JavaScript

PART I. FUNDAMENTAL

Ths. Hồ Đức Lĩnh
Mail: duclinh47th@gmail.com
Phone: 0905 28 68 87

Tools

Visual Studio Code

Sublime Text

Content

- **Chapter 1: JavaScript Fundamental**
  - Introduction: Variable, If-else statements, Loops,
  - String, Date time, Math
  - Arrays
- **Chapter 2: Function**
- **Chapter 3: Objects, The Document Object Model (DOM) and Event**
- **Chapter 4: Forms and Validation**
- **Chapter 5: Object-Oriented Programming**


npm
TS
REQUIRE·JS
BABEL
JS
JS
A

Why Study JavaScript?

JavaScript is one of the **3 languages** all web developers **must** learn:

- **HTML** to define the content of web pages
- **CSS** to specify the layout of web pages
- **JavaScript** to program the behavior of web pages

Where JavaScript is used

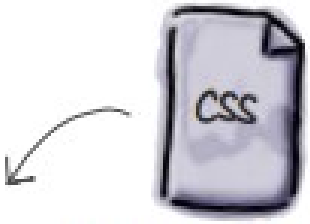
- Client-side validation
- Dynamic drop-down menus
- Displaying data and time
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

What JavaScript CANNOT Do?

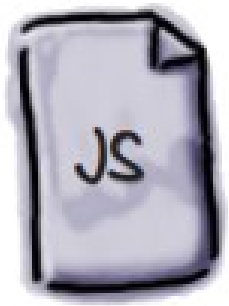
- Remember that JavaScript is a client-side program that you downloaded from a server, and run inside the browser of your (client) machine.
- What to stop someone from writing a JavaScript that wipes out your hard disk, or triggers a denial-of-service attack to another server? As a result, for security purpose,
  - It cannot read file from the client's machine.
  - It can only connect to the server that it come from. It can read file from the server that it come from. It cannot write file into the server machine.
  - It cannot connect to another server.
  - It cannot close a window that it does not open.

The way JavaScript works

You already know we use HTML, or Hypertext Markup Language, to specify all the content of your pages along with their structure, like paragraphs, headings and sections.



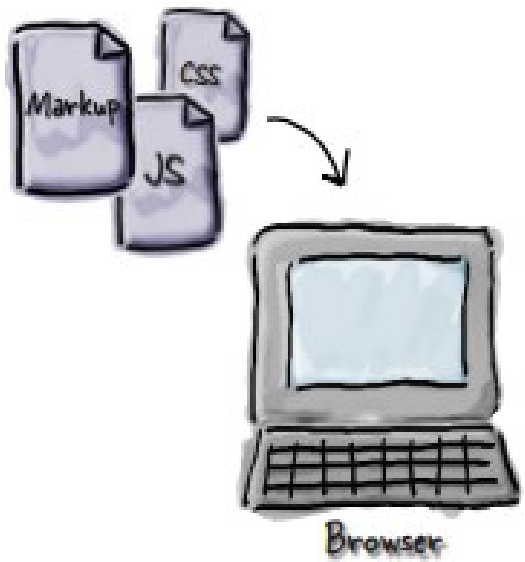
And you already know that we use CSS, or Cascading Style Sheets, to specify how the HTML is presented...the colors, fonts, borders, margins, and the layout of your page. CSS gives you **style**, and it does it in a way that is separate from the structure of the page.

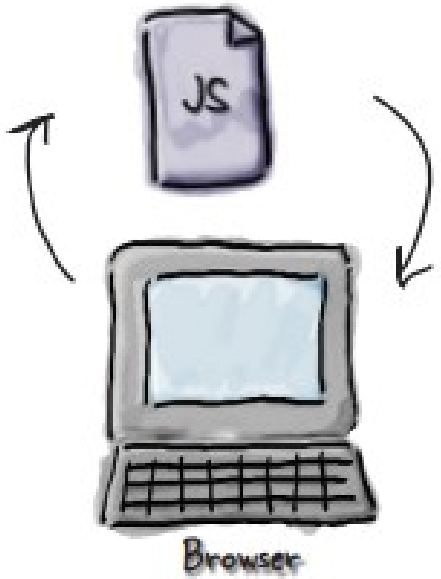


So let's introduce JavaScript, HTML & CSS's computational cousin. JavaScript lets you create **behavior** in your web pages. Need to react when a user clicks on your "On Sale for the next 30 seconds!" button? Double check your user's form input on the fly? Grab some tweets from Twitter and display them? Or how about play a game? Look to JavaScript. JavaScript gives you a way to add programming to your page so that you can compute, react, draw, communicate, alert, alter, update, change, and we could go on... anything dynamic, that's JavaScript in action.

