第２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(どらふぃあ, どらふぁー)
- 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

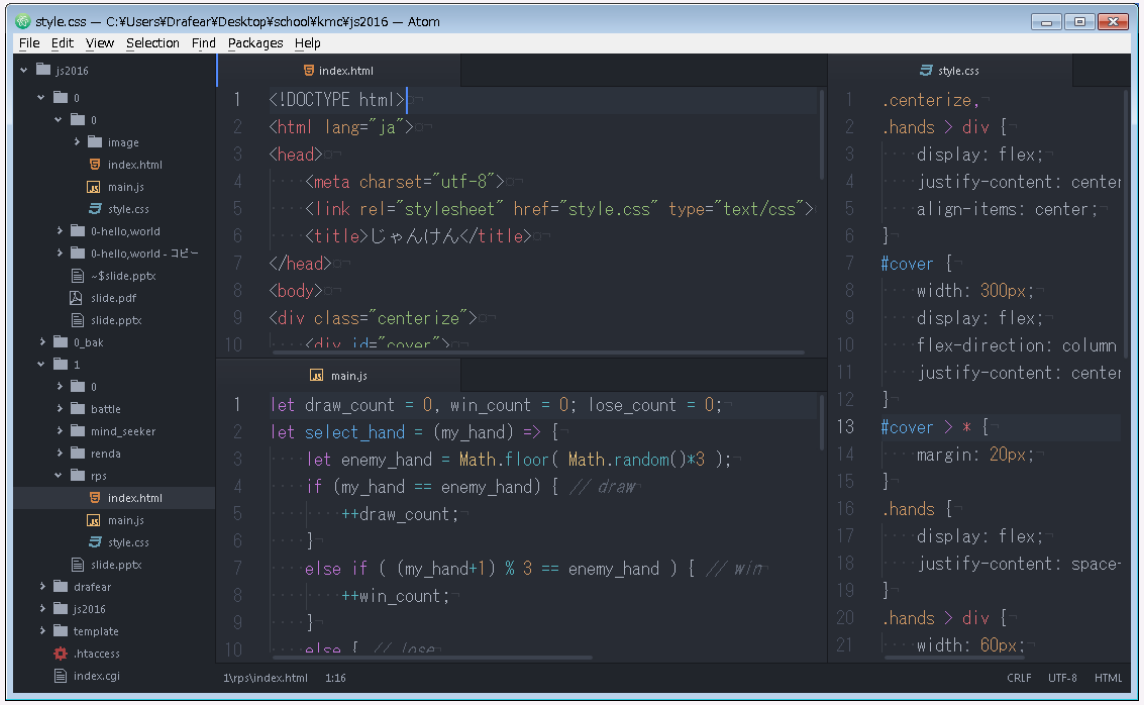
自己紹介

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
- 京都で好きな飲食店(あれば)

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

- HTML, CSS の違いを理解する
- CSSをいじってデザインしてみる
- じゃんけんを作る

本日の内容

- HTML
  - div要素 と span要素 の違い
- CSS
  - テキストの装飾
  - ボックス
  - ホバー, アニメーション
- JavaScript
  - 数学関数(特に乱数)
  - 論理値
  - if ~ else 文
  - return の挙動

てんぷれ

- 以下から雛形をダウンロードしてください
  - https://github.com/kmc-jp/js2016

注意

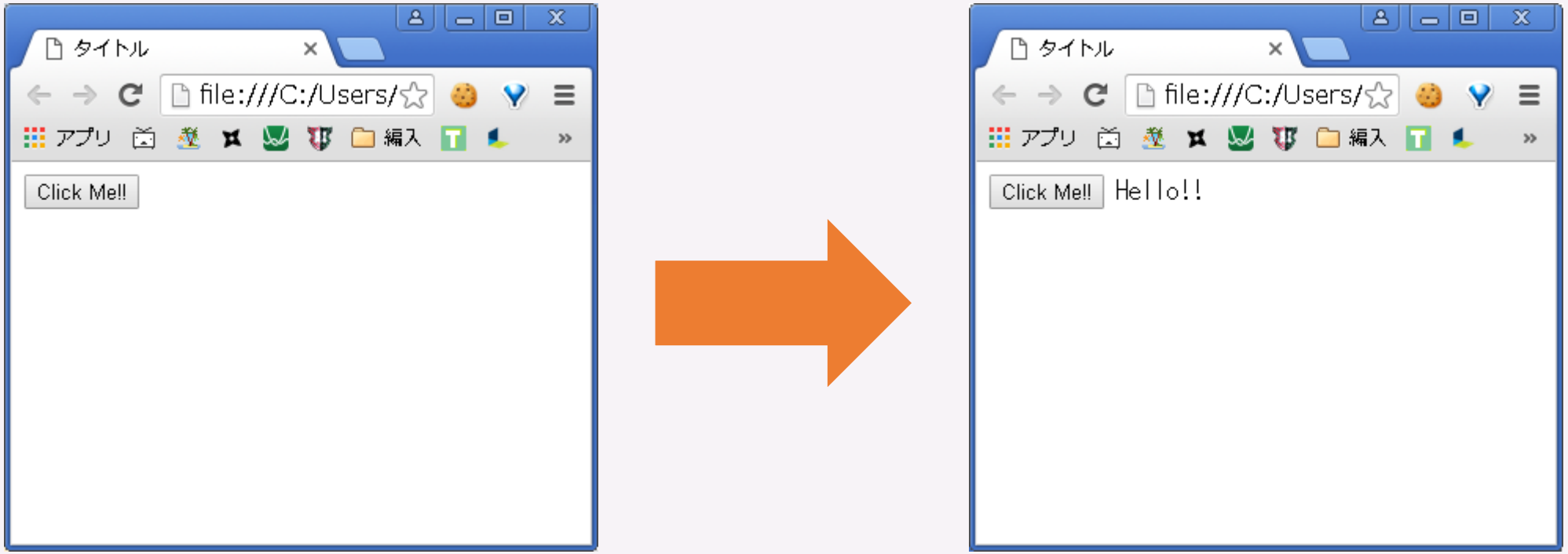
- 新しいことがたくさん出てきますが
  全部覚える必要はありません
- こんなのあったなーくらいでおｋです
- 必要になれば調べれば良いので
- 重要な項目は、やってるうちに覚えます

復習

- 第１回のスライドを見て復習していきます

復習問題

- ボタンをクリックすると Hello! と表示する
  HTML, JavaScriptを書いて下さい



復習問題

- ボタンをクリックすると Hello! と表示するプログラムを書いて下さい

index.html

```
<input type="button" id="btn" value="Click Me!!">
<span id="text"></span>
```

main.js

```
document.getElementById("btn").addEventListener("click", (e) => {
    document.getElementById("text").innerText = "Hello!!";
});
```

Mathオブジェクト (数学関数)

- Math.max(a, b)
  - a と b の大きい方を返します
- Math.min(a, b)
  - a と b の小さい方を返します

main.js

```
console.log( Math.max(2, 10) ); // 10
console.log( Math.min(2, 10) ); // 2
```

Mathオブジェクト (数学関数)

- Math.floor(x)
  - xを切り捨てた値を返します
- Math.ceil(x)
  - xを切り上げた値を返します
- Math.round(x)
  - xを四捨五入した値を返します

main.js

```
console.log( Math.floor(3.5) ); // 3
console.log( Math.ceil(3.5) ); // 4
console.log( Math.round(3.5) ); // 4
console.log( Math.round(3.49) ); // 3
console.log( Math.floor(-3.5) ); // -4
```

Mathオブジェクト (数学関数)

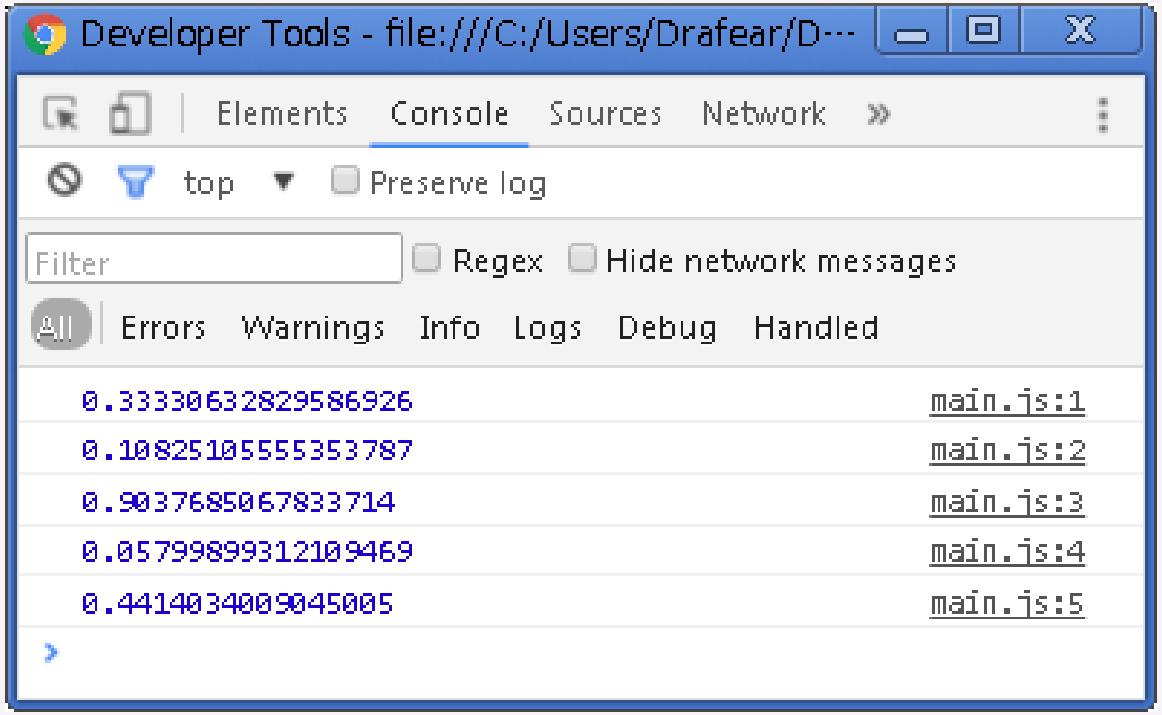
- Math.pow(x, y)
  - x^y を返します
- Math.sqrt(x)
  - $\sqrt{x}$ を返します
- Math.sin(x)
  - sin(x) を返します
  - xはラジアン単位です
- Math.abs(x)
  - |x| を返します
- Math.PI
  - π = 3.14159... (定数)

Mathオブジェクト (数学関数)

- Math.random()
  - 0以上1未満の乱数(ランダムな数)を返します

main.js

```
console.log( Math.random() );
console.log( Math.random() );
console.log( Math.random() );
console.log( Math.random() );
console.log( Math.random() );
```



整数値の乱数

- 0 から n-1 までの乱数を生成します

main.js

```
let randN = (n) => {
    return Math.floor( Math.random()*n );
}
```

```
> rand_n(10)
< 0
> rand_n(10)
< 4
> rand_n(10)
< 7
```

演習

1. Math.round を使わずに四捨五入する関数 round を作って下さい
   - let round = (x) => { ... }
   - $x \geq 0$ と仮定して良い

2. 1～6の乱数を生成する関数 diceRoll を作って下さい
   - let diceRoll = () => { ... }

3. 三角形の3辺の長さから面積を計算する関数 S を作って下さい
   - let S = (a, b, c) => { ... }

演習

1. Math.round を使わずに四捨五入する関数 round を作って下さい
   - 0.5 足して切り捨て

main.js

```
let round = (x) => {
    return Math.floor(x + 0.5);
}
```

演習

2. 1～6の乱数を生成する関数 diceRoll を作って下さい
   - randN(6) + 1

main.js

```
let diceRoll = () => {
//  return randN(6) + 1;
    return Math.floor( Math.random()*6 ) + 1;
}
```

演習

3. 三角形の3辺の長さから面積を計算する関数 S を作って下さい
   - ヘロンの公式
     - $S = \sqrt{s(s-a)(s-b)(s-c)},\ s = \frac{a+b+c}{2}$

main.js

```
let S = (a, b, c) => {
    let s = (a + b + c) / 2;
    return Math.sqrt(s * (s - a) * (s - b) * (s - c));
}
```

論理値

- 論理値には 真(true) と 偽(false) の2種類があります

main.js

```
let flag = true;
console.log(flag); // true
```

比較演算子

- a < b
  - a が b より小さいとき true
  - そうでないとき false
- a > b
  - a が b より大きいとき true
  - そうでないとき false

main.js

```
let a = 10;
let b = 5;
console.log(a < b); // false
console.log(a > b); // true
console.log(a < a); // false
```

比較演算子

- a <= b
  - a が b 以下のとき true
  - そうでないとき false
- a >= b
  - a が b 以上のとき true
  - そうでないとき false

main.js

```
let a = 10;
let b = 5;
console.log(a <= b); // false
console.log(a >= b); // true
console.log(a <= a); // true
```

比較演算子

- a == b
  - a と b が等しいとき true
  - そうでないとき false
- a != b
  - a と b が等しくないとき true
  - そうでないとき false

main.js

```
let a = 10;
let b = 5;
console.log(a == b); // false
console.log(a != b); // true
console.log(a == a); // true
```

if 文

- if (論理値) { 処理 }
  - 論理値 が true のとき 処理 を行います

main.js

```
let a = 10;
let b = 5;
if (a > b) {
    console.log("a > b");
}
```

if ~ else 文

- if (論理値) { 処理1 } else { 処理2 }
  - 論理値 が true のとき 処理1 を行います
  - 論理値 が false のとき処理2 を行います

main.js

```
let a = 10;
let b = 5;
if (a > b) {
    console.log("a > b");
}
else {
    console.log("a <= b");
}
```

if ~ else if ~ else 文

- { 処理 } の命令が1つのとき { 処理 } の { } を省略できとくに, if が続くときに省略します

main.js

```
let a = 10;
let b = 5;
if (a == b) {
    console.log("a == b");
}
else if (a < b) {
    console.log("a < b");
}
else {
    console.log("a > b");
}
```

