

エンジニア体験インターンシップ 基礎編

2022/08/02

株式会社ナビタイムジャパン



概要

ナビタイムジャパンがサービスを提供する上で開発、および活用している経路探索技術や地図、地点のデータなどを用いて、当社が用意した課題に沿ってWebアプリケーションの開発を体験していただきます。

日程

- 8/2 : 講義 (HTML / CSS / JavaScript / デバッグ方法)
- 8/8～8/10 : 開発① (課題に沿ってWebサイトを作成)
- 8/23～8/25 : 開発② (課題に沿ってWebサイトを作成)

基礎編アジェンダ

01 基礎編概要
今日のゴール確認

02 HTML

03 CSS

04 JavaScript

05 jQuery

06 デバッグについて

01_Summary
Chapter 01

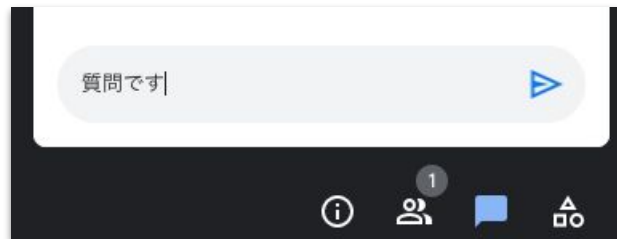
今日のゴール

HTML, CSS, JavaScript / jQuery の基礎を学ぶ

今回の講義や演習を通して基礎を理解し
今後のインターンの講義や開発に役立てる

講義や演習中でわからないことがあればいつでも聞いてください。
どんな些細なことでも質問してくれて構いません！

- 挙手をして口頭で質問する
- 質問したいことをコメントに書く
- 『質問です』と書き込んで口頭で質問する



02_HTML
Chapter 01

HTMLの書き方

Hyper Text Markup Language

- Webの基本となるハイパーテキストを記述するための
マークアップ言語
 - ウェブページを作成するために開発された言語
 - 命令を書く Java などのプログラミング言語と違い、構造を書く言語
 - Markdown に近い
- HTML の役割は文書の中で見出し・段落・リスト等の各部分が果たしている役割をコンピュータが理解できるように**構造**を定義すること

sample-1.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8"/>
    <title>NAVITIME インターンシップ</title>
  </head>

  <body>
    <h1>HTMLについて</h1>
  </body>
</html>
```

headタグ

タイトルなどのページ情報と、CSSなど外部ファイルの読み込みを記述する

bodyタグ

文章や画像など、実際にブラウザ上に表示される要素を記述する

result

HTMLについて

基本的にHTMLの要素は上から下へ配置される

sample-2.html

```
<body>
  <div>あいうえお</div>
  <div>かきくけこ</div>
  <span>さしすせそ</span>
  <span>たちつてと</span>
</body>
```

result

あいうえお

かきくけこ

さしすせそ

たちつてと

ブロック要素

中身に係わらず親要素いっぱい
に広がる

インライン要素

中身に応じて範囲が定められて
いる

| | 特徴 |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ブロック要素 | <ul style="list-style-type: none">● 範囲が親要素いっぱい広がる● 代表的なタグ<ul style="list-style-type: none">○ div○ h1 |
| インライン要素 | <ul style="list-style-type: none">● 範囲は中身に応じて決まる● CSSで幅と高さを指定できない● 上下左右の余白は指定できる<ul style="list-style-type: none">○ span○ a |

- 見出しで利用するタグ
- <h1>～<h6>までである
- hタグは数字が小さい順(h1 → h2 → h3)で利用する

sample-3.html

```
<body>  
  <h1>インターンシップ</h1>  
  <h2>基礎編</h2>  
  <h3>HTML編</h3>  
</body>
```

result

インターンシップ

基礎編

HTML編

- href="URL"で書いたURLのページを表示する
- target="_blank"を指定すると、別タブでページを開くことができる

sample-4.html

```
<body>
  <!-- aタグ(リンク) -->
  <a href="https://www.navitime.co.jp/">NAVITIME</a>
  <!-- aタグ(別タブで開く) -->
  <a href="https://github.co.jp/" target="_blank" rel="noopener">github</a>
</body>
```

result

[NAVITIME](https://www.navitime.co.jp/) [github](https://github.co.jp/)

ulタグとliタグ

- ulタグ
 - 箇条書きリストの表示ができる
- liタグ
 - リストの項目を表示する。ulタグの子要素として利用する

sample-5.html