How you're going to write JavaScript

```
<html>
<head>
<title>Icecream</title>
<script>
 var x = 49;
</script>
<body>
<h1>Icecream Flavors</h1>
<h2><em>49 flavors</em></h2>
<p>All your favorite
flavors!</p>
</body>
</html>
```





Writing ❶

Loading ❷

Executing ❸

How you're going to write JavaScript

Writing

1

You create your page just like you always do, with HTML content and CSS style. And you also include JavaScript in your page. As you'll see, just like HTML and CSS, you can put everything together in one file, or you can place JavaScript in its own file, to be included in your page.

We'll talk about the best way in a bit...

Loading

2

Point your browser to your page, just as you always do. When the browser sees code, it starts parsing it immediately, getting ready to execute it. Note that like HTML and CSS, if the browser sees errors in your code, it will do its best to keep moving and reading more JavaScript, HTML and CSS. The last thing it wants to do is not be able to give the user a page to see.

For future reference, the browser also builds an "object model" of your HTML page that JavaScript can make use of. Put that in the back of your brain, we'll come back to it later...

Executing

3

The browser starts executing your code as soon as it encounters it in your page, and continues executing it for the lifetime of your page. Unlike early versions of JavaScript, today's JavaScript is a powerhouse, using advanced compilation techniques to execute your code at nearly the same speed as many native programming languages.

JavaScript Syntax

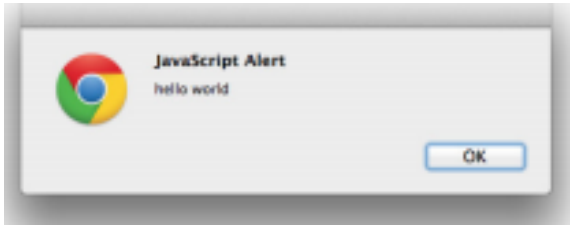
- In HTML, JavaScript code **must be inserted** between **<script>** and **</script>** tags.

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>My Web Page</title>
<script>
</script>
</head>
```

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>My Web Page</title>
<script>
  alert('hello world!');
</script>
</head>
```

```
<script>
  // Your JavaScript programming codes here!
</script>
```

Example: displays the message "Hello world!"



Add JavaScript to a Page

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>My Web Page</title>
<script src="navigation.js"></script>
</head>
```

JavaScript Variables and Operator

```
var firstName = 'Peter';
var lastName = 'Parker';
var age = 22;
var isSuperHero = true;
```

```
var score=0,
    highScore=0,
    player='';
```

```
var x, y, z;
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

OPERATOR	WHAT IT DOES	HOW TO USE IT
+	Adds two numbers	5 + 25
-	Subtracts one number from another	25 - 5
*	Multiplies two numbers	5 * 10
/	Divides one number by another	15/5

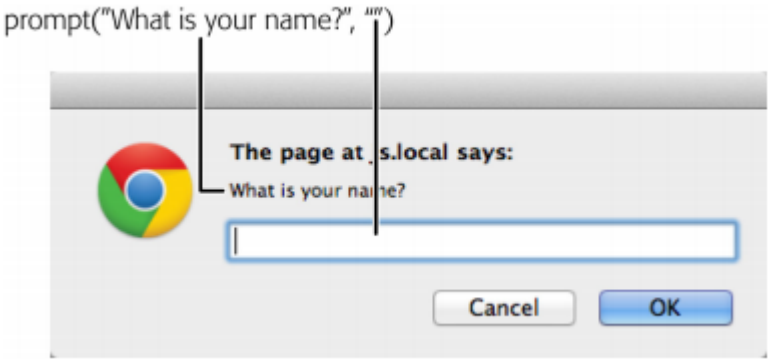
JavaScript Variables and Operator

- **Example 1:**

```
<script type="text/javascript">
var firstName = 'Cookie';
var lastName = 'Monster';
document.write('<p>');
document.write(firstName + ' ' + lastName);
document.write('</p>');
</script>
```