論理演算子

- && 演算子
  - and をとります
  - true && true のときはtrue, それ以外のときはfalse
- || 演算子
  - or をとります
  - false || false のときはfalse, それ以外のときはtrue

main.js

```
let a = 10;
let b = 5;
let flagAnd = a != 10 && b >= 0; // a ≠ 10 かつ b ≥ 0
let flagOr = a != 10 || b >= 0; // a ≠ 10 または b ≥ 0
console.log(flagAnd); // false
console.log(flagOr); // true
```

論理否定演算子

- !a
  - aが true のとき false
  - aが false のとき true

main.js

```
let flag = true;
flag = !flag;
console.log(flag); // false
```

return

- returnを実行するとそこで関数は終了します

main.js

```
let func = () => {
    console.log(1);
    return;
    console.log(2);
    return;
}
func(); // 1のみ出力される
```

演習

1. Math.max(a, b) を使わずに a と b の大きい方を返す関数 max を作って下さい
   - let max = (a, b) => { ... }

2. 実行すると, 50%の確率で "表" , 50%の確率で "裏" とコンソールに出力される関数 flip を作って下さい
   - let flip = () => { ... }

演習

1. Math.max(a, b) を使わずに a と b の大きい方を返す関数 max を作って下さい

main.js

```
let max = (a, b) => {
    if (a > b) return a;
    return b;
}
```

main.js (別解)

```
let max = (a, b) => {
    return a + b - Math.min(a, b);
}
```

演習

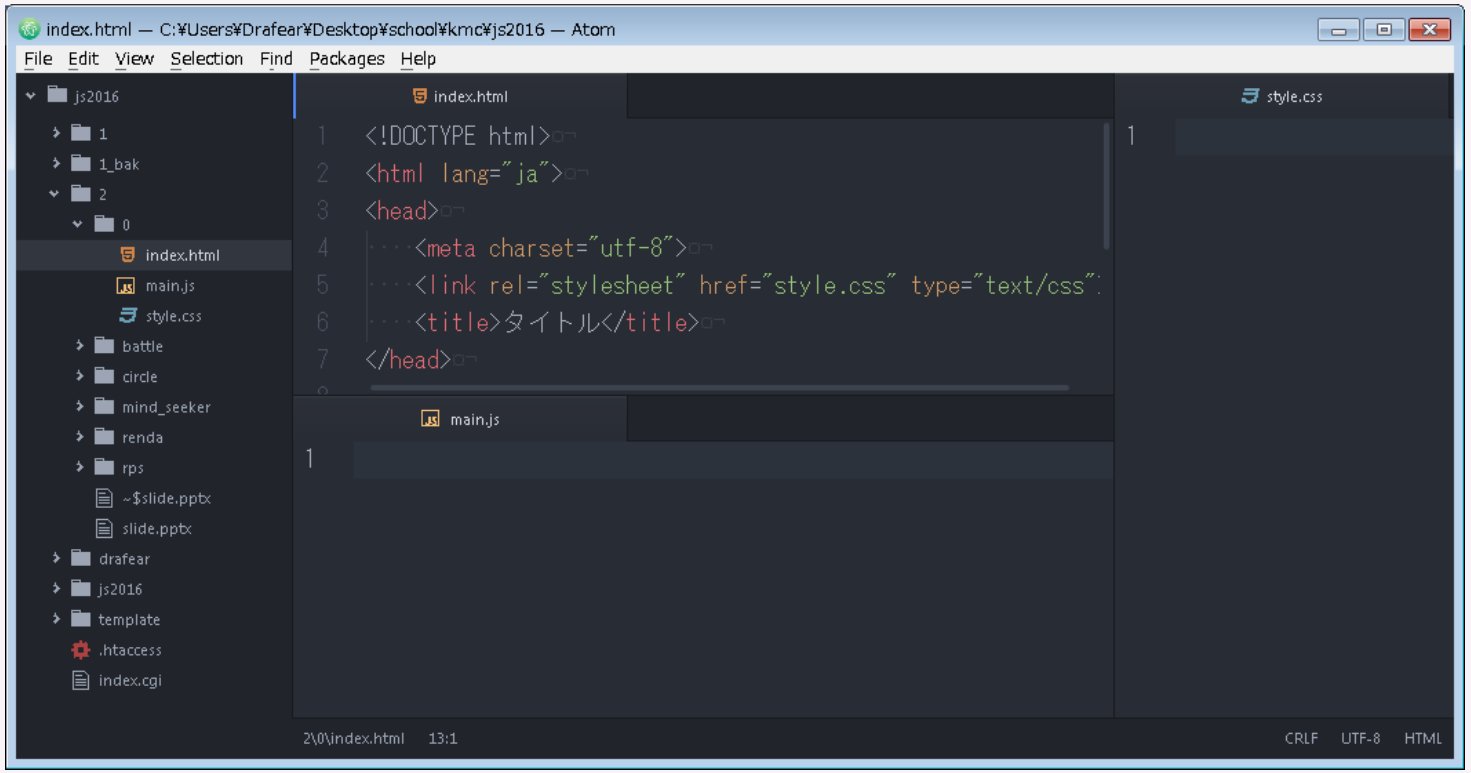
2. 実行すると, 50%の確率で “表” , 50%の確率で “裏” とコンソールに出力される関数 flip を作って下さい

main.js

```
let flip = () => {
    if ( Math.random() < 0.5 ) {
        console.log(“表”);
    }
    else {
        console.log(“裏”);
    }
}
```

CSS

- 次は, CSS を学んでいきます
- CSSは骨組みのHTMLにデザインしていくものです
- Atomの画面を3分割しましょう (右クリック → Split hoge)



CSS

- CSSでどんなことができるの？
  - http://www.nxworld.net/tips/css-only-button-design-and-hover-effects.html
  - http://www.nemuchan.com/css3/wk_base04.html
  - http://coliss.com/articles/build-websites/operation/css/css3-form-styling-cheat-sheet.html

class属性

- CSSを適用するためにまずclass属性を設定します
  - 雛形の「内容」の部分だけいじっていきます

index.html

```
<span class="class1">text</span>
```

text装飾

- style.css に次のように記述します

style.css

```
.class1 {
    color: red;
    font-weight: bold;
    font-size: 40px;
    text-decoration: underline;
}
```

index.html

```
<span class="class1">text</span>
```

text装飾

- 文法
  - .クラス名 { プロパティ名1: 値1; プロパティ名2: 値2; }
- プロパティ (順不同)
  - color: red
    - 色を指定します
    - rgb(180, 220, 180) や #aabbcc でRGB指定できます
  - font-weight: bold
    - 太字にします
  - font-size: 40px
    - 文字の大きさを指定します. 単位はここではpxで指定します(他の単位は割愛).
  - text-decoration: underline
    - 下線を引きます

複数のクラス指定

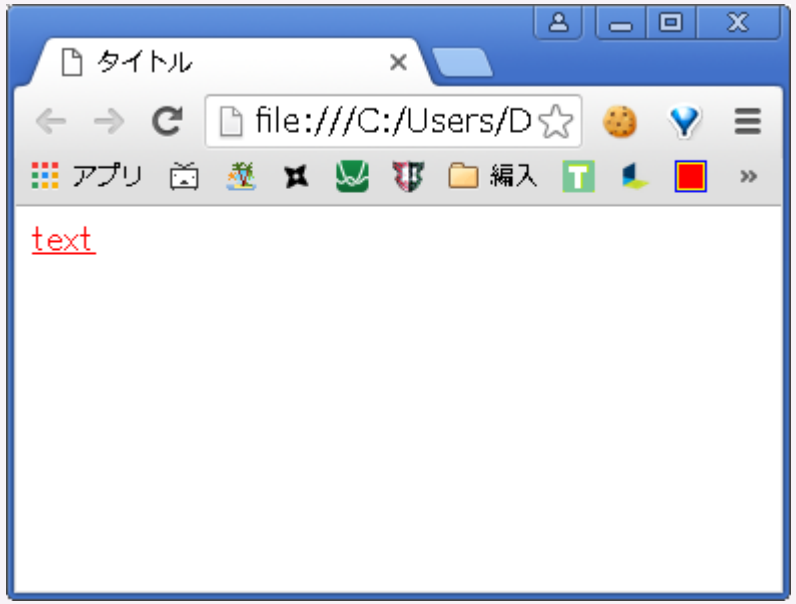
- 1つの要素に複数のクラスを指定できます

index.html

```
<span class="red underline">text</span>
```

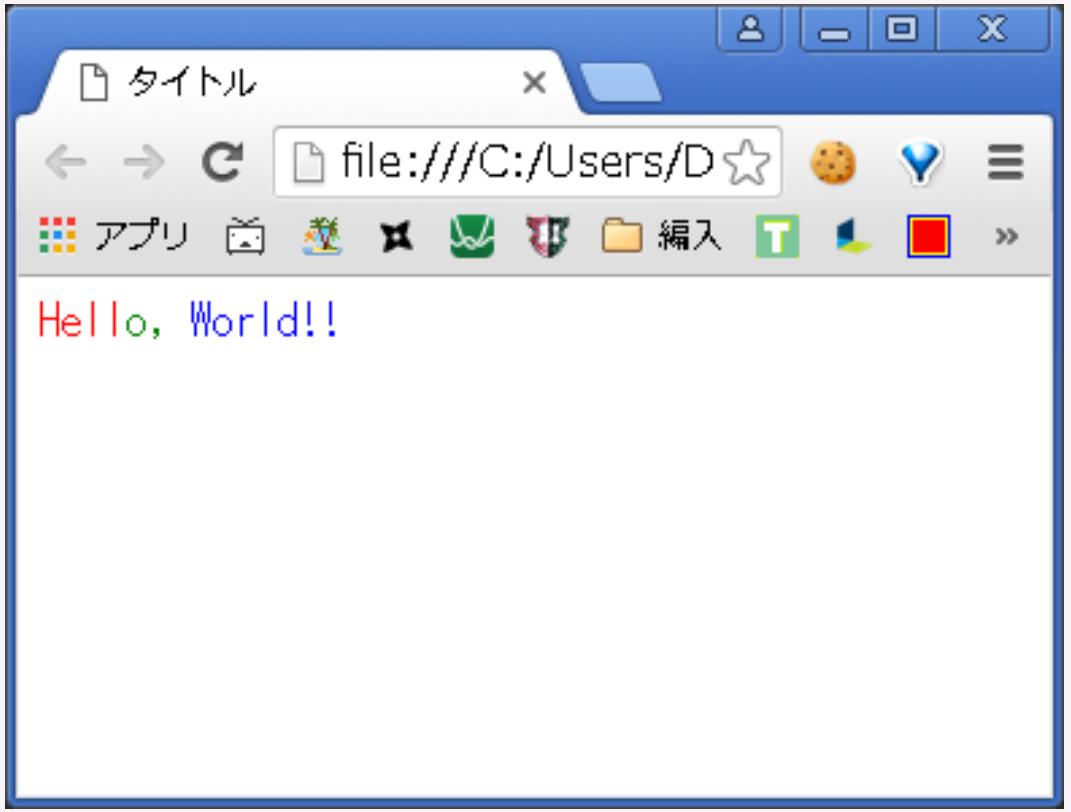
style.css

```
.red {
    color: red;
}
.underline {
    text-decoration: underline;
}
```



演習

- 次のような表示をするものを作って下さい



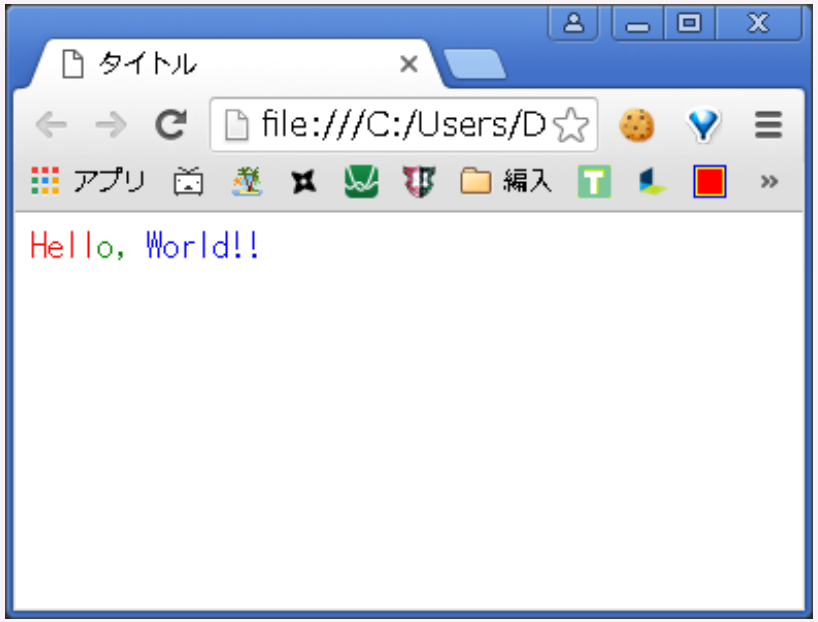
演習

index.html

```
<span class="red">Hell</span><span class="green">o, </span><span class="blue">World!!</span>
```

style.css

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



JavaScriptでclassをいじる

- elem.classList.add("className")
  - 要素にクラス [className] を追加する
- elem.classList.remove("className")
  - 要素からクラス [className] を削除する
- elem.classList.toggle("className")
  - 要素がクラス [className] を持っていれば remove
    持っていなければ add

