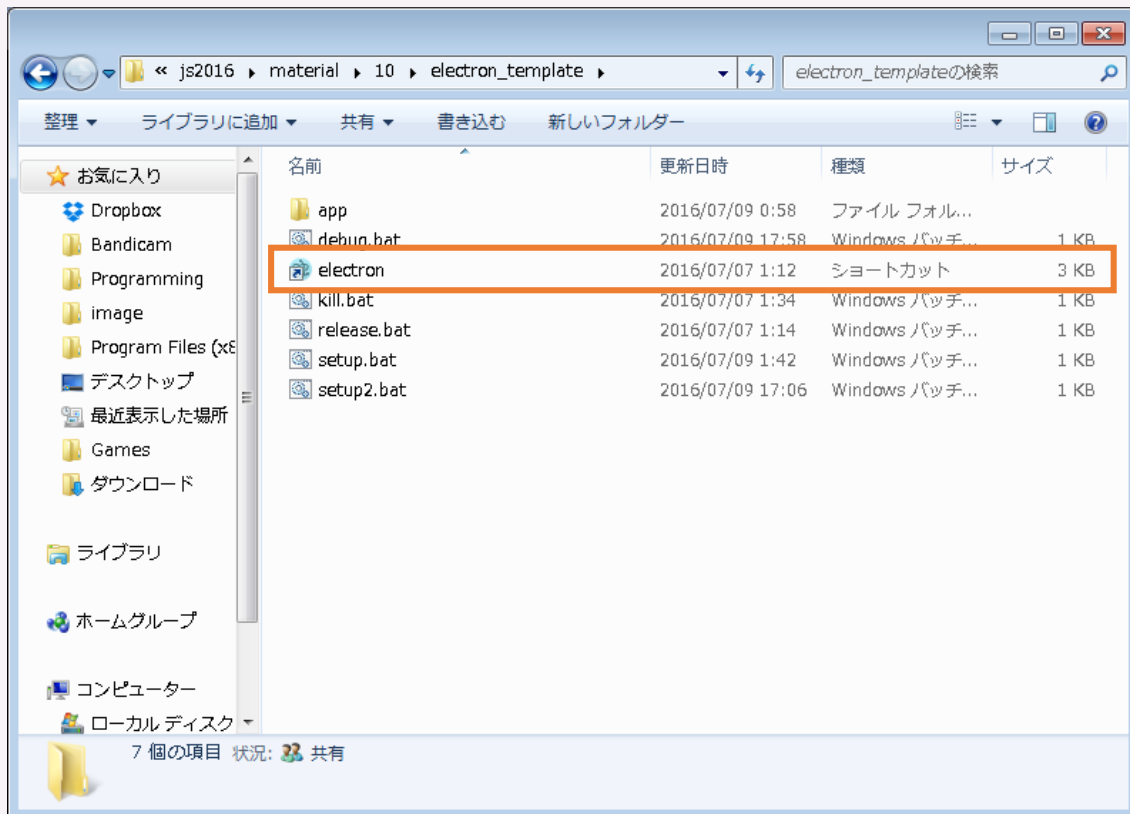
環境構築

- `setup.bat`および`setup2.bat`を実行
または以下のコマンドで環境構築

```
npm install electron-prebuilt -g  
npm install electron-packager -g
```

確認

- ショートカット「electron」を起動してみてください
 - ウィンドウが現れ, Hello, World と表示されたら成功
 - windowsでない方は端末で electron と打つと良いかも

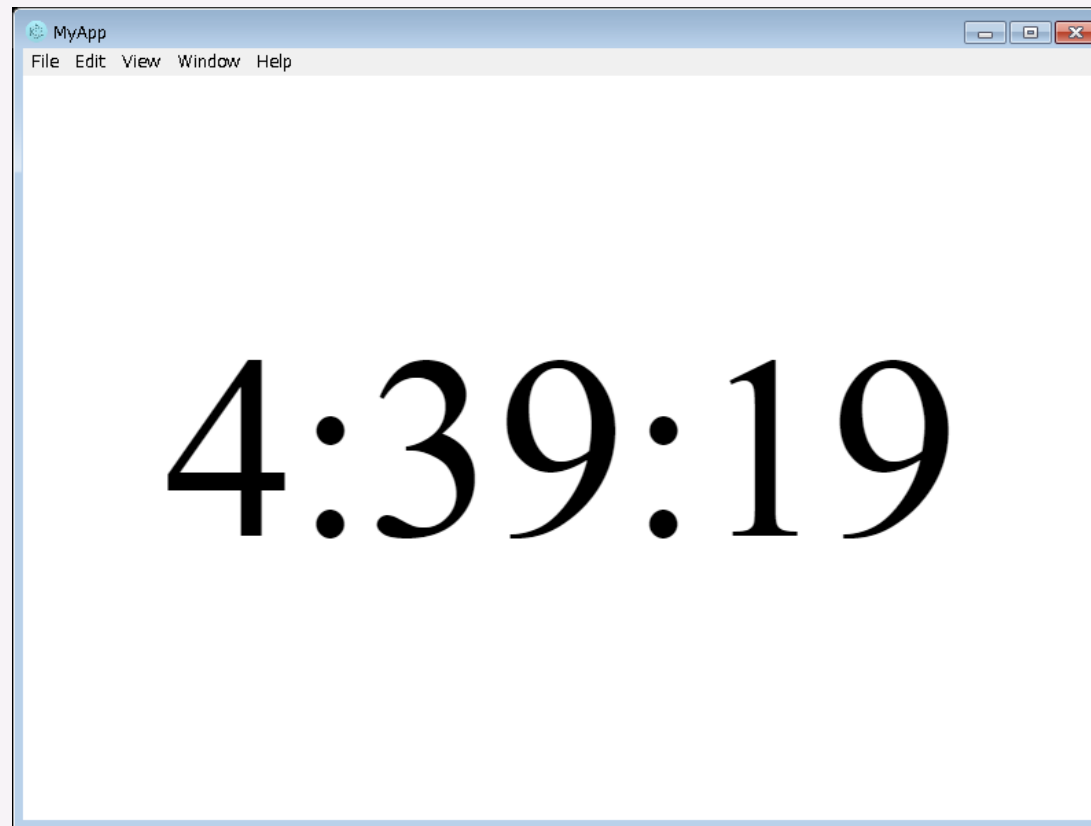


ディレクトリ構造・各ファイルの説明

```
app/
├── app.js ..... アプリ本体
├── icon.ico ..... アプリのアイコン
├── package.json ..... アプリ設定ファイル
├── page/
│   ├── index.html ..... 初期画面
│   ├── main.js ..... index.htmlが読み込むjsファイル
│   └── style.css ..... index.htmlが読み込むcssファイル
└── release.js ..... アプリをリリースするときの設定ファイル
```

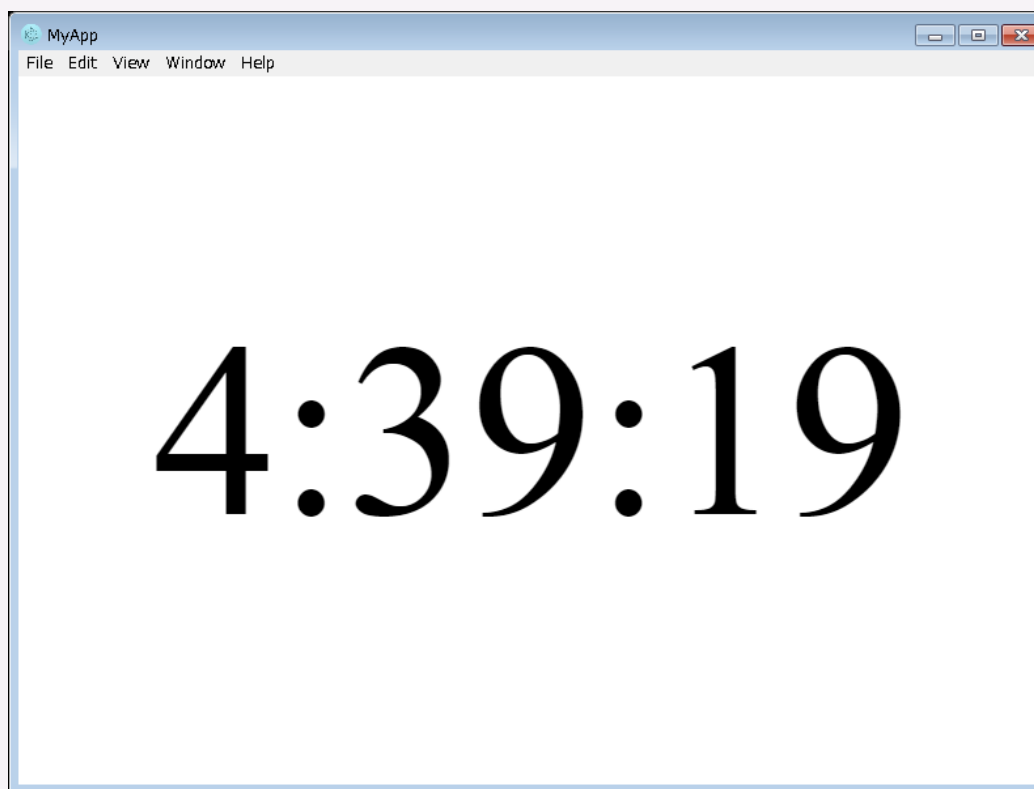
実は…

- 実は `app/page/index.html` が表示されてます！！
- `app/page` 以下のファイルをいじって時計を作ってみよう



演習

- Electron上に現在時刻をリアルタイムに表示してみよう
- 現在時刻は`new Date().toLocaleTimeString()`で文字列として取得できるよ



演習

- Electron上に現在時刻をリアルタイムに表示してみよう

index.html

```
<div id="time"></div>
```

main.js

```
const loop = () => {  
  setTimeout(loop, 100);  
  document.getElementById("time").textContent = new Date().toLocaleTimeString();  
};  
loop();
```

style.css

```
body { margin: 0; padding: 0; }  
#time {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  width: 100vw; height: 100vh;  
  font-size: 35vh;  
}
```

Dateオブジェクト

https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Global_Objects/Date



Electronの基本API

～其の壱～

フチなしウィンドウ

- `app.js`をいじります
- 18行目を以下のように変更します

app.js (一部)

```
mainWindow = new BrowserWindow({ width: 800, height: 600, frame: false });
```

Frameless Window

- フチがなくなりました！
- × ボタンもないので，閉じるときはALT+F4などで強制終了

16:07:58

Frameless Window

- 自分でフチを作ってみる

style.css

```
body {  
    margin: 0; padding: 0;  
}  
#time {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    box-sizing: border-box;  
    width: 100vw; height: 100vh;  
    font-size: 35vh;  
    border: 3px double black; /* 追加 */  
}
```

Frameless Window

A rectangular frameless window with a thin black border. Inside the window, the time "16:52:50" is displayed in a large, black, serif font, centered horizontally and vertically.

