中学校技術における
プログラミングの基礎と応用

（プログラミング言語）

プログラミング言語とは

- 言語の役割を「コミュニケーションのための道具」としてとらえた場合，自然言語と人工言語に大別される。
  - 自然言語は人間同士のコミュニケーションのための道具として用いられるものでる。（日本語，英語，ドイツ語など）
  - 人工言語は人間とコンピュータのコミュニケーションのための道具であり，これはプログラミング言語(programming language)またはプログラム言語とよばれる。
- プログラミング言語は人間とコンピュータ間の情報伝達の道具である。
  - 具体的には，コンピュータに仕事を依頼するときに用いられる。
  - 依頼する仕事の内容は主に文書によってコンピュータに伝えられる。このときの文書をプログラム(program)とよぶ。

プログラムとは

- ある仕事の操作手順を特定の言語で記述して得られるもの
  - 青字部分の中を情報技術で用いられる用語に対応させるとつぎのようになる。
    - 仕事 → 仕様(specification)
    - 操作手順 → アルゴリズム(algorithm)
    - 言語 → プログラミング言語(programming language)
    - 記述する → コーディング(coding)
    - 文書 → プログラム(program)またはコード(code)
  - 置き換えた結果
    - ある仕様のアルゴリズムを特定のプログラミング言語でコーディングして得られるもの

プログラムと言語処理系

- ある言語によって記述されたプログラムは，その言語に対応した言語処理系によって実行される。
  - 翻訳系(コンパイラ:compiler)
  - 解釈実行系(インタープリタ:interpreter)
- コンパイラは，与えられたプログラムを一括してコンピュータが実行可能な形式のプログラムに変換する。
- インタープリタは，与えられたプログラムを(他の形式に変換することなしに)順番に実行する。
- アルゴリズムの考案から，それの言語による記述，プログラムの動作確認と間違い修正(デバッグ:debug)の作業全体をプログラミング(programming)という。

プログラミングの流れ

人　間　　仕様の自然言語による記述
↓
↓（コーディング）
↓
プログラム　　プログラミング言語による記述
↓
↓（翻訳系，解釈実行系）
↓
コンピュータ　　論理回路による実行

プログラミング言語の分類（1）

- 水準による分類
  - 高水準言語
    - 人間によって理解しやすい形式のプログラミング言語
    - C++, Basic, Java, JavaScript, Pythonなど
  - 低水準言語
    - CPUの命令語と対応するプログラミング言語
    - アセンブリ言語，機械語など
- 記述方式による分類
  - テキスト・プログラミング言語
    - プログラムをテキストで記述する言語
  - ビジュアル・プログラミング言語
    - プログラムを視覚的なオブジェクトで記述する言語
    - Scratch, LabVIEW, VISCUITなど

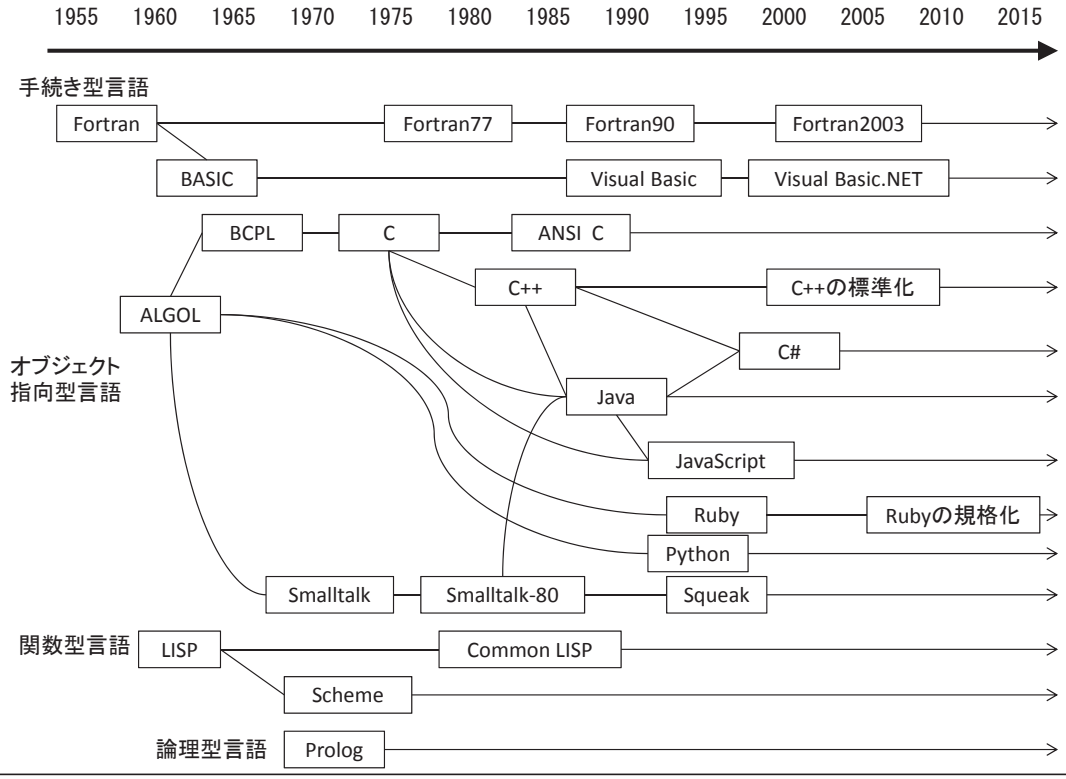
プログラミング言語の分類（2）

- 用途による分類
  - ➢汎用言語
    - 用途を想定せず仕様が定められたプログラミング言語
    - C++, Basic, Java, Pythonなど
  - ➢特定目的言語
    - 用途を特定して仕様が定められたプログラミング言語
    - データベース言語

      SQLなど
    - ページ記述言語

      Postscriptなど

プログラミング言語(抜粋)の略歴



アルゴリズム (algorithm)

- Set of rules or procedures that must be followed in solving a problem. (OXFORD Advanced Learner's Encyclopedic DICTIONARY)
- 日本工業規格(JIS:Japanese Industrial Standard)での定義:
  問題を解くためのものであって，明確に定義され，順序付けられた有限個の規則からなる集合
- どんな問題も，必ず解決する手順が必要となる。
  この手順が，「アルゴリズム」である
- 日本語訳:「算法」（あまり定着していない）

アルゴリズムの表記方法（1）

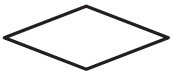
- 一般的な文書
- 図を用いる表記

  フローチャート（流れ図）
  - 長方形の中に処理を書き込んだ要素
  - ひし形の中に判断分岐条件を書き込んだ要素
  - 処理や判断の要素を結ぶ線と矢印
  - 処理の開始と終了などを示す円形の端子

  木構造チャート　PAD（Problem Analysis Diagrams)
  - 構造的プログラムの考え方に沿った流れ図
  - 連続，選択，反復の構造が明確

PAD：Problem Analysis Diagram（直訳：問題解析図）
構造化チャートの一種　（日立製作所の二村良彦氏が開発）

フローチャートの記号（JISX0121）

記号と名称	意味	記号と名称	意味
 端子	プログラムの開始，終了または外部からの入口，出口を表わす。	 定義済み処理	別の場所で定義された処理を表わす。
 準備	初期値設定など，事前の準備を行う。	 ループ始端	繰り返し処理の始まりを表わす。終わりと同じ名前を記述する。
 データ	媒体を指定しないデータからの入力や出力を表わす。	 ループ終端	繰り返し処理の終わりを表わす。始まりと同じ名前を記述する。
 書類	人間が読める媒体上のデータであり，プリンタ出力や帳票などを表わす。	 判断	記号の中に記述された条件に従って，処理を分岐する。
 表示	ディスプレイなどの表示デバイスに表示されるデータを表わす。	線	データや制御の流れを表わす。流れの向きを明示するときは矢印を付ける。
 処理	演算や転送など任意の種類の処理を表わす。	 結合子	流れ図の別の場所への出口，または別の場所からの入口を表わす。対応する結合子には同じ名前を付ける。

アルゴリズムの表記方法（2）

- UML（統一モデリング言語，Unified Modeling Language）
  - オブジェクト指向の分析や設計においてシステムをモデル化するための図法を規定した言語
  - UML 1.4.2　　ISO/IEC 19501:2005
  - UML 2.4.1　　ISO/IEC 19505-1:2012
    ISO/IEC 19505-2:2012
- UML図の種類
  - 構造図（静的構造を表現）

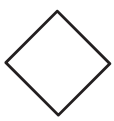
    クラス図，コンポーネント図，複合構造図，オブジェクト図，パッケージ図，プロファイル図
  - 振る舞い図（動的構造を表現）

    アクティビティ図，コミュニケーション図，インタラクション図，シーケンス図，ステートマシン図，タイミング図，ユースケース図

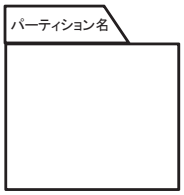
アクティビティ図

- 連続する「実行」の遷移を表現する図
- ある事象の開始から終了までの機能を実行される順序に従って記述
- フローチャートで簡潔に記述できない処理であるオブジェクト志向，並列処理，例外処理（割込処理等），状態遷移，メッセージ処理等が記述可能