JavaScriptでclassをいじる

- elem.classList.contains("className")
  - 要素がクラス [className] を持っていたら true, なければ false

- elem.className = "";
  - 要素のクラスをクリアする

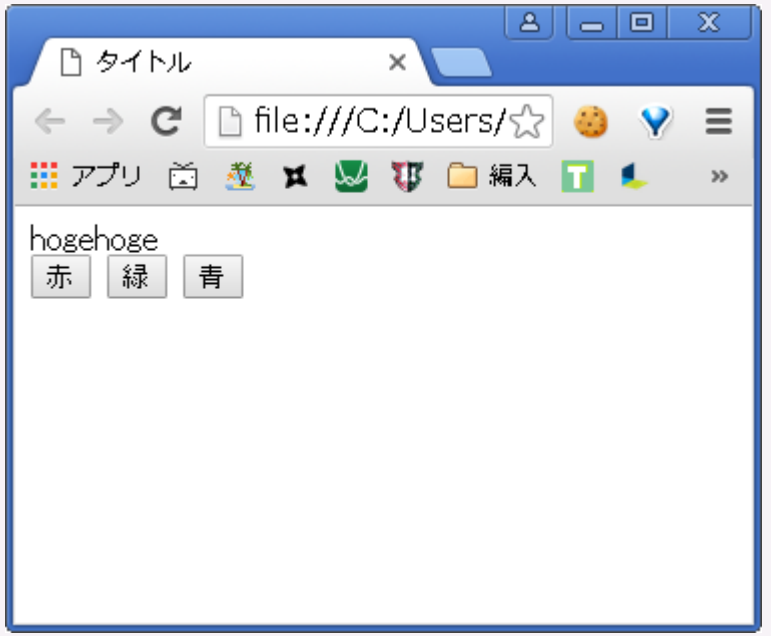
JavaScriptでclassをいじる

index.html

```
<div id="text">hogehoge</div>
<button id="btnRed">赤</button>
<button id="btnGreen">緑</button>
<button id="btnBlue">青</button>
```

style.css

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



JavaScriptでclassをいじる

main.js

```
let prevClass = "";
let changeClass = (c) => { // change class to c
    let target = document.getElementById("text");
    if (prevClass != "") {
        target.classList.remove(prevClass);
    }
    target.classList.add(c);
    prevClass = c;
}
document.getElementById("btnRed").addEventListener("click", (e) => {
    changeClass("red");
});
document.getElementById("btnGreen").addEventListener("click", (e) => {
    changeClass("green");
});
document.getElementById("btnBlue").addEventListener("click", (e) => {
    changeClass("blue");
});
```

display: none;

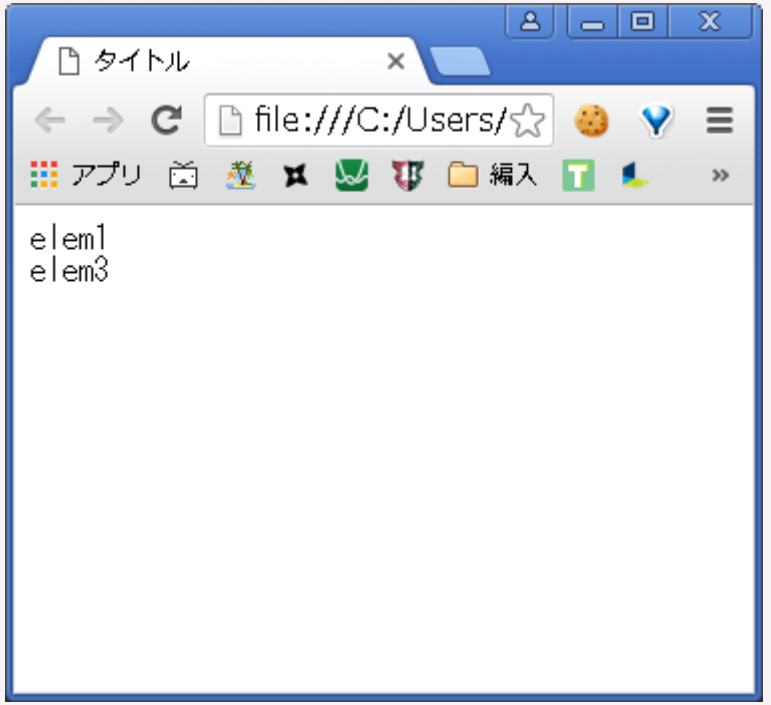
- cssのプロパティ display に none を指定すると...
  - その要素が非表示になる

index.html

```
<div>elem1</div>
<div class="hide">elem2</div>
<div>elem3</div>
```

style.css

```
.hide {
    display: none;
}
```

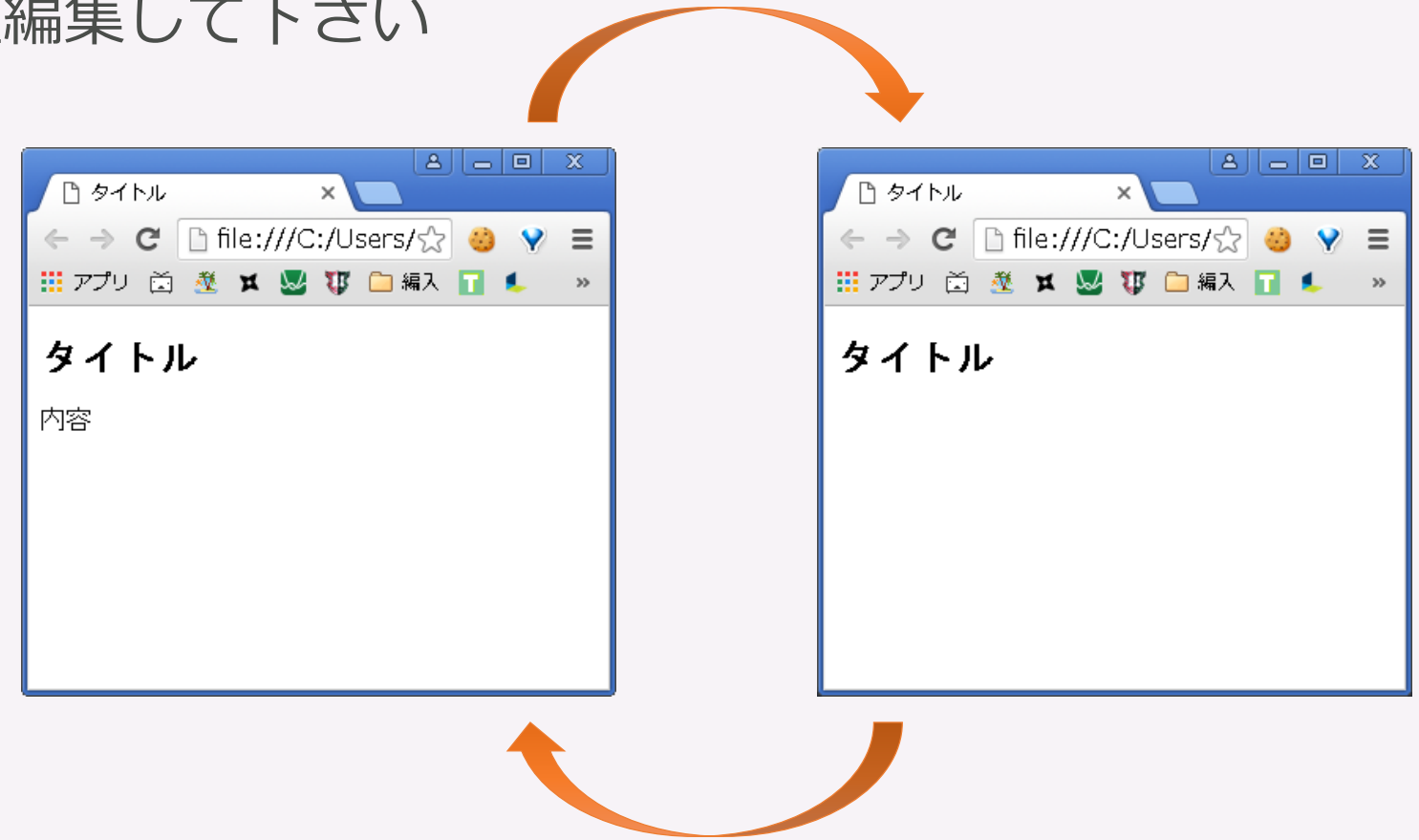


演習

- タイトル<h2>～</h2>をクリックすると内容 ～ の表示非表示が切り替わるようにしてみよう
  - index.html も適宜編集して下さい

index.html

```
<h2>タイトル</h2>
<div>内容</div>
```



演習

- タイトル<h2>～</h2>をクリックすると内容 ～ の表示非表示が切り替わるようにしてみよう

index.html

```
<h2 id="title">タイトル</h2>
<div id="content">内容</div>
```

style.css

```
.hide {
    display: none;
}
```

main.js

```
document.getElementById("title").addEventListener("click", (e) => {
    document.getElementById("content").classList.toggle("hide");
});
```

演習 (別解)

- タイトル<h2>～</h2>をクリックすると内容 ～ の表示非表示が切り替わるようにしてみよう

index.html

```
<h2 id="title">タイトル</h2>
<div id="content">内容</div>
```

style.css

```
.hide {
    display: none;
}
```

main.js

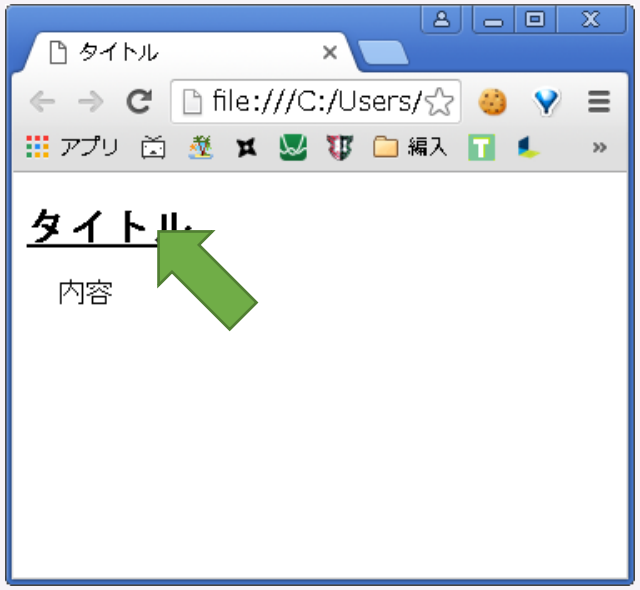
```
document.getElementById("title").addEventListener("click", (e) => {
    let elem = document.getElementById("content");
    if (elem.classList.contains("hide")) {
        elem.classList.remove("hide");
    }
    else {
        elem.classList.add("hide");
    }
});
```

演習のものに装飾してみる

- タイトルにカーソルを合わせたときの挙動を変えよう
  - main.js はそのまま

index.html

```
<h2 id="title" class="title">タイトル</h2>
<div id="content" class="content">内容</div>
```



style.css

```
.hide {
    display: none;
}
.title {
    cursor: pointer;
}
.title:hover {
    text-decoration: underline;
}
.content {
    margin-left: 20px;
}
```