16:52:50

Global Shortcut

- ESCで閉じるようにする

```
app.js

'use strict'

const {app, BrowserWindow, globalShortcut} = require("electron"); // 変更
let mainWindow = null;
...
// Electronの初期化完了後に実行
app.on('ready', () => {
  // ショートカット追加
  globalShortcut.register('Escape', () => { // 追加
    mainWindow.close(); // 追加
  }); // 追加
  ...
});
```

Global Shortcut

- `globalShortcut.register("キ一名", () => { ... });`
で設定したキーが押されたら ... が実行される
- キ一名一覧
 - <http://electron.atom.io/docs/api/accelerator/>

BrowserWindowのオプション

- タスクトレイに表示しないようにする

```
new BrowserWindow({ width: 800, height: 600, frame: false, skipTaskbar: true });
```

BrowserWindowのオプション

- オプション一覧（アイコンを設定するなど）
 - <http://electron.atom.io/docs/api/browser-window/#new-browserwindowoptions>

draggable

- framelessなwindowをドラッグできるようにする
- ドラッグ可能領域をcssで指定する

style.css

```
body {  
    margin: 0; padding: 0;  
}  
#time {  
    -webkit-app-region: drag; /* 追加 */  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    box-sizing: border-box;  
    width: 100vw; height: 100vh;  
    font-size: 35vh;  
    border: 3px double black;  
}
```



アプリリリース

アプリリリース

- `release.bat`を実行すれば，アプリを配布用の形にします
- `dist`ディレクトリの中に(`release.js`で指定した)プラットフォーム用のディレクトリができてその中に実行ファイルなどができます
- windows以外の方は`release.js`の`platform`をいい感じに設定して下さい
 - <https://github.com/electron-userland/electron-packager>



Electronの基本API

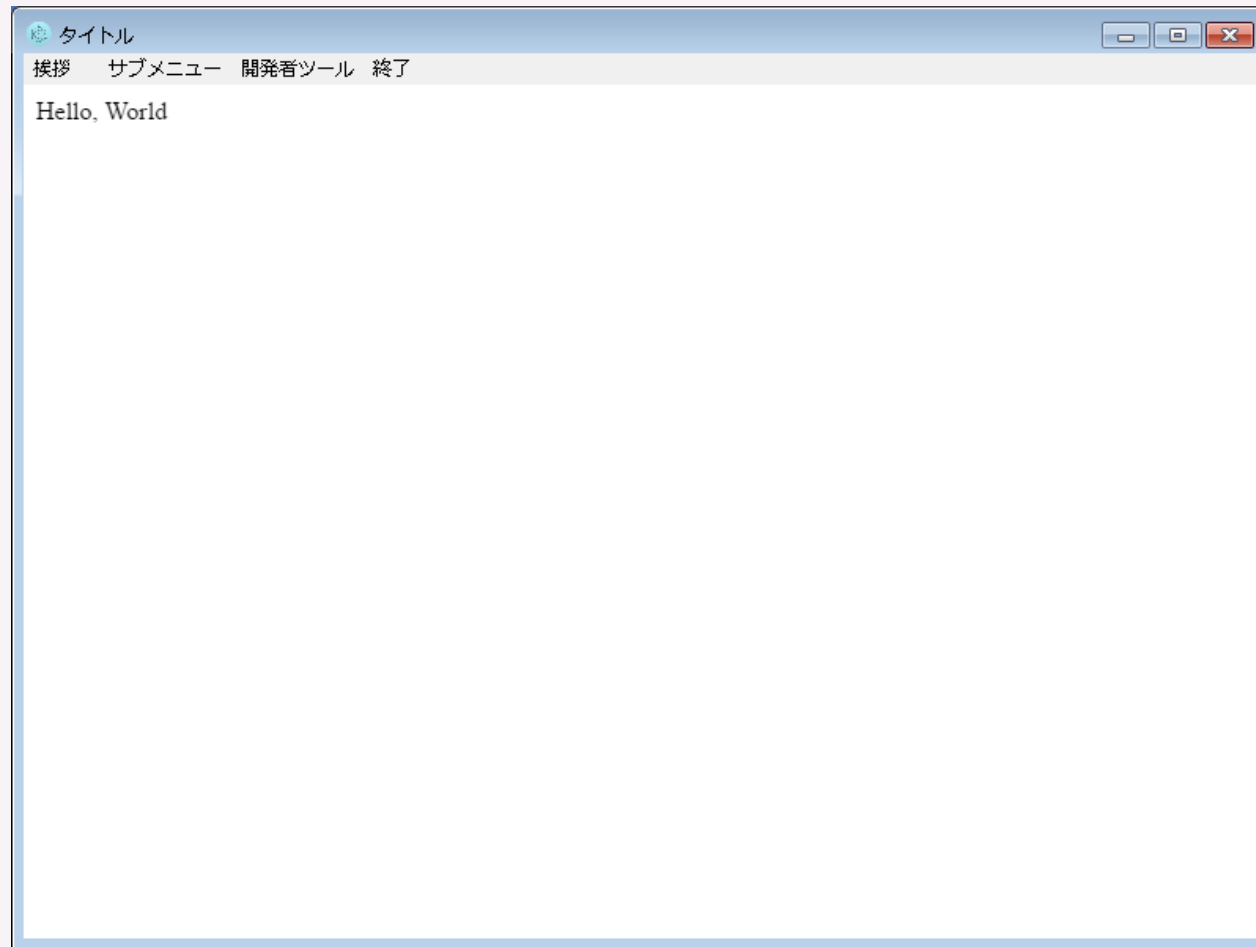
～其の弐～

新しいアプリ

- 新しいアプリを作ります
- またelectron_templateをコピーして下さい
- setup.bat, setup2.bat は実行しなくて大丈夫です

Menu

- メニューバーのメニューを設定します



app.js

```
'use strict'
```

```
const {app, BrowserWindow, Menu, MenuItem} = require('electron'); // 変更
```

```
let mainWindow = null;
```

```
...
```

```
// Electronの初期化完了後に実行
```

```
app.on('ready', () => {
```

```
...
```

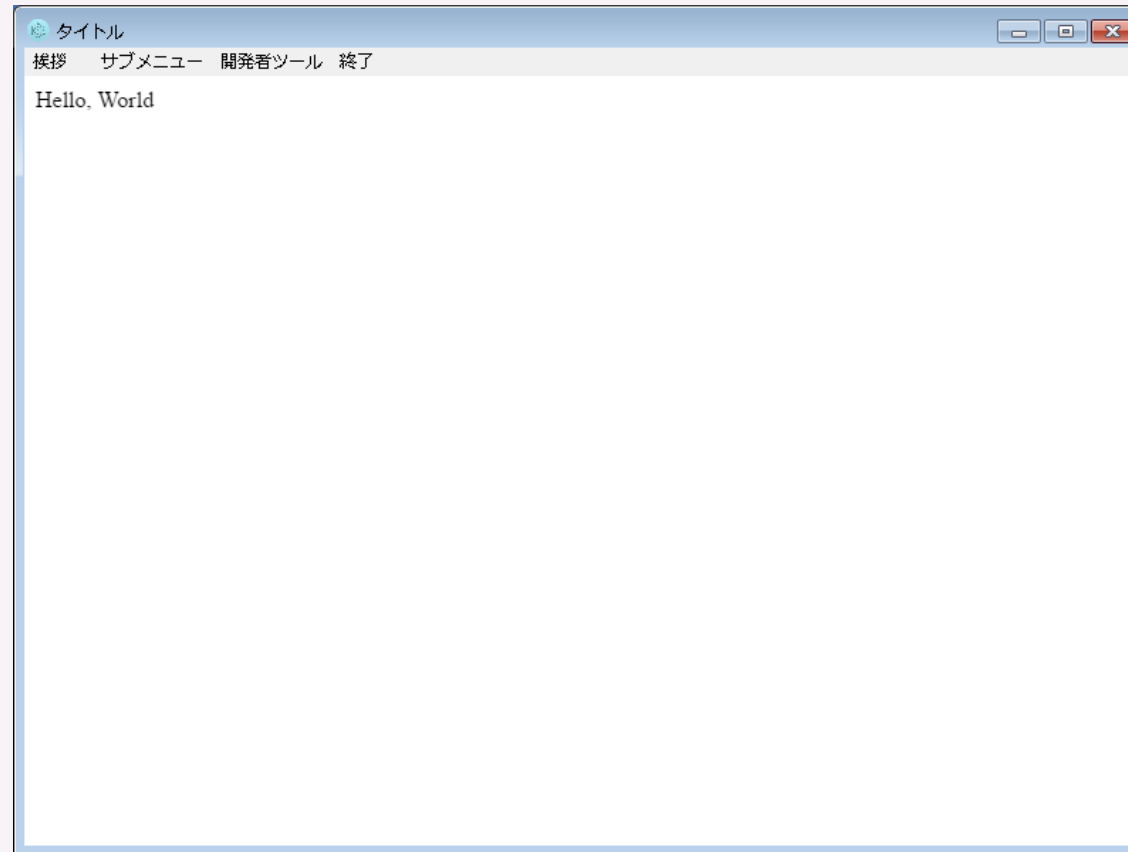
```
// ウィンドウのメニュー設定 (追加)
```

```
Menu.setApplicationMenu(Menu.buildFromTemplate([  
  { label: "挨拶", click: () => { console.log("hello"); } },  
  { label: "サブメニュー", submenu: [  
    { label: "a", type: "checkbox" },  
    { label: "b", type: "checkbox" },  
  ] },  
  { label: "開発者ツール", accelerator: 'Ctrl+Shift+I',  
    click: () => { mainWindow.toggleDevTools(); } },  
  { label: "終了", role: "close" },  
]));
```

```
});
```