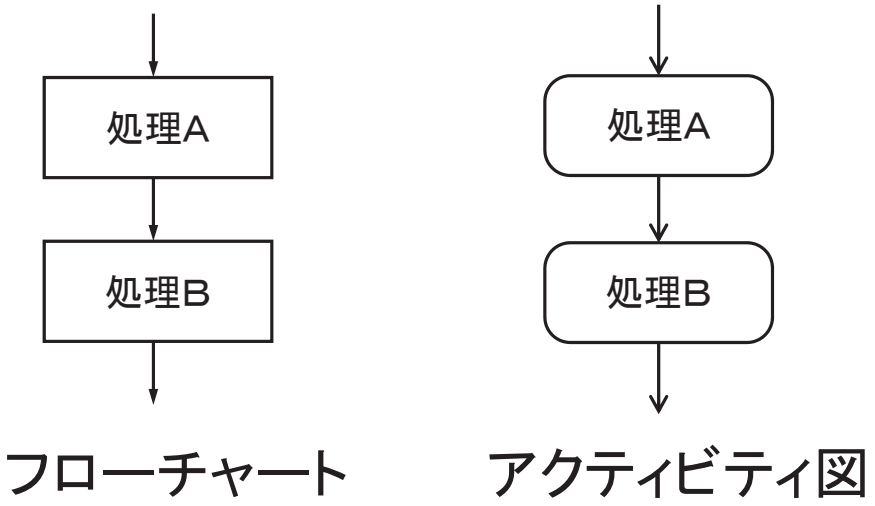
アクティビティ図の構成要素（抜粋1）

要素	図形	説　明
初期ノード		処理の開始を表す。
終了ノード		処理の終了を表す。
アクションノード		処理を表す。
デシジョンノード マージノード		条件によるフロー分岐（デシジョンノード）を表す。 複数のフローの合流（マージノード）を表す。

アクティビティ図の構成要素（抜粋2）

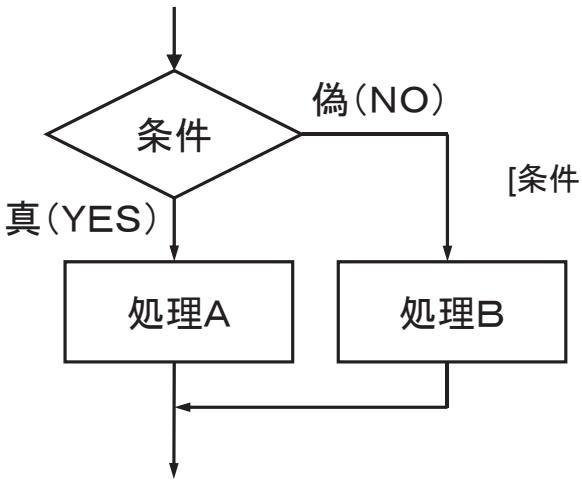
要素	表示形式	意味
フォークノード ジョインノード		複数のフローが非同期に実行される（フォークノード）ことを表す。 複数の非同期処理が終了する（ジョインノード）ことを表す。
パーティション		処理が行われる区画のために，パーティション名を表記する。 区画の中に，アクティビティ図の一部を記載する。
遷移		状態の遷移を示す。

基本構造の記述例 （連続）

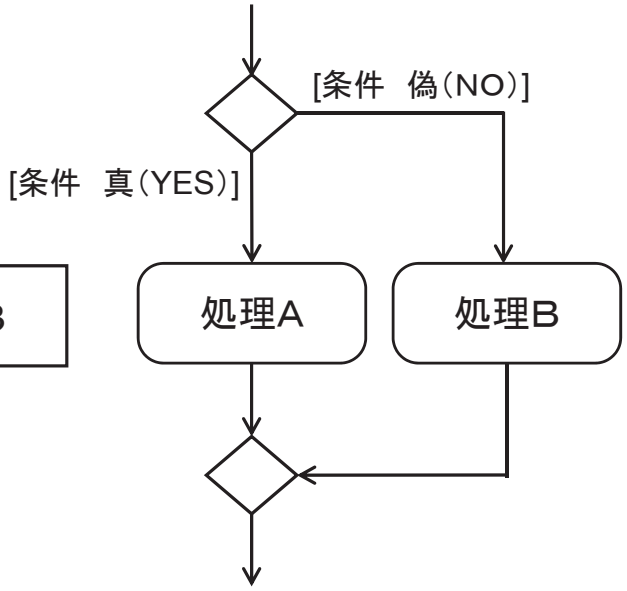


フローチャート　　アクティビティ図

基本構造の記述例（選択）

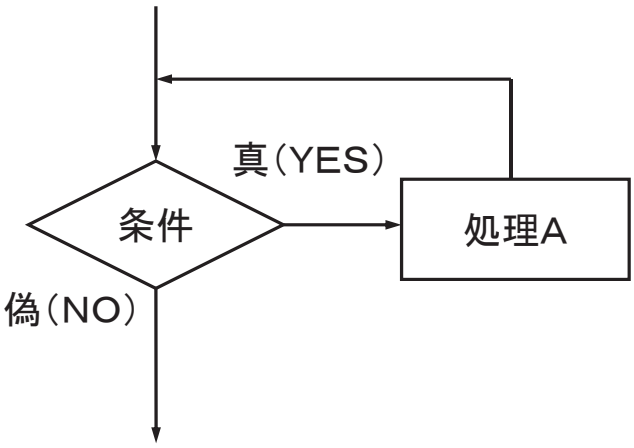


フローチャート

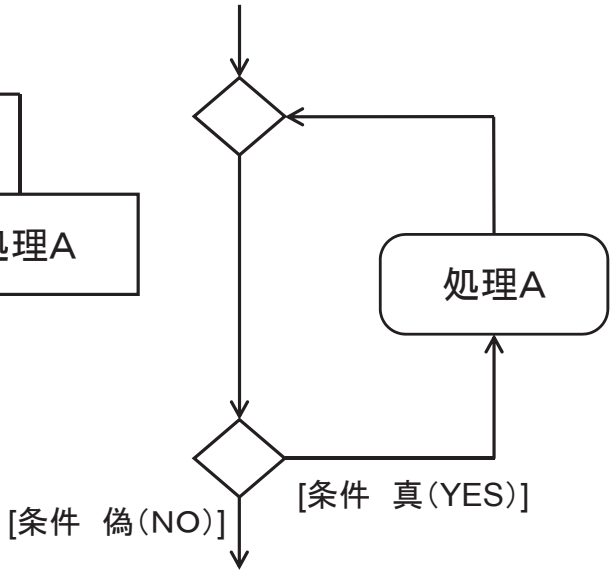


アクティビティ図

基本構造の記述例（反復）



フローチャート



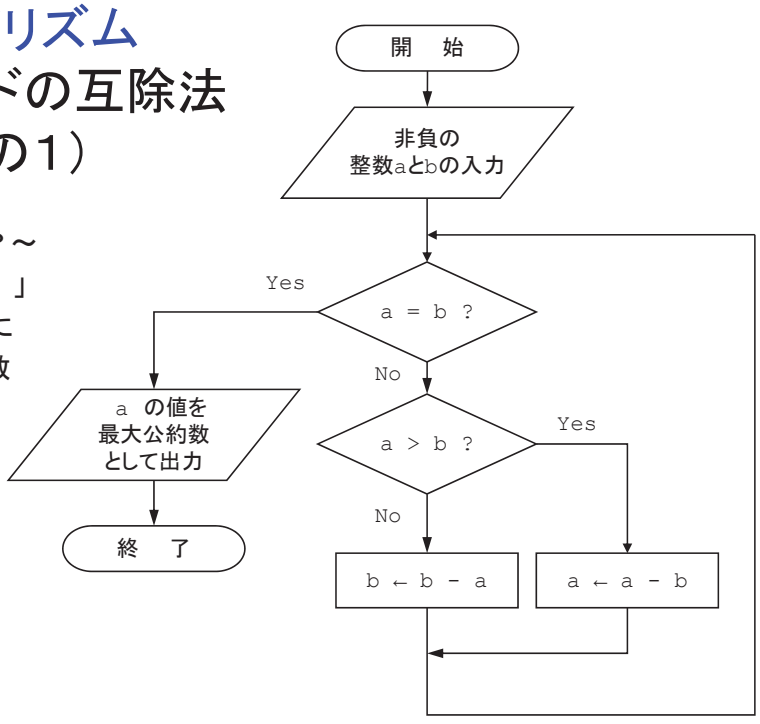
アクティビティ図

アルゴリズムの例

- 最大公約数
  - ユークリッドの互除法
- 線形探索
  - 単純な線形探索
  - 番兵を用いた線形探索
- 整列（ソート）
  - 記憶容量や実行速度に応じて多数のアルゴリズムが提案されている
  - 挿入ソート，交換ソート，選択ソート
  - クイックソート，ヒープソート，マージソート
- フーリエ変換
  - 高速フーリエ変換(FFT)