演習のものに装飾してみる

- cursor: pointer;
  - 要素にマウスカーソルを合わせたときのアイコンの種類
  - 一覧
    - http://www.tagindex.com/stylesheet/page/cursor.html
- .title:hover
  - titleクラスが設定された要素にマウスオーバーされた状態の設定
- margin-left: 20px;
  - 要素の左側に20pxだけ余白を設ける
  - margin-top/right/bottom/left

style.css

```
.hide {
    display: none;
}
.title {
    cursor: pointer;
}
.title:hover {
    text-decoration: underline;
}
.content {
    margin-left: 20px;
}
```

hoverのサンプル

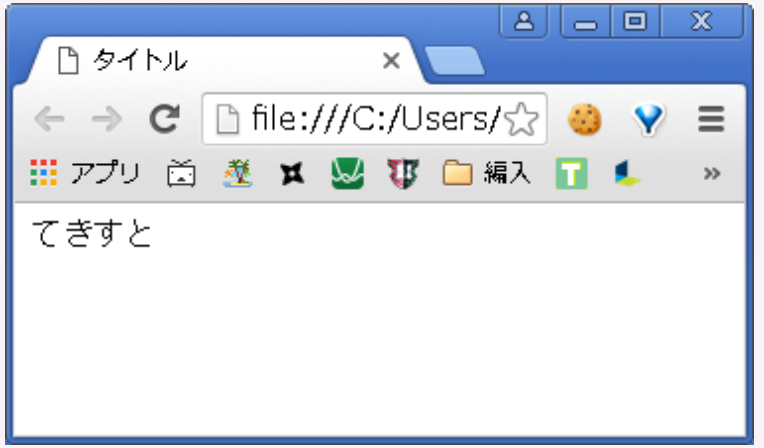
- マウスオーバーで赤文字にするサンプルです

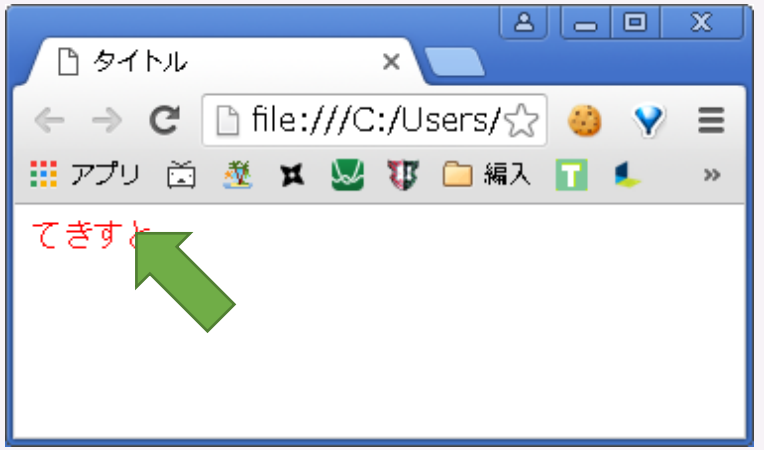
index.html

```
<span class="hoverRed">てきすと</span>
```

style.css

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





ボックス

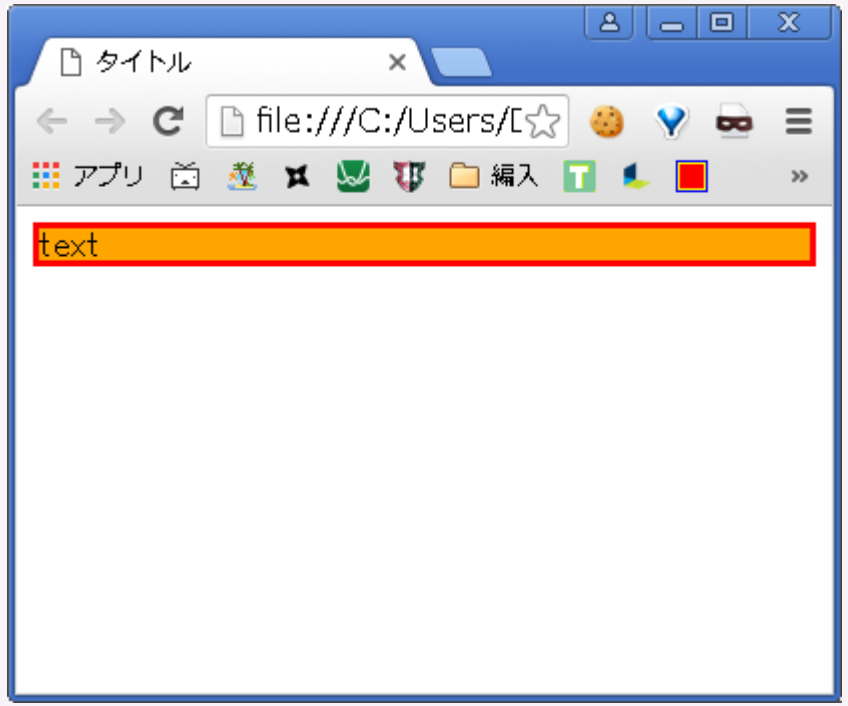
- HTML要素は全て長方形の領域を持っています
- その領域に装飾してみます

index.html

```
<div class="class1">text</div>
```

style.css

```
.class1 {
    border: 3px solid red;
    background-color: orange;
}
```



border, background-color

- border: 3px solid red;
  - 枠線の 太さが3px で 一重線 で 赤色
  - solid
    - 一重線
    - double(二重線), none(なし) など
- background-color: orange;
  - 背景色
  - 背景に画像を使う場合
    - background-image: url(“画像へのpath”);
    - 背景画像の縦ループ, 横ループなどはまた別のプロパティ指定でできます

色の指定

1. white, black, red, ...
   - 一覧: http://www5.plala.or.jp/vaio0630/hp/c_code.htm
2. #000000 ～ #FFFFFF (or #ffffff)
3. #000 ～ #FFF (or #fff)
4. rgb(0～255, 0～255, 0～255)
5. rgb(0%～100%, 0%～100%, 0%～100%)
6. hsl(色相(0以上360未満), 彩度(0～100%), 輝度(0～100%))

色の指定(alpha値: 透明度)

- 書式
  - rgba(red, green, blue, alpha)
  - hsla(hue, saturation, lightness, alpha)
- 値
  - 0(透明) ～ 1(不透明)

色の指定(alpha値: 透明度)

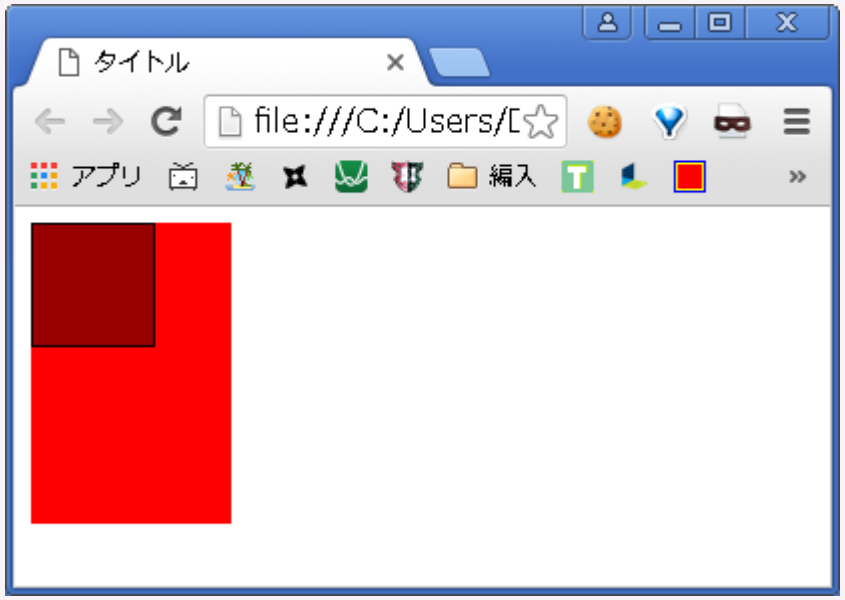
- 例

index.html

```
<div class="box1">
    <div class="box2"></div>
</div>
```

style.css

```
.box1 {
    width: 100px;  /* 横幅 */
    height: 150px; /* 縦幅 */
    background-color: red;
}
.box2 {
    width: 60px;  /* 横幅 */
    height: 60px; /* 縦幅 */
    background-color: rgba(0, 0, 0, 0.4);
    border: 1px solid black;
}
```

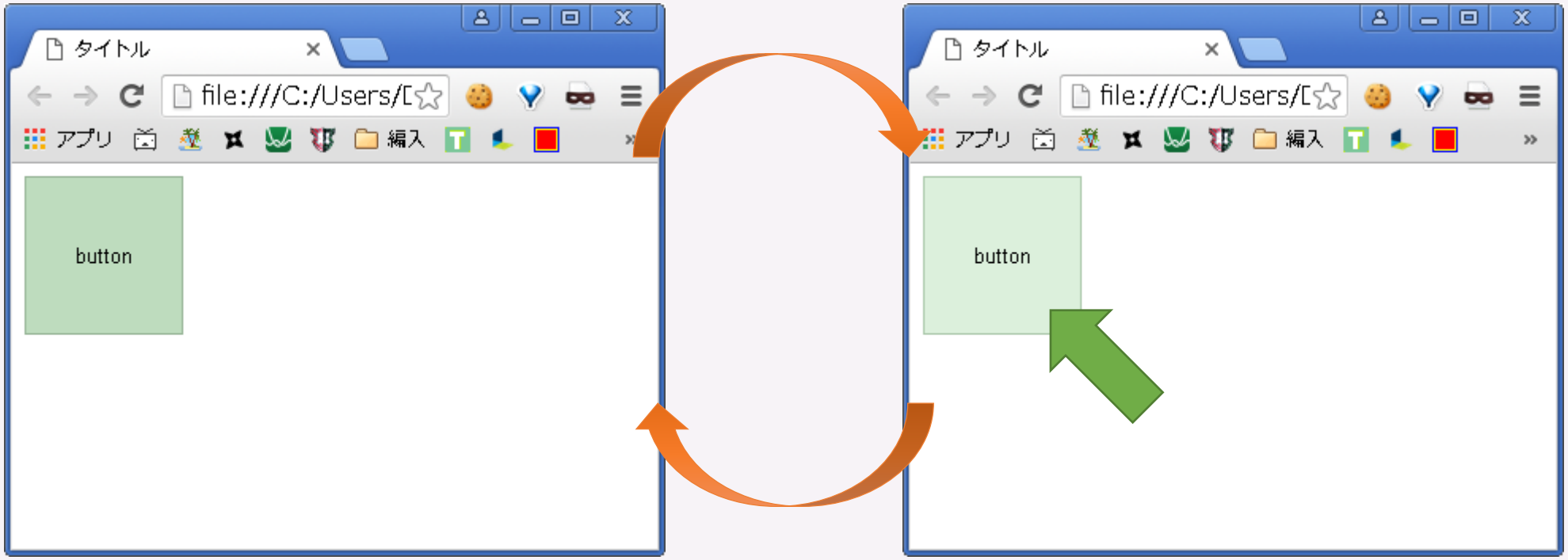


演習

- マウスオーバーで背景色が変化するボタンを作ってみましょう

index.html

```
<button class="button1">button</button>
```



演習

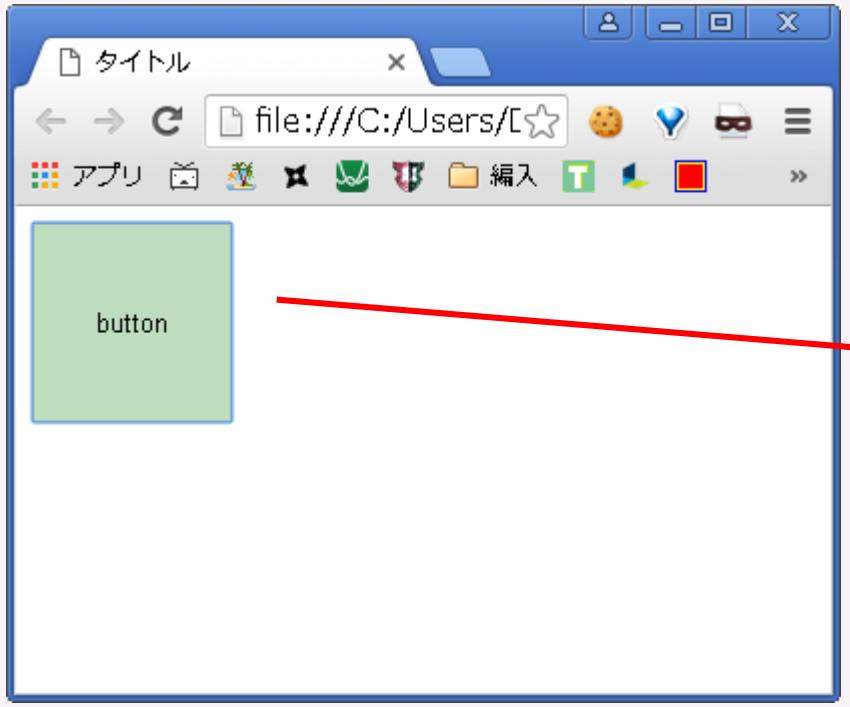
- 例 (枠線の色も変えていますが)

style.css

```
.button1 {
    width: 100px; /* 横幅 */
    height: 100px; /* 縦幅 */
    background-color: rgb(190, 220, 190);
    border: 1px solid rgb(150, 180, 150);
}
.button1:hover {
    background-color: rgb(220, 240, 220);
    border: 1px solid rgb(170, 200, 170);
}
```

豆知識

- クリックした時に表示される青い縁取りが気になる場合は outline: none; を追加すると表示されなくなります



style.css

```css
.button1 {
    width: 100px; /* 横幅 */
    height: 100px; /* 縦幅 */
    background-color: rgb(190, 220, 190);
    border: 1px solid rgb(150, 180, 150);
    outline: none;
}
.button1:hover {
    background-color: rgb(220, 240, 220);
    border: 1px solid rgb(170, 200, 170);
}
```

アニメーション

- マウスオーバー時に ふわっ と背景色を変化させてみる
  - transition: プロパティ 時間;
  - [プロパティ] の値が変化するとき [時間] の間でなめらかに変化する
  - 全てのプロパティに対して設定する場合は all を指定する

style.css

```css
.button1 {
    width: 100px; /* 横幅 */
    height: 100px; /* 縦幅 */
    background-color: rgb(190, 220, 190);
    border: 1px solid rgb(150, 180, 150);
    outline: none;
    transition: all 0.3s; /* これだけ！ */
}
.button1:hover {
    background-color: rgb(220, 240, 220);
    border: 1px solid rgb(170, 200, 170);
}
```