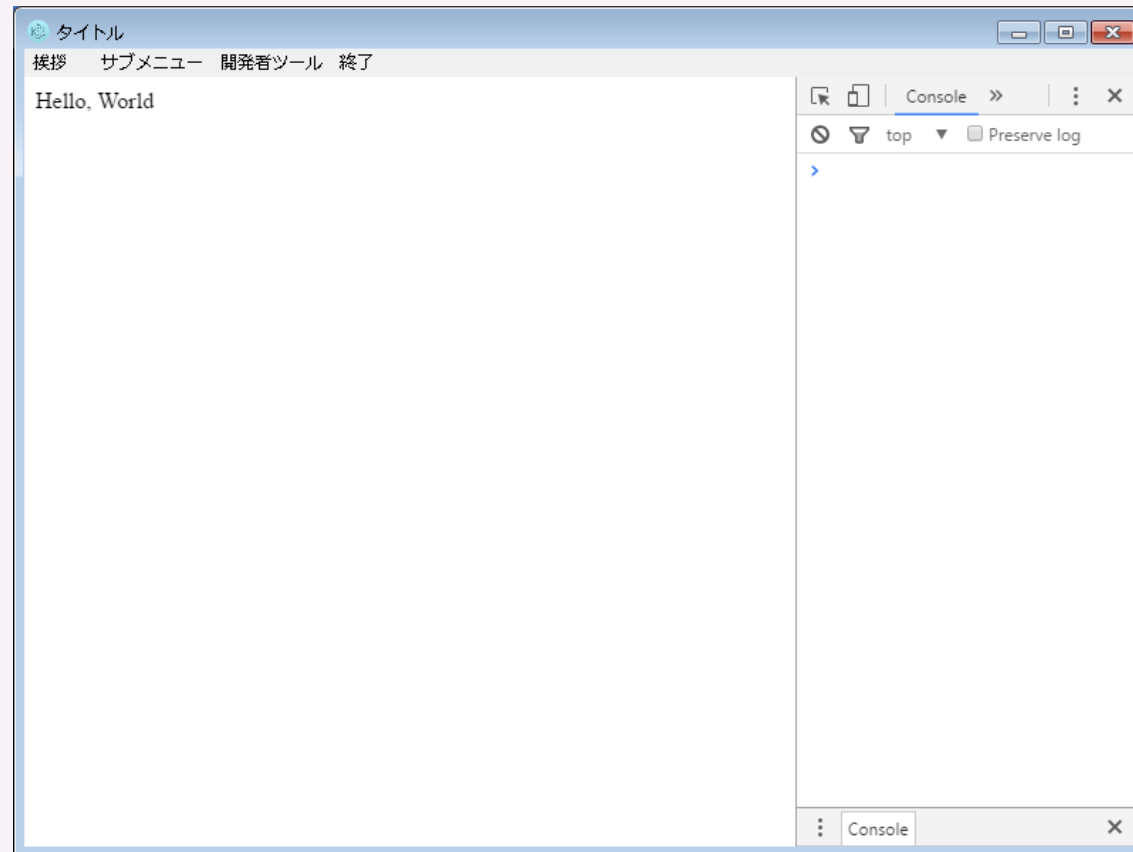
Menu

- "挨拶" をクリックしてみましょう



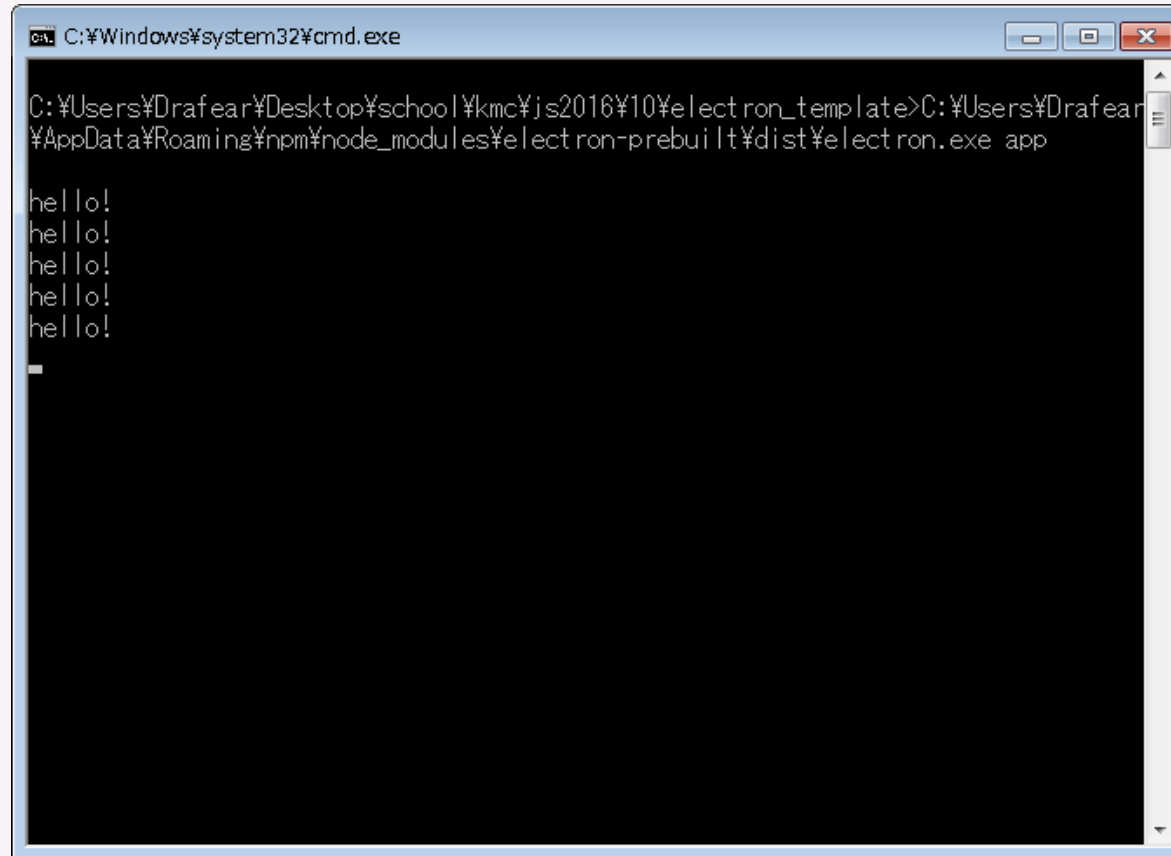
Menu

- 開発者ツール(Ctrl+Shift+I)の中のconsoleには表示されず
- このconsoleはmainWindow(renderer)のconsoleだから



Menu

- debug.batからelectronを起動して
挨拶をクリックすると黒い画面上に表示されるはず



A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The command prompt shows the following text:

```
C:\Users\Drafean\Desktop\school\kmc\js2016\10\electron_template>C:\Users\Drafean\AppData\Roaming\npm\node_modules\electron-prebuilt\dist\electron.exe app
```

Below the command, the output shows five lines of "hello!" followed by a cursor:

```
hello!  
hello!  
hello!  
hello!  
hello!  
_
```


