HTML基礎
スタイルシート
Javascript
form

HTML

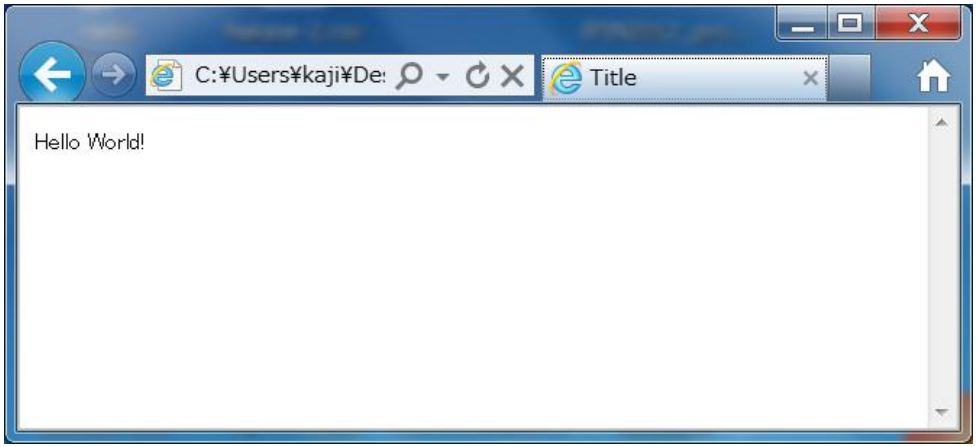
- Webページ表示用の記述形式
- タグ（<xxx> … </xxx>）で構造を表現

- XMLに似ている
  - HTML：文法があまり厳しくない，Web表示専用
  - XML：文法が厳密，タグに独自の意味を持たせる

例

```
<html>
  <head>
    <title>Title</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

1. テキストエディタに張り付け
2. test.htmlという名前で保存
3. ブラウザで開いてみる



主なタグの説明

- <html> HTML全体を囲む
- <head>ページのヘッダ（メタ情報）
- <title>タイトル
- <body>本文

- <p> 段落
-
 改行
- 列挙（リスト）表現
-   ブロック化
- <table> <tr> <td> 表組み

CSS（スタイルシート）

- Webページの表示形式（レイアウトや色）を指定するための記述形式
- タグごとや，独自に定義したクラスごとにスタイルを適用できる

styletest.html

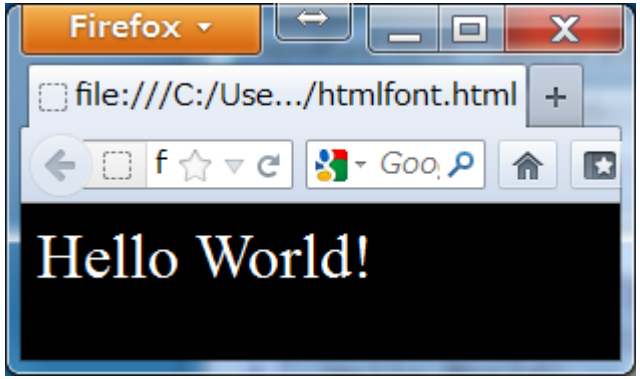
```
<html>
 <head>
 <link href="style.css" rel="stylesheet"
    type="text/css">
</head>
 <body>Hello World! </body>
</html>
```

style.css

```
body {
  background-color: black;
  color: white;
  font-size:30px;
  font-family:'Times New Roman';
}
```

HTMLタグだけでもできるけど．．．

```
<html>
 <body bgcolor="black">
 <font color="white" size="6"
   face="Times New Roman">
 Hello World!
 </font>
 </body>
</html>
```



HTMLのファイルがキタナイ！

表示方法と文書構造の分離

- 前例のHTMLは文書構造と表示方法が混在

- 文書構造をHTMLで記述
- 表示方法は外部ファイルのCSS（スタイルシート）で指定

- メリット：
  - HTML自体がシンプルになり保守性が高まる
  - 複数ページの表示方法を一括で変更できる

CSSの例

styletest.html