アニメーション

- マウスオーバー時に ふわっ と背景色を変化させてみる
  - 複数のプロパティを指定したい場合以下のようにする

style.css

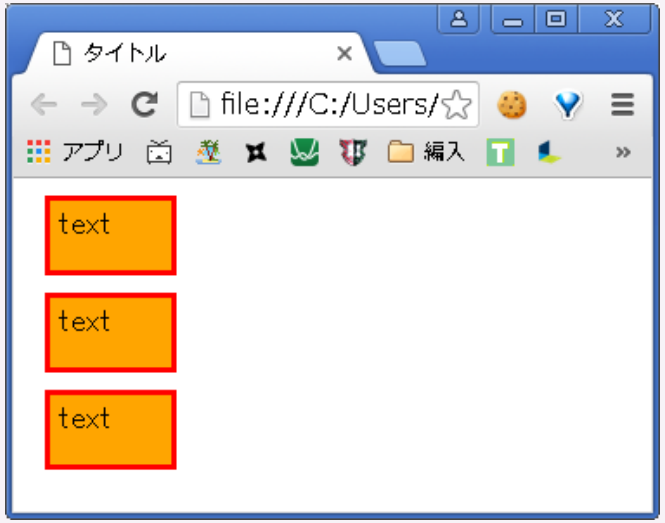
```
.button1 {
    width: 100px; /* 横幅 */
    height: 100px; /* 縦幅 */
    background-color: rgb(190, 220, 190);
    border: 1px solid rgb(150, 180, 150);
    outline: none;
    transition-property: background-color, border-color;
    transition-duration: 0.3s;
}
.button1:hover {
    background-color: rgb(220, 240, 220);
    border: 1px solid rgb(170, 200, 170);
}
```

padding, border, margin (重要)

- 先ほど出てきた width, height などについて説明します

index.html

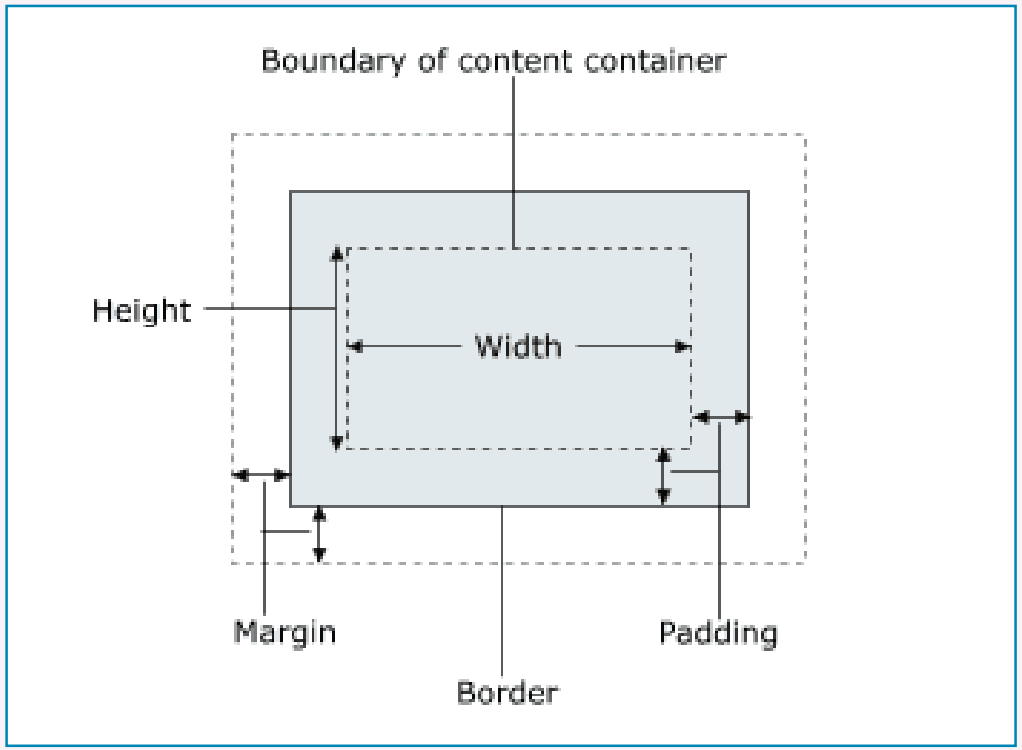
```
<div class="class1">text</div>
<div class="class1">text</div>
<div class="class1">text</div>
```



style.css

```
.class1 {
    width: 60px;
    height: 30px;
    margin: 10px;
    padding: 5px;
    border: 3px solid red;
    background-color: orange;
}
```

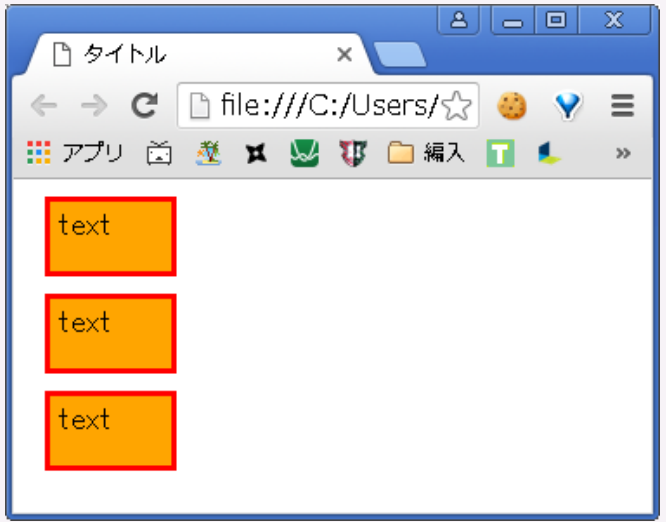
padding, border, margin (重要)



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

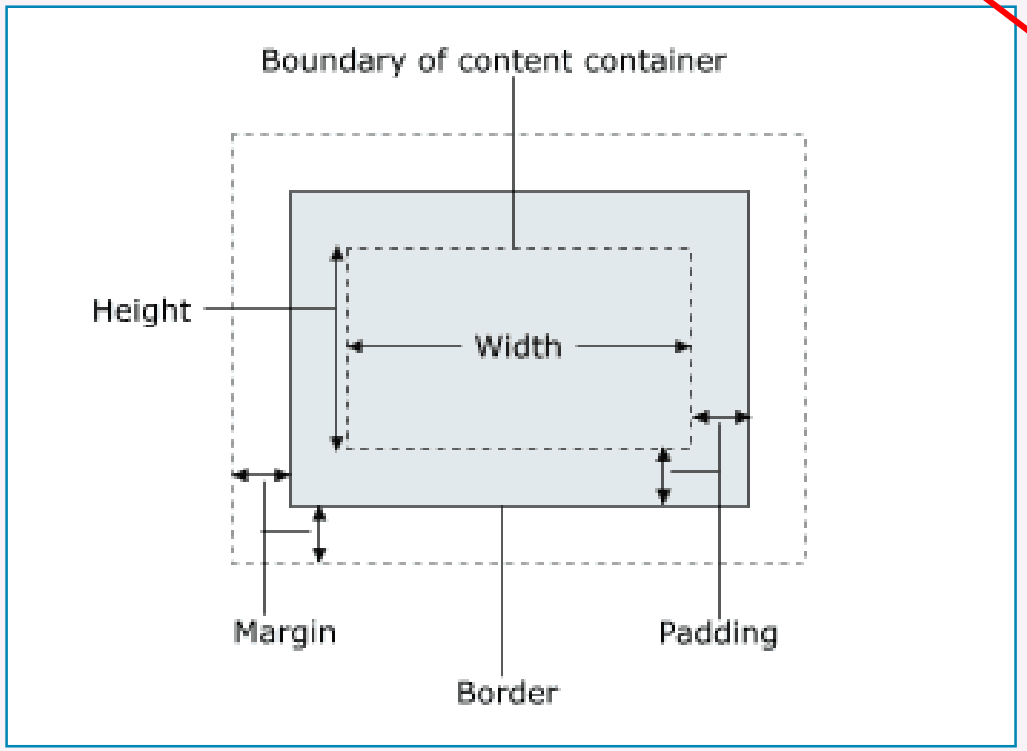
style.css

```css
.class1 {
    width: 60px;
    height: 30px;
    margin: 10px;
    padding: 5px;
    border: 3px solid red;
    background-color: orange;
}
```



問題

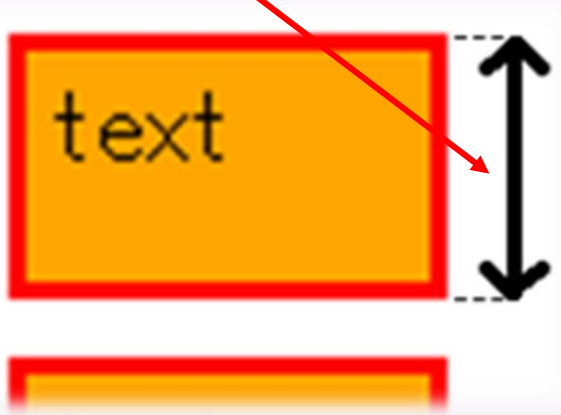
- この幅は 何px でしょう



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

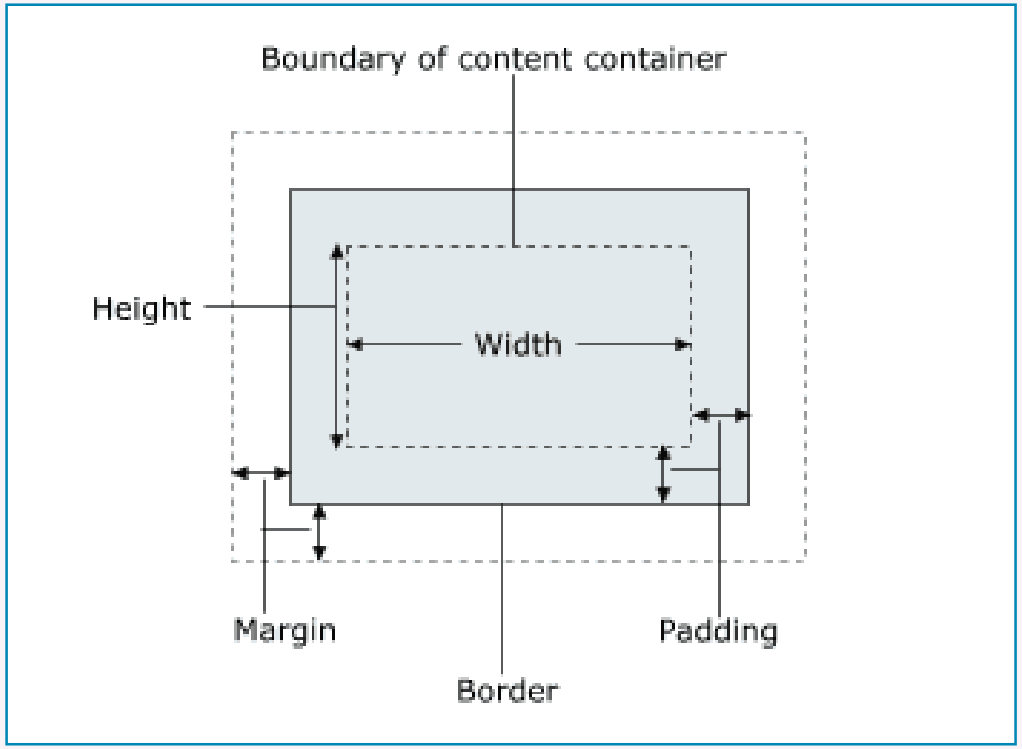
style.css

```
.class1 {
    width: 60px;
    height: 30px;
    margin: 10px;
    padding: 5px;
    border: 3px solid red;
    background-color: orange;
}
```



答え

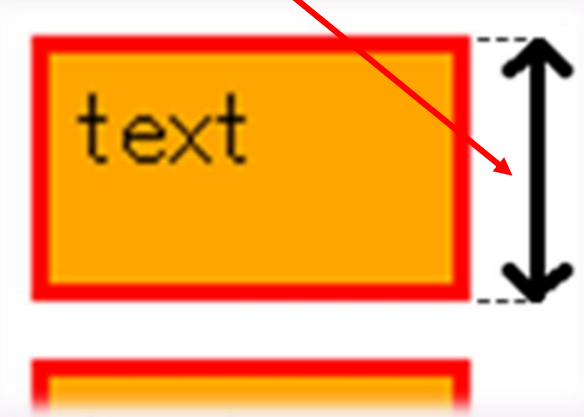
height + padding*2 + border*2
= 46px



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

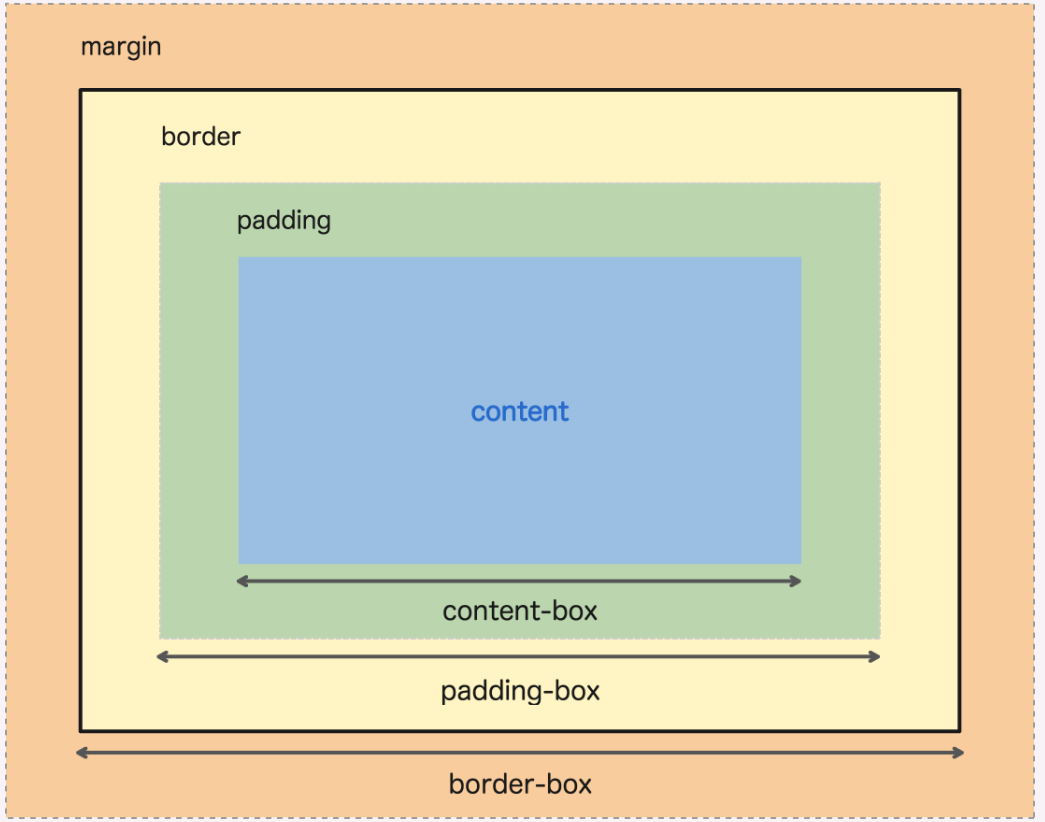
style.css

```
.class1 {
    width: 60px;
    height: 30px;
    margin: 10px;
    padding: 5px;
    border: 3px solid red;
    background-color: orange;
}
```



