

第4回

JavaScriptから始める プログラミング

京都大学工学部情報学科

計算機科学コース3回

KMC2回 drafear

自己紹介

- id
 - drafear(どらふいあ, どらふあー)
- 所属
 - 京都大学 工学部 情報学科 計算機科学コース 3回
- 趣味
 - ゲーム(特にパズルゲー), ボドゲ, ボカロ, twitter
- 参加プロジェクト ※青: 新入生プロジェクト
 - **これ**, **競プロ**, ctf, 終焉のC++, coq, 組み合わせ最適化読書会



@drafear



@drafear_ku



@drafear_carryok



@drafear_evolve



@drafear_sl



@gekimon_1d1a



@cuigames

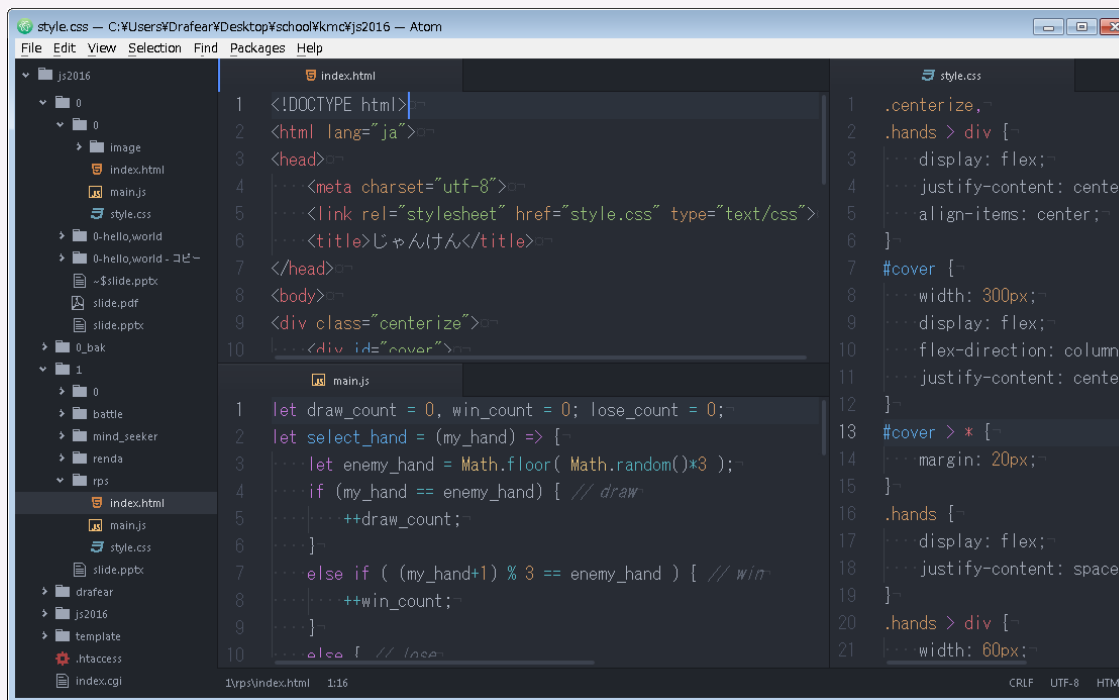
自己紹介

1. KMC 2回生
2. 新入生
3. KMC 3回生
4. KMC 4回生
5. KMC n回生

- 学部学科
- id (入部してたら) or 名前
- 趣味・宣伝

この講座で使用するブラウザとエディタ

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



今日の目標

- 復習する

本日の内容

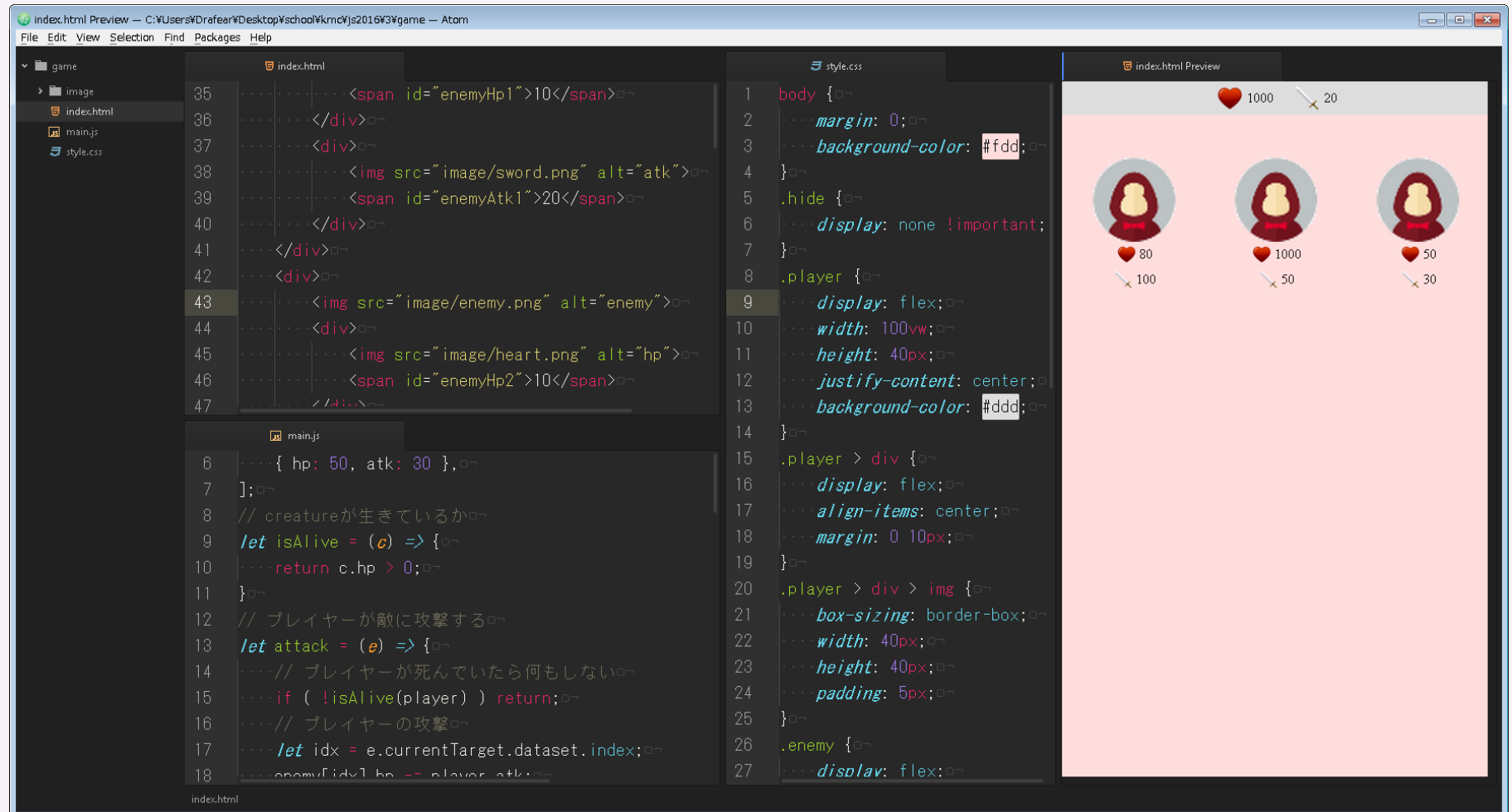
- atomの便利なプラグイン紹介！！
- 復習(座学)
- 簡単なゲーム・ツールを作る