```
<body>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
  </ul>
</body>
```

result

- HTML
- CSS
- JavaScript

- inputタグ
 - 文字を入力する枠ができる
- buttonタグ
 - ボタンを押した時に何か処理を行いたい場合は、Javascriptを書く必要がある

sample-6.html

```
<body>
  <input type="text" placeholder="目的地を入力"/>
  <button type="button">検索</button>
</body>
```

result

03_CSS
Chapter 01

CSSの基本構文

セレクタとプロパティ

CSSを書くとうなる？

NAVITIME

- 周辺検索
- - 駅/店舗/住所
 - [現在地](#)
- 駅
- 駐車場
- グルメ/レストラン
- 賃貸物件
- 東京23区・三鷹市・武蔵野市 限定
タクシーを呼ぶ
- 機能一覧
- [TOPページ](#)
- トータルナビ・地図
 - [トータルナビ](#)
 - [地図](#)
 - [スポット検索](#)
 - [住所検索](#)
 - [ジャンル検索](#)
 - [テイクアウト・デリバリー・ド
ライブスルー店舗検索](#)
- 乗換検索・時刻表
 - [乗換案内](#)

CSSで見栄えを整えると...



Cascading Style Sheets

- ウェブページのスタイルを指定するためのスタイルシート言語
 - HTMLと同様に命令を書くJavaなどのプログラミング言語とは異なる
- HTMLで作った文書構造に対して、デザインを施し、見栄えを整える役割

- CSSによるデザイン指定をどのHTML要素に適用するのかを指定するために使われる
- JavaScriptから特定のHTML要素を対象にして処理を行いたい場合に要素を選択する目的でも使われる

セレクトタ

```
h1 {  
    background-color: #dddddd;  
    padding: 24px;  
}
```

| | HTML | 指定方法 | メモ |
|------------|-------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTMLタグ指定 | <code>NAVITIME</code> | <pre>span { color: red; }</pre> | スコープが広いので、基本的に使用しない |
| class名指定 | <code>NAVITIME</code> | <pre>.red-text { color: red; }</pre> | classの重複はOKなので、一気に複数箇所にスタイル指定できる |
| id名指定 | <code>NAVITIME</code> | <pre>#main { color: red; }</pre> | 1ファイルに対して同じidは1つしか利用できない |
| 複数のセレクトタ指定 | <code>NAVITIME</code> <code>NAVITIME</code> | <pre>.red-text, .blue-text { color: red; }</pre> | カンマ(,)でつなげると同時指定できる 他にもいろいろな指定方法がある 参考 :https://webliker.info/css-selector-cheat-sheet/ |

- CSSで適用するスタイルの種類を指す
- セレクタで指定した要素に対して、どのような変化をさせるのかを指定する

```
h1 {  
  background-color: #dddddd;  
  padding: 24px;  
}
```

プロパティとその値

| | |
|------------------|------------------------|
| font-size | 文字の大きさ |
| font-weight | 文字の太さ |
| color | 文字色指定 |
| background-color | 背景色 |
| width | 要素の幅を指定 |
| height | 要素の高さを指定 |
| margin | 要素の外側の余白 |
| padding | 要素の内側の余白 |
| border | 要素の枠線(線の色、線のスタイル、線の太さ) |

参考: <https://developer.mozilla.org/ja/docs/Web/CSS>

03_CSS
Chapter 02

スタイル指定

sample-css-1.html

```
<head>
  <meta charset="UTF-8"/>
  <link rel="stylesheet" href="sample-css1.css">
</head>
<body>
  <ul>
    <li class="red">RED</li>
    <li id="blue">BLUE</li>
  </ul>
</body>
```

linkタグ

headタグの中で、cssのファイルを読み込む。cssファイルの位置はhrefに記述する

sample-css-1.css

```
.red {
  color: red;
}
#blue {
  color: #0000ff;
}
```


sample-css-1.html

```
<head>
  <meta charset="UTF-8"/>
  <link rel="stylesheet" href="sample-css1.css">
</head>
<body>
  <ul>
    <li class="red">RED</li>
    <li id="blue">BLUE</li>
  </ul>
</body>
```

result

- RED
- BLUE

sample-css-1.css