最大公約数を求めるアルゴリズム
ユークリッドの互除法（その1）

ユークリッド（BC 330？～BC 275）の著作「原論」第7巻命題1から命題3に書かれている最大公約数を計算する方法



ユークリッドの互除法（その1）の計算例

a=2431, b=2002の計算例

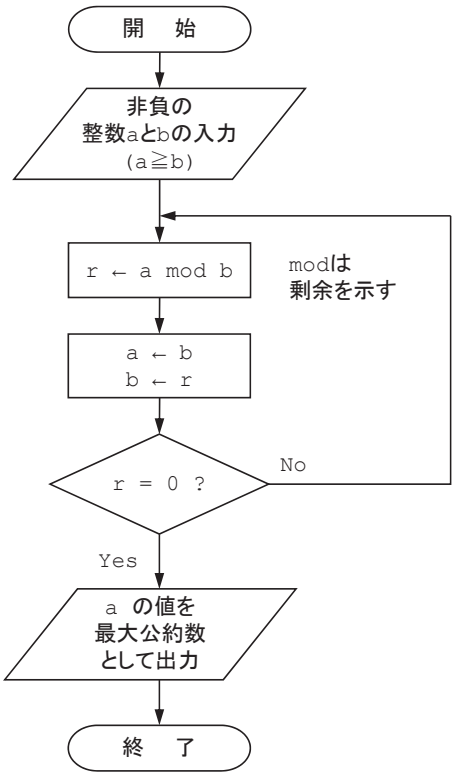
順番	a	b
1	2431	2002
2	429	2002
3	429	1573
4	429	1144
5	429	715
6	429	286
7	143	286
8	143	143

a=2431, b=2002
の最大公約数は，
143となる

最大公約数を求める アルゴリズム ユークリッドの互除法 （その2）

a=2431, b=2002の計算例

a	b	r=a mod b
2431	2002	429
2002	429	286
429	286	143
286	143	0
143	0	---



JavaScriptの概要

- 1995年　Netscape Communications社によって開発
- ECMAScript(エクマスクリプト)として標準化

  ISO/IEC 16262 (1998年)，JIS X 3060 (2000年)
- Webブラウザなどに内蔵されたインタープリタによって実行されるプログラミング言語
  - ➢表示されるコンテンツを操作する。
  - ➢表示されたコンテンツの情報を読み取る。
  - ➢クリックやタップなどの操作によるイベントに対応した処理を行う。

JavaScriptとHTML，CSSの関係

- HTML (HyperText Markup Language)
  - ➢Webページのコンテンツ(文字や画像等)を記述
- CSS (Cascading Style Sheets)
  - ➢Webページのコンテンツに対して，レイアウトやデザインを設定
- JavaScript
  - ➢HTMLやCSSで表現されたコンテンツをリアルタイムで書き換える

HTMLの概要

- HTMLの種類
  - ➢W3C (The World Wide Web Consortium)によって標準化
    - HTML4.01　1999年
    - XHTML1.0　2000年
    - HTML5.0　2012年　　今後の主流
- HTMLタグによる記述

  <!DOCTYPE html>タグで**HTML5.0**の文書型を宣言する

  <html>タグと**</html>**タグで囲む

  <head>タグと**</head>**タグの間に，コンテンツに関する情報を記述する

  <body>タグと**</body>**タグの間に，コンテンツ自体を記述する

HTMLの基本構造

- コンテンツの言語設定

  <html lang="ja">　　※日本語の場合
- コンテンツの文字コード設定

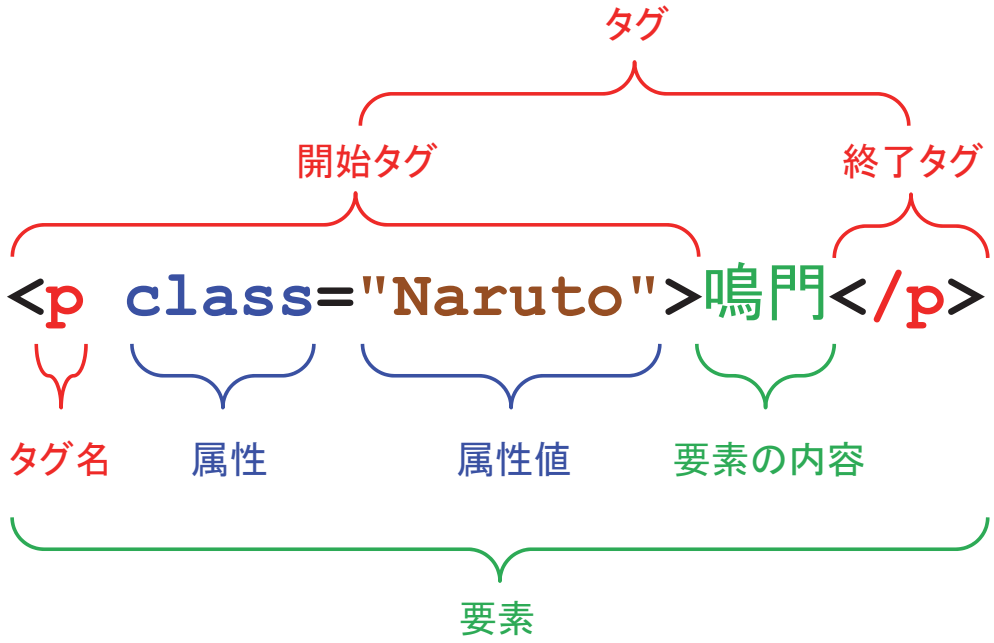
  <meta charset="utf-8">　※多言語対応

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="utf-8">
    </head>
    <body>

    コンテンツ本体の記述

    </body>
</html>
```

HTMLタグの書式



HTMLタグ一覧（抜粋）（1）

HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
<a>	ハイパーリンク	ほぼすべてのタグ	href	URL	リンク先のURL
			target	_blank, _self, _top	リンクを開くウィンドウ
<b>	太字	テキスト，テキストを修飾するタグ			
<blockquote>	引用	ほぼすべてのタグ			
<body>	ドキュメントのコンテンツ	ほぼすべてのタグ			
 	強制改行	なし			
<cite>	作品名	テキスト，テキストを修飾するタグ			
	汎用ブロック	ほぼすべてのタグ			

HTMLタグ一覧（抜粋）（2） HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
	強調	テキスト，テキストを修飾するタグ			
<form>	フォーム	ほぼすべてのタグ	action	URL	データ送信先URL
			method	GET， POST	データ送信方法
<h1>，<h2>，<h3>，<h4>，<h5>，<h6>	見出し	テキスト，テキストを修飾するタグ			
<head>	ドキュメントのメタデータ	<title>，<meta>，<link>，<style>など			
<html>	ルート要素	<head>，<body>			
<i>	斜体	テキスト，テキストを修飾するタグ			

HTMLタグ一覧（抜粋）（3） HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
<img>	画像	なし	alt	テキスト	代替表示テキスト
			height	数値	表示画像の高さ
			src	URL	画像ファイルのURL
			width	数値	表示画像の幅
<input type="checkbox">	チェックボックス	なし	name	テキスト	送信されるキー値
<input type="radio">	ラジオボタン		value	テキスト	送信される値
<input type="text">	テキストフィールド				
<input type="submit">	送信ボタン				

HTMLタグ一覧（抜粋）（4） HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
<label>	フォームのラベル	テキスト，テキストを修飾するタグ	for	テキスト（id名）	対応するコントロールのid属性値
	リスト項目	ほぼすべてのタグ			
<link>	リンク	なし	href	URL	スタイルシートへのURL
			rel	stylesheet	スタイルシートへのリンクであることを示す
			type	text/css	スタイルシートのタイプ
<meta>	メタデータ	なし	charset	文字コードの種類	文字コード
			content	テキスト	キーワードまたは説明の内容
			name	keyword，description	キーワードまたは説明
	序列リスト				

HTMLタグ一覧（抜粋）（5） HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
<option>	プルダウンメニュー項目	テキスト	selected	なし	デフォルトで選択されている
			value	テキスト	選択している場合に送信される値
<p>	段落	テキスト，テキストを修飾するタグ			
<q>	引用	テキスト，テキストを修飾するタグ			
<select>	プルダウンメニュー	<option>，<optgroup>	name	テキスト	送信されるキー値
	汎用インライン	テキスト，テキストを修飾するタグ			
	重要	テキスト，テキストを修飾するタグ			

HTMLタグ一覧（抜粋）（6）

HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
<sub>	下付き文字	テキスト，テキストを修飾するタグ			
<sup>	上付き文字	テキスト，テキストを修飾するタグ			
<table>	表	表関係のタグ			
<td>	表のセル	ほぼすべてのタグ	colspan	数値	列方向に結合するセル数
			rowspan	数値	行方向に結合するセル数
<textarea>	テキストエリア	テキスト	name	テキスト	送信されるキー値
<th>	表の見出しセル	ほぼすべてのタグ	colspan	数値	列方向に結合するセル数
			rowspan	数値	行方向に結合するセル数

HTMLタグ一覧（抜粋）（7）

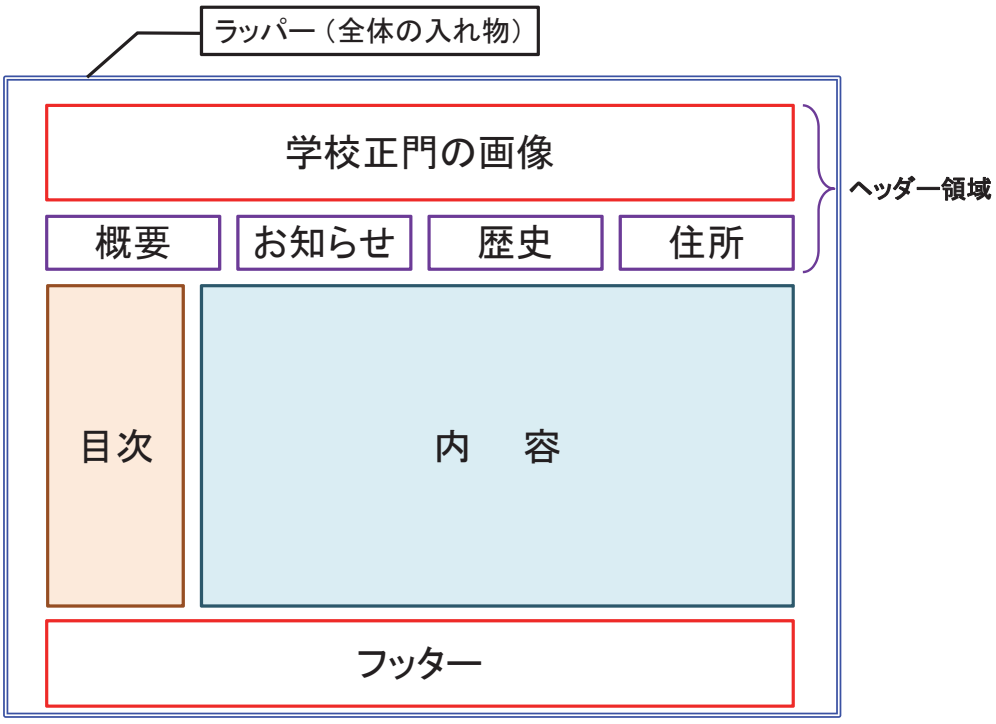
HTML5

<タグ名>	機能	「子」として記述可能な主な要素	主な属性	主な属性値	属性の内容
<title>	ドキュメントのタイトル	テキスト			
<tr>	表の行	<td>, <th>			
	非序列リスト				
<!--	コメント開始				
-->	コメント終了				

特殊な文字 （文字実態参照）

表示される文字	文字実態参照	説　明
<	<	小なり　不等号
>	>	大なり　不等号
&	&	アンパサンド
"	"	二重引用符
␣		半角の空白
¥	¥	円記号

Webコンテンツの枠組み構成例



HTMLの記述例

ファイル名：**html_basic1.html**

行	HTMLドキュメント
1	<!DOCTYPE html>
2	<html lang="ja">
3	<head>
4	<meta charset="utf-8">
5	</head>
6	<body>
7	学校正面の画像
8	概要
9	お知らせ
10	住所
11	歴史
12	目次
13	内容
14	フッター
15	</body>
16	</html>

ボックスの表示方法

- **HTML**の表示内容は，矩形領域毎で管理できる。
- この矩形領域を「ボックス」と呼ぶ。
- ボックスを赤枠で表示したい場合は，
  <head>から**</head>**の間に，以下の指定を追加する。