1. atomの便利なプラグイン

atom-html-preview

- Ctrl + Shift + H (default)



run-in-browser

- HTMLファイル編集集中に Ctrl + Alt + R を押すとブラウザで開く

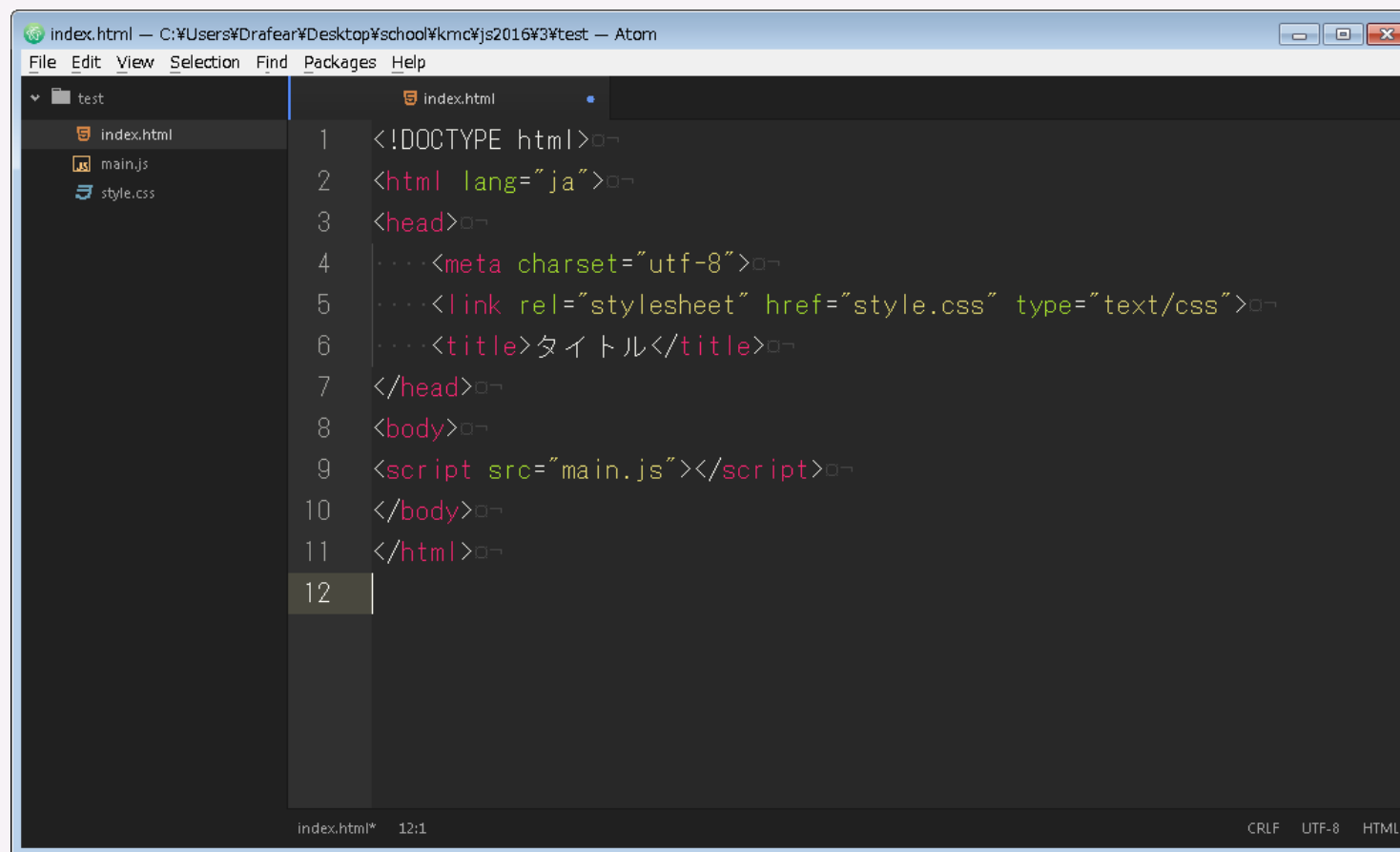
autocomplete-paths

- ファイルやディレクトリ名を補完！最高！



file-icons

- ファイル名にアイコンが付く！
- 分かりやすいしやる気に繋がる！！



The screenshot shows the Atom text editor interface. On the left, a file explorer pane displays a folder named 'test' containing three files: 'index.html' (with a document icon), 'main.js' (with a JavaScript icon), and 'style.css' (with a CSS icon). The main editor area shows the content of 'index.html', which is an HTML document with the following code:

