計画行政学会 GIS 分科会 Webプログラミングセミナー

アジェンダ

- Git / Githubの説明 (こちらは飛ばしていただいてOKです)
- Webプログラミング入門
- ソースコードの公開とは？

Gitの説明に関しては飛ばしても大丈夫です。

Gitとは

プログラムの共有用に、時系列で
使い慣れることはないが、使わないと多人数の開発は成り立たないので、コードベースのやり取りを行う場合ほぼ必須。

テキストフォーマットであれば全てを差分管理できる都合、プログラムに限らず、ドキュメントやスライド、回路図やデータフォーマットを共有することにも使える

Gitコマンド

```
git init
```

```
git add .
git commit -m "ここにコメントを入れる"
```

GitHubとは？

GitHub(現　Microsoft傘下)が運営する開発者向けのソースコード管理サービス。
エンタープライズの仕様であるが、オープンソースの公開場所として世界的なシェアを誇り、ボランティアベースの活動の支援もされている。

GitLabとは？

上記のGitHubの代替として利用されることも多いサービス。
エンタープライズ仕様のものもあるが、セルフホストとして利用できるオープンソースも公開されており、例えば大学や企業のサーバーでクローズドな利用をすることもできる。

Bitbucket や Giteeなどの商用代替もある。
また、大手のSaaS(GAFAM)が提供するクラウドに含まれていたり、Gitの管理ソフトをプロジェクト管理に組み込んでいるパターンもある(Nulabなど)。

どこでコードを書くか

1. Web上のエディター

インストール不要、すぐに試せる

- **CodePen** - codepen.io
- **JSFiddle** - jsfiddle.net
- **Wandbox** - wandbox.org
- **CodeSandbox** - codesandbox.io

メリット：

- すぐに始められる
- 結果がリアルタイムで見える
- 共有が簡単

デメリットではないが、htmlの書き方が省略されていたりするため、詳しい人と一緒に使い始めるのがなお良い。

2. テキストエディタ

自分のPCで開発する

シンプル：

- メモ帳（Windows）
- テキストエディット（Mac）

高機能：

- **Visual Studio Code**

他にもテキストエディタはたくさんありますが、Microsoftから出されているVisual Studio Codeが現在ではかなりよくまとまっていて拡張も高くおすすめ。

エディタとIDE(統合開発環境)の垣根は曖昧だが、コンパイルが必要なプログラミング言語を利用したい場合はIDEを使うことが多い。R Studioもその類。

HTML と CSS

HTML/CSS入門

初心者のためのWeb制作

(関連概念としてWeb開発、というものもある)

Webページの3要素

- **HTML** → 構造（骨組み）
- **CSS** → 見た目（装飾）
- **JavaScript** → 動き

HTMLの基本構造 (HTML : HyperText Markup Language)

```html
<!DOCTYPE html> <html> <head> <title>ページタイトル</title> </head> <body> ここに内容を書く </body> </html>```
```

よく使うHTML構文

リスト (List)

箇条書き（順序なし）

```
<ul>
  <li>項目1</li>
  <li>項目2</li>
  <li>項目3</li>
</ul>
```

番号付きリスト（順序あり）

```
<ol>
  <li>最初</li>
  <li>次</li>
  <li>最後</li>
</ol>
```

リンク (Link)

基本のリンク

```
<a href="https://example.com">リンクテキスト</a>
```

新しいタブで開く

```
<a href="https://example.com" target="_blank">外部リンク</a>
```

同じページ内リンク

```
<a href="#section1">セクション1へ</a>
```

3. 画像挿入 (Image)

基本の画像

```
<img src="image.jpg" alt="画像の説明">
```

サイズ指定

```
<img src="image.jpg" alt="説明" width="300" height="200">
```

💡 **alt属性は必須！** 画像が表示されない時の代替テキスト

4. テーブル (Table)

```
<table>
  <tr>
    <th>見出し1</th>
    <th>見出し2</th>
  </tr>
  <tr>
    <td>データ1</td>
    <td>データ2</td>
  </tr>
  <tr>
    <td>データ3</td>
    <td>データ4</td>
  </tr>
</table>
```