```
<style>
  * {
    margin: 5px;
    border: 1px solid #ff0000;
  }
</style>
```

ファイル名：**html_basic1_box.html**

HTMLによるGUIコンポーネントの記述例

ファイル名：**html_gui1.html**

html_gui1_box.html

行	HTMLドキュメント
1	<!DOCTYPE html>
2	<html lang="ja">
3	<head>
4	<meta charset="utf-8">
5	<title>学習アンケート</title>
6	</head>
7	<body>
8	<input type="checkbox" checked>アンケート結果をメールで受け取る
9	<input type="checkbox">アンケート結果を手紙で受け取る
10	教　　科：<input type="radio" name="r1" checked>技術・家庭(技術分野)
11	<input type="radio" name="r2">技術・家庭(家庭分野)
12	学　　年：
13	<select>
14	<option value="y1">第１学年</option>
15	<option value="y2">第２学年</option>
16	<option value="y3">第３学年</option>
17	</select>
18	
19	名　　前：<input type="text" size="40" value="姓名のローマ字">
20	パスワード：<input type="password" size="20">
21	教科の学習内容について意見を入力してください。
22	<textarea cols="40" rows="5" ></textarea>
23	
24	<button>送る</button>
25	</body>
26	</html>

CSSの概要

- HTMLで記述したコンテンツのスタイル(体裁)を設定
  - ➢コンテンツの配置を設定
  - ➢隣接するコンテンツの隙間を設定
  - ➢書体の種類やサイズを設定
  - ➢コンテンツの背景に色や画像を設定
- 「CSS」を単に「スタイルシート」と呼ぶこともある。

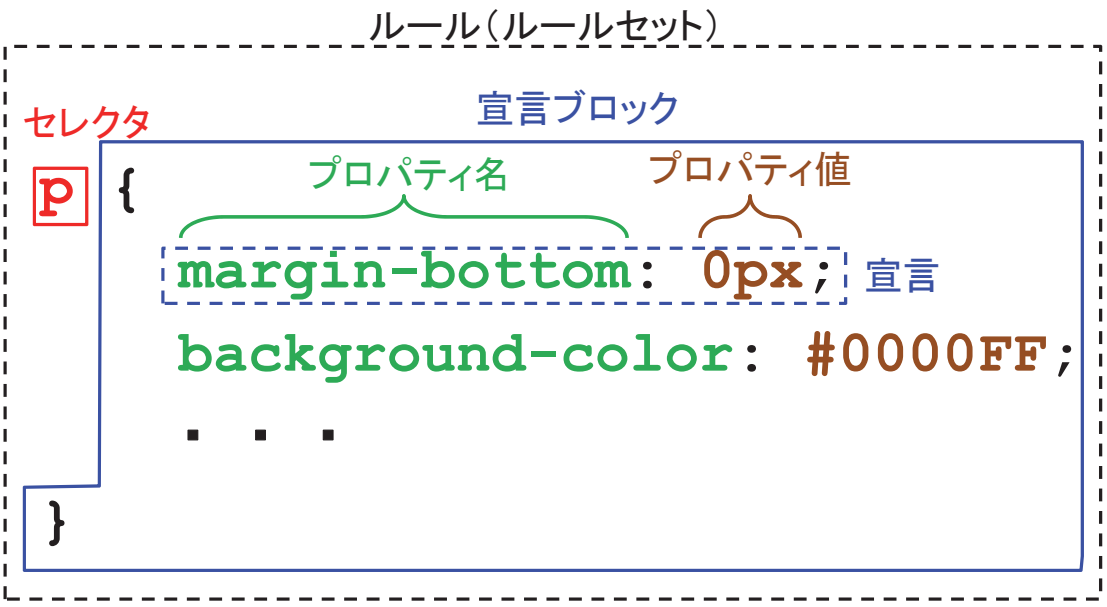
CSSの記述方法

- <head>と</head>タグの間に記述
  - <style>と</style>タグの間に記述
  - 別のファイルから読み込む場合