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4   <meta charset="utf-8">
5   <link rel="stylesheet" href="style.css" type="text/css">
6   <title>タイトル</title>
7 </head>
8 <body>
9   <script src="main.js"></script>
10 </body>
11 </html>
12
```

The status bar at the bottom indicates the current file is 'index.html*' at line 12, column 1, with encoding set to UTF-8 and line endings set to CRLF.

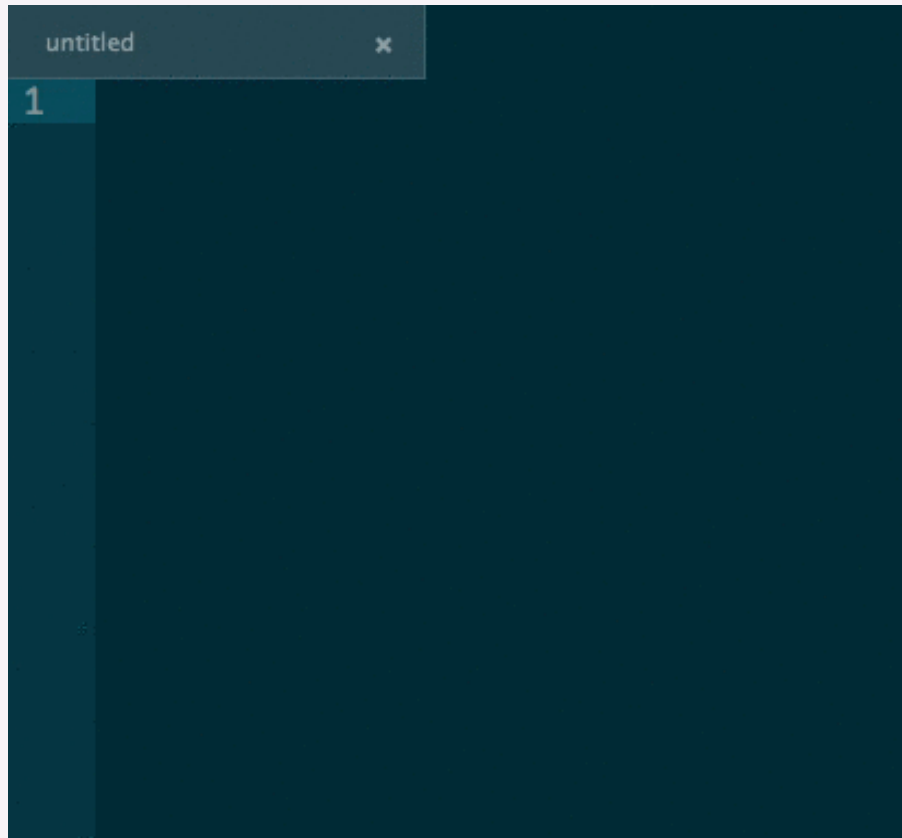
highlight-selected

- 同じ単語をハイライト！
- ミスが減る！！

```
9  initialize: (editorView) ->
10      @views = []
11      @editorView = editorView
12
13  attach: =>
14      @editorView.underlayer.append(this)
15      @editorView.on "dblclick", @handleDbClick
16      @editorView.on "click", @removeMarkers
17
```

less-than-slash

- 最初に入れてもらったプラグイン
- " </ " と入力すると閉じタグが自動的に入る！



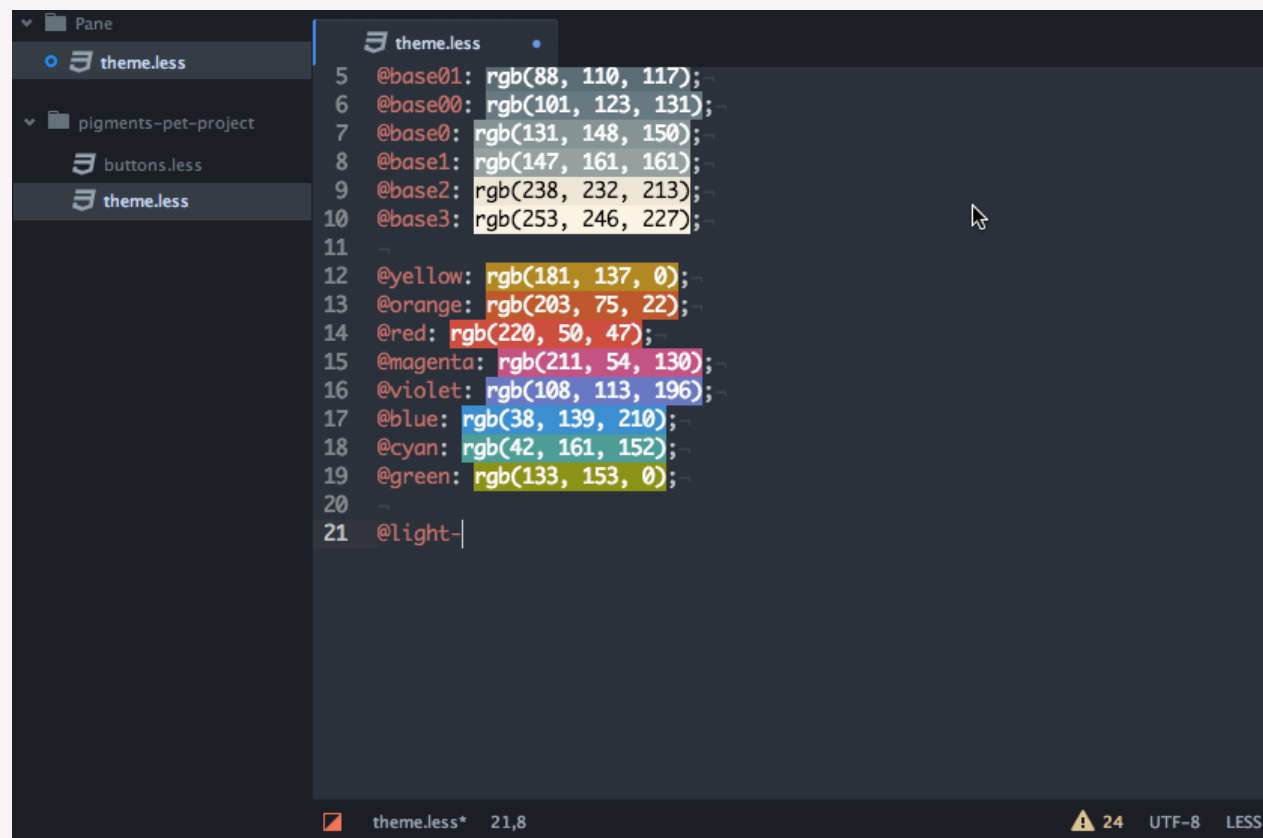
color-picker

- RGBの値を見て, 正確に色を想像できますか？

```
4  
5  .Content {  
6     background: $color-blue;  
7     color: #39b813;  
8     margin: auto;  
9     position: relative; }  
10
```

pigments

- rgb(...) に色が付く
- 分かりやすい！