contextmenu

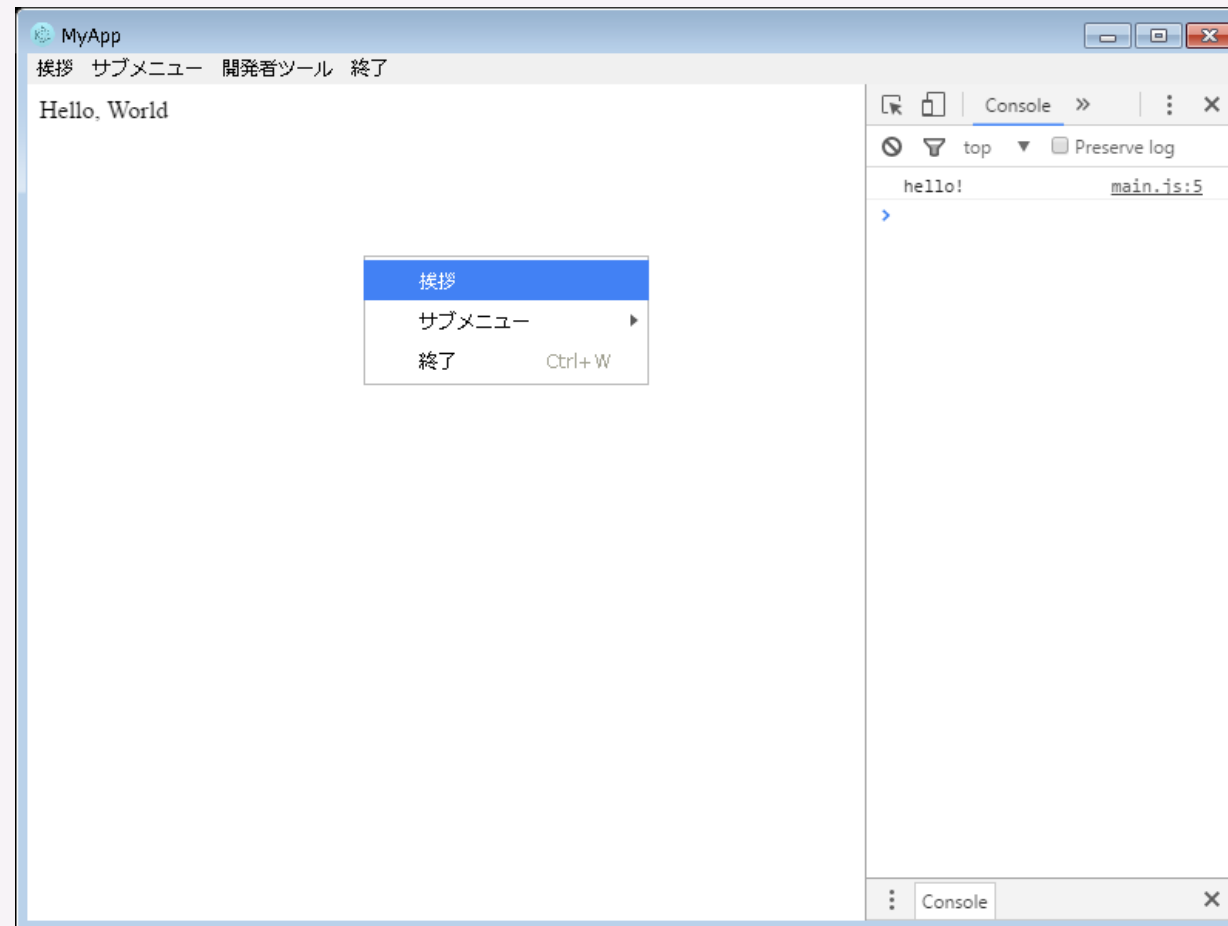
- 右クリックしたときのメニューを設定する

main.js

```
const {remote} = require('electron');
const {Menu, MenuItem} = remote;
const menu = Menu.buildFromTemplate([
  { label: "挨拶", click: () => { console.log("hello!"); } },
  { label: "サブメニュー", submenu: [
    { label: "a", type: "checkbox" },
    { label: "b", type: "checkbox" },
  ] },
  { label: "終了", role: "close" },
]);
window.addEventListener('contextmenu', (e) => {
  e.preventDefault();
  menu.popup(remote.getCurrentWindow());
});
```

contextmenu

- こっこの挨拶はちゃんとconsoleに表示される



Tray

- タスクトレイのアイコンとメニュー



app.js

```
'use strict'
const {app, BrowserWindow, Menu, MenuItem, Tray} = require('electron'); // 変更
...
// Electronの初期化完了後に実行
app.on('ready', () => {
  ...
  // タスクトレイ (追加)
  const tray = new Tray(`_${dirname}/icon.ico`);
  tray.setContextMenu(Menu.buildFromTemplate([
    { label: "挨拶", click: () => { console.log("hello!"); } },
    { type: "separator" },
    { label: "終了", role: "close" },
  ]));
  tray.setToolTip("tooltip");
});
```

ipc (プロセス間通信: main→renderer)

- メニューで背景色を変更してみる

app.js

```
...
app.on('ready', () => {
  ...
  Menu.setApplicationMenu(Menu.buildFromTemplate([
    { label: "背景色", submenu: [
      { label: "白", type: "radio", checked: true,
        click: () => { mainWindow.webContents.send("setbgcolor", "white"); } },
      { label: "赤", type: "radio",
        click: () => { mainWindow.webContents.send("setbgcolor", "red"); } },
    ] },
    { label: "開発者ツール", accelerator: 'Ctrl+Shift+I',
      click: () => { mainWindow.toggleDevTools(); } },
    { label: "終了", role: "close" },
  ]));
});
```

ipc (プロセス間通信: main→renderer)

- メニューで背景色を変更してみる

main.js

```
const {remote, ipcRenderer} = require('electron');  
  
ipcRenderer.on("setbgcolor", (e, data) => {  
  document.body.style.backgroundColor = data;  
});
```

ipc

- 逆方向の通信を行いたい場合やレスポンスを返したい場合などは以下参照
 - <https://github.com/electron/electron/blob/master/docs-translations/jp/api/ipc-main.md>
 - <https://github.com/electron/electron/blob/master/docs-translations/jp/api/ipc-renderer.md>
 - <http://electron.atom.io/docs/api/ipc-main/>
 - <http://electron.atom.io/docs/api/ipc-renderer/>
 - <http://sourcechord.hatenablog.com/entry/2015/11/03/124814>
 - ここは情報が古いので一部(require部分)コードを書き換える必要あり



外部サイトにアクセスする

new template

- また新しいelectron_templateを準備して下さい

KMCの例会日程を表示してみる

- ここから情報を取得することにする

Kyoto University Micro Computer Club

京大マイコンクラブ

トップ イベント プロジェクト 部員 入部案内 リンク集

例会日程

07/11 (月) 4共14
07/14 (木) 4共14

お知らせ

03 Apr 2016 KMCのことが気になる新入生の方へ

履歴 >

<https://www.kmc.gr.jp/>

KMCの例会日程を表示してみる

- 以下を記述して実行

main.js

```
const {remote} = require('electron');  
fetch("https://www.kmc.gr.jp/").then((res) => {  
  return res.text();  
}).then((html) => {  
  alert(html);  
});
```

KMCの例会日程を表示してみる

```
MyApp
<!DOCTYPE html>
<html lang="ja" data-path="index">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/assets/kmc-153bf88ab4a7b3f99f1caf45f6f4849e.css">
    <title>京大マイコンクラブ (KMC)</title>
    <meta content="京大マイコンクラブ (KMC)" name="author">
    <meta content="京大マイコンクラブ (KMC)の公式ページ。活動記録や製作中のゲーム・ソ  
フトに関する情報を公開しています。" property="og:description">
    <meta content="京大マイコンクラブ" property="og:title">
    <meta content="website" property="og:type">
    <meta
      content="https://www.kmc.gr.jp/assets/logo-6acd7dc5d4e946cb26e502ee8e1937d0.png"
      property="og:image">
    <meta content="https://www.kmc.gr.jp/" property="og:url">

    <meta name="twitter:card" content="summary">
    <meta name="twitter:site" content="@KMC_JP">
    <link href="/assets/favicon-ba900f71790c45f9c90b910d822de10b.ico" rel="icon"
      type="image/vnd.microsoft.icon">
    <link href="/assets/apple-touch-icon-f8b2de78499c9c1a9a2496df1c5fde6c.png"
      rel="apple-touch-icon" type="image/png">
    <!--[if lt IE 9]>
    <script
      src="/assets/vendor/html5shiv-57ea15ceccc00b7440629a9f697249c.js"> </script>
    <![endif]-->
    <script type="text/javascript">
      var _gaq = _gaq || [];
      _gaq.push(['_setAccount', 'UA-39763858-1']);
      _gaq.push(['_trackPageview']);
      (function() {
        var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async =
true;
        ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') +
'.google-analytics.com/ga.js';
        var s = document.getElementsByTagName('script')[0];
        s.parentNode.insertBefore(ga, s);
      })();
    </script>
  </head>
  <body>
    <header role="banner">
      <h1>京大マイコンクラブ</h1>
      <nav>
        <a href="/" rel="start"></a>
        <ul><li class="navlink-index">
          <a href="/">トップ</a>
        </li><li class="navlink-events">
          <a href="/events/">イベント</a>
        </li><li class="navlink-projects">
          <a href="/projects/">プロジェクト</a>
        </li></ul>
      </nav>
    </header>
  </body>
</html>
```

KMCの例会日程を表示してみる

- <https://www.kmc.gr.jp/> のソースが取得できたので
後はいい感じに処理する

```
53 
54 <aside class='meetings'>
55   <h1><a href='/guidance/meetings.html'>例会日程</a></h1>
56
57
58   <table>
59     <thead>
60       <tr><th scope='col'>日付</th>
61       <th scope='col'>場所</th></tr>
62     </thead>
63
64     <tr><td>07/11 (月)</td>
65     <td>4共14</td></tr>
66
67     <tr><td>07/14 (木)</td>
68     <td>4共14</td></tr>
69
70   </table>
71
72 </aside>
73
```

演習

- この続きを書いていい感じに表示してみてください

main.js

```
const {remote} = require('electron');
fetch("https://www.kmc.gr.jp/").then((res) => {
  return res.text();
}).then((html) => {
  html = html.replace(/[¥r¥n]/g, "");
  const meetingstr = html.match(/<aside class='meetings'>(.*?)<¥/aside>/)[1];
  console.log(meetingstr);
  /* fill in here */
});
```

演習

- パース部分だけ．表示するところはいい感じにやってください．

main.js

```
const {remote} = require('electron');
fetch("https://www.kmc.gr.jp/").then((res) => {
  return res.text();
}).then((html) => {
  html = html.replace(/[\r\n]/g, "");
  const meetingstr = html.match(/<aside class='meetings'>(.*?)</aside>)[1];
  console.log(meetingstr);
  const datastr = meetingstr.match(/<\/thead>(.*?)<\/table>)[1];
  const data = datastr.split("<\/tr>");
  for (let i = 0; i < data.length-1; ++i) {
    const [,date, loc] = data[i].match(/<td>(.*?)<\/td>.*<td>(.*?)<\/td>/);
    console.log(`date: ${date}, location: ${loc}`);
  }
});
```

演習

- 別解. こっちの方が綺麗.

main.js

```
const {remote} = require('electron');
fetch("https://www.kmc.gr.jp/").then((res) => {
  return res.text();
}).then((html) => {
  const doc = new DOMParser().parseFromString(html, 'text/html');
  const tr = doc.querySelectorAll(".meetings > table tr");
  for (let i = 0; i < tr.length; ++i) {
    const tds = tr[i].querySelectorAll("td");
    if (tds.length === 0) continue; // headerの場合
    const date = tds[0].textContent;
    const loc = tds[1].textContent;
    console.log(`date: ${date}, location: ${loc}`);
  }
});
```

- 曲別ランキング | 精密集計DX

clubdam.info/ranking/index/

精密集計DX

君の知らない物語(生音) / supercell

100,000点
全国平均: 83.071点
2015/08/12

音程 97
V&L 97
安定性 99
リズム 99
表現力 100

分析レポート
文句なし、パーフェクトです。プロも夢ではないでしょう。…もしかしてプロの方ですか？

安定性
裏えがち 100% まっすぐ

表現力
抑揚 100% しゃくり40回
こぶし9回 フォール0回

リズム
タメ 100% 走り

ロングトーン
上手さ 100%

ビブラート
上手さ 100% 合計27.4秒 タイプボックス型

音域
m2A~hiD キー±0

1. 君の知らない物語(生音)