```
<html>
 <head>
 <link href="style.css" rel="stylesheet"
    type="text/css">
  <title>Style Test</title>
 </head>
 <body>
  <div class="block1"> ブロック1</div>
  <div class="block2"> ブロック2</div>
  <div class="block3"> ブロック3</div>
</body>
</html>
```

文書構造のまとまりをdivを使って記述

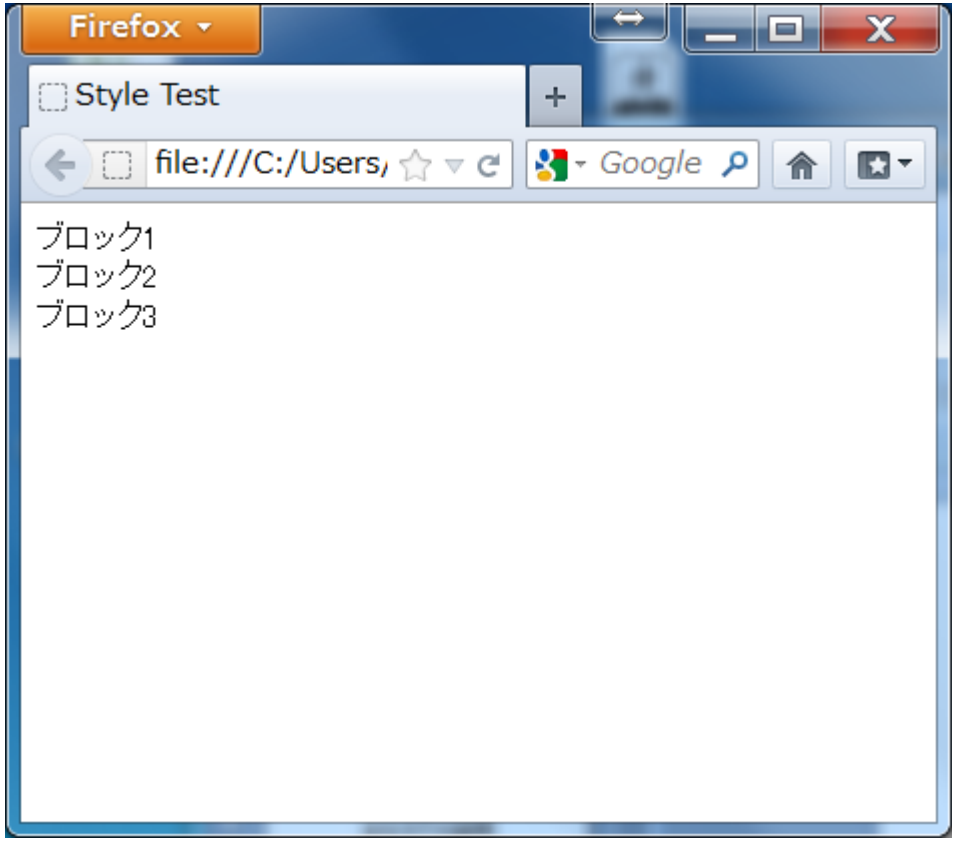
style.css