```
5 @base01: rgb(88, 110, 117);
6 @base00: rgb(101, 123, 131);
7 @base0:  rgb(131, 148, 150);
8 @base1:  rgb(147, 161, 161);
9 @base2:  rgb(238, 232, 213);
10 @base3:  rgb(253, 246, 227);
11
12 @yellow:  rgb(181, 137, 0);
13 @orange:  rgb(203, 75, 22);
14 @red:     rgb(220, 50, 47);
15 @magenta: rgb(211, 54, 130);
16 @violet:  rgb(108, 113, 196);
17 @blue:    rgb(38, 139, 210);
18 @cyan:    rgb(42, 161, 152);
19 @green:   rgb(133, 153, 0);
20
21 @light-
```



2. 復習 (座学)

全般

- 第1回

- 拡張子 : hoge.fuga.png
- 絶対パス : http://google.com/
- 相対パス : image/hoge.png

JavaScript

- 第1回

- alert(val)
- console.log(val)
- エラー : コンソールに出力される
- 変数 : `let a = 10;`
- 代入 : `a = 5;`
- 四則演算 : `(a * 5 + 8 / 2 - 1) % 10`
- 糖衣構文 : `a = a + 5 ⇔ a += 5`
- コメント : `// 一行コメント, /* 複数行コメント */`
- 関数 : `(param1, param2) => { ... }`
- 文字列 : `"to" + "kyoto"`
- オブジェクト : `{ a: 10, b: 5 }.a`
- テンプレート文字列 : ``${a} + ${b} = ${a + b}``

JavaScript

- 第2回

- `Math.max(a, b)`, `Math.min(a, b)`
- `Math.floor(x)`, `Math.ceil(x)`, `Math.round(x)`
- `Math.pow(x, n)`
- `Math.sqrt(x)`
- `Math.sin(x)`, `Math.cos(x)`, `Math.tan(x)`
- `Math.asin(x)`, `Math.acos(x)`, `Math.atan(x)`
- `Math.atan2(y, x)`
- `Math.abs(x)`
- `Math.PI`
- `Math.random()`
- `a > b`, `a < b`, `a >= b`, `a <= b`, `a == b`, `a != b`

JavaScript

- 第2回 (+ α)
 - `Math.max(a, b)`, `Math.min(a, b)` : aとbの小さい方/大きい方
 - `Math.floor(x)`, `Math.ceil(x)`, `Math.round(x)` : 切り捨てなど