```
.red {
  color: red;
}
```

```
#blue {
  color: #0000ff;
}
```

文字色はcolorで指定する
.red {} のスタイルは
class="red" の要素に適用される

#blue {} のスタイルは
id="blue" の要素に適用される

sample-css-2.html

```
<body>
  <ul>
    <li class="red">RED</li>
    <li id="blue">BLUE</li>
    <li class="green">GREEN</li>
    <li class="gray">GRAY</li>
  </ul>
</body>
```

result

- RED
- BLUE
- GREEN
- GRAY

sample-css-2.css

```
.red { color: red; }
#blue { color: #0000ff; }
.green { background-color: rgb(0, 255, 0); }
.gray { background-color: gray; }
```

背景色はbackground-colorで指定する
RGB値で色を指定することもできる
[カラーコード一覧表](#)

要素の幅と高さを変更する

sample-css-3.css

```
.top {  
  width: 100px;  
  height: 100px;  
  background-color: lightblue;  
}
```

幅が100pxで横が100pxの指定

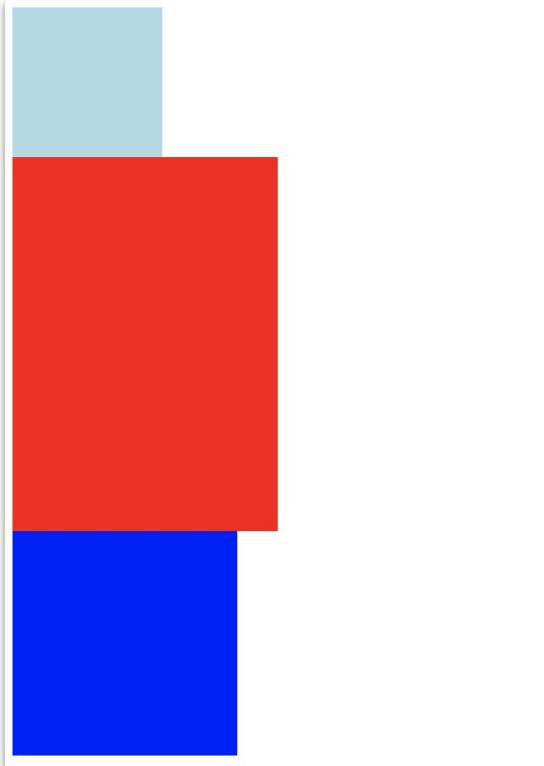
```
.middle {  
  width: 50%;  
  height: 250px;  
  background-color: red;  
}
```

幅は表示可能な領域の50%の長さに指定

```
.bottom {  
  width: calc(300px / 2);  
  height: 150px;  
  background-color: blue;  
}
```

幅は300pxを2で割った数

result





横並びのレイアウトを作るにはどうしたらいいか？

sample-css-4.html

```
<body>
  <ul>
    <li>あいうえお</li>
    <li>かきくけこ</li>
    <li>さしすせそ</li>
    <li>たちつてと</li>
  </ul>
</body>
```

result

| | | | |
|-------|-------|-------|-------|
| あいうえお | かきくけこ | さしすせそ | たちつてと |
|-------|-------|-------|-------|

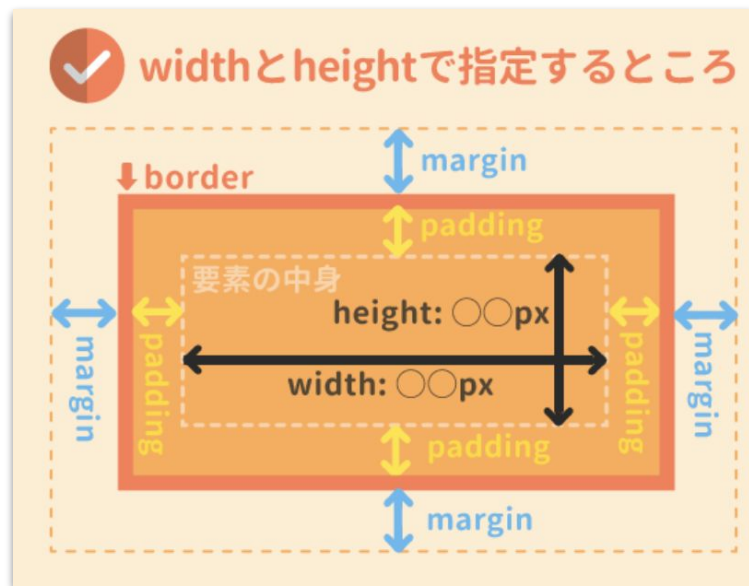
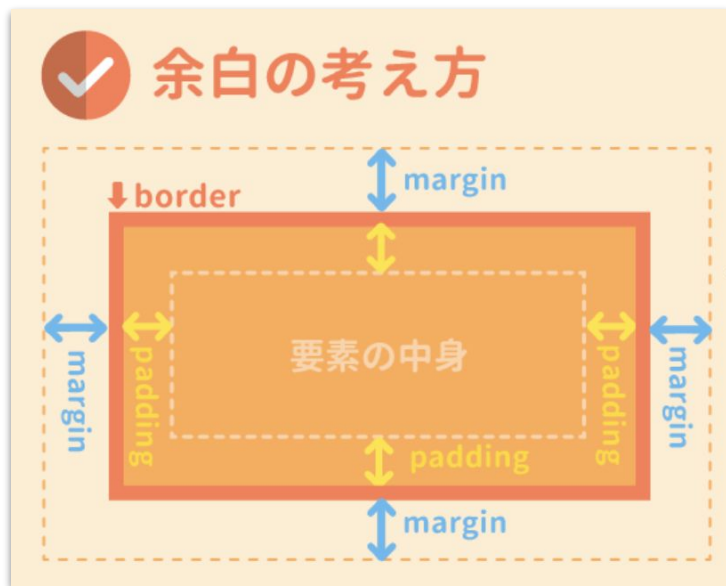
sample-css-4.css