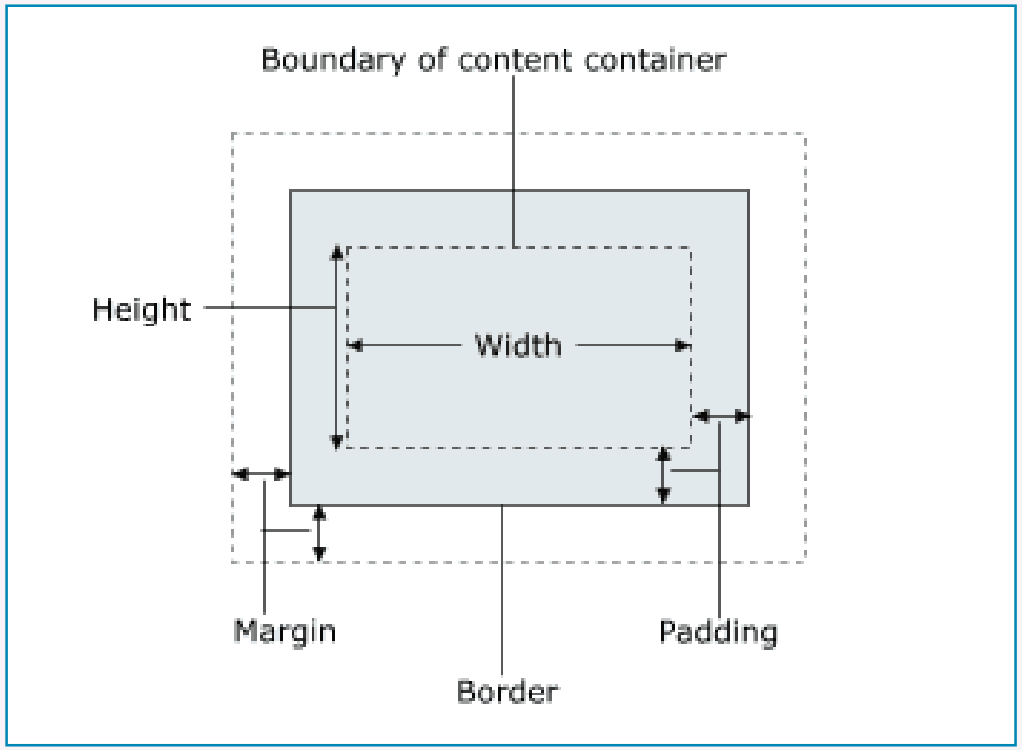
計算つらひ

- box-sizing プロパティ
  - width, heightの計算方法を変更する
  - 値
    - content-box (default)
    - padding-box
    - border-box



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

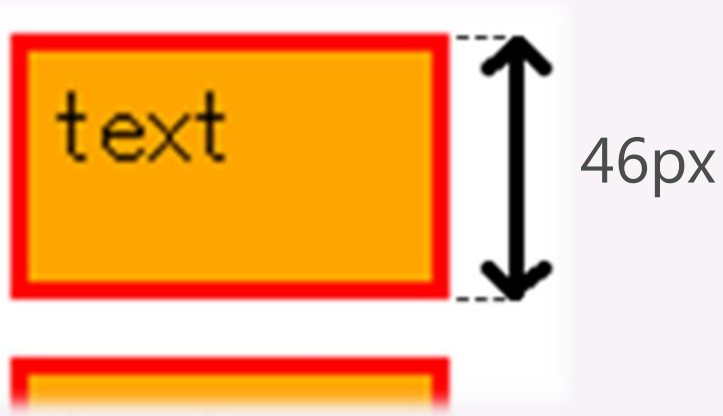
box-sizing



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

style.css

```css
.class1 {
    width: 76px;
    height: 46px;
    margin: 10px;
    padding: 5px;
    border: 3px solid red;
    background-color: orange;
    box-sizing: border-box;
}
```



content

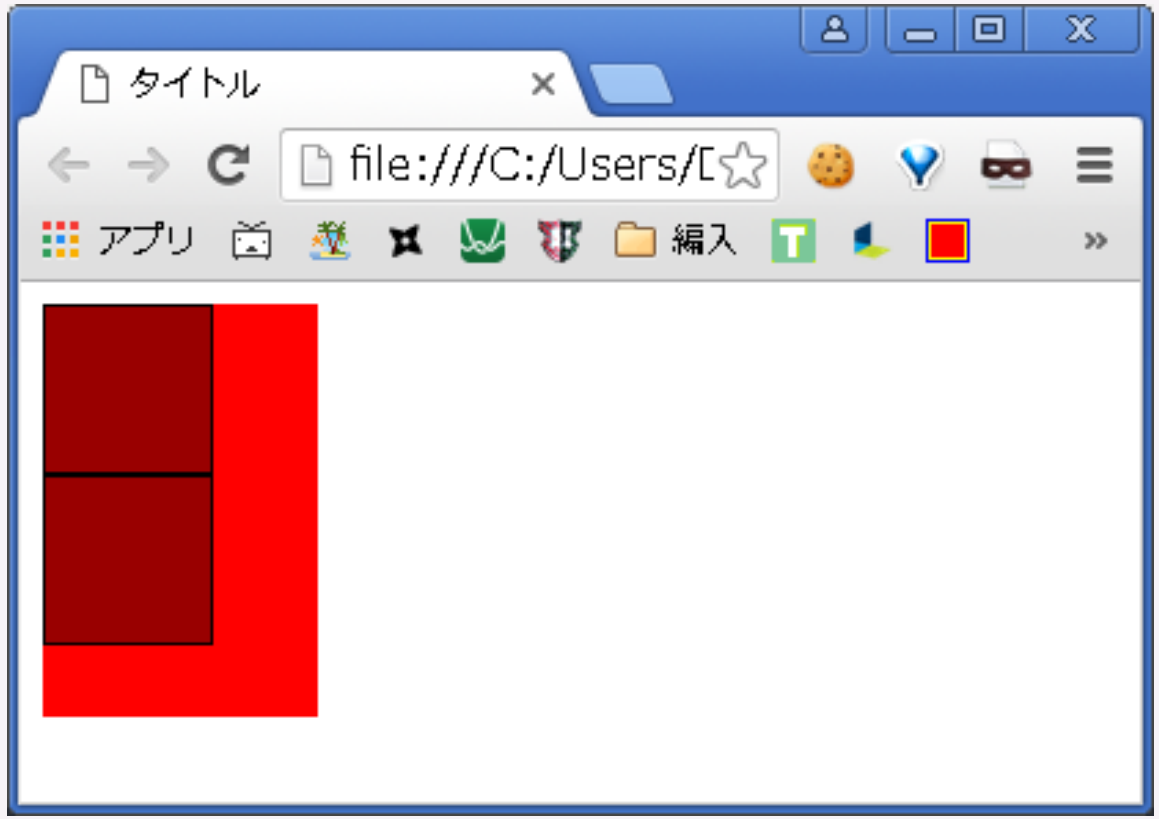
- 子要素は親要素のcontent領域の左上から順に配置されます

index.html

```
<div class="box1">
    <div class="box2"></div>
    <div class="box2"></div>
</div>
```

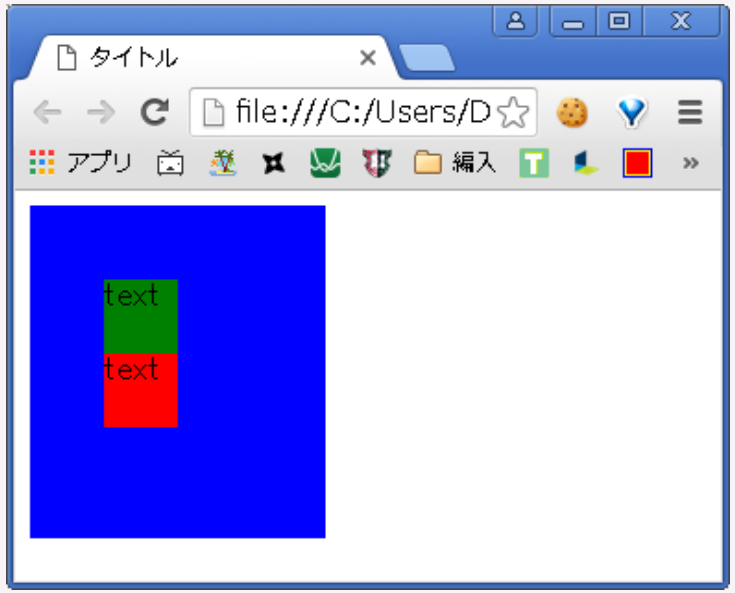
style.css

```
.box1 {
    width: 100px; height: 150px;
    background-color: red;
}
.box2 {
    width: 60px; height: 60px;
    background-color: rgba(0, 0, 0, 0.4);
    border: 1px solid black;
}
```



演習

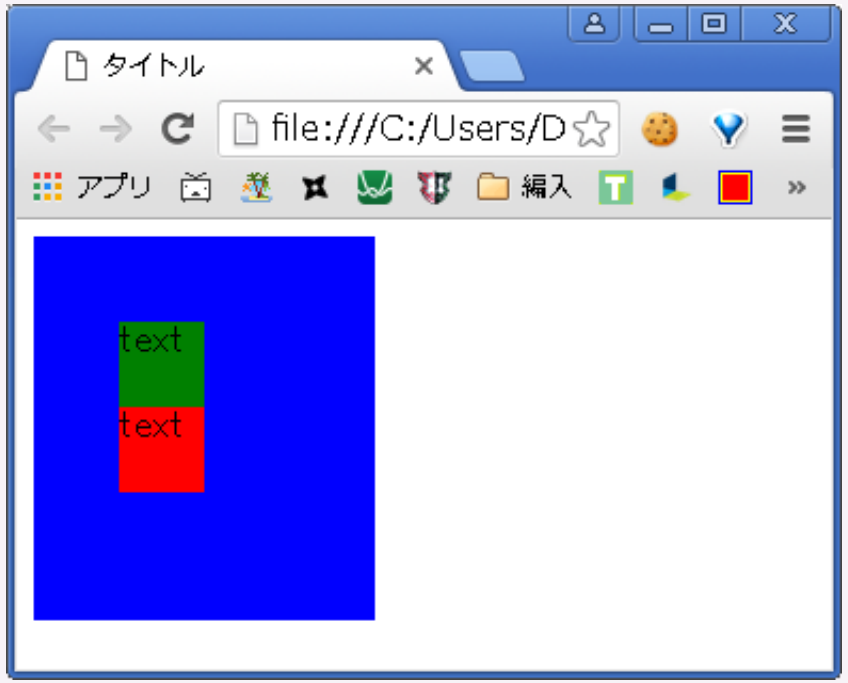
- 次のような表示をするものを作って下さい
- 時間が余ったら装飾してみましょう
  - 半透明にする
  - カーソルを合わせると背景色が変わる
  - カーソルを合わせるとボックスの大きさが変わる
  - アニメーションを付ける



演習

index.html

```
<div class="box1">
    <div class="box2">text</div>
    <div class="box3">text</div>
</div>
```



style.css

```
.box1 {
    width: 80px;
    height: 100px;
    padding: 40px;
    background-color: blue;
}
.box2 {
    width: 40px;
    height: 40px;
    background-color: green;
}
.box3 {
    width: 40px;
    height: 40px;
    background-color: red;
}
```