 - `Math.pow(x, n)` : x^n
 - `Math.sqrt(x)` : \sqrt{x}
 - `Math.sin(x)`, `Math.cos(x)`, `Math.tan(x)`
 - `Math.asin(x)`, `Math.acos(x)`, `Math.atan(x)`
: $\sin^{-1} x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ $\cos^{-1} x \in [0, \pi]$ $\tan^{-1} x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$
 - `Math.atan2(y, x)` : $\tan^{-1} \frac{y}{x} \in [-\pi, \pi]$
 - `Math.abs(x)` : $|x|$
 - `Math.PI` : π
 - `Math.random()` : $[0, 1)$ の乱数

JavaScript

- 第2回

- true, false
- $a > b$, $a < b$
- $a \geq b$, $a \leq b$
- $a == b$, $a != b$
- $!a$: NOT a
- $a \&\& b$: a AND b
- $a || b$: a OR b
- if (条件1) { 処理1 } else if (条件2) { 処理2 } else { 処理3 }
- return : そこで中断

JavaScript

- 第3回

- 変数のスコープ : 変数の見える範囲
- 再帰 : 関数の中で自分自身を呼ぶ
- インクリメント : `++i, i++`
- デクリメント : `--i, i--`
- 無限ループ : `while (1) { 処理 }, for (;;) { 処理 }`
- while文 : `while (継続条件) { 処理 }`
- for文 : `for (let i = 0; i < n; ++i) { 処理 }`
- continue : 次の繰り返しに移る
- break : 繰り返し文を抜ける
- 配列 : `a = [114, 514, 1919, 810]; b = a[3]; // 1919`
- 配列とfor文 : `for (let i = 0; i < a.length; ++i) { 処理 }`

HTML

- 第1回

- `<h1> ~ </h1>` : 見出し
- 開始タグ : `<tagName>`
- 終了タグ(閉じタグ) : `</tagName>`
- HTML要素 : 開始タグ~終了タグ で1つの要素
- `<タグ名 属性1="属性値1" 属性2="属性値2">`
- `` : src属性で指定した画像を表示
閉じタグ不要, alt属性(代替テキスト)必須
- `<div>, ` : 汎用コンテナ

HTML

- 第3回
 - `<p>` : 段落
 - `` : 順序なしリスト
 - `` : ulの要素

CSS

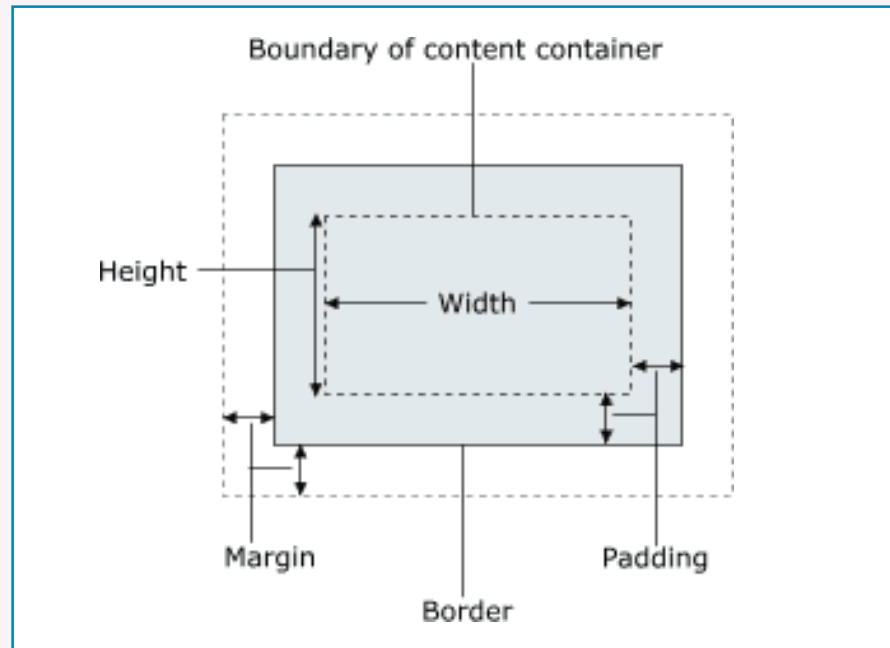