```
li {
  border: solid 1px #888888;
  list-style: none;
}

ul {
  display: flex;
}
```

displayプロパティで要素の表示
形式をflexboxに変更

| | |
|---------|----------|
| margin | 要素の外側の余白 |
| padding | 要素の内側の余白 |



出典: <https://saruwakakun.com/html-css/basic/margin-padding>

04_ JavaScript
Chapter 01

JavaScriptとは

- Webページの動きを記述するプログラミング言語
- 要素を変更したり、通信処理を行うことができる
- ブラウザ上で動作する

トータルナビ 車 徒歩 自転車 トラック

📍 (株)ナビタイムジャパン 🔍

↑↓ (+) 経由地の追加

📍 東京タワー 🔍

出発時刻 ⬆️ 2021年03月24日 17時 ⬆️ 54分 ⬆️

🕒 現在時刻を設定

(+) 詳細条件の設定



トータルナビ 車 徒歩 自転車 トラック

📍 (株)ナビタイムジャパン 🔍

↑↓ (+) 経由地の追加

📍 東京タワー 🔍

🔍 東京タワーで検索

📍 東京タワー
東京都港区芝公園4-2-8

📍 中国料理 美林華飯店 六本木 東京タワー店
東京都港区麻布台3-4-10 2F

入力内容に合わせてスポットの候補を出す

04_ JavaScript
Chapter 02

JavaScriptの基本

変数は、**let** または **const** を用いて宣言する

- `const`, `let` が誕生する前は `var` のみで書かれていた

| | 特徴 |
|--------------------|----------------|
| <code>const</code> | 再代入不可 / スコープあり |
| <code>let</code> | 再代入可能 / スコープあり |
| <code>var</code> | 再代入可能 / スコープなし |

```
let text1 = '1回目';  
alert(text1);           // 1回目  
text1 = 'もう1回';     -> 代入可能  
  
const text2 = '1回目';  
alert(text2);           // 1回目  
text2 = 'もう1回';     -> エラー
```

| 型 | 意味 |
|-----------|-----------------|
| String | 文字列(' や " で囲む) |
| Number | 数値(整数・小数) |
| Boolean | 真偽値 true/false |
| undefined | 未定義状態 |
| null | null |

```
const a = 'Hello';    // String
const b = 12.34;      // Number
const c = true;       // Boolean

const d = undefined;  // undefined
const e = null;       // null
```

オブジェクトは、キーとバリューの関係で成り立ちます
変数.キー の形でバリューにアクセスできます

```
const company = {  
  name: 'NAVITIME',  
  since_year: 2000  
  is_remote: true  
};  
  
console.log(company.name); // NAVITIME  
console.log(company.id); // undefined
```

配列は、複数の値を1つのデータ(変数)として扱えることが特徴です