1位	16/05/24	ボムボム	100,000	496
2位	16/01/11	クオンマ	100,000	493
3位	16/04/22	お味噌汁	100,000	492
4位	15/08/12	***	100,000	491
5位	16/02/12	みかん	100,000	491

3. ひまわりの約束(生音)

1位	15/08/31	やらいで	100,000	496
2位	15/09/03	レイン	100,000	491
3位	16/03/24	***	100,000	491
4位	16/05/26	しおまる	100,000	490
5位	15/11/20	بوي اوغورا	100,000	489

5. 残酷な天使のテーゼ / 高橋洋子

1位	16/06/19	☆REIYA☆	100,000	494
2位	16/03/29	さいじゃく	100,000	492

6. only my railgun / fripSide

1位	16/06/19	はぁマジ糞	100,000	495
2位	16/06/18	みかん	100,000	493

他の応用例

- webllioから辞書データ取得（音声も）
 - <http://ejje.webllio.jp/content/kmc>



他の応用例

- ニコ動のクライアントを作る
- 2ch監視用クライアントを作る
- ポケモンの画像を集める
- オレオレブラウザを作る



データの保存

localStorage

- 昔ちょっと触れたlocalStorageで同じようにデータを保存できる

index.html

```
<button id="btnAdd"></button>
```

main.js

```
'use strict'  
const {remote} = require('electron');  
let val = localStorage.times || 0;  
const update = () => {  
  document.getElementById("btnAdd").textContent = val;  
};  
document.getElementById("btnAdd").addEventListener("click", (e) => {  
  ++val;  
  localStorage.times = val;  
  update();  
});  
update();
```

app.js側でデータを保存するには

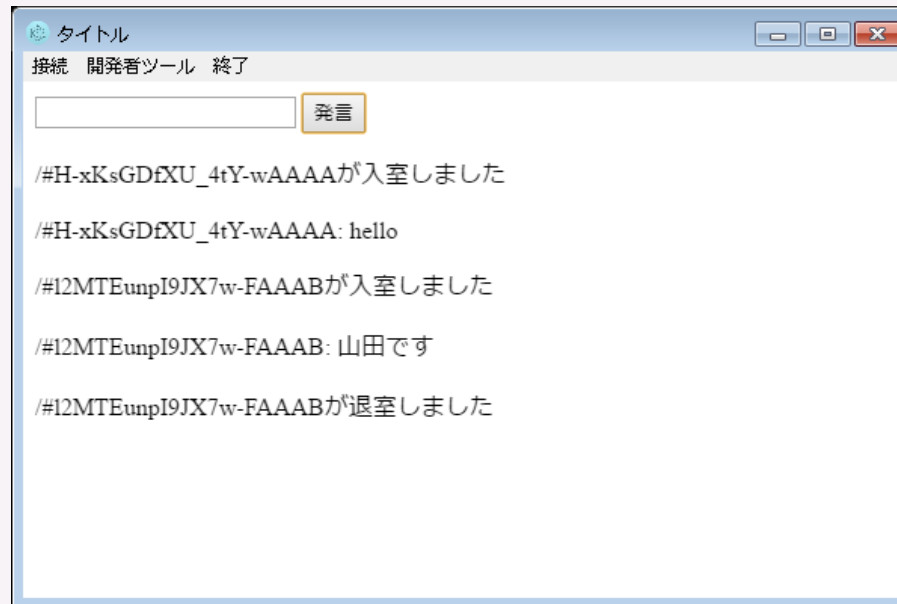
- 新しくモジュールが必要なので紹介だけ
- electron-json-storageを使うと良い
 - <http://qiita.com/KimuraTakaumi/items/fcae3fb9ca62143a00b4>



socket.io + electron

socket.io + electron

- マイクラのように、ゲームのクライアントと共にサーバも配布し、ユーザーがサーバを立て、接続して遊ぶ形態
- チャットを作ったので、配布したコードを見ながら解説する
 - works/10_chat



socket.io + electron

- これをreleaseするときは注意が必要
- 何も考えずreleaseを動かすと, socket.ioがないと言われる
- そこで, socket.ioのモジュールも一緒に含める

```
$ cd app  
$ npm install socket.io --save  
$ node release.js
```




electron + canvas

electron + canvas

- canvasは普通のブラウザでも使える機能だけど面白いのでここでとりあげる

app.js

```
mainWindow = new BrowserWindow({  
  width: 640,  
  height: 480,  
  useContentSize: true,  
  resizable: false,  
});
```

electron + canvas

- `useContentSize: true`
 - サイズ指定(`width`, `height`)の値が示すものが
ウィンドウの枠やメニューバーを含めたサイズではなく
ページの表示領域(クライアント領域)のサイズとする

app.js

```
mainWindow = new BrowserWindow({  
  width: 640,  
  height: 480,  
  useContentSize: true,  
  resizable: false,  
});
```

electron + canvas

- `resizable: false`
 - ウィンドウのリサイズ不可

app.js

```
mainWindow = new BrowserWindow({  
  width: 640,  
  height: 480,  
  useContentSize: true,  
  resizable: false,  
});
```

electron + canvas

- `resizable: false`
 - ウィンドウのリサイズ不可

app.js

```
mainWindow = new BrowserWindow({  
  width: 640,  
  height: 480,  
  useContentSize: true,  
  resizable: false,  
});
```

electron + canvas

index.html

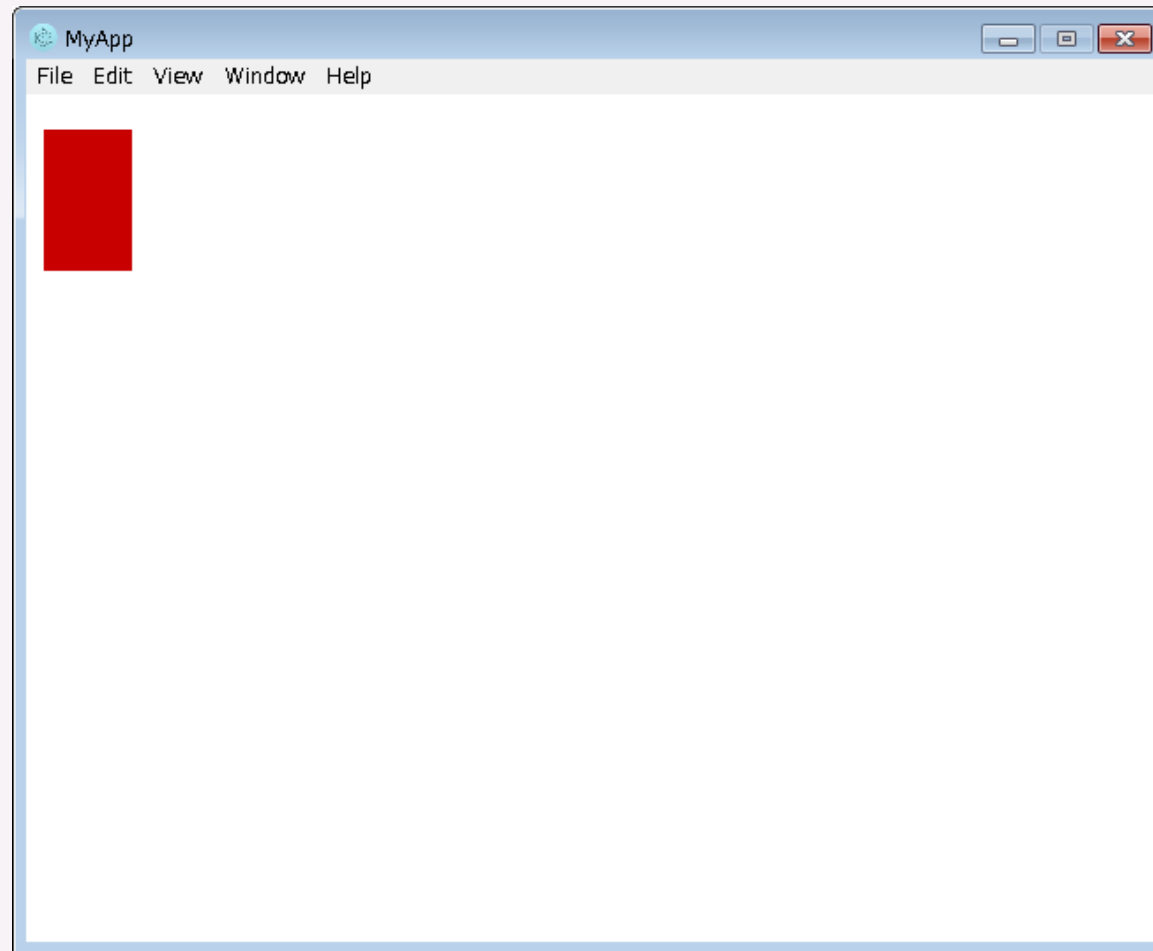
```
<canvas id="canvas" width="640" height="480"></canvas>
```

main.js

```
'use strict'  
const canvas = document.getElementById("canvas");  
const ctx = canvas.getContext('2d');  
ctx.fillStyle = "rgb(200, 0, 0)";  
ctx.fillRect(10, 20, 50, 80);
```

長方形の描画

- 長方形が表示されるはず



長方形の描画

- `ctx.fillStyle = "rgb(200, 0, 0)";`
 - 色を指定
 - もちろん`rgba(R,G,B,alpha)`で半透明もできる
- `ctx.fillRect(10, 20, 50, 80);`
 - 左上の頂点を(10,20)とする横幅50,縦幅80の長方形を描画