- **Example 2:**

```
<script>
var name = prompt("What is your name:","");
document.write("Your name is:"+name);
</script>
```



Bài tập

- **Bài 1:** Viết chương trình nhập điểm của một sinh viên cho các môn: Vật lý, Hóa học, và Sinh học. Sau đó hiển thị điểm trung bình và tổng của những điểm này.
- **Bài 2:** Viết chương trình nhập một giá trị là độ 0C (Celsius) và chuyển nó sang độ 0F (Fahrenheit). [Hướng dẫn: C/5 = (F-32)/9]
- **Bài 3:** Viết chương trình tính diện tích hình tròn
- **Bài 4:** Viết chương trình chu vi hình tròn

JavaScript Variables and Operator

OPERATOR	WHAT IT DOES	HOW TO USE IT	THE SAME AS
+=	Adds value on the right side of equals sign to the variable on the left.	`score += 10`	`score = score + 10`
-=	Subtracts value on the right side of the equals sign from the variable on the left.	`score -= 10`	`score = score - 10`
*=	Multiplies the variable on the left side of the equals sign and the value on the right side of the equals sign.	`score *= 10`	`score = score * 10`
/=	Divides the value in the variable by the value on the right side of the equals sign.	`score /= 10`	`score = score / 10`
++	Placed directly after a variable name, ++ adds 1 to the variable.	`score++`	`score = score + 1`
--	Placed directly after a variable name, -- subtracts 1 from the variable.	`score--`	`score = score - 1`

Arithmetic cum Assignment Operators

Operator	Description	Example	Result
+=	Addition cum Assignment	x += y;	Same as: x = x + y;
-=	Subtraction cum Assignment	x -= y;	Same as: x = x - y;
*=	Multiplication cum Assignment	x *= y;	Same as: x = x * y;
/=	Division cum Assignment	x /= y;	Same as: x = x / y;
%=	Modulus cum Assignment	x %= y;	Same as: x = x % y;

The Number Operations

- `parseInt(aString)`: Parse the *aString* until the first non-digit, and return the number; or `NaN`.
- `parseFloat(aString)`:
- `Math.round(aNumber)`, `Math.floor(aNumber)`, `Math.ceil(aNumber)`:
- `Math.random()`: Generate a random number between 0 (inclusive) and 1 (exclusive).
- `isNaN(aString)`: return true if the *aString* is not a number. For example,

```
console.log(isNaN('123'));     // false
console.log(isNaN('1.23'));    // false
console.log(isNaN('123abc'));  // true
```

JavaScript Math Reference

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

JavaScript Math Reference (Cont …)

Method	Description
floor(x)	Returns x, rounded downwards to the nearest integer
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sqrt(x)	Returns the square root of x

JavaScript **Number** Reference

Method	Description
isFinite()	Checks whether a value is a finite number
isInteger()	Checks whether a value is an integer
isNaN()	Checks whether a value is Number.NaN
isSafeInteger()	Checks whether a value is a safe integer
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length
toString()	Converts a number to a string
valueOf()	Returns the primitive value of a number

String's Operations

- **.toUpperCase():** returns the uppercase string.
- **.toLowerCase():** returns the lowercase string.
- **.charAt(*index*):** returns the character at the *index* position. Index begins from 0. Negative index can be used, which counts from the end of the string.
- **.substring(*beginIndex, endIndex*):** returns the substring from *beginIndex* (inclusive) to *endIndex* (exclusive).
- **.substr(*beginIndex, length*):** returns the substring from *beginIndex* of *length*.

String's Operations (Cont …)

- **.indexOf(*searchString, fromIndex*?):** Return the beginning index of the *first occurrence* of *searchstring*, starting from an optional *fromIndex* (default of 0); or -1 if not found.
- **.lastIndexOf(*searchString, fromIndex*?):** Return the beginning index of the *last occurrence* of *substring*, starting from an optional *fromIndex* (default of string.length); or -1 if not found.
- **.slice(*beginIndex, endIndex*):** Return the substring from *beginIndex* (inclusive) to *endIndex* (exclusive).
- **.split(*delimiter*):** returns an array by splitting the string using *delimiter*.