演習

- 同じ設定はまとめられます

style.css

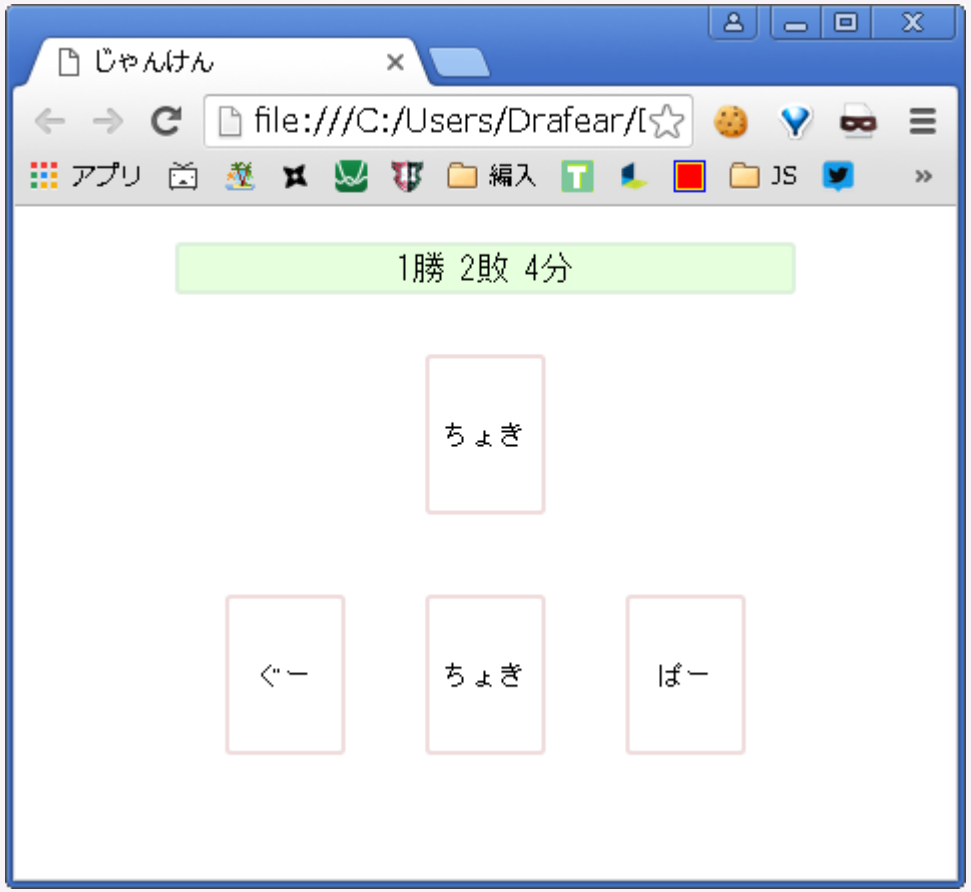
```
.box1 {
    width: 80px;
    height: 100px;
    padding: 40px;
    background-color: blue;
}
.box2, .box3 {
    width: 40px;
    height: 40px;
}
.box2 { background-color: green; }
.box3 { background-color: red; }
```

div と span の違い (重要)

- spanは囲むだけ, divは四角いブロックを生成する
- だからspanにはwidth等が設定できない
- divの後には改行が入る
- 次回、より詳しく説明します

じゃんけん

- 作ります

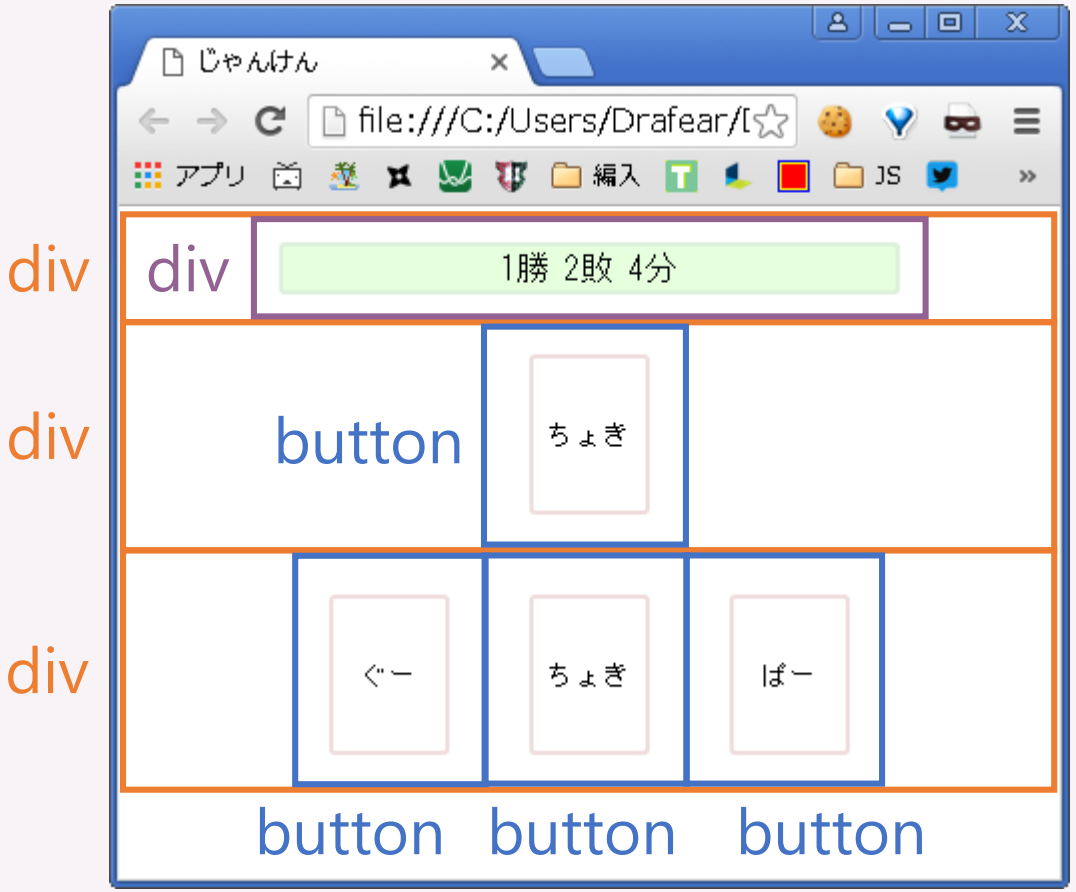


じゃんけん

- 構造を定めます

index.html

```
<div>
    <div>0勝 0敗 0分</div>
</div>
<div>
    <button></button>
</div>
<div>
    <button>ぐー</button>
    <button>ちょき</button>
    <button>ぱー</button>
</div>
```

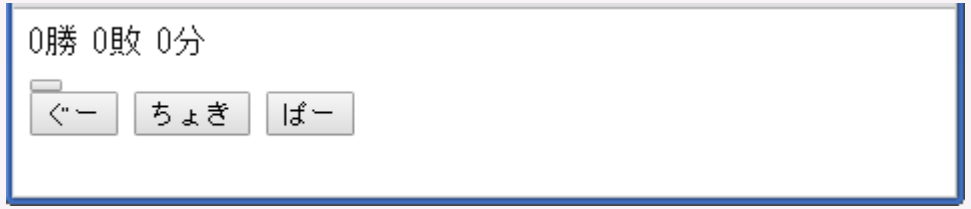


じゃんけん

- id, classを振っていきます

index.html

```
<div class="centering">
    <div id="result" class="result">0勝 0敗 0分</div>
</div>
<div class="centering">
    <button id="enemyHand" class="hand disable"></button>
</div>
<div class="centering">
    <button id="rock" class="hand">ぐー</button>
    <button id="scissors" class="hand">ちょき</button>
    <button id="paper" class="hand">ぱー</button>
</div>
```



じゃんけん

- 中央寄せします
- 次回ちゃんと説明するので
  今のところは "おまじない" で大丈夫です

style.css

```
.centering {
    display: flex;
    justify-content: center;
    align-items: center;
}
```

じゃんけん

- .hand, .result をデザインします
  - .centering { ... } は消さずに追記して下さい

style.css

```
.hand {
    margin: ...;
    width: ...;
    height: ...;
    background-color: ...;
    border: ...;
    border-radius: 3px; /* 角を丸める */
    outline: none;
}
.result {
    ...
    text-align: center; /* テキスト中央寄せ */
    cursor: default;
}
```

style.css (サンプル)

```
.hand {
    margin: 20px;
    width: 60px;
    height: 80px;
    background-color: white;
    border: 2px solid rgb(240, 220, 220);
    border-radius: 3px;
    outline: none;
}
.result {
    width: 300px;  /* height は指定しない */
    margin: 10px;
    padding: 3px;
    border: 2px solid rgb(220, 240, 220);
    background-color: rgb(230, 255, 220);
    border-radius: 3px;
    text-align: center;
    cursor: default;
}
```

じゃんけん

- :not(条件)
  - 条件に一致しないもの
  - 例) .a:not(.b)
    - クラスa が設定されていて クラスb が設定されていない要素

style.css (サンプル)

```
.hand:not(.disable) {
    cursor: pointer;
}
.hand:not(.disable):hover {
    background-color: rgb(255, 230, 230);
}
```

じゃんけん

- 次はJavaScript部分になります

main.js

```
let drawCount = 0; // 引き分けになった回数
let winCount = 0;   // 勝利した回数
let loseCount = 0;  // 敗北した回数
let selectHand = (myHand) => { /* myHand番の手を出したときの処理 */ }
document.getElementById("rock").addEventListener("click", (e) => {
    selectHand(0);
});
document.getElementById("scissors").addEventListener("click", (e) => {
    selectHand(1);
});
document.getElementById("paper").addEventListener("click", (e) => {
    selectHand(2);
});
```

じゃんけん

- selectHandを実装しましょう

main.js

```
let drawCount = 0; // 引き分けになった回数
let winCount = 0;   // 勝利した回数
let loseCount = 0;  // 敗北した回数

// [myHand] 0:ぐー / 1:ちょき / 2:ぱー
let selectHand = (myHand) => {
    let enemyHand = Math.floor( Math.random()*3 ); // 相手の手
    /*
        drawCount, winCount, loseCount を更新する
        (fill in here)
    */
    update(enemyHand); // 画面に反映する
}
let update = (enemyHand) => { ... }
```

じゃんけん

- selectHandを実装しましょう

main.js

```
let selectHand = (myHand) => {
    let enemyHand = Math.floor( Math.random()*3 );
    if (myHand == enemyHand) { // draw
        drawCount += 1;
    }
    else if ( (myHand+1) % 3 == enemyHand ) { // win
        winCount += 1;
    }
    else { // lose
        loseCount += 1;
    }
    update(enemyHand);
}
```

じゃんけん

- 画面に反映する関数 update を実装しましょう

main.js

```
let update = (enemyHand) => {
    let enemyHandName = "";
    /*
        enemyHand(0, 1, 2) から enemyHandName("ぐー", "ちょき", "ぱー") に変換
        (fill in here)
    */
    document.getElementById("enemyHand").innerText = /* (fill in here) */;
    document.getElementById("result").innerText = /* (fill in here) */;
}
```

じゃんけん

- 画面に反映する関数 update を実装しましょう

main.js

```
let update = (enemyHand) => {
    let enemyHandName = "";
    if (enemyHand == 0) {
        enemyHandName = "ぐー";
    }
    else if (enemyHand == 1) {
        enemyHandName = "ちょき";
    }
    else if (enemyHand == 2) {
        enemyHandName = "ぱー";
    }
    document.getElementById("enemyHand").innerText = enemyHandName;
    document.getElementById("result").innerText =
        `${winCount}勝 ${loseCount}敗 ${drawCount}分`;
}
```

じゃんけん

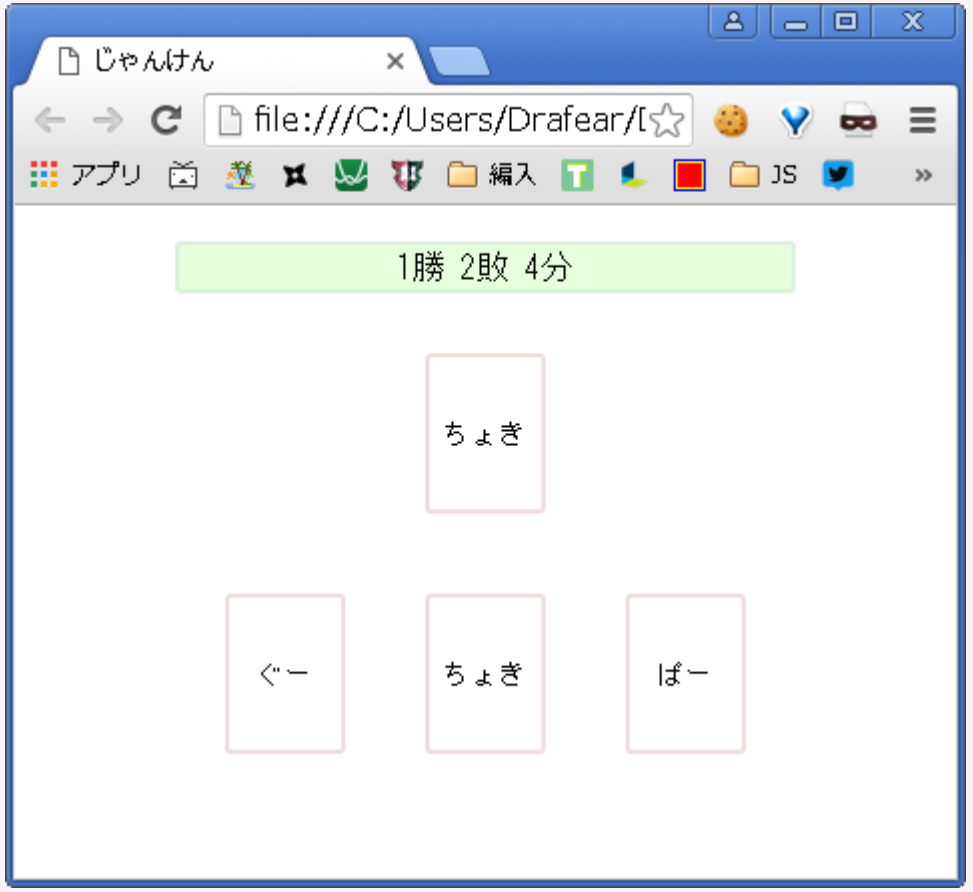
- 最初にupdate(-1)を呼んで画面を初期化するようにしましょう

main.js

```
let drawCount = 0; // 引き分けになった回数
let winCount = 0;   // 勝利した回数
let loseCount = 0;  // 敗北した回数

let selectHand = (myHand) => { ... }
let update = (enemyHand) => { ... }

document.getElementById("rock").addEventListener("click", (e) => { ... });
document.getElementById("scissors").addEventListener("click", (e) => { ... });
document.getElementById("paper").addEventListener("click", (e) => { ... });

update(-1); // 追加
```