```
<link href="ファイル名.css"
      rel="stylesheet">
```

- コメント
  - /* で開始し，*/で終了

CSSの書式



セレクタの種類

- HTMLタグ名を条件に選択

  タグ名 { ・・・ }
- すべてのHTMLタグを選択

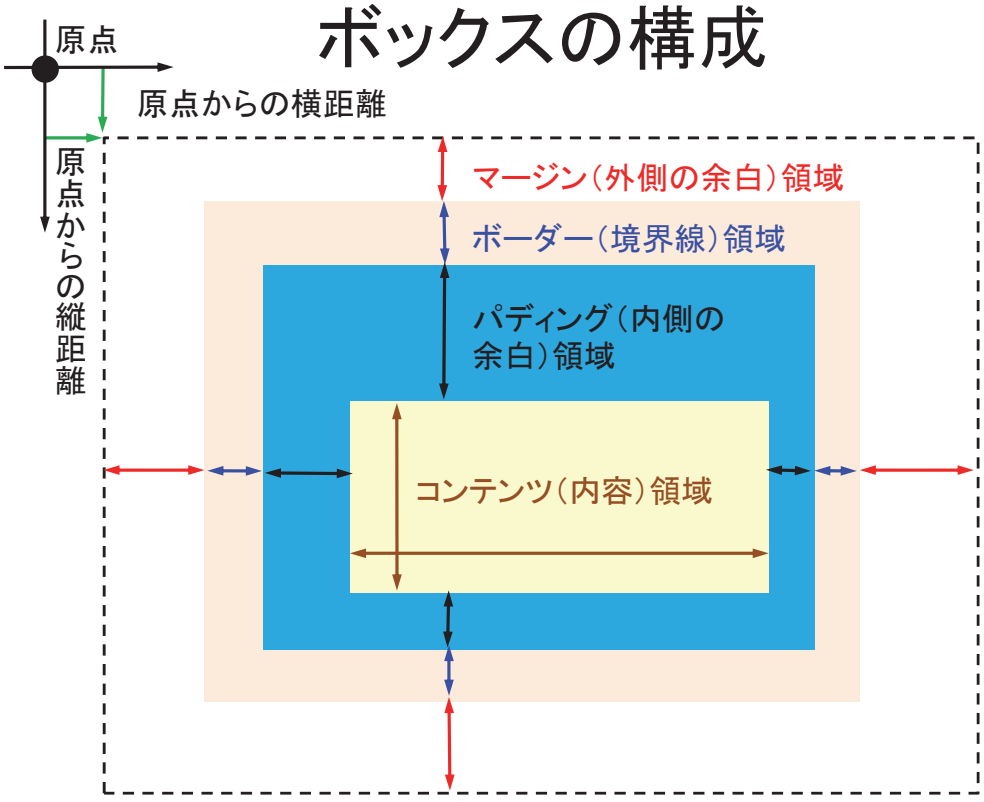
  ＊　（アスタリスク）
- HTMLタグで設定したid属性値を条件に選択

  #id名 { ・・・ }

  タグ名#id名 { ・・・ }
- HTMLタグで設定したclass属性値を条件に選択

  .class名 { ・・・ }

ボックスの構成



プロパティの単位（絶対値）

- 画素（ピクセル） **px**
- インチ（25.4mm） **in**
- 1／72インチ＝1ポイント **pt**
  （約0.35mm）
- 1／6インチ＝12ポイント
  ＝1パイカ **pc**
  （約4.23mm）
- ミリメートル **mm**
- センチメートル **cm**

プロパティの単位（相対値）

- 比率（パーセント） **%**
  スタイル指定しなかった場合の
  ボックスの幅やフォントサイズに対する比率
- 英大文字「M」の幅（エム） **em**
- 英小文字「x」の幅（エックス） **ex**
- 英大文字「M」の幅（ルート・エム） **rem**
  <html>で指定されたフォントサイズ
  多くのブラウザでは，**16pt**となっている。

プロパティの色指定

#RRGGBB

青色の濃度
緑色の濃度
赤色の濃度

16進数表記
00 （薄い）
～
ff （濃い）

光の三原色

カラーネーム
white, black, purple, indigoなど

CSSプロパティー覧（抜粋）（1）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
background-color	#RRGGBB		しない	背景色を指定
background-image	url(URL) none		しない	背景画像を指定
background-repeat	repeat repeat-x repeat-y no-repeat		しない	背景画像の繰り返し方式を指定
border-color	#RRGGBB		しない	ボックス4辺の境界線の色を指定
border-top-color				ボックス上辺の境界線の色を指定
border-right-color				ボックス右辺の境界線の色を指定
border-bottom-color				ボックス下辺の境界線の色を指定
border-left-color				ボックス左辺の境界線の色を指定

CSSプロパティー覧（抜粋）（2）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
border-style	none dotte dashed solid double などの種類名		しない	ボックス4辺の境界線の種類を指定
border-top-style				ボックス上辺の境界線の種類を指定
border-right-style				ボックス右辺の境界線の種類を指定
border-bottom-style				ボックス下辺の境界線の種類を指定
border-left-style				ボックス左辺の境界線の種類を指定
border-width	数値（単位）		しない	ボックス4辺の境界線の太さを指定
border-top-width				ボックス上辺の境界線の太さを指定
border-right-width				ボックス右辺の境界線の太さを指定
border-bottom-width				ボックス下辺の境界線の太さを指定
border-left-width				ボックス左辺の境界線の太さを指定

CSSプロパティー覧（抜粋）（3）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
border	数値（太さ），種類名，#RRGGBB を空白で区切って記述		しない	ボックス4辺の境界線の太さ，種類，色を指定
border-top				ボックス上辺の境界線の太さ，種類，色を指定
border-right				ボックス右辺の境界線の太さ，種類，色さを指定
border-bottom				ボックス下辺の境界線の太さ，種類，色を指定
border-left				ボックス左辺の境界線の太さ，種類，色を指定
border-collapse	collapse separate	表に関連する要素	する	表の罫線の描画方式を指定
clear	none left right both	ブロックレベル要素	しない	テキストの回り込みを解除
color	#RRGGBB		する	テキストの色を指定

CSSプロパティー覧（抜粋）（4）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
display	**none** **inline** **block** **list-item** **inline-block** **など**		しない	ボックスの表示状態を変更
float	**left** **right** **none**	すべての要素	しない	テキストの回り込みを設定
font-family	**sans-serif** **serif** **monospace** など		する	書体を指定
font-size	値（単位） 数値（%） **small** **medium** **large** など		する	書体の大きさを指定

CSSプロパティー覧（抜粋）（5）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
font-style	**normal** **italic** **oblique**		する	書体の修飾を指定
font-weight	数値		する	書体の指定
height	数値（単位） 数値（%） **auto**	一部のインラインレベル要素を除く	しない	ボックスの高さを指定
left	数値	一部のインラインレベル要素を除く	しない	ボックスの原点からの横距離を指定
line-height	**normal** 数値 数値（単位） 数値（%）		する	行の送り幅を指定
list-style-type	**disc** **circle** **square** **none** など	リスト項目 ** **	する	箇条書きの先頭に表示する記号を指定

CSSプロパティー覧（抜粋）（6）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
margin	上，右，下，左の各余白幅を空白で区切って記述	一部の表に関係する要素を除く	しない	ボックス4辺の外側の余白幅を指定
margin-top	数値（単位） 数値（%） auto			ボックス上辺の外側の余白幅を指定
margin-right				ボックス右辺の外側の余白幅を指定
margin-bottom				ボックス下辺の外側の余白幅を指定
margin-left				ボックス左辺の外側の余白幅を指定
overflow	visible hidden scroll auto	ブロックレベル要素	しない	ボックスに要素の内容が表示しけれないときの表示を指定
outline	数値（太さ），種類，#RRGGBBを空白で区切って記述		しない	ボックスの輪郭線の太さ，種類，色を指定 ※Internet Explorerでは無効

CSSプロパティー覧（抜粋）（7）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
padding	上，右，下，左の各余白幅を空白で区切って記述	一部の表に関係する要素を除く	しない	ボックス4辺の内側の余白幅を指定
padding-top	数値（単位） 数値（%）			ボックス上辺の内側の余白幅を指定
padding-right				ボックス右辺の内側の余白幅を指定
padding-bottom				ボックス下辺の内側の余白幅を指定
padding-left				ボックス左辺の内側の余白幅を指定
position	absolute fixed relative	一部のインラインレベル要素を除く	しない	ボックスの位置の指定方式を指定

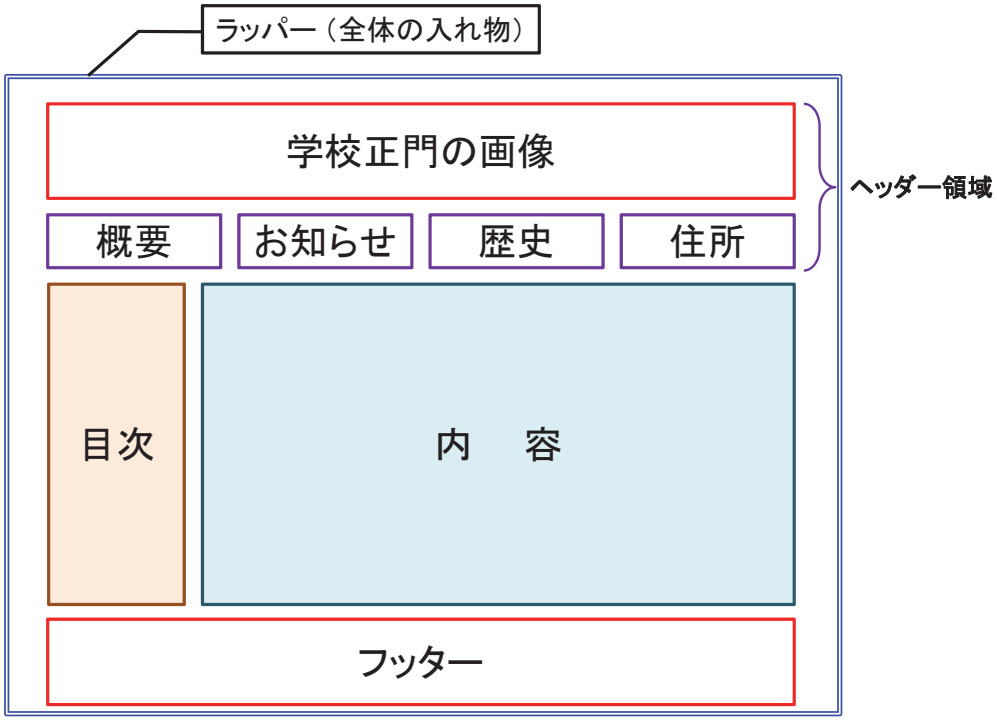
CSSプロパティ一覧（抜粋）（8）CSS3

プロパティ名	プロパティ値	適用可能な要素	継承	機能
text-align	left right center justify	ブロックレベル要素	する	テキストの整列位置を指定
text-decoration	none underline overline line-through **など**		しない	テキストの装飾を指定
top	数値	一部のインラインレベル要素を除く	しない	ボックスの原点からの縦距離を指定
vertical-align	top middle bottom など	インラインレベル要素または表のセル	しない	表のセルに適用した場合，テキストの縦方向の整列位置を指定
width	数値（単位） 数値（%） auto	一部のインラインレベル要素を除く	しない	ボックスの幅を指定

ボックス4辺のプロパティ値の指定方法

- 設定値が1つの場合　⇒「上下左右」
- 設定値が2つの場合　⇒「上下」,「左右」
- 設定値が3つの場合　⇒
  「上」,「左右」,「下」
- 設定値が4つの場合　⇒
  「上」,「右」,「下」,「左」

Webコンテンツの枠組み構成例



ボックスを構成したHTML例

ファイル名：css_basic1.html
css_basic1_box.html

行	HTMLドキュメント
1	<!DOCTYPE html>
2	<html lang="ja">
3	<head>
4	<meta charset="utf-8">
5	<title>〇〇学校へようこそ！</title>
6	</head>
7	<body>
8	
9	
10	学校正面の画像
11	概要
12	お知らせ
13	住所
14	歴史
15	
16	目次
17	内容
18	フッター
19	
20	</body>
21	</html>

ボックスのスタイル設定（1）

ファイル名：**css_basic2.html**

行	HTMLドキュメント
1	<style>
2	#wrapper {
3	width: 600px;
4	height: 400px;
5	margin: 4px;
6	border: solid 2px dimgray;
7	background-color: silver;
8	}
9	#header {
10	height: 120px;
11	margin: 4px;
12	border: solid 2px black;
13	}
14	#index {
15	float: left;
16	width: 180px;
17	height: 200px;
18	margin: 4px;
19	border: solid 2px red;
20	text-align: center;
21	}

ボックスのスタイル設定（2）

行	HTMLドキュメント
22	#contents {
23	float: left;
24	width: 396px;
25	height: 200px;
26	margin: 4px;
27	border: solid 2px chocolate;
28	text-align: center;
29	}
30	#footer {
31	float: left;
32	width: 588px;
33	height: 44px;
34	margin: 4px;
35	border: solid 2px navy;
36	text-align: right;
37	}
38	</style>

メニューのスタイル設定（1）

ファイル名：**css_basic3.html**

行	HTMLドキュメント
1	`#hdimg {`
2	`    height:   60px;`
3	`    margin:    4px;`
4	`    border: solid 2px black;`
5	`    text-align: center;`
6	`    font-size: 32pt;`
7	`}`
8	
9	`.menu {`
10	`    float: left;`
11	`    width:   133px;`
12	`    height:   35px;`
13	`    margin:    5px;`
14	`    border: solid 2px blue;`
15	`    text-align: center;`
16	`    font-size: 16pt;`
17	`}`

JavaScriptの特徴

- Webブラウザの各部品や情報をオブジェクト(Object)として取り扱う
  - ➢オブジェクトの持っている属性値
    - ⇒ プロパティ (Property)
  - ➢オブジェクトに対して処理するプログラム
    - ⇒ メソッド (Method)
  - ➢プロパティや処理の手順を動的に変更可能

JavaScriptプログラムの記述方法

- **<script>**と**</script>**の間に記述
- 別のファイルから読み込む場合
  <script src="ファイル名.js">
  </script>
- **記述場所**
  - **<head>**と**</head>**タグの間に記述
  - **<body>**と**</body>**タグの間に記述
    最下部に記述することが推奨されている。
  - タグ内に記述
    推奨されない。

JavaScriptプログラムのコメント

// で開始し，行末まで
/* で開始し，***/**で終了　　※複数行のコメント

JavaScriptによるプログラミング環境

- JavaScriptインタープリタ内蔵のWebブラウザ
  - Mozilla Firefox
  - Google Chrome
  - Microsoft Internet Explorer
  - Microsoft Edge
- テキストエディタ
  - 文字コードとして「UTF-8」に対応したもの
    - メモ帳(Windows)
    - Tera Pad
    - サクラエディタ

※「**UTF-8**」を指定して保存

Webブラウザ付属の開発ツール

- Mozilla Firefox

  「ツール」⇒「ウェブ開発」⇒「開発ツールを表示」
- Google Chrome

  「・」が縦に3つのアイコン⇒「その他のツール」⇒「デベロッパーツール」
- Microsoft Internet Explorer

  「ツール」⇒「F12 開発者ツール」
- Microsoft Edge

  「詳細」⇒「F12 開発者ツール」

プログラムのデバッグ

- Google Chromeの開発ツール
  - Elements　HTML／CSSの状態を確認
  - Network　ブラウザで発生した通信を走査
  - Sources　JavaScriptのデバッグ
    ブレークポイントの設定や変数の監視・状態表示など
  - Timeline　実行速度を計測
  - Profiles　JavaScriptが使用しているCPUやメモリの情報
  - Application　クッキーなどの内容確認
  - Audits　ページを分析し，最適化のための情報表示
  - Console　エラーメッセージやJavaScriptによる文字出力結果を表示

開発ツールのコンソールへの出力

- JavaScriptのデバッグにおいて，プログラムの実行状態を把握するために，コンソールに文字列を出力する関数