```
const array = [1, 2, 3];  
console.log(array[0]);    // 1
```

```
const mix = [true, 'a', 3.14];  
console.log(mix[1]);      // a
```

```
const list = [1, 2, 3, 4, 5, 6, 7, 8];  
console.log(list.length); // 8
```

```
const items = [  
  { id: 1, name: 'HTML' },  
  { id: 2, name: 'CSS' }  
];  
console.log(items[0].id); // 1  
items.forEach(function(item) {  
  console.log(item.name) // HTML  
})
```

配列のいろいろな扱い方 https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Global_Objects/Array

入力に対して出力を返す処理のまとめ

関数自体のメリット：同じ処理を一つにまとめることができる

```
function sum(a, b) {  
  return a + b;  
}  
  
console.log(sum(1, 2)); // 3
```

```
function get(params) {  
  return axios.get(path + params)  
}  
  
get(path, params);
```

ページ読み込み時や
クリックイベント時に使える

04_ JavaScript
Chapter 03

DOM操作

- HTML文書を階層構造で表現し、
- プログラムで操作できるようにした仕組み

HTMLとJavaScriptが接続されることで

文書構造やスタイルをJSから変更できるようになる

DOM操作

- 指定したIDの要素を取得する
- documentはJSを実行しているHTMLドキュメントを指す

```
document.getElementById('menu-list');           // id="menu-list"  
document.getElementsByClassName('container');    // class="container"  
document.getElementsByTagName('p');              // <p>
```

getElementsByClassName と getElementsByTagNameは配列のような形で複数返ってくるので注意

JavaScript/sample-1.html

```
<body>
  <main class="main">
    ~~~省略~~~
  </main>
  <script src="./static/sample-1.js"></script>
</body>
```

scriptタグをbodyタグの最後に記述

ボタンの見た目をJavaScriptから変更してみる

ふつうのボタン

成功ボタン

キャンセルボタン

見えないボタン

Sample-1

NAVITIME

JavaScript/static/sample-1.js

```
document.querySelector('.success').innerText = '文字が変わるボタン';  
document.getElementById('cancel').style.color = '#006400';  
document.getElementById('premium').style.display = 'none';
```

DOMのテキストを変更

文字色を変更

CSSを変更

要素を非表示

Result

ふつうのボタン

文字が変わるボタン

キャンセルボタン

ボタンが押されたら検索ワードをポップアップ表示する



The image shows a UI mockup within a light gray rectangular frame. On the left is a white rectangular input field with a thin blue border. To its right is a dark green rectangular button with rounded corners and the white Japanese text '検索' (Search).

JavaScript/sample-2.html JavaScript/static/sample-2.js

クリック時にsearch関数を実行

<input>の文字列を取得

ポップアップを表示

```
<main class="main">
  <form class="form">
    <input type="text" id="search-window"/>
    <button type="button" id="search">検索</button>
  </form>
</main>
-----
<script>
  document.getElementById('search').addEventListener('click', function(){
    const searchWord = document.getElementById('search-window').value;
    alert(searchWord);
  })
</script>
```

Result



05_jQuery
Chapter 01

jQuery

- DOM操作や通信処理などをシンプルな記述で行える
- JavaScriptのライブラリ



- jQuery自体はJSのコードを集めたライブラリ
- <head>タグ内で、他のJSファイルより先に読み込む必要がある

jQuery/sample-1.html

jQueryのライブラリを読み込む

```
<head>
  <meta charset="utf-8">
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <title>jQuery sample-1</title>
</head>
```

| | JavaScript | jQuery |
|-------|------------------------------------------------|-------------------------------------------|
| DOM取得 | <code>document.querySelector('#status')</code> | <code>\$('#status')</code> |
| DOM追加 | <code>document.createElement('div')</code> | <code>\$("<div>")</code> |
| CSS取得 | <code>element.style.border</code> | <code>element.css('border','none')</code> |
| 表示の制御 | <code>element.style.display='block'</code> | <code>element.show()</code> |
| 要素の編集 | <code>element.innerHTML='表参道'</code> | <code>element.text('表参道')</code> |

画像を押したら文字列を反転する

東京



名古屋

画像を押したら文字列を反転する

jQuery/static/sample-2.js

Result

ボタンクリック時