- 第2回

- class属性 : 複数クラス指定可
- `.className { property1: parameter1; ... }`
- 文字色 : `color: red;`
- 文字サイズ : `font-size: 40px;`
- 太字 : `font-weight: bold;`
- 下線 : `text-decoration: underline;`
- 要素非表示 : `display: none;`
- カーソルの形 : `cursor: pointer;`
- マウスオーバー : `.className:hover { ... }`
- notフィルター : `.className1:not(.className2) { ... }`

CSS

- 第2回

- ボックス : width, height, margin, padding, border
- 背景色 : background-color: rgb(114, 51, 4);
- 背景画像 : background-image: url(*imagePath*);

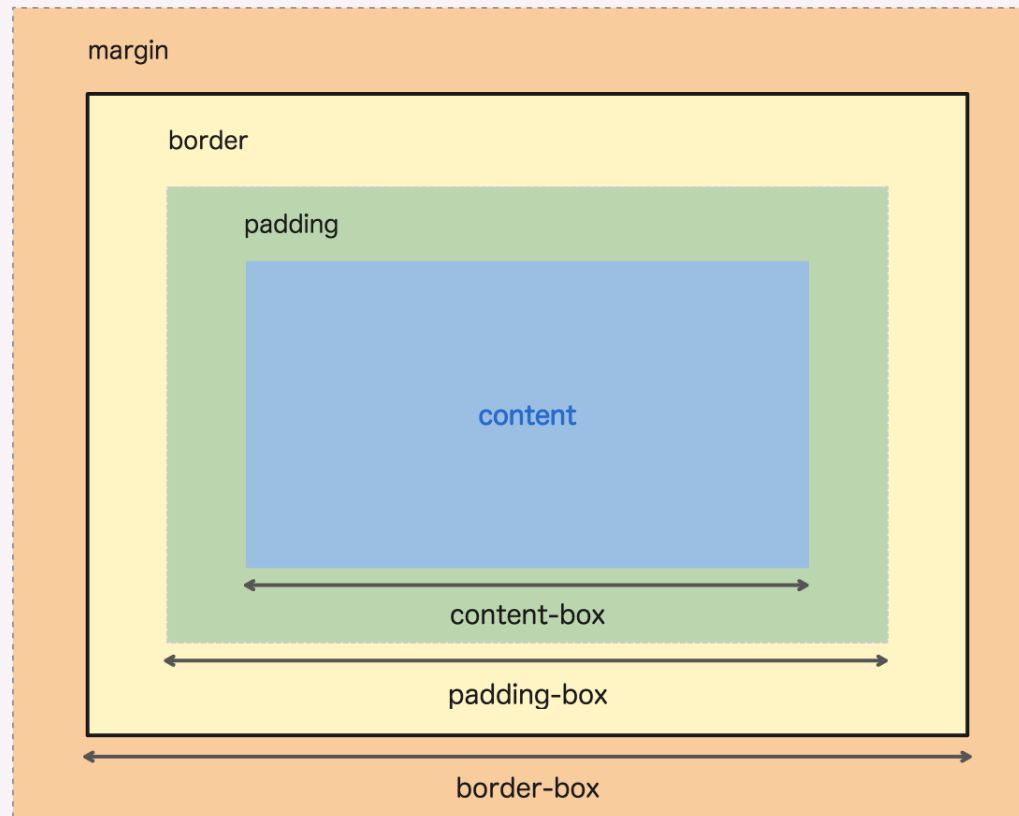


<https://www.addedbytes.com/articles/for-beginners/the-box-model-for-beginners/>

CSS

- 第2回

- width等の計算方法 : `box-sizing: border-box;`



CSS

- 第2回

- 色の指定 : red, #abcdef, #fff, rgb(255,255,255)
- 半透明な色 : rgba(255,255,255,1.0)
- 縁取り非表示 : outline: none;
- アニメーション : transition: all .3s;

CSS

- 第3回

- セレクタ

#id

.className

tagName

selector1 selector2 : 子孫

selector1 > selector2 : 子

selector1 + selector2 : 直後の兄弟

selector1 ~ selector2 : それ以降の兄弟

selector.className : *selector* かつ *className*

selector#id : *selector* かつ *id*

CSS

- 第3回

- ブロックボックス : □, 直後に改行
`display: block;`
- インラインボックス : 囲むだけ
`display: inline;`



<https://nulab-inc.com/ja/blog/nulab/css-basics-for-engineer-boxmodel/>



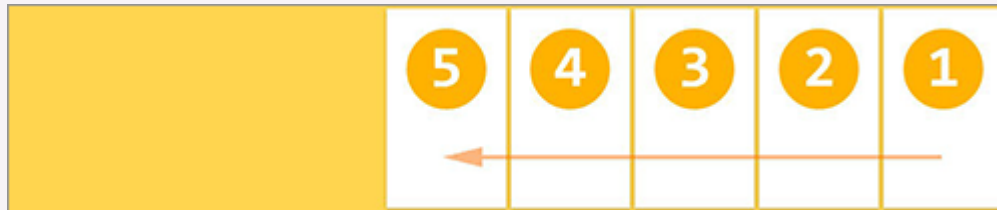
<https://nulab-inc.com/ja/blog/nulab/css-basics-for-engineer-boxmodel/>

CSS

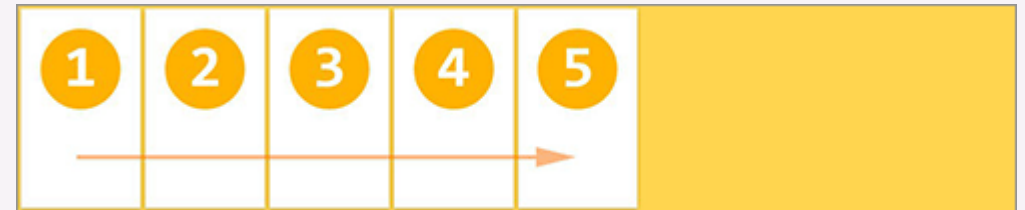