長方形の描画

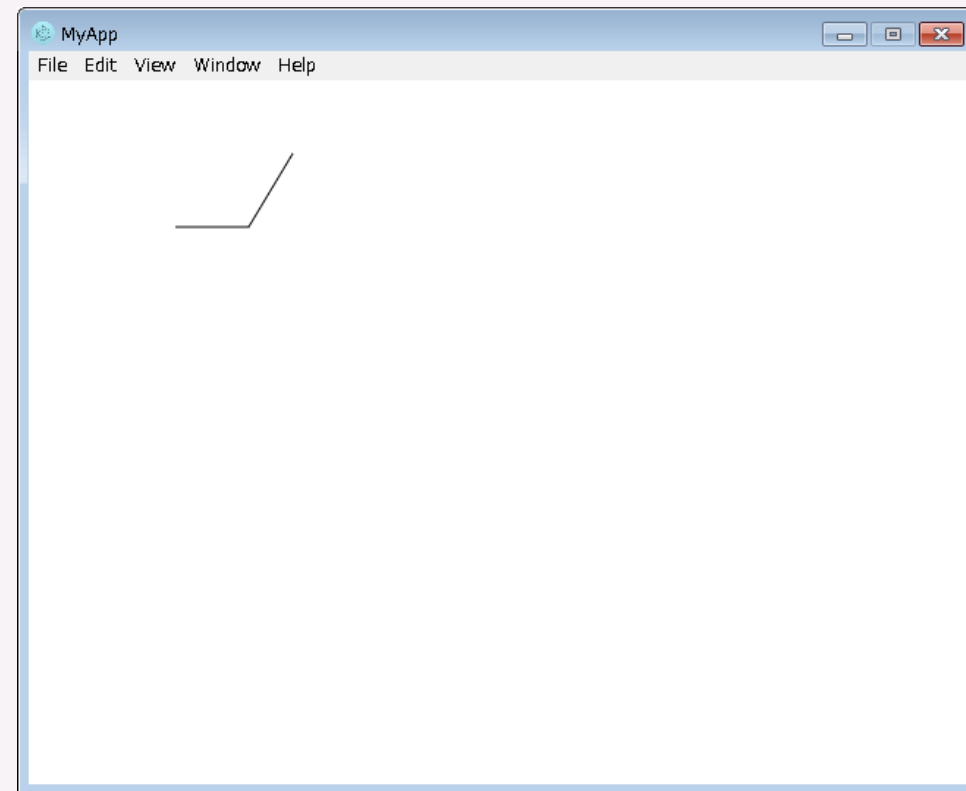
- `ctx.fillRect(x, y, w, h)`
 - 中也塗る
- `ctx.strokeRect(x, y, w, h)`
 - 枠だけ
- `ctx.clearRect(x, y, w, h)`
 - その領域を消去する

直線と円弧

- 曲線を設定した後に一気に描画する

main.js

```
'use strict'  
const canvas = document.getElementById("canvas");  
const ctx = canvas.getContext('2d');  
// 曲線の設定  
ctx.beginPath();  
ctx.moveTo(100, 100);  
ctx.lineTo(150, 100);  
ctx.lineTo(180, 50);  
// 描画  
ctx.stroke();
```



直線と円弧

- `ctx.beginPath()`
 - 新しい曲線の開始を表す
- `ctx.moveTo(x, y)`
 - ペン先を(x, y)に瞬間移動する
- `ctx.lineTo(x, y)`
 - (x, y)に向かって線分を描く(設定するだけで画面には変化なし)
 - ペン先は(x, y)に移動する

直線と円弧

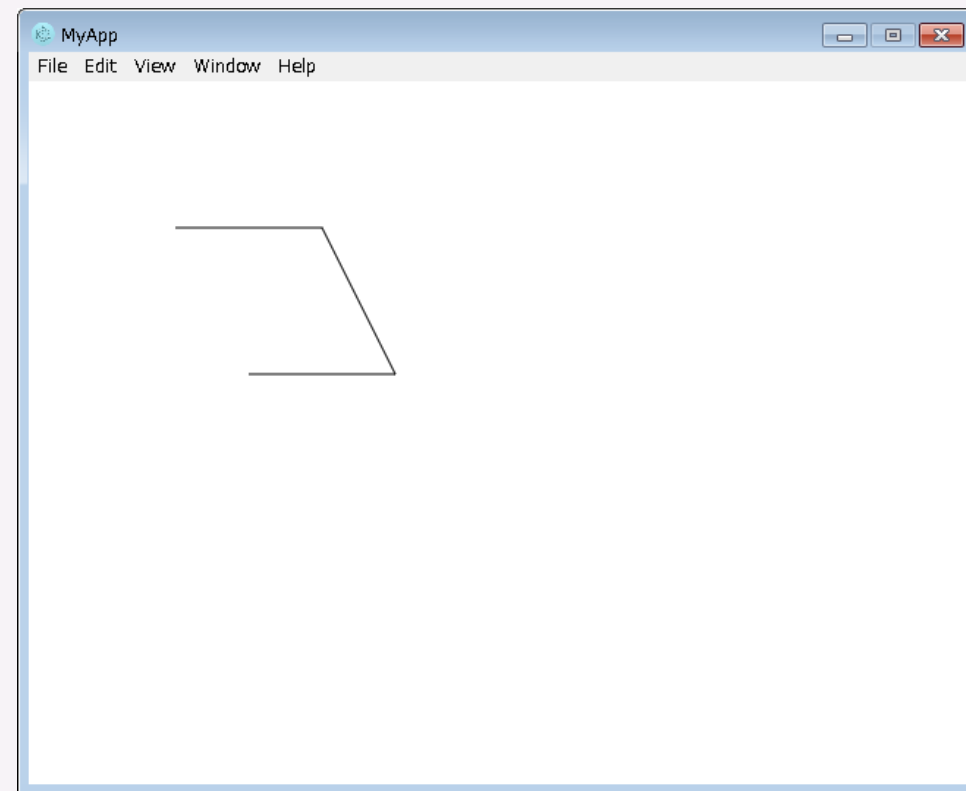
- `ctx.stroke()`
 - 設定した曲線の輪郭を描く (実際に画面に反映する)
- `ctx.closePath()`
 - 前回の`moveTo`の位置に`lineTo`する (閉じて領域にする)
- `ctx.fill()`
 - 自動的に`closePath`し, その領域を塗りつぶす

直線

- 例 (stroke)

main.js

```
'use strict'  
const canvas = document.getElementById("canvas");  
const ctx = canvas.getContext('2d');  
// 曲線の設定  
ctx.beginPath();  
ctx.moveTo(100, 100);  
ctx.lineTo(200, 100);  
ctx.lineTo(250, 200);  
ctx.lineTo(150, 200);  
// 描画  
ctx.stroke();
```

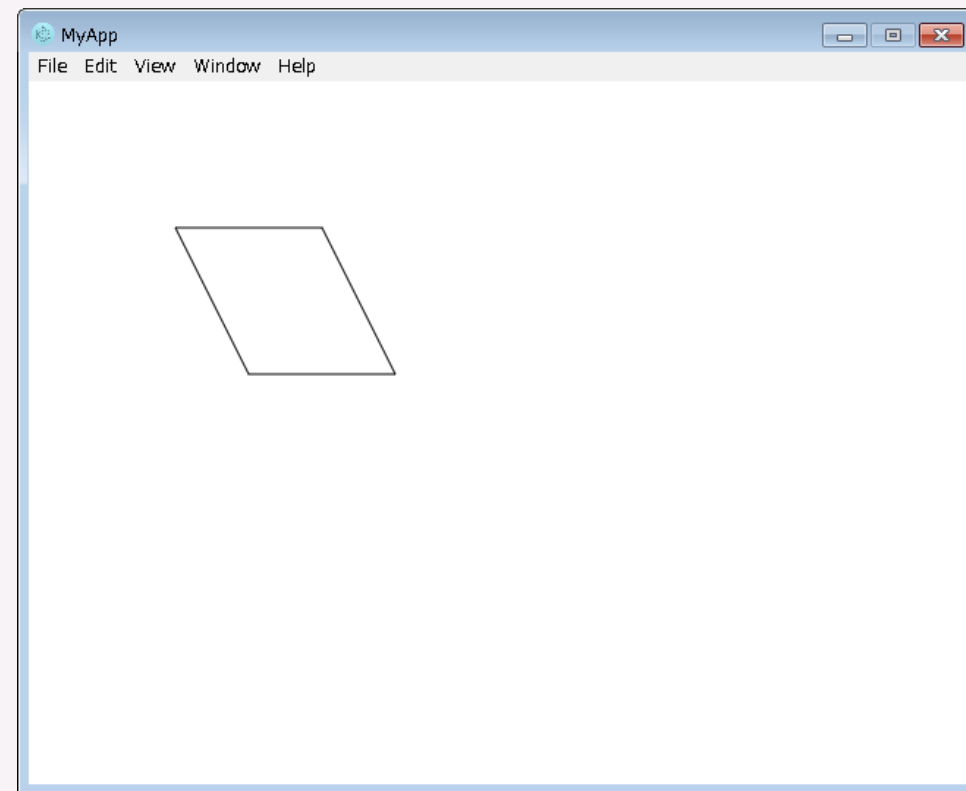


直線

- 例 (closePath, stroke)

main.js

```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
// 曲線の設定
ctx.beginPath();
ctx.moveTo(100, 100);
ctx.lineTo(200, 100);
ctx.lineTo(250, 200);
ctx.lineTo(150, 200);
ctx.closePath();
// 描画
ctx.stroke();
```