```
$("#switch-button").on('click', (function() {  
  const departure = $("#departure").val();  
  const arrival = $("#arrival").val();  
  $("#departure").val(arrival);  
  $("#arrival").val(departure);  
});
```

テキストを取得

テキスト入れ替え

A screenshot of a web interface. It features a central button with a double-headed arrow icon. To the left of the button is a text input field containing the text '名古屋'. To the right of the button is another text input field containing the text '東京'.

入力した文字列をリストの要素に足す

クチコミ投稿

追加

入力した文字列をリストの要素に足す

jQuery/static/sample-3.js

```
$('#add-button').on('click', (function() {  
    const inputMessage = $('#input-message').val();  
    const li = $('<li>').text(inputMessage)  
    $('#comment-list').append(li);  
}));
```

タグを追加

Result

クチコミ投稿

余裕で座れる

普通に立てる

普通に立てる

圧迫される

席はいっぱい

Mac

// を出すコマンド

command + /

/* */ を出すコマンド

shift + option + A

Windows

// を出すコマンド

Ctrl + /

/* */ を出すコマンド

Shift + Alt + A

| | Mac | Windows |
|---------|--------------------|-----------------|
| 1行コメント | Command + / | Ctrl + / |
| 複数行コメント | Shift + Option + A | Shift + Alt + A |
| JSDoc | /** + Enter | /** + Enter |
| ファイル内検索 | Command + F | Ctrl + F |
| ファイル保存 | Command + S | Ctrl + S |

| | Mac | Windows |
|---------------|--------------------------|---------------------------------------------|
| コピー | command + C | Ctrl + C |
| ペースト | command + V | Ctrl + V |
| 文字選択 | shift + → (矢印) | shift + → (矢印) |
| 文末へ カーソル移動 | command + → (矢印) | Ctrl + → (矢印) ※windowsは単語ごとの選択になる |
| 1行選択 | command + shift + → (矢印) | Ctrl+ shift + → (矢印) ※windowsは単語ごとの選択になる |

06_デバッグ
Chapter 01

