G's ACADEMY TOKYO

function(関数)

functionが理解できると見える世界が変わる

function(関数)

関数 (function)

※サンプル：function01.html

【 関数定義】

関数とは、一つの処理（再利用可能な処理など）をまとめたもので、
関数名だけで、関数定義した処理を実行することが可能です。

```
<script>
      //ブラウザに読み込まれたら処理開始
       let str = “関数をしらない。";
       alert( str );
</script>
```

```
<script>
   //関数定義 (定義はページが読み込まれても実行しません。)
   function test() {
        let str = "関数を知った。";
        alert( str );
   }
  //関数実行（関数名を呼ぶことで実行できます！）
  test();
</script>
```

※関数内外で同名の変数があっても違う変数です。

関数練習１（function）

```
<script>
  //関数定義(読み込まれても実行しない)
  function alt(str){
      alert( str);
  }
   //関数実行（関数名を呼んで実行開始！）
 alt("アラート表示関数！");
</script>
```

alert関数をもっと短くして呼びましょう！

※関数内外で同名の変数があっても違う変数です。

関数 (function)

※サンプル：function02.html

【 引数（Argument）と戻り値（Return value） 】

関数を呼び出す時に「引数」を使って値を渡す事できます。
※関数内の変数は外部から参照できません。

```
<script>
//関数定義
function addition(a, b) {
    const total= a + b;
    return total;
}

//関数実行（ t は戻り値を受け取ります）
const val= addition(10, 20);
console.log( val );

</script>
```

関数定義：引数
関数名(第一引数,第二引数...)

関数定義：戻り値
return 変数; で関数の外に値を出す

関数実行：引数
関数名(第一引数,第二引数...)

関数定義：戻り値
return 変数; を関数の外で受け取る

【引数のPOINT】
・引数は少ないほうがいいです。
・関数内の変数は外部から参照できません。

【練習】関数～これで完璧～

5分

◇仕様：

- “function_ex.html” サンプルコードで既に用意されてます
  先程練習した内容を見本として、同様に関数を定義してください！！

- 足し算の関数定義 //定義→実行(10+10); 結果をconsole.log表示（ここは一緒に）
- 引き算の関数定義 //定義→実行(20-30); 結果をconsole.log表示
- 掛け算の関数定義 //定義→実行(40*50); 結果をconsole.log表示
- 割り算の関数定義 //定義→実行(20/5); 結果をconsole.log

上記を定義後、関数を実行してconsole.logで表示しましょう！！

【練習2】関数～これで完璧～

5分

◇仕様：

- “function_ex2.html” サンプルコードで既に用意されてます
  先程練習した内容を見本として、同様に関数を定義してください！！

- 足し算の関数定義 //定義→実行(10+10); 結果をreturn、変数aで受け取る
- 引き算の関数定義 //定義→実行(20-30); 結果をreturn、変数bで受け取る
- 掛け算の関数定義 //定義→実行(40*50); 結果をreturn、変数cで受け取る
- 割り算の関数定義 //定義→実行(20/5); 結果をreturn、変数dで受け取る
- 変数a,b,c,d を足し算し、h1要素へ表示する！！

乱数生成を関数化
問題

関数２（function）

【 汎用的な、乱数発生関数を作って見ましょう！】

rand関数を作成して、引数に「数値幅」、戻り値「1~数値幅内の数値」
console.logにて表示して確認します。

```
<script>
//関数定義しましょう！

 const num= Math.ceil( Math.random() * 5 );

</script>
```

「数値幅：１～５」

関数化してみましょう！

関数

― selectbox 汎用的使用例―

【 例）反復処理の関数化　　select_create.html】

```
<select id="year"></select>
<select id="month"></select>
<select id="date"></select>
<!– ここにセレクトボックスの値が生成されます -->

<script>
function ymd( start, end, id ){  //関数宣言(開始値, 終了値, 表示する場所)
    let str = "";
    for( let i=start; i<=end; i++ ) {
       str +=  '<option>' + i + '</option>';
    }
    $(id).html(str); //表示して終わるのでreturnはしない
}
//関数実行
ymd(2011, 2099, "#year");
ymd(1, 12, "#month");
ymd(1, 31, "#date");
```

Clickイベントのfunction()？

補足: Clickイベントのfunction?

```
<script>

 //関数定義
 function alt() {
       alt("アラート表示関数！");
 }
 //Clickイベントに関数名で紐づけ
 $("#btn").on("click",alt);

 //これまでの書き方：クリックイベント内に処理を記述。
 $("#btn").on("click",function(){
       alt("アラート表示関数！");
 });
```

別けて書ける！！