Conditional Statement Basics

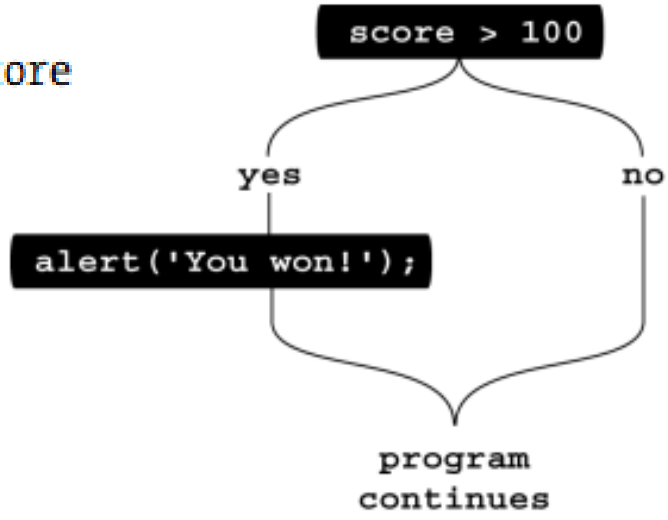
```
if ( condition ) {
    // some action happens here
}
```

```
if (condition) {
  // door #1
} else if (condition2) {
 // door #2
} else {
 // door #3
}
```

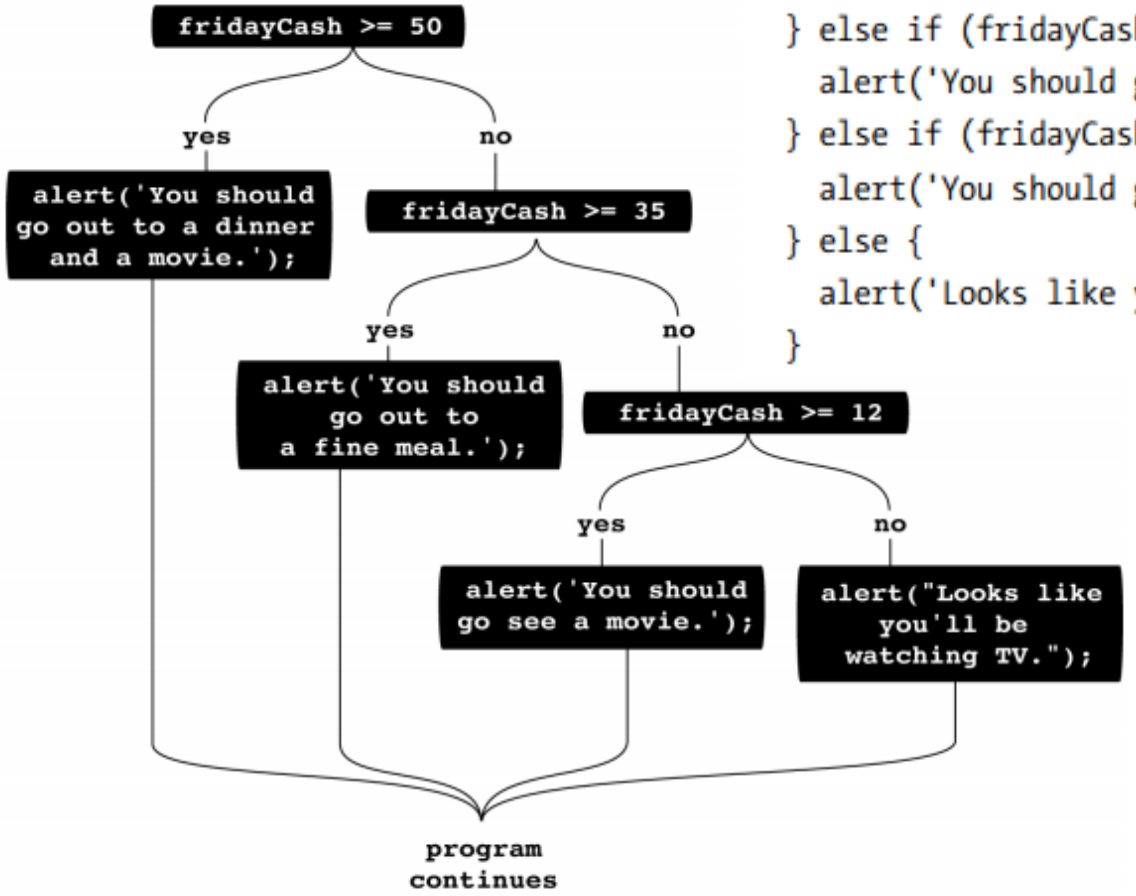
- Example:

```
var score = 0; //stores 0 into the variable score

if (score > 100) {
 alert('You won!');
}
```