```
body {
  background-color: black;
  color: white;
}
div.block1{
  background-color: #00FF00;
  width:200px;
  text-align:center;
  float:left;
  color:black;
}
div.block2{
  background-color: #0000FF;
  width:100px;
  text-align:center;
  float:left;
}
div.block3{
  background-color: #FF0000;
  width:300px;
  text-align:center;
  float:clear;
}
```

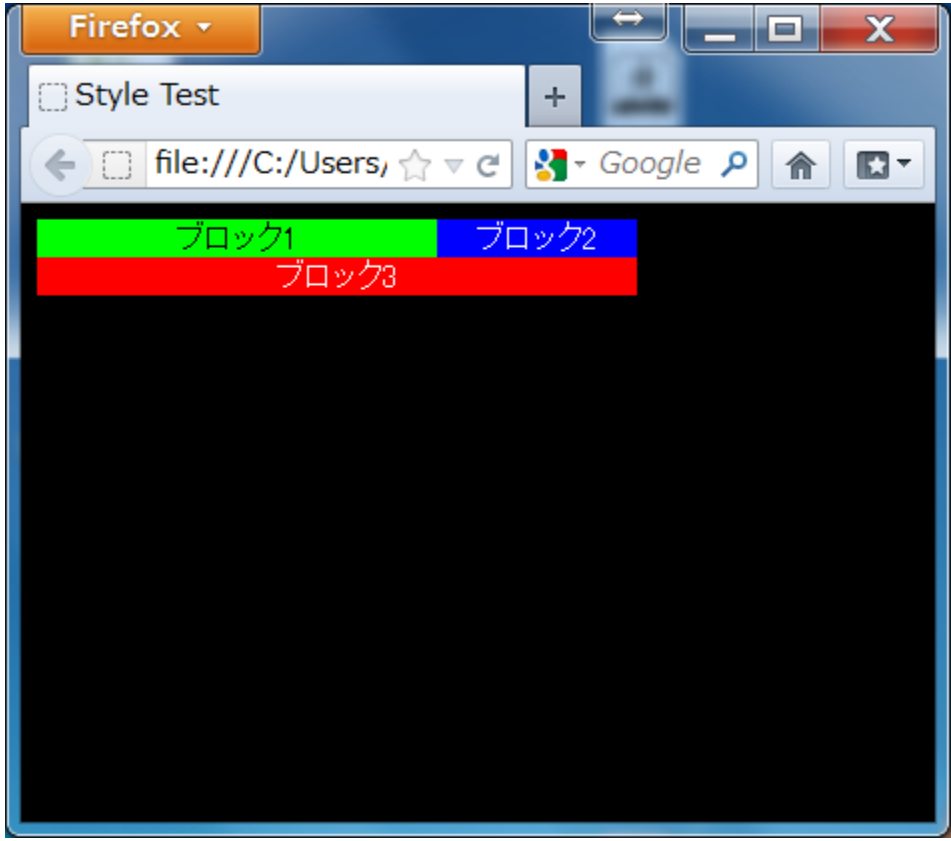
各divのレイアウトや色を指定

CSSの例

スタイルシート適用前



適用後



Javascript

- Webブラウザ上で動作するスクリプト言語
- ページ遷移を必要としない
  ダイナミックなユーザインタフェースの実現
  - ボタンをクリックしたら情報が表示される
  - マウスをドラッグしたら線が描画される

- 文法がJavaに似ている（オブジェクト指向）

例：ボタンを押したらダイアログを出す

```
<html>
 <head>
  <script>
   function buttonProcess(value){
    alert(value);//ダイアログを出す
   }
  </script>
</head>
 <button onclick="buttonProcess('A')">A dayo</button>
 <button onclick="buttonProcess('B')">B desu</button>
 <button onclick="buttonProcess('C')">C kana</button>
</html>
```

Javascriptの関数を定義

ボタン押下で関数呼び出し

マウスがある領域に入ったら別の領域に情報を出す

```
<html>
 <head>
  <script>
    function mouseOverProcess(){
      var div=document.getElementById("infobox");
      div.innerHTML="Mouse<br/>Over!!!";
    }
    function mouseOutProcess(){
      var div=document.getElementById("infobox");
      div.innerHTML="";
    }
  </script>
</head>
 <div id="infobox" style="height:100px;"></div>
 <br/>
 <div style="width:300px;height:300px;background:blue"
    onMouseOver="mouseOverProcess()" onMouseOut="mouseOutProcess()"></div>
</html>
```

マウスが領域に入ったときの処理

タグをid名から取得

内を直接編集

領域から出たときの処理

マウスが領域に入ったとき，出たときに関数を呼ぶ

form

- ユーザからの入力をサーバに送信するタグ

- 用意されているインタフェース
  - 一行テキスト，テキストエリア
  - パスワード
  - チェックボックス
  - ラジオボタン
  - セレクトボックス

例：ユーザの入力文字を送信

test.html

```
<html>
  <body>
    <form action="test.html" method="get">
      id:<input type="text" name="id" value=""/>
      pass:<input type="text" name="pass" value=""/>
      <input type="submit" value="SUBMIT!!"/>
    </form>
  </body>
</html>
```

送信するURL

送信方法（get/post）

＝行入力のテキストボックス

送信ボタン

<form>..</form>に囲まれたinputの内容が送信される

?以降がパラメータ．複数ある場合は&で区切られる

送信後にURLに現れる文字列：　….test.html?id=kaji&pass=hoge

GETとPOST

- GET
  - URLに入力されたパラメータが現れる
  - ちょっとした入力を送信する場合はこちら
- POST
  - URLに入力されたパラメータが現れない
  - 表示されると困る情報がある場合（パスワードなど）
  - 大きなデータを送信する場合（画像ファイルなど）

Servletでパラメータを受け取る

関数：doGet内で

```
String id=request.getParameter("id");
System.out.println(id);
```

まとめ

- HTMLでは文書構造のみを記述
- CSSでレイアウトや見栄えを記述
- Javascriptで動的なUIを実現
- formを介してServletと情報をやり取り