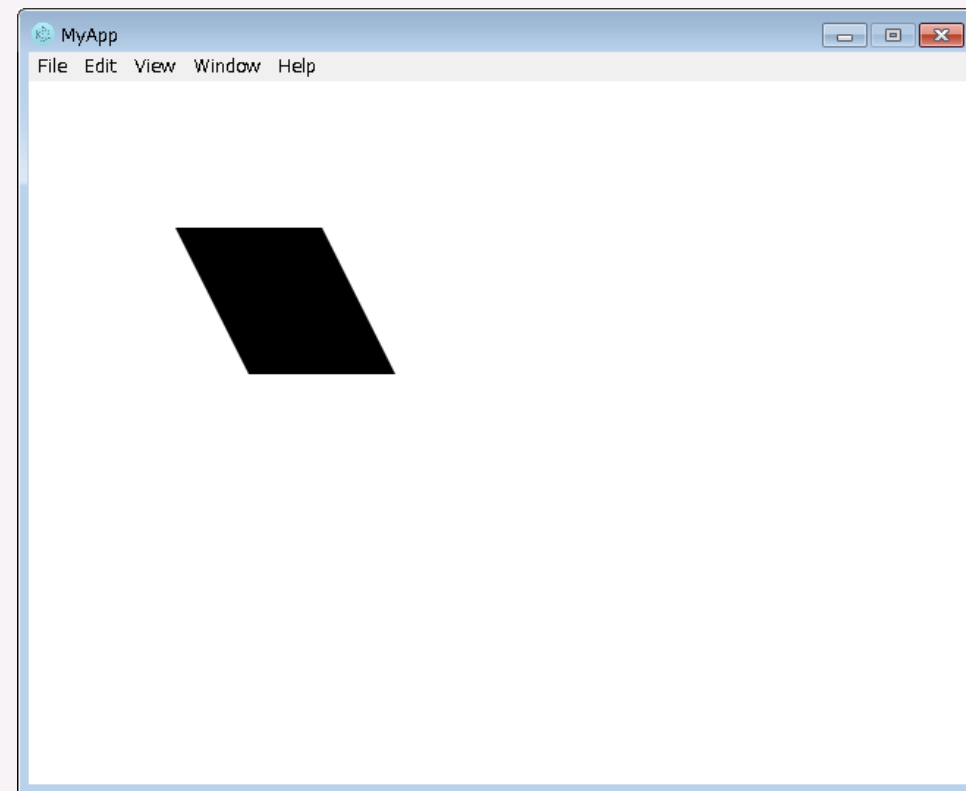


直線

- 例 (stroke)

main.js

```
'use strict'  
const canvas = document.getElementById("canvas");  
const ctx = canvas.getContext('2d');  
// 曲線の設定  
ctx.beginPath();  
ctx.moveTo(100, 100);  
ctx.lineTo(200, 100);  
ctx.lineTo(250, 200);  
ctx.lineTo(150, 200);  
// 描画  
ctx.fill();
```

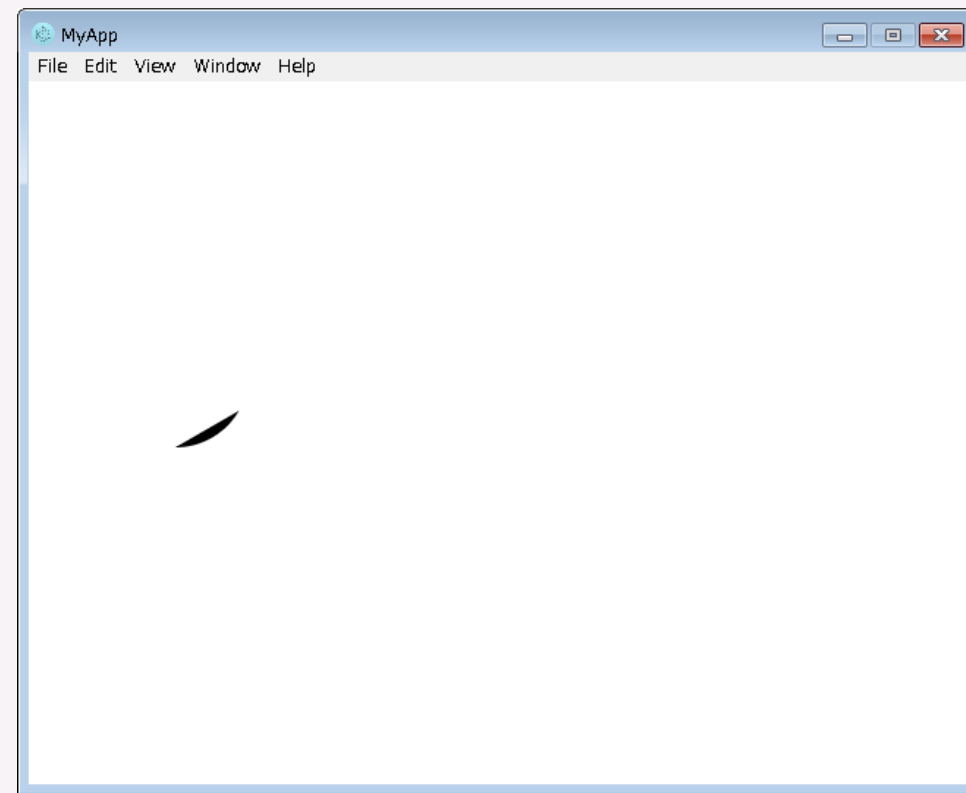


円弧

- `ctx.arc(x, y, radius, startAngle, endAngle, isAnticlockwise)`

main.js

```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
// 曲線の設定
ctx.beginPath();
ctx.arc(100, 200, 50, Math.PI/6, 3*Math.PI/6, false);
// 描画
ctx.fill();
```

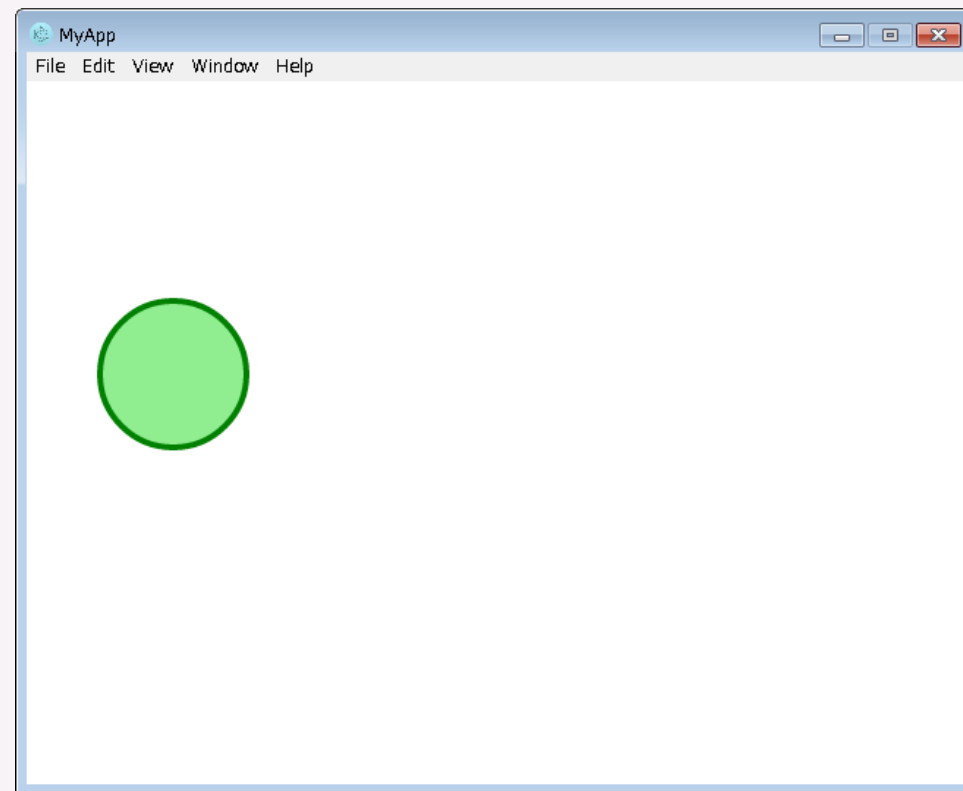


描画スタイル

- 例 (stroke)

main.js

```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
// 曲線の設定
ctx.beginPath();
ctx.strokeStyle = "green";
ctx.fillStyle = "lightgreen";
ctx.lineWidth = 4;
ctx.arc(100, 200, 50, 0, 2*Math.PI, false);
// 描画
ctx.fill();
ctx.stroke();
```



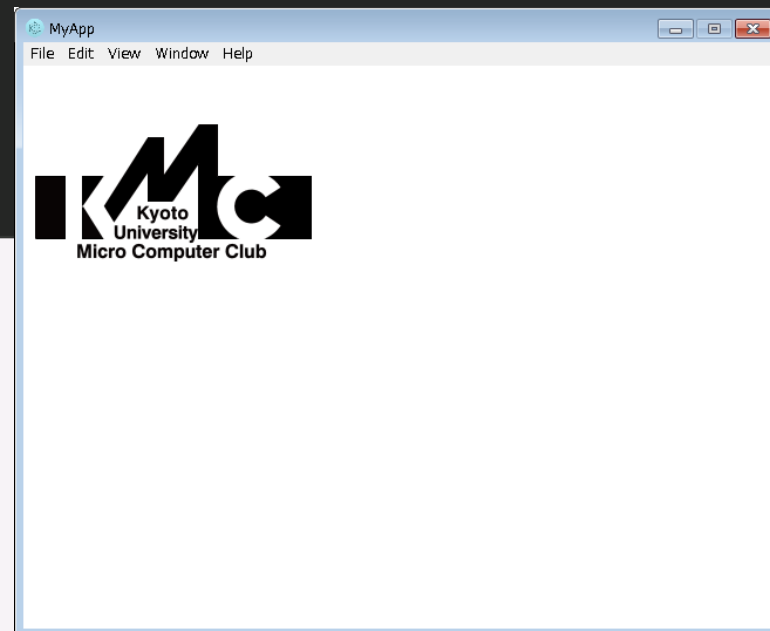
描画スタイル

- `ctx.strokeStyle` = "線の描画色"
- `ctx.fillStyle` = "塗りつぶしの描画色"
- `ctx.lineWidth` = 線の太さ
- その他のスタイル
 - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Applying_styles_and_colors

画像の描画

main.js

```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
new Promise((fulfilled, rejected) => {
  const imgkmc = new Image();
  imgkmc.src = "https://www.kmc.gr.jp/assets/logo-6acd7dc5d4e946cb26e502ee8e1937d0.png";
  imgkmc.onload = () => { fulfilled(imgkmc); };
}).then((img) => {
  ctx.drawImage(img, 10, 50);
});
```

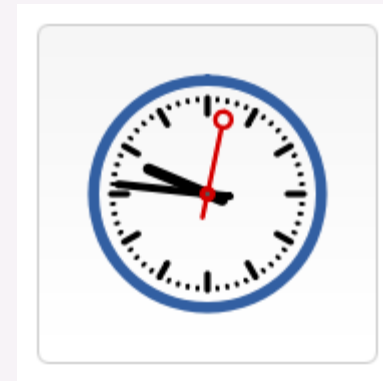


画像の描画

- `ctx.drawImage(imageElement, x, y)`
 - 左上座標を (x, y) として画像を描画
 - 画像 `imageElement` はHTML要素
- `ctx.drawImage(imageElement, x, y, w, h)`
 - 画像を横幅 w , 縦幅 h で上と同じように描画