```
console.log(文字列)　　黒字
console.error(文字列)　赤字
console.info(文字列)　!マーク付き黒字
console.warn(文字列)　!マーク付き黄字
```

※文字列の代わりに変数名を記述するとその内容が表示される。

JavaScriptで取り扱えるオブジェクト（1）

ナビゲータ・オブジェクト (Navigator Object)

ブラウザが本来持っている部品や情報を扱う。

オブジェクト名	主な機能
navigator	Webブラウザに関する情報 (名称，バージョン等)
screen	ディスプレイに関する情報 (幅，高さ等)
event	マウスやキーボード操作等のイベントに関する処理
window	ウィンドウの処理 (ウィンドウの生成，移動，サイズ変更，スクロール，タイマー等)
location	表示URLに関する制御
link	リンク情報に関する処理
history	履歴情報に関する処理 (前ページ，次ページへの移動等)
document	HTMLコンテンツに関する処理
form	フォームに関する処理
image	画像に関する処理

JavaScriptで取り扱えるオブジェクト（2）

ビルトイン・オブジェクト (Built-in Object)

JavaScript 自身に組み込まれた情報や機能を扱う。

オブジェクト名	主な機能
Date	日付や時間に関する情報
Math	数学関係の定数や数学関数
String	文字列の操作
Array	配列の操作

トップレベル関数 (ビルトイン関数)

文字列の評価，数値への変換，値か否かの判定等

ダイナミックHTML

属性値やStyleの動的な変更等

階層構造しているオブジェクト

- 階層構造をしているオブジェクトは，「.」（ピリオド）で，上から順に区切って記述する。

（例） window オブジェクトの下にある document オブジェクトのプロパティを参照する場合

- **window.document.**プロパティ名
- window オブジェクトは，一番上のオブジェクトのため省略できる。

予約語とグローバル変数・関数

- ECMAScript6の予約語

```
break, case, catch, class, const, continue, debugger,
default, delete, do, else, export, extends, false,
finally, for, function, if, import, in , instanceof,
new, null, return, super, switch, this, throw, true,
try, typeof, var, void, while, with, yield
```

- ECMAScript6の予約語（将来の言語拡張用）

```
await, enum, implemnts, package, protected,
interface, private, public
```

- 定義済みグローバル変数・関数

```
arguments, Array, Boolean, Date, decodeURI,
decodeURIComponent, encodeURI, encodeURIComponent,
Error, eval, EvalError, Function, Infinity, isFinite,
isNaN, JSON, Math, NaN, Number, Object, parseFloat,
parseInt, RangeError, RefferenceError, RegExp, String,
SyntaxError, TypeError, undefined, URIError
```

変数

- 様々な値を記憶しておく「変数」を

  `var 変数名, 変数名, ・・・;`

  と宣言することで利用できる。
  - C言語 や Java のように，データ型を指定して宣言する必要はない。
  - 変数に代入されたデータによって，自動的にその変数のデータ型が決定される。
- 名前のルール
  - 使用できる文字

    英文字(大文字と小文字を区別する)，数字(1文字目を除く)，下線(_)，ドル記号($)
  - 予約語でない
  - グローバル変数名でない，関数名でない

    使用すると，本来の機能が使えなくなる

データ型

- 数値型

  整数，浮動小数点の区別はない
  - `1～9`で始まる整数　　10進数表記
  - `0o`から始まる整数　　8進数表記　※紛らわしいので使わない方がよい。
  - `0x`から始まる整数　　16進数表記
  - 小数点や指数表記を含む数値　　浮動小数点表記
- 文字列型
  - 二重引用符 " で囲まれた文字

    一重引用符 ' を文字として含めることができる。
  - 一重引用符 ' で囲まれた文字

    二重引用符 " を文字として含めることができる。
- 真理値型

  `true`または`false`のみの値をとる

特別な意味をもつ文字

- エスケープシーケンス （Escape sequence)
  - ➢バックスラッシュ ＼ または 円記号 ¥ に続く文字で表す。
  - ➢主なエスケープシーケンス

¥n 改行

¥¥ バックスラッシュ または 円記号

¥' 一重引用符

¥" 二重引用符

¥t タブ文字

¥uXXXX Unicode文字(XXXXは16進数文字コード)

特別な定数

➢**null**

何も設定されていないことを表す。

➢**undefined**

定義されていないことを表す。

➢**NaN**

数値でないことを表す。

➢**Infinity**

無限であることを表す。

有効範囲と変換

- 変数の有効範囲
  - 関数内で宣言するとローカル変数となり，その関数内でのみ利用できる。
  - それ以外の箇所で宣言すると，グローバル変数になり，どこでも利用できる。
- データ型の変換
  - 文字列⇒整数　　parseIntメソッド

（例）
```
var n = parseInt(strn);
```

  - 文字列⇒浮動小数点　　parseFloatメソッド

（例）
```
var x = parseFloat(strx);
```

  - 数値⇒文字列　　toStringメソッド

（例）
```
var str = num.toString(10);
```

※ 10は基数（進数）を示す。

配列（1）

- 配列の宣言方法
  - 要素数を指定した宣言

（例）
```
var x = new Array(10);
```

  - 要素数を指定しない宣言

（例）
```
var x = new Array();
```

※ 必要に応じて要素数を決めてくれる。

  - 要素の値を指定した宣言

（例）
```
var x = new Array("one", "two",
                  "three");
```

または
```
var x = ["one", "two", "three"];
```

配列（2）

- 配列の要素の指定方法
  - ➢要素番号を「添字」と呼ぶ。
  - ➢添字は，0から始まる。
  - ➢添字は，整数値である。

➢配列の連結

➢concatメソッド

（例）　配列名A.concat(配列名B，配列名C)

➢配列要素の連結

➢joinメソッド

（例）　`var s =` 配列名A`.join(":");`

配列（3）

➢配列要素の順番の逆転

➢reverseメソッド

（例）　配列名A`.reverse()`

➢配列要素の整列（ソート）

➢sortメソッド

（例）　**昇順に整列**