テーブルのタグ構造

タグ	意味	用途
`<table>`	テーブル全体	表の囲み
`<tr>`	table row	行
`<th>`	table header	見出しセル
`<td>`	table data	データセル

まとめ

要素	タグ	用途
リスト	`<ul>`, `<ol>`, `<li>`	箇条書き
リンク	`<a>`	他ページへ移動
画像	`<img>`	画像を表示
テーブル	`<table>`, `<tr>`, `<td>`	表形式データ

見出し

```
<h1>最も重要な見出し</h1>
<h2>中見出し</h2>
<h3>小見出し</h3>
<h4>さらに小さい見出し</h4>
<h5>h5見出し</h5>
<h6>最も小さい見出し</h6>
```

段落

```
<p>これは段落です。文章をまとまりで囲みます。</p>
<p>別の段落です。自動的に上下に余白ができます。</p>
```

ブロック要素

```
<div>
  <h2>セクションタイトル</h2>
  <p>このエリアをまとめてグループ化</p>
</div>
```

インライン要素

```
<p>この<span style="color: red;">部分だけ</span>赤くしたい</p>
```

フォーム

```
<form action="送信先URL" method="post">
  <!-- ここに入力欄を配置 -->
</form>
```

フォーム実用版

```
<!-- テキスト入力 -->
<input type="text" placeholder="名前を入力">

<!-- メールアドレス -->
<input type="email" placeholder="メールアドレス">

<!-- パスワード -->
<input type="password">

<!-- チェックボックス -->
<input type="checkbox"> 同意する

<!-- ラジオボタン -->
<input type="radio" name="gender"> 男性
<input type="radio" name="gender"> 女性

<!-- 送信ボタン -->
<input type="submit" value="送信">
```

ボタン

```
<button type="button">クリック</button>
<button type="submit">送信</button>
<button type="reset">リセット</button>
```

その他 (セマンティック要素)

```
<header> - ヘッダー
<nav> - ナビゲーション
<main> - メインコンテンツ
<footer> - フッター
<article> - 記事
<section> - セクション
<br> - 改行
```

CSSの使い方

インライン

```
<p style="color: red; font-size: 20px;">赤い文字</p>
```

内部スタイルシート（headに書く）

```
<head>
  <style>
    p {
      color: blue;
      font-size: 16px;
    }
  </style>
</head>
```

外部スタイルシート（別ファイル）

```
<!-- HTML側 -->
<head>
  <link rel="stylesheet" href="style.css">
</head>
```

```
/* style.css */
p {
  color: green;
  font-size: 18px;
}
```

基本構造

```
セレクタ {
  プロパティ: 値;
  プロパティ: 値;
}
```

複数のプロパティを並べて書くことができる

```
p {
  color: red;
  font-size: 20px;
  background-color: yellow;
}
```

適用範囲(セレクタ)

タグ

```
p {
  color: blue;
}
```

```
<p class="important">重要な段落</p>
```

クラス

```
.important {
  color: red;
  font-weight: bold;
}
```

```
<div id="header">ヘッダー</div>
```

クラス

```
#header {
  background-color: navy;
  color: white;
}
```

Markdownのすすめ

別添資料をご覧ください。
https://alt9800.github.io/Slide-test/2022-12-01-md-handson/

研究室単位での浸透に関して青山学院大学の古橋先生のゼミが参考になるかも。
https://github.com/furuhashilab

ボタンを押したら何かが起こる、計算する、データを保存する...
Webページを動かすのがJavaScriptの役割！

1. ユーザーの操作に反応する

クリック、入力、スクロールなどに反応

```
// ボタンをクリックしたらメッセージ表示
button.addEventListener('click', function() {
  alert('ボタンが押されました！');
});
```

できること：

- ボタンを押したら画面が変わる
- フォームに入力したら自動チェック
- メニューの開閉
- 画像のスライドショー

2. HTMLを動的に変更する

ページの内容をリアルタイムで書き換え

```
// 文字を変更
document.getElementById('text').textContent = '新しいテキスト';

// 要素を追加
const newItem = document.createElement('li');
newItem.textContent = '新しい項目';
list.appendChild(newItem);

// スタイルを変更
element.style.color = 'red';
```