アニメーション例

https://developer.mozilla.org/ja/docs/Web/Guide/HTML/Canvas_tutorial/Basic_animations



演習

1. 正三角形を描いてみよう
2. クリックしたところに円が表示されるようにしてみよう
 - クリックする度に円が増えていく
 - Hint: canvasもHTML要素
3. 円をプレイヤーにして
十字キーで移動できるようにしてみよう

演習

1. 正三角形を描いてみよう

main.js

```
'use strict'  
const canvas = document.getElementById("canvas");  
const ctx = canvas.getContext('2d');  
ctx.beginPath();  
ctx.moveTo(200, 200);  
ctx.lineTo(300, 200);  
ctx.lineTo(250, 200-50*Math.sqrt(3));  
ctx.closePath();  
ctx.stroke();
```

演習

1. 正三角形を描いてみよう

- 正N角形を描く例

main.js (ctx=…まで省略)

```
const N = 10, R = 100, centerX = 300, centerY = 250;
const getX = (n) => {
  return centerX + R * Math.cos(-Math.PI/2 + Math.PI*n*2/N);
};
const getY = (n) => {
  return centerY + R * Math.sin(-Math.PI/2 + Math.PI*n*2/N);
};
for (let i = 0; i < N; ++i) {
  ctx.beginPath();
  ctx.moveTo(getX(i), getY(i));
  ctx.lineTo(getX((i+1)%N), getY((i+1)%N));
  ctx.stroke();
}
```


演習

2. クリックしたところに円が表示されるようにしてみよう

main.js

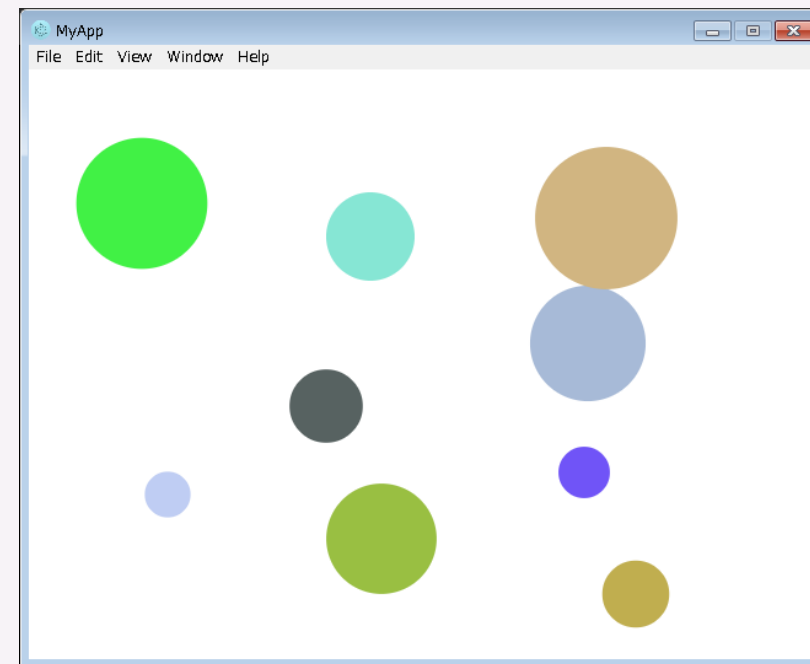
```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
canvas.addEventListener("click", (e) => {
  ctx.beginPath();
  ctx.arc(e.clientX, e.clientY, 30, 0, Math.PI*2);
  ctx.fill();
});
```

演習

2. クリックしたところに円が表示されるようにしてみよう

main.js (装飾版)

```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
canvas.addEventListener("click", (e) => {
  const r = Math.floor(Math.random()*200)+55;
  const g = Math.floor(Math.random()*200)+55;
  const b = Math.floor(Math.random()*200)+55;
  const radius = Math.random()*50+10;
  ctx.beginPath();
  ctx.fillStyle = `rgb(${r}, ${g}, ${b})`;
  ctx.arc(e.clientX, e.clientY, radius, 0, Math.PI*2);
  ctx.fill();
});
```



演習

3. 円をプレイヤーにして 十字キーで移動できるようにしてみよう

main.js [1/2]

```
'use strict'
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
const SPEED = 5;
const key = [];
let px = 100, py = 100;
document.addEventListener("keydown", (e) => { key[e.keyCode] = true; });
document.addEventListener("keyup", (e) => { key[e.keyCode] = false; });
```

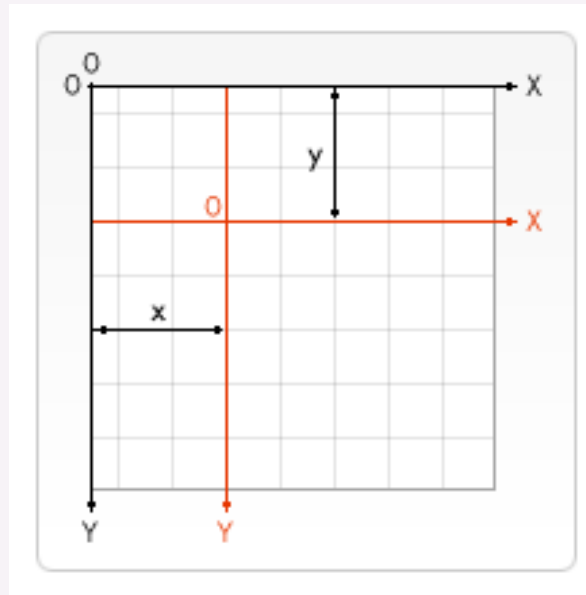
演習

main.js [2/2]

```
const mainloop = () => {
  setTimeout(mainloop, 1000/60);
  let dx = 0, dy = 0;
  if (key[37]) --dx;
  if (key[38]) --dy;
  if (key[39]) ++dx;
  if (key[40]) ++dy;
  px += dx*SPEED, py += dy*SPEED;
  draw();
};
const draw = () => {
  ctx.clearRect(0, 0, 800, 600);
  ctx.beginPath();
  ctx.arc(px, py, 50, 0, Math.PI*2);
  ctx.fill();
};
mainloop();
```

平行移動

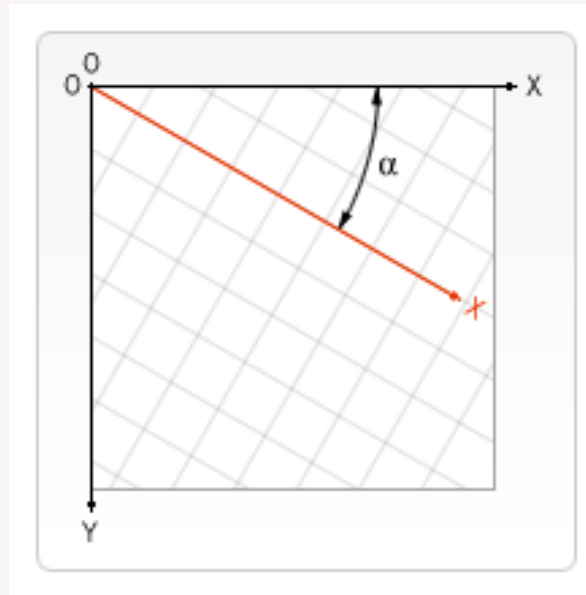
- `ctx.translate(dx, dy)`
 - これ以降の描画ではこの座標系になる



https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Transformations

回転

- `ctx.rotate(angle)`
 - これ以降の描画ではこの座標系になる



https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Transformations

拡大・縮小・反転

- `ctx.scale(scaleX, scaleY)`
 - x方向にscaleX倍, y方向にscaleY倍するような座標系となる
 - これ以降の描画ではこの座標系になる
 - scale値に負の値を指定すると上下や左右が反転する

状態の一時保存

- `ctx.save()`
 - 今の状態(スタイル等設定状態)を一時保存する
- `ctx.restore()`
 - 前回保存した状態を復元する
 - 既にそれが復元されたことがあれば,
前々回保存した状態を復元する



Document

electronのdocument

- English
 - <http://electron.atom.io/docs/>
- Japanese
 - <https://github.com/electron/electron/blob/master/docs-translations/jp/api/ipc-main.md>



今後

今後の予定

- これでこのプロジェクトは糸冬了ですが来週にあほげーというのがあるので忙しいとは思いますがよかったら参加して下さい
- 「JavaScriptから始めるプログラミング2016」は終了してもまだまだJavaScriptに関してやり残したことがあるので夏休みに勉強会を開いてもいいかも(開きたい)

あほげーとは

- 公式サイト
 - <http://ahoge.info/>
- ブラウザで動くあほなゲームを24時間で作るイベント
- 予定
 - 7月15日(金) 21:00 お題発表
 - 7月16日(土) 21:00 作品投稿締切
 - 7月17日(日) 19:00頃 発表
 - 7月18日(月) 寝る

あほげーとは

- 7/15(金) 21:00 に部室に集まってわいわいゲームを作ります
- 自宅からでも参加できますが
部室に来て下されるとサポートできます
- 7/16(土)昼までに完成させて投稿して帰って寝る予定です

after this project

- 夏休みに続編やりたい
- やり残していること
 - ライブラリ・フレームワーク紹介
 - Three.js (3D)
 - jQuery (DOM)
 - Angular.js (framework)
 - React (framework)
 - phina.js (Game)
 - enchant.js (Game)
 - テンプレートエンジン・ラップ言語
 - Jade, EJS (HTML)
 - Less, Scss, Sass (CSS)
 - TypeScript (JS)

after this project

- やり残していること
 - ビット演算
 - Promise
 - prototype汚染
 - データ構造
 - メタプログラミング
 - フォームアプリケーション
 - Ajax
 - 演習
 -
 -
 -

after this project

- まだまだやることがあるので夏休みやりたさ
- 毎回誰かが何かについて発表する輪講形式
- 週1くらいでやりたい（8/9から毎週火曜日を考えています）
- お題を決めてそれぞれ何か作ってきて発表もしたい