```
function cmp(x, y) {
  return (y - x);
}
var s = 配列名A.sort(cmp);
```

オブジェクト

- 配列とオブジェクトの違い
  要素の選択に「キー」を使うこと
- 要素は「キー」と「値」がセットになっている。
- 「キー」と「値」の区切りは「:」を使う。

➢すべてのキーを取得し，配列に格納

➢keysメソッド

（例）　`var k = Object.keys(`オブジェクト名`);`

演算子（1）

- 算術演算子

+	加算，どちらかが文字列型の場合，文字列の連結
-	減算
*	乗算
/	除算
%	剰余(余り)
++	1増加
--	1減少

演算子(2)

- 代入演算子

=	代入	^=	排他的論理和して代入
+=	加算して代入	<<=	左シフトして代入
-=	減算して代入	>>=	右シフトして代入
*=	乗算して代入		※ 符号なし
/=	除算して代入	>>>=	右シフトして代入
%=	剰余をとって代入		※ 符号付き
|=	論理和して代入		
&=	論理積して代入		

演算子(3)

- 関係演算子

>	より大きい	
<	より小さい	
>=	以上	
<=	以下	
==	等しい	※異なるデータ型であっても その値を比較
!=	等しくない	
===	等しい	※データ型と値の両方を比較
!==	等しくない	

演算子(4)

- 論理演算子
  - `||` 論理和 （または）
  - `&&` 論理積 （かつ）
  - `!` 論理否定 （でない）
- ビット演算子
  - `|` ビット毎の論理和
  - `&` ビット毎の論理積
  - `^` ビット毎の排他的論理和
  - `~` ビット毎の論理否定 （1の補数）
- シフト演算子
  - `<<` 左にシフト
  - `>>` 右にシフト ※符号なし
  - `>>>` 右にシフト ※符号付き

演算子(5)

- その他の演算子
  - `new` オブジェクトの生成
  - `delete` オブジェクトの削除
  - `typeof` 変数等のデータ型
  - `in` オブジェクト内に指定した値が含まれていればtrue(真)
- 演算子の優先順位

  括弧 `(　)` で囲むことによって，規定の優先順位を変更できる。

文字列の操作

- 連結　　+
- 比較　　< = > 文字コードの順序に基づく
- 長さ　　変数名.length
- 部分文字列
  - ➢k文字目からj文字目の部分文字列

    変数名.substring(k, j)
  - ➢k文字目から長さn文字の部分文字列

    変数名.substr(k, n)
- 文字列の分割
  - ➢指定した区切り文字（列）を境にして分割し，配列を作成

    var slist = 変数名.split(",");

制御文（1）

- 条件分岐
  - ➢ if文（その1）

```
if （論理式） {
    論理式が「true（真）」のとき実行するプログラム
}
```

  - ➢ if文（その2）

```
if （論理式） {
    論理式が「true（真）」のとき実行するプログラム
} else {
    論理式が「false（偽）」のとき実行するプログラム
}
```

制御文(2)

- 条件分岐
  - if文(その3)

```
if (論理式1) {
    論理式1が「true(真)」のとき実行するプログラム
} else if (論理式2){
    論理式2が「true(真)」のとき実行するプログラム
} else if (論理式3){
    論理式3が「true(真)」のとき実行するプログラム
...
} else {
    すべての論理式が「false(偽)」のとき実行するプログラム
}
```

制御文(3)

- 条件分岐
  - switch文

```
switch (演算式) {
case 値1 :
    演算式の値が値1のとき実行するプログラム
    break;
case 値2 :
    演算式の値が値2のとき実行するプログラム
    break;
...
default :
    演算式の値がどの値とも異なるとき実行するプログラム
}
```

制御文(4)

- 繰り返し
  - while文

```
while (論理式) {
```

論理式が「true(真)」の間，実行するプログラム

※ 1回も実行されない場合がある。

```
}
```

  - do-while文

```
do {
```

論理式が「true(真)」の間，実行するプログラム

※ 必ず1回は実行される。

```
} while (論理式);
```

制御文(5)

- 繰り返し
  - for文

```
for (初期化処理; 論理式; 各回の後処理) {
```

論理式が「true(真)」の間，実行するプログラム

※ 1回も実行されない場合がある。

```
}
```

  - for-of文

```
for (変数名 of 配列またはオブジェクト名) {
```

配列またはオブジェクト内の要素が順番に変数に代入され実行するプログラム

※ 1回も実行されない場合がある。

```
}
```

制御文(6)

- ジャンプ
  - ラベル名

    命令文の前に「ラベル名：」を記述する。

  - break文

    switch文と繰り返し文の中で使用できる。

    ラベル名を指定しないと，最も内側の繰り返しを抜け出す。

    ラベル名を指定すると，そのラベル名の命令文の最後にジャンプする。

  - continue文

    繰り返し文の中で使用できる。

    ラベル名を指定しないと，繰り返し処理を中断し，次の繰り返しから処理を行う。

    ラベル名を指定すると，そのラベル名の命令文の次の繰り返しから処理を行う。

制御文(7)

- その他
  - with文

```
with （オブジェクト名） {
        実行するプログラム
        ※ オブジェクト名を省略して記述できる。
}
```

関数

- 定義の記述方法

```
function 関数名(仮引数, ・・・) {
    様々な処理
}
```

- 値を返したい場合は，**return**を使用する
- 一般に，関数の定義は**<head>と</head>**の間に記述する。
- 関数の呼び出し方法

```
関数名(実引数, ・・・)
戻り値=関数名(実引数, ・・・)
```

ページの読み込み完了時に実行する処理

- ページの読み込み完了時に実行される関数を定義する。

```
window.onload = function( ) {
    処理内容
}
```

※複数回定義すると，最後に定義した関数のみが実行される。

図形描画

- HTMLコンテンツに配置されたcanvasという描画領域のコンテキストに備えられた様々なメソッドを使って図形を描画する。

（HTMLコンテンツの記述例）

```
<canvas id="e_csv"
      width="400" height="300"></canvas>
```

（JavaScriptの記述例）

```
var cvs = document.getElementById("e_cvs");
var ctx = cvs.getContext("2d");
```

図形描画の考え方

- 「パス(Path)」を設定し，そのパスに沿って，指定した色や太さ，接続形状で線を描画したり，指定した色で塗りつぶししたりする。
- パスに関する主なメソッド
  - **beginPath( )**　　パス開始
  - **moveTo(x, y)**　　パスの座標位置を設定
  - **lineTo(x, y)**　　直線でつながるパスの座標位置を設定
  - **closePath( )**　　開始位置と直線でつながるパス
  - **arc(x, y, r, s, e, cw)**　　円弧のパス
    - **x, y** 中心座標
    - **r**　半径
    - **s**　開始角度（ラジアン）
    - **e**　終了角度（ラジアン）
    - **cw**　描画方向（**true**：反時計回り，**false**：時計回り）

図形描画メソッド（1）

- 太さや色の設定
  - **linewidth**　太さ（単位：px）
  - **strokeStyle**　線の色#RRGGBBまたはrgb(R,G,B)
  - **fillStyle**　塗りつぶし色#RRGGBBまたは rgb(R,G,B)
- 図形の描画
  - **stroke( )**　パスに沿った線分を描画
  - **fill( )**　パスを囲むように塗りつぶし
  - **strokeRect(x, y, w, h)**
    座標**(x, y)**に幅**w**，高さ**h**の矩形の枠線を描画
  - **fillRect(x, y, w, h)**
    座標**(x, y)**に幅**w**，高さ**h**の矩形を塗りつぶし

図形描画メソッド（2）

- 画像の描画
  - **drawImage(img, x, y)**　画像の描画
    - **img　画像オブジェクト**
    - **x, y** 画像を描画する座標

      （例）