- 第3回
 - flexbox : 便利
display: flex;

flexbox

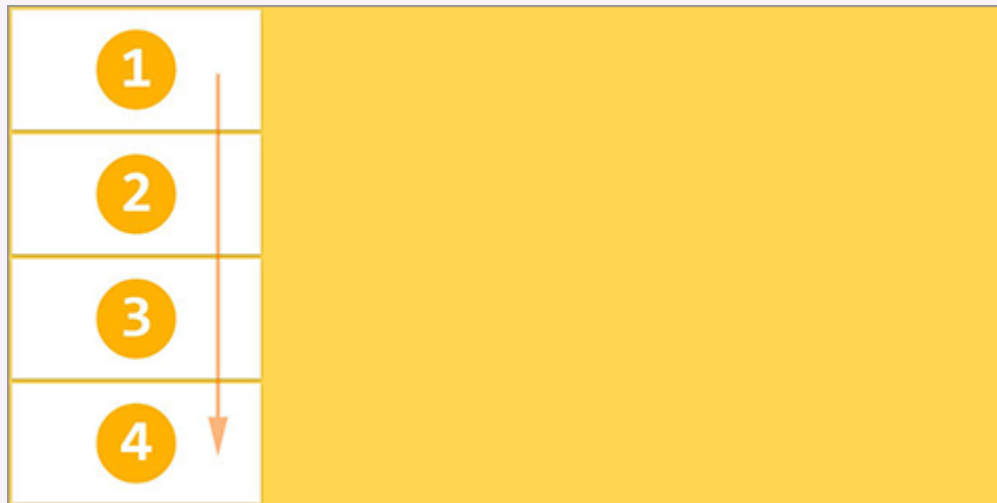
- flex-direction 向き



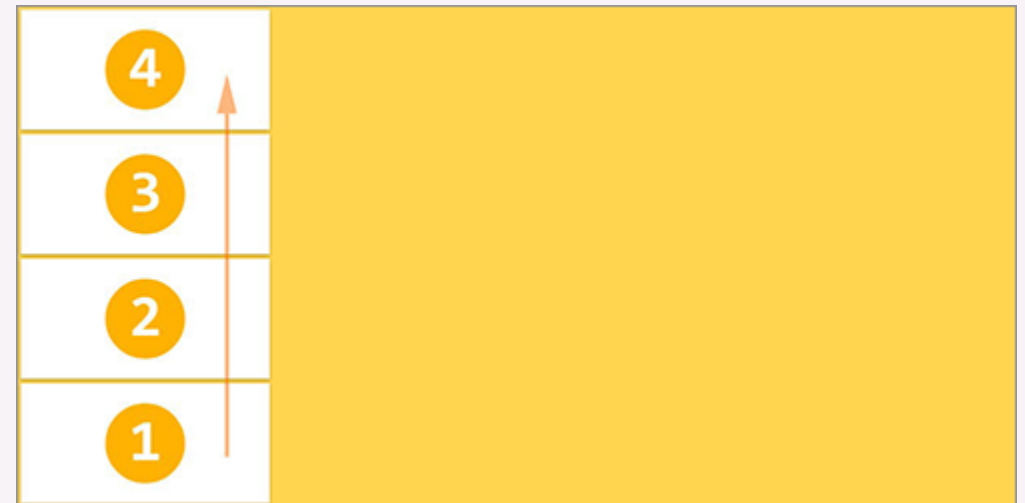
row(default)



row-reverse



column



column-reverse

flexbox

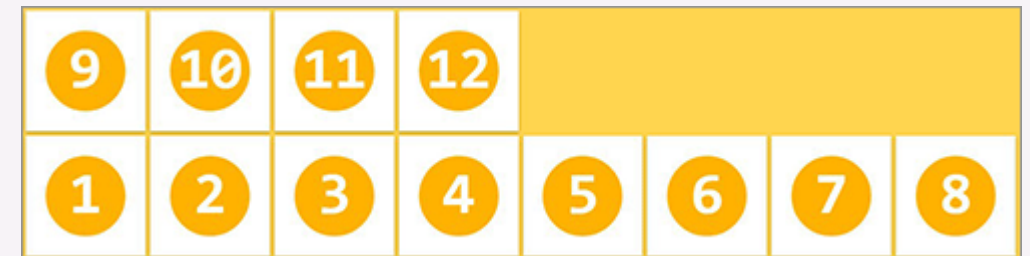
- flex-wrap 折り返し



nowrap(default)



wrap



wrap-reverse

flexbox

- justify-content 主軸方向の余白の使い方



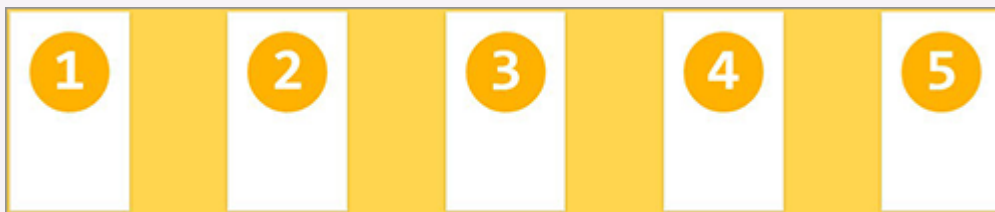
flex-start(default)



flex-end



center



space-between



space-around

flexbox

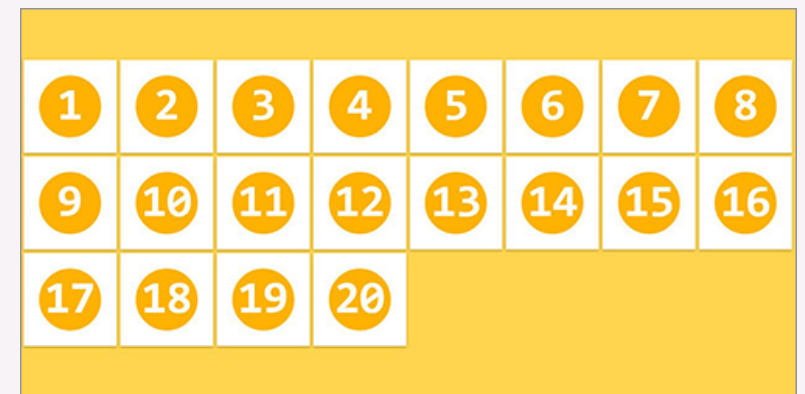
- align-content クロス軸方向全体としての余白の使い方



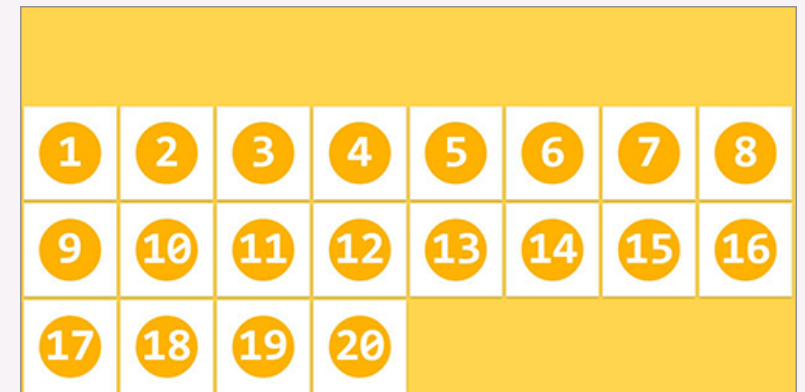
stretch(default)



flex-start



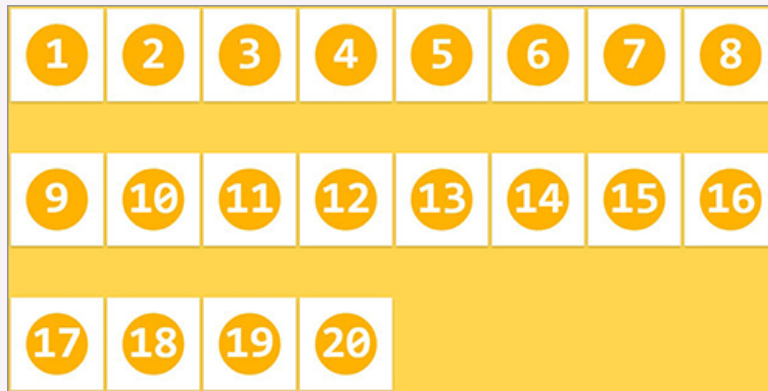
center



flex-end

flexbox

- align-content クロス軸方向全体としての余白の使い方



space-between



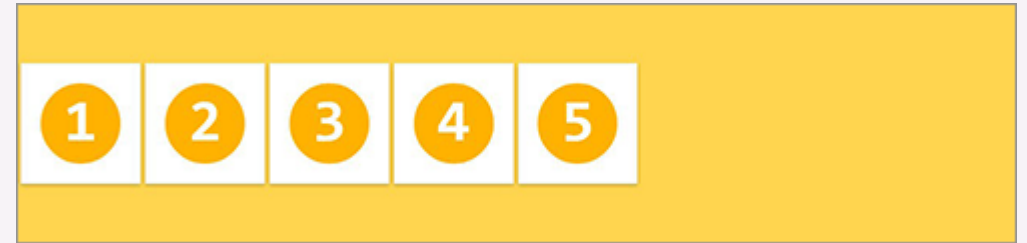
space-around

flexbox

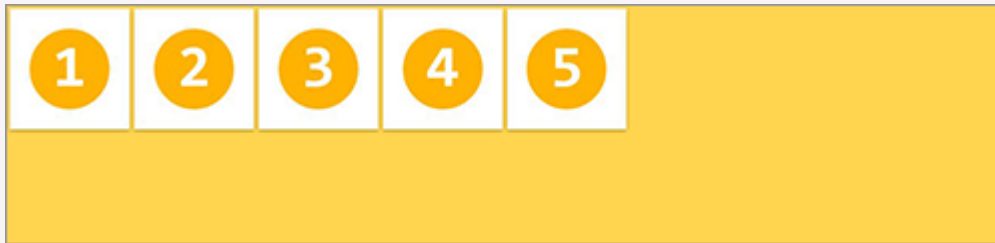
- align-items 各行でのクロス軸方向の余白の使い方



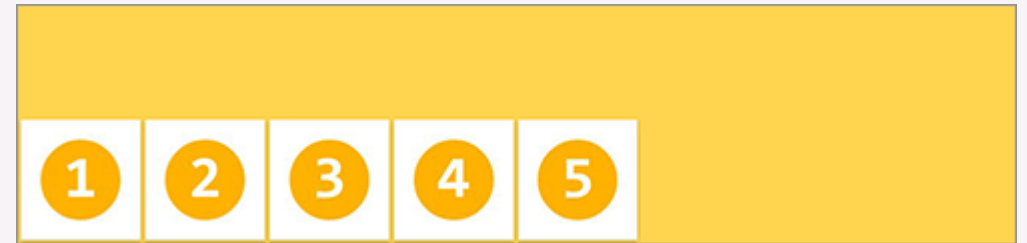
stretch(default)



center



flex-start



flex-end



baseline

CSS

- 第3回

- body : デフォルトでpaddingを持つ
- ブラウザ表示領域に対する単位
 - width: 100vw;
 - height: 100vh;
- marginの指定方法
 - margin: [上下左右];
 - margin: [上下] [左右];
 - margin: [上] [右] [下] [左];
 - padding等も同様

DOM操作

- `document.getElementById(id)`
 - *id*属性が*id*の要素を得る
- `document.getElementsByTagName(tagName)`
 - タグ名が*tagName*の要素を配列で得る
- `document.getElementsByClassName(className)`