できること：

- いいねボタンの数を増やす
- コメントを追加・削除
- ダークモード切り替え

3. 計算・データ処理

複雑な計算や条件判断

```
// 合計金額を計算
let price = 1000;
let quantity = 3;
let total = price * quantity;

// 条件分岐
if (age >= 18) {
  console.log('成人です');
} else {
  console.log('未成年です');
}
```

できること：

- ショッピングカートの合計計算
- 入力チェック（メールアドレス形式など）
- フィルター・検索機能
- ゲームのスコア計算

4. サーバーとデータのやり取り

ページを更新せずにデータを取得・送信

```
// サーバーからデータを取得
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    // データを画面に表示
    console.log(data);
  });
```

できること：

- 天気予報の表示 (WebAPIの読み込み)
- リアルタイムチャット
- 無限スクロール (遅延しながら読み込んだりユーザアクションの検知)
- ログイン・会員登録

JavaScriptの活躍場所

Webで動くものにはほぼJavascriptが介在していると言えます。

利用

HTMLにおける使い方は二種類あり、

```
<script src="script.js"></script>
```

のように`<script>`タグ内に記述してリソースを呼び出すパターンと、

```
<script>
  // コードを記述
</script>
```

のように記述するパターンとある。

地図ライブラリ入門

Leaflet / MapLibre / Deck.gl

地図ライブラリの比較

Leaflet
軽量でシンプル、技術者以外にもチェーンで記載する書き方はわかりやす...かった
が、最近メジャーアップデートされて文法が変わった。

MapLibre
高機能、ベクトルタイル対応、WebGLベース

Deck.gl
大量データの可視化に強い

Leaflet

Leafletとは

最も人気のある地図ライブラリ

特徴：

- 軽量（39KB）
- シンプルで学びやすい
- プラグインが豊富
- モバイル対応

使用例：GitHub、Pinterest、Flickr

Leaflet 基本の書き方

```
const map = L.map('map').setView([35.6812, 139.7671], 13);
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);
```

とりあえずv1の文法を覚えておこう。

Leafletでできること

- マーカー表示
- ポップアップ
- ポリゴン描画
- GeoJSON読み込み
- イベント処理
- 背景地図タイルの読み込み

シンプルな地図アプリからモバイル用のアプリまで使える。

MapLibre

MapLibreとは

Mapbox GL JSのオープンソース版

特徴：

- ベクトルタイル対応
- 3D表示可能
- カスタムスタイル
- 高速レンダリング

使用例：位置情報サービス、ダッシュボード

MapLibre 基本の書き方

```
const map = new maplibregl.Map({
  container: 'map',
  style: 'https://demotiles.maplibre.org/style.json',
  center: [139.7671, 35.6812],
  zoom: 13
});
```

MapLibreでできること

- ベクトルタイル表示
- 3D建物表示
- カスタム地図デザイン
- 地図の回転・傾き
- アニメーション

デザインにこだわる地図に最適

Deck.gl

Deck.glとは

大量データ可視化に特化したライブラリ

特徴：

- WebGL活用で高速
- 数百万点のデータも表示可能
- 多彩なレイヤー
- 3D表現

開発元：Uber

Deck.gl 基本の書き方

```
new deck.DeckGL({
  container: 'map',
  initialViewState: {
    longitude: 139.7671,
    latitude: 35.6812,
    zoom: 13
  },
  controller: true,
  layers: []
});
```

Deck.glでできること

- ヒートマップ
- 3Dヘキサゴン
- 軌跡アニメーション
- 大量ポイント表示
- データビジュアライゼーション

ビッグデータの可視化に最適

3つのライブラリ比較

	Leaflet	MapLibre	Deck.gl
難易度	易しい	中程度	やや難
用途	一般的な地図	デザイン重視	データ可視化
サイズ	軽量	中程度	重い
3D	×	○	◎

まとめ

まず試すなら

Leaflet

デザインにこだわるなら

MapLibre

大量データを扱うなら

Deck.gl

すべてオープンソースかつ無料で使える

おすすめサイト

kepler.gl

https://kepler.gl/

Dekart - SQL to Map Instantly - Visualize BigQuery, Snowflake & Wherobots Data.

https://dekart.xyz/