じゃんけん

- 完成！！ お疲れ様です！！



じゃんけん

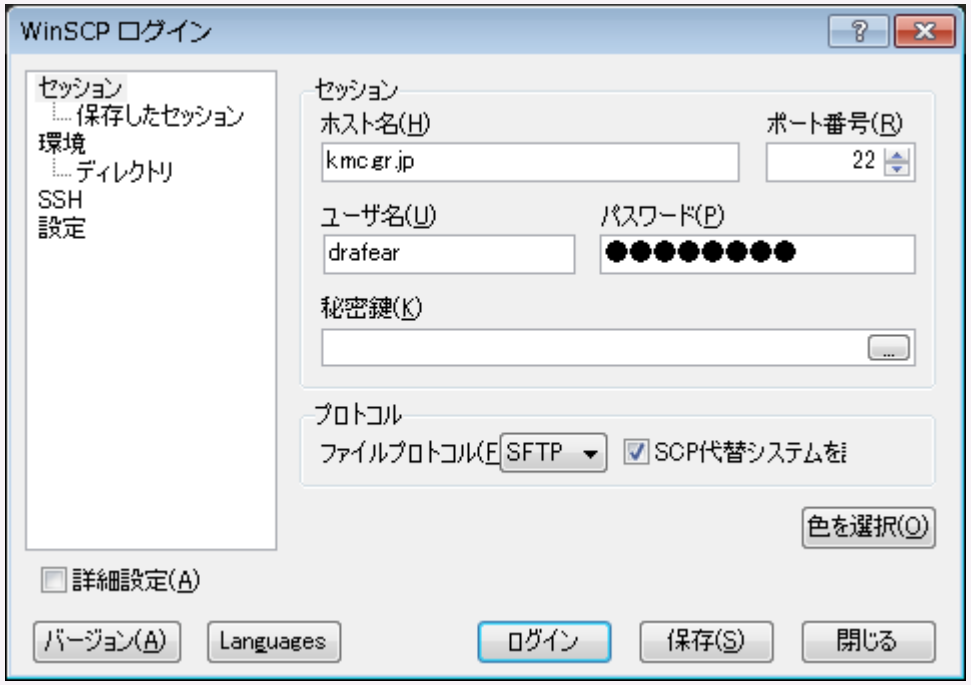
- 余力があれば改造してみてください
  - AIを変える
    - "ぐー", "ちょき", "ぱー" からランダムに選んでいるが
      「特定の手しか出さない」「絶対に勝つ手を出す」「ローテーション」など
  - 前に自分が選んだ手が分かりやすいようにデザインする
  - 勝率を出す
  - ルールを追加する
    - 「前に出した手は出せない」など

作品をアップロードしてみよう

- WinSCPを使って内部ページにアップロードしてみる
  - http://www.forest.impress.co.jp/library/software/winscp/

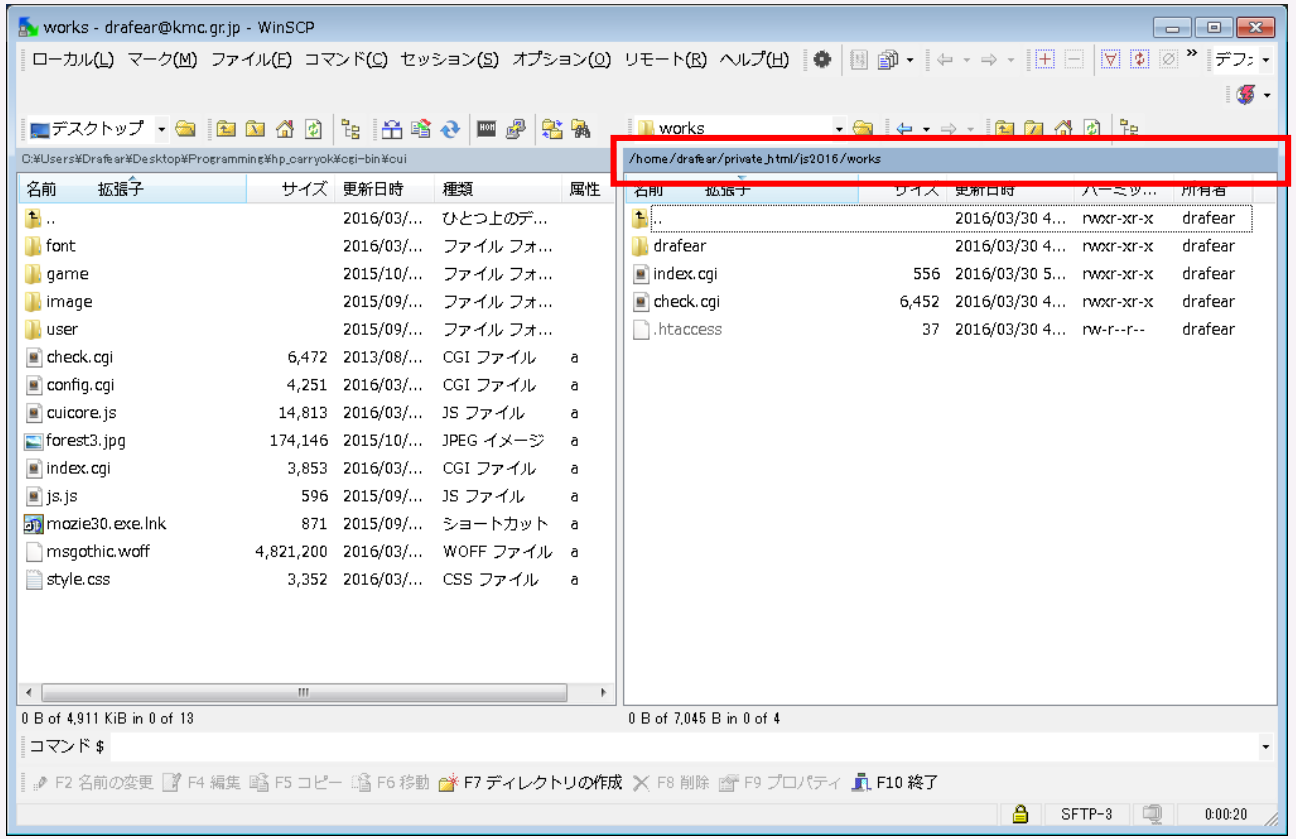
作品をアップロードしてみよう

- 次のように入れてログインして下さい
  - ユーザ名, パスワード は KMC の wiki にログインするときに用いるものを入れて下さい



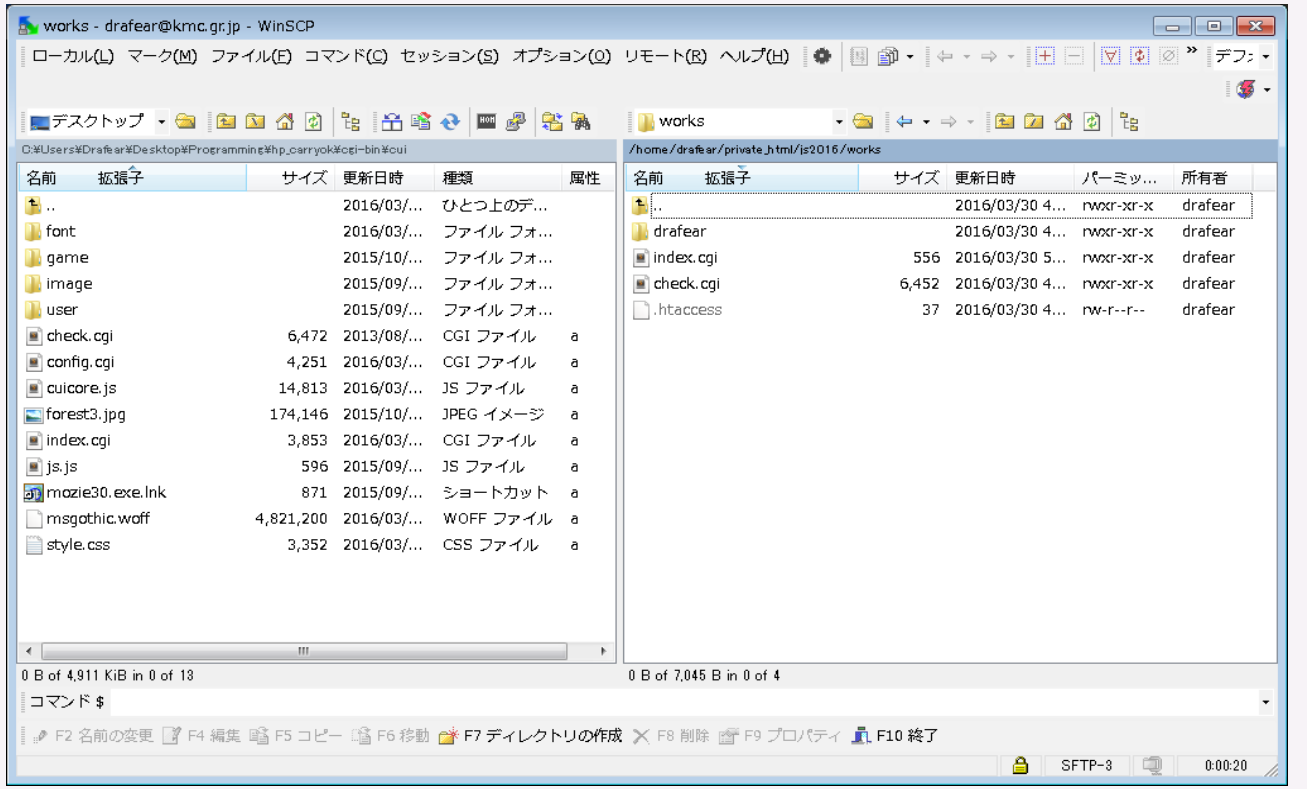
作品をアップロードしてみよう

- ここをダブルクリックして次のパスを入力して下さい
  - /home/drafear/private_html/js2016/works



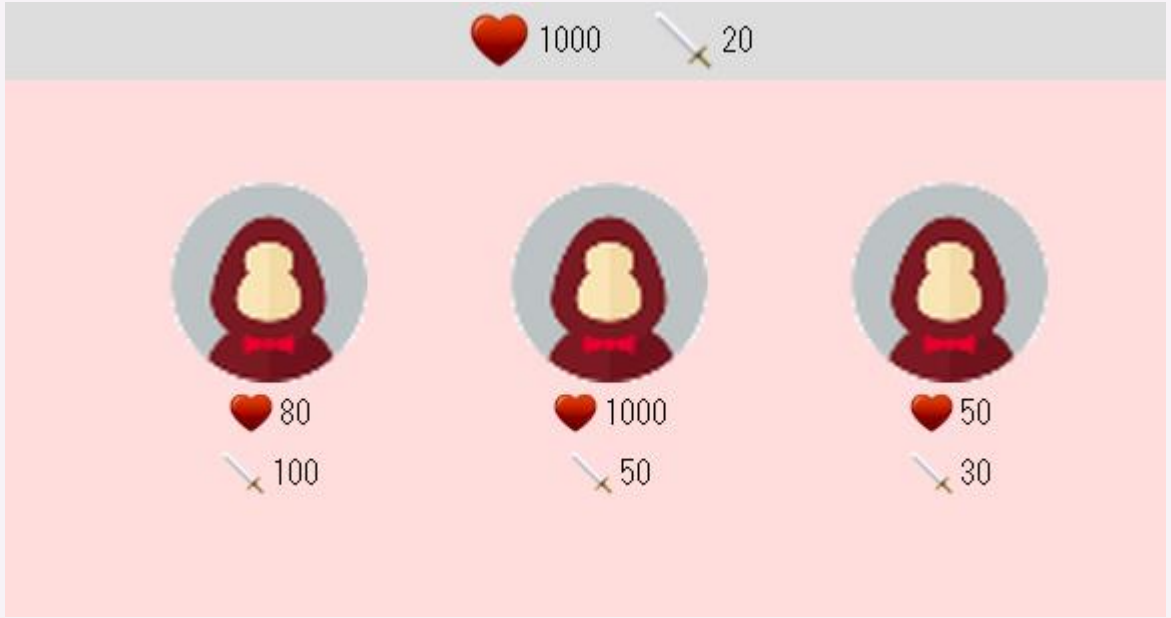
作品をアップロードしてみよう

- そこに自分のidの名前のディレクトリを作って,
  その中に名前 rps でディレクトリを作って,
  その中に index.html と main.js と style.css を入れて下さい



今後の予定

- 5/22(日) 13:00 ～ 16:00
  - JavaScriptの基礎をマスターする
  - CSSでレイアウトを学ぶ
  - ゴリラを倒すゲームを作る



- 5/29(日) 13:00 ～ 16:00
  - 避けゲーを作る

- 6/5(日) 13:00 ～ 16:00
  - 復習回
  - 何かを作ろう