 - *className*クラスが設定されている要素を配列で得る
- `elem.querySelector(selector)`
 - *selector*にマッチする最初の要素を得る
- `elem.querySelectorAll(selector)`
 - *selector*にマッチする全ての要素を配列で得る

DOM操作

- `elem.addEventListener(eventName, function)`
 - `elem`で`eventName`イベントが発生したら`function`を実行する
 - イベント監視
 - イベント例) "click", "mousedown", "mouseup", "keydown", "keyup", "keypress", "mousemove", "mouseover", "mouseout", "dragstart", "dblclick"

DOM操作

- `elem.classList.add(className)`
 - `elem` に `className` クラスを追加する
- `elem.classList.remove(className)`
 - `elem` の `className` クラスを削除する
- `elem.classList.toggle(className)`
 - `elem` の `className` クラスを反転(なければ追加, あれば削除)する
- `elem.classList.contains(className)`
 - `elem` が `className` クラスを持っているか(true/false)
- `elem.className = "";`
 - `elem` のクラスをクリアする

DOM操作

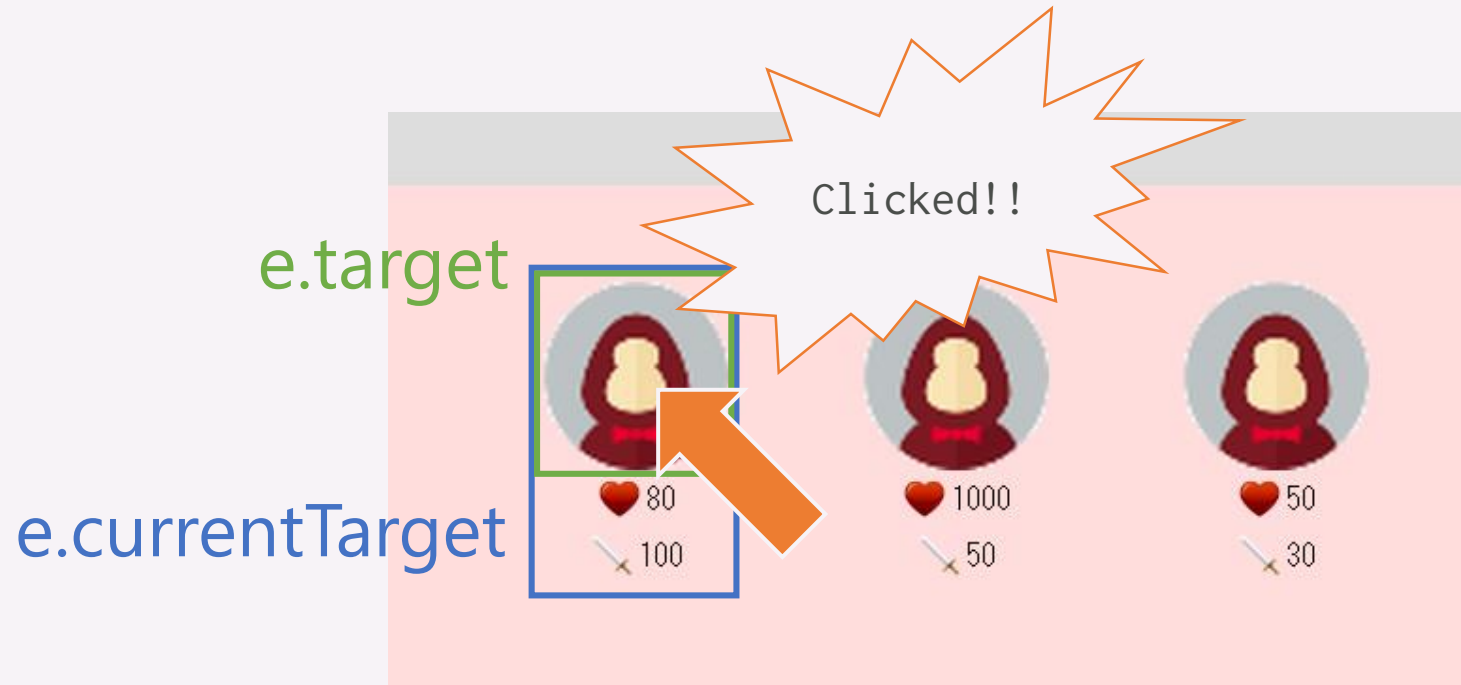
- `elem.children`
 - 子要素(配列)
- `elem.parentElement`
 - 親要素

DOM操作

- `elem.dataset.dataName = value`
 - 要素にデータを設定できる(文字列のみ)
- `elem.dataset.dataName`
 - 要素のデータを参照できる

DOM操作

- `e.currentTarget`
 - そのイベントが設定された要素
- `e.target`
 - そのイベントが起きた一番深い要素





3. 簡単なゲーム・ツールを作る

めんせきくん

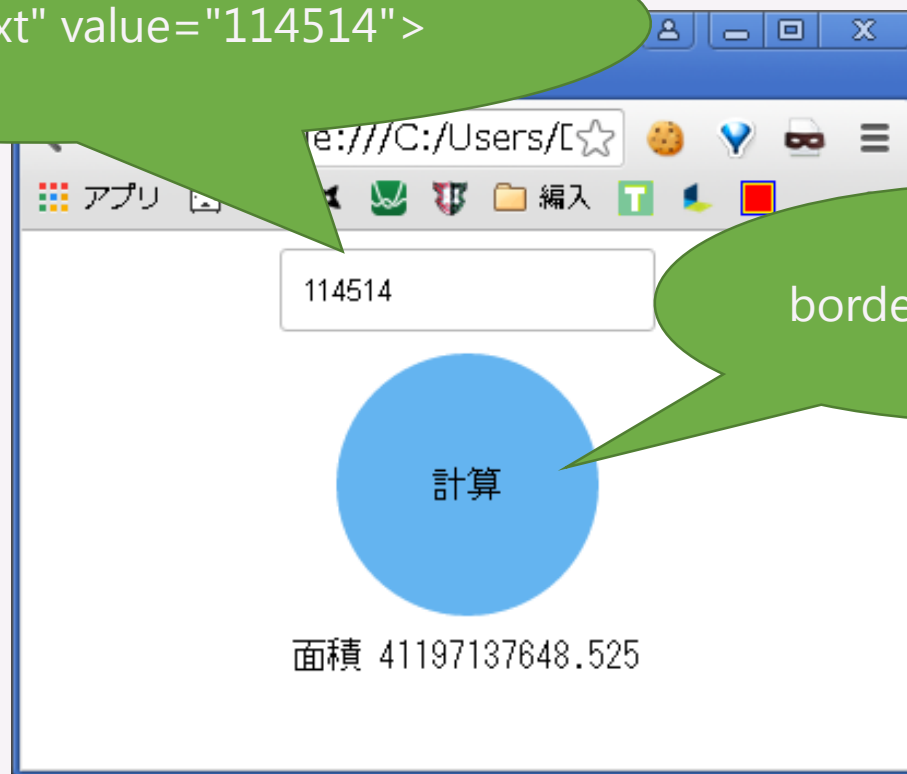
- 計算ボタンを押すと円の面積を計算してくれるめんせきくん



めんせきくん

- 計算ボタンを押すと円の面積を計算してくれるめんせきくん

`<input type="text" value="114514">`



`border-radius: 50px;`

簡単なゲーム・ツールを作る

- ゲーム
 - 宝探しゲーム
 - Hit and Brow
 - High or Low
 - ○×クイズ
- ツール
 - 計算ツール
 - カウンタ(+1, -1), ポケモンの努力値カウンタ
 - 西暦 ⇔ 平成 変換器