デバッグ

- バグの原因を見つけて直すこと
 - プログラムの不具合・誤り

- うまく動いてくれない

- 変数の中身、エラー内容を確認できる

- APIのリクエスト内容が見たい

- リクエストの状態、レスポンスの中身を確認できる

- 画面のレイアウトが崩れてしまう

- 効いているCSSの値が見れる

debug.html を**ブラウザ**で開いてください

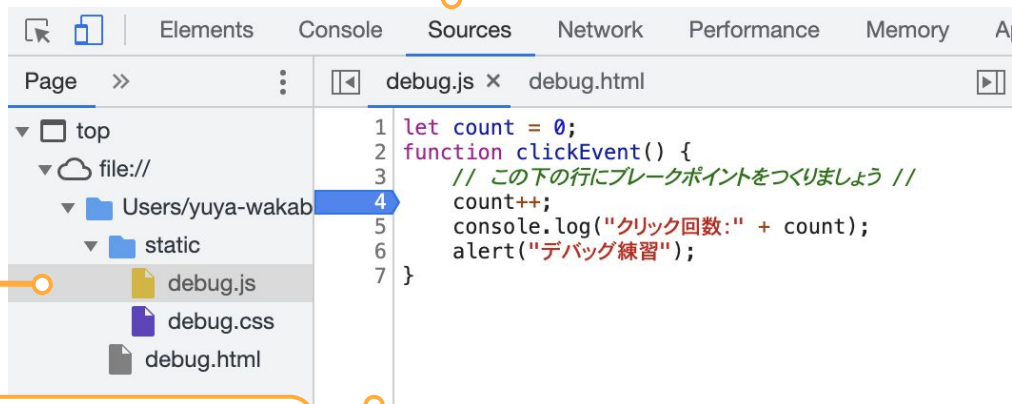
① ブラウザで右クリック

② 検証 を選択

③ Sources を選択

④ファイル を選択

⑤処理を止めたい
行を選択



次のブレークポイント
まで進める

処理を一つ進める

console.log()の
値が表示



レイアウトの確認

1. 画面上のどこかで右クリック

2. 検証を選択

3. `li.debug-box` 106×54
Background ■ #F08080

4. ACCESSIBILITY
Name
Role listitem

5. Keyboard-focusable ☐ リック

6.

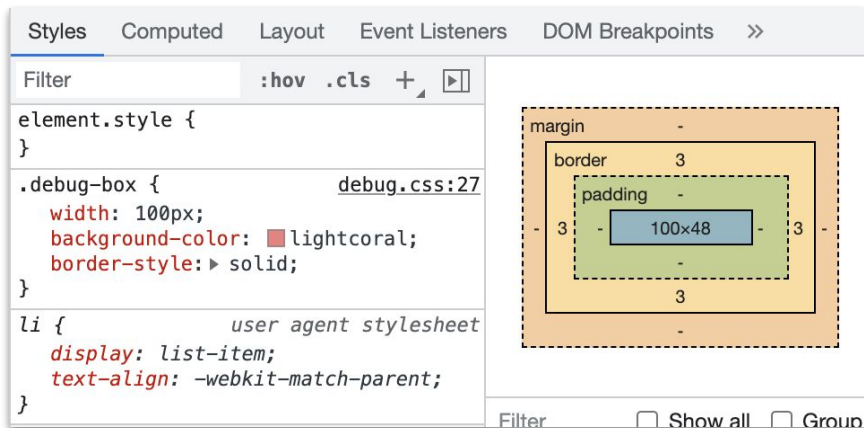
7. `.debug-box` の `{}` の中にcssの記載をしてみよう

③クリックした要素の
CSSがわかる

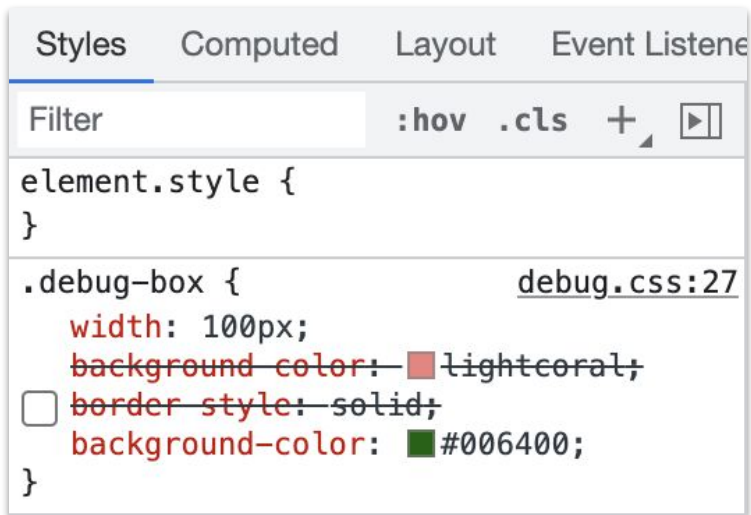
①矢印マークをクリック



②クリックした要素の
範囲がわかる




debug.html



Result

レイアウトの確認

1. 画面上のどこかで右クリック
2. 検証を選択
3. 左上の矢印をクリック
4. その状態で各htmlタグ要素の上にカーソルを合わせてみよう
5. その状態で下の箱をクリック
6. 

デベロッパーツールで編集した値は、リロードするとリセットされます

- 書いたはずの処理が実行されない
 - ファイルは保存しましたか？
 - ブラウザは再読み込みしましたか？
 - 更新したものとは違うHTMLファイルを開いていませんか？
- ブレークポイントで止まらない
 - 置いた箇所を通る操作になっていますか？

07_ 演習問題
Chapter 01

演習問題

要件

見出し・段落・リストの要素を使って自己紹介ページを作りなさい。
※CSSは不要です

見本

自己紹介



名前

Mr. ナビタイム

性別

男

出身

東京都

趣味

- ナビゲーション
- プログラミング
- 移動する

要件

index.cssをindex.htmlで読み込みなさい。また、CSSをあてて見本と同じような見た目にしなさい。

薄い緑: #ddeadd
濃い緑: #006400
文字色: #4b4b4b

見本

自己紹介

名 前 Mr. ナビタイム
性 別 男
出 身 東京都
趣 味

- ナビゲーション
- プログラミング
- 移動する

要件

index.jsを、index.htmlで読み込みなさい。

画像をクリックしたら背景色が白色になるようにjQueryで実装なさい。

見本

自己紹介



名前 Mr. ナビタイム
性別 男
出身 東京都
趣味

- ナビゲーション
- プログラミング
- 移動する

本日のまとめです



Webページの構成

- HTML
 - ページの構造を作成
- CSS
 - ページの見た目を整える
- JavaScript / jQuery
 - ページの動きを制御

Webアプリケーションの開発ではデバッグを多用するので使いこなせるようになりましょう！

**本日の講義は終了です
お疲れ様でした！**