```
var img = new Image( );
img.src = "画像のファイル名(URLを含んでもよい。)";
ctx.drawImage(img, 100, 150);
```

  - **putImageData(gd, x, y)**
    画素データの集合(**ImageData**オブジェクト)の描画
    - **gd**　**ImageDataオブジェクト**
    - **x, y**　**ImageData**オブジェクトを描画する座標

図形描画メソッド（3）

- コンテキスト（キャンバスの座標系，太さ，色等）の操作
  - **save**　現在のコンテキストをスタックに保存
  - **restore**　スタックからコンテキストを取り出し，設定

- 座標系の変更
  - **scale(kx, ky)**　**x**軸を**kx**倍，**y**軸を**ky**倍
  - **translate(dx, dy)**　**x**軸を**dx**，**y**軸を**dy**移動
  - **rotate(th)**　原点を中心として，**th**ラジアン時計回りに回転

イベント駆動型プログラミング

- 主なイベント発生要因
  - ポインティングデバイスの操作によって発生
    - ✓クリック，ダブルクリック，ポインタの移動等
  - キーボードの操作によって発生
    - ✓キーを押したとき，離したとき等
  - ウィンドウの状態が変化することによって発生
    - ✓読み込み完了，閉じたとき，更新したとき，ページの切り替え時等
    - ✓サイズ変更，フォームの選択や入力，フォーカスされたとき，フォーカスが外れたとき，値の変化したとき，値が変化しつつあるとき等

イベントリスナの設定

- イベントが発生したときに実行する関数をaddEventListenerメソッドを使って設定する。
- 通常，イベントリスナの設定は，ページの読み込み完了時に実行される関数で行う。

```
window.onload = function( ) {
    イベントリスナの設定
}
```

- または，イベントが発生したときに実行する関数をオブジェクトに設定する。

主なイベント（1）

ウィンドウ操作関係

イベント名	イベントの発生時
onload	ページや画像の読み込みが完了したとき
onupload	ウィンドウを閉じたときや他のページに切り替えたとき，ページを更新したとき
onresize	ウィンドウのサイズが変更されたとき
onchange	フォームの要素選択や入力した内容が変更されたとき
onselect	テキストが選択されたとき
onblur	ページやフォームの要素からフォーカスが外れたとき
onfocus	ページやフォームの要素がフォーカスされたとき
onchange	GUIコンポーネントの値が変化したとき
oninput	GUIコンポーネントの値が変化しつつあるとき
onsubmit	フォームを送信しようとしたとき
onreset	フォームがリセットされたとき
onabort	画像等の読み込みを中断したとき
onerror	画像等の読み込み中にエラーが発生したとき

主なイベント（2）

ポインティングデバイス操作関係

イベント名	イベントの発生時
onclick	要素やリンクをクリックしたとき
ondblclick	要素をダブルクリックしたとき
onmouseout	ポインタが要素の上から離れていくとき
onmouseover	ポインタが要素の上に移動してきたとき
onmouseup	デバイスのボタンが要素の上で放されたとき
onmousedown	デバイスのボタンが要素の上で押し下げられたとき
onmousemove	ポインタが要素の上を移動しているとき

キーボード操作関係

イベント名	イベントの発生時
onkeyup	押していたキーをあげたとき
onkeydown	キーを押したとき
onkeypress	キーを押しているとき

イベントオブジェクト

- イベントによって起動された関数の引数にわたされるもの
- イベントに関する各種情報を対応するメソッドを使って取得する。

（ポインティングデバイスによるイベントに関わる主なメソッド）

メソッド名	イベントの発生時
pageX	HTMLコンテンツの左上を原点とするX座標値(単位：px)
pageY	HTMLコンテンツの左上を原点とするY座標値(単位：px)
clientX	Webブラウザの描画領域の左上を原点とするX座標値(単位：px)
clientY	Webブラウザの描画領域の左上を原点とするY座標値(単位：px)
screenX	ディスプレイの全体領域の左上を原点とするX座標値(単位：px)
screenY	ディスプレイの全体領域の左上を原点とするY座標値(単位：px)
offsetX	HTML要素の左上を原点とするX座標値(単位：px)
offsetY	HTML要素の左上を原点とするY座標値(単位：px)

プログラムの実行順序

- JavaScriptは，シングルスレッド（同時に一つの処理のみ可能）で動作する。
  ⇒ 並列処理はできない
- キュー（実行する関数の待ち行列）に関数が登録され，順番に一つずつ実行される。
  ⇒ キューに関数が登録されても，必ずしもすぐに実行されるわけではなく，待ち行列の順番通りに関数が実行される。
  ⇒ 指定時間後の処理や指定した周期ごとの処理を行う関数をキューに登録しても，時間通りに処理されず「待たされる」ことが生じる。

タイミングイベント

- 指定した時間が経過した後，指定した関数を「キュー（実行する関数の待ち行列）」に登録する。

（例） `var to_id = setTimeout(関数名， 時間)`
時間の単位:ミリ秒
※ `to_id`は，タイミングイベントIDの値

- 指定した時間ごとに，指定した関数を「キュー（実行する関数の待ち行列）」に登録する。

（例） `var ti_id = setInterval(関数名， 時間)`
時間の単位:ミリ秒
※ `ti_id`は，タイミングイベントIDの値

- 仕掛けたタイミングイベントをキャンセルする。

（例） `clearTimeout(to_id)`
（例） `clearInterval(ti_id)`

地図を表示するWebページ

- 地図データの提供元
  - ➢無償提供
  - ➢日本のみならず世界まで対応
- 地図データの取り扱い
  - ➢地図API (Application Programming Interface)
  - ➢JavaScript地図ライブラリ

無償で利用できる地図データ（例）

- OpenStreetMap
  - ➢不定期に閲覧できなくなる可能性はある。
- 地理院地図(国土地理院)
  - ➢日本国外の地図情報が少ない。
- Google
  - ➢1日当たりの閲覧回数に制限ある。
  - ➢Keyを別途取得する必要がある。
- Yahoo! Open Local Platform
  - ➢1日当たりの閲覧回数に制限ある。
  - ➢IDを取得する必要がある。

地図データを利用するためのAPI

- OpenStreetMap
- 地理院タイル (国土地理院)
- Google Map
- Yahoo! JavaScript Map

JavaScript地図ライブラリ

- leaflet
  様々な地図データを取り扱い可能
- OpenLayers
  様々な地図データを取り扱い可能
- Google Maps API
  基本的には，Google提供の地図データのみ対応

leafletの使い方

- <head>と</head>の間に記述
- スタイルシートの読み込み

```
<link rel="stylesheet" href="https:/
/unpkg.com/leaflet@1.0.3/dist/leafle
t.css">
```

- JavaScriptライブラリの読み込み

```
<script src="https://unpkg.com/leafl
et@1.0.3/dist/leaflet.js">
</script>
```

地図オブジェクトの作成

- ID名で指定したHTML要素に対して，地図オブジェクトを作成する。

（例）

【JavaScript】

```
var map = L.map("h_map");
```

【HTML】

```
<div id="h_map" style='width: 800px;
height: 600px; border: solid 2px #808
080;'> </div>
```

※ Lは，leafletのオブジェクト名で変更できない。

※ スタイルシートの設定は必須

地図データのレイヤ追加

- 地図に表示内容を追加するメソッド

  `addTo(地図オブジェクト)`

- OSM(Open Street Map)を使う場合

```
var tileLayer_osm = L.tileLayer(
  "http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
  {attribution: "&copy; OpenStreetMap</a>"}
);
tileLayer_osm.addTo(map);
```

- 国土地理院の地図データを使う場合

```
var tileLayer_gsi = L.tileLayer(
  "http://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.png",
  {attribution: "国土地理院"}
);
tileLayer_gsi.addTo(map);
```

地図の表示

- 表示する地図の中心位置を緯度と経度で設定し，指定した表示倍率で地図を表示するメソッド

  `setView([緯度, 経度], 表示倍率)`

  緯度：10進数の角度（北緯＋，南緯－）

  経度：10進数の角度（東経＋，西経－）

  表示倍率：整数　(例：10～15程度)

（例） `map.setView([34.0, 134.6], 14);`

- 表示する地図の中心位置を緯度と経度で設定し地図を表示するメソッド

  `panTo([緯度, 経度])`

マーカの追加

- マーカを追加するメソッド
  `marker([緯度, 経度])`

- ポップアップを表示するメソッド
  `bindpopup("HTMLコンテンツ")`

（例）

```
var marker = L.marker([34.2, 134.6]);
marker.bindPopup("鳴門教育大学");
marker.addTo(map);
```

地図上に図形を描画（1）

- 複数地点を線分で結ぶ。

（例）

```
var points = [
    [第1番目地点の緯度, 第1番目地点の経度],
    ....
    [第n番目地点の緯度, 第n番目地点の経度]
];
var polyline = L.polyline(points, {color: "red"});
polyline.addTo(map);
```

地図上に図形を描画（2）

- 矩形を描画する。

（例）

```
var bounds = [
    [第1番目地点の緯度, 第1番目地点の経度],
    [第2番目地点の緯度, 第2番目地点の経度]
];
var rectangle = L.rectangle(bounds, {color: "red",
weight:1});
rectangle.addTo(map);
```

- 円を描画する。

（例）

```
var circle = L.circle([中心地点の緯度, 中心地点の経度],
{color: "red", fillOpacity:透明度, radius:半径});
circle.addTo(map);
```

地図表示Webページの応用例

- 地域の歴史的な建物
- 地域のバス路線図と時刻表
- 地域の交通乗り換え案内
- 地域の防災マップ（避難所や消防署）
- 修学旅行や遠足の訪問先
- 山の標高や特徴
- 岬の特徴や周辺の特産物