Conditional Statement Basics




```
var fridayCash = prompt('How much money can you spend?', '');
if (fridayCash >= 50) {
  alert('You should go out to dinner and a movie.');
} else if (fridayCash >= 35) {
  alert('You should go out to a fine meal.');
} else if (fridayCash >= 12) {
  alert('You should go see a movie.');
} else {
  alert('Looks like you will be watching TV.');
}
```

Bài tập IF - Else

- **Bài 1:** Nhập vào hai số a và b, và kiểm tra xem a có chia hết cho b hay không
- **Bài 2:** Nhập tuổi và in ra kết quả nếu tuổi học sinh đó không đủ điều kiện vào học lớp 10.
- **Bài 3:** Nhập một số nguyên bất kỳ và in kết quả ra màn hình để nói cho người dùng biết số đó là lớn hay nhỏ hơn 0
- **Bài 4:** Nhập 3 số nguyên và tìm giá trị lớn nhất của ba số nguyên đó
- **Bài 5:** Xếp hạng học lực của học sinh dựa trên các điểm bài kiểm tra, điểm thi giữa kỳ, điểm thi cuối kỳ
- **Bài 6:** Tính hoa hồng nhận được tuỳ theo mức doanh số bán hàng
- **Bài 7:** Tính cước điện thoại cho một hộ gia đình với các thông số đã cho

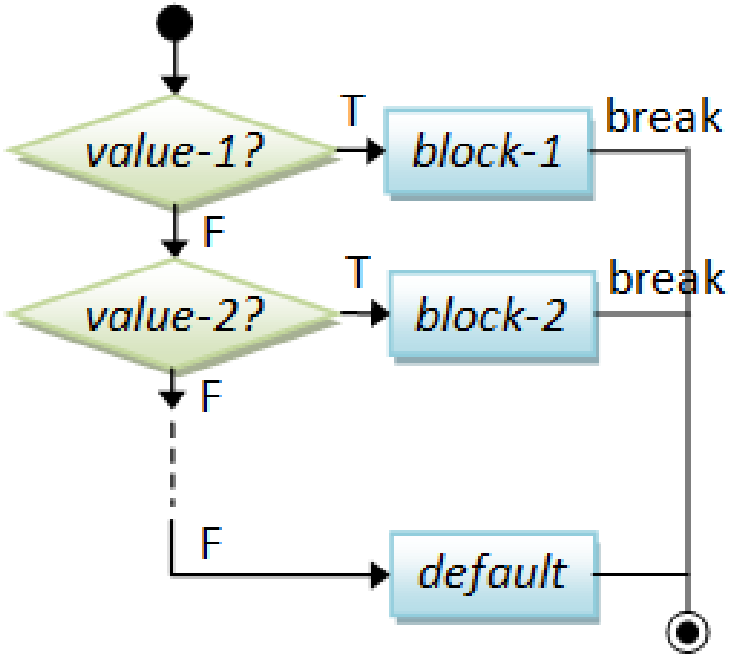
Bài tập IF – Else (Tt)

- **Bài 8:** Tính diện tích hình chữ nhật khi biết 02 cạnh a, b.
- **Bài 9:** Tính diện tích tam giác vuông khi biết 02 cạnh kề a, b.
- **Bài 10:** Giải phương trình bậc 1.
- **Bài 11:** Giải phương trình bậc 2.
- **Bài 12:** Kiểm tra xem một số nhập vào có phải là tuổi của một người không. Một số nguyên là tuổi của một người khi nhỏ 120 và lơn hơn 0.
- **Bài 13:** Viết chương trình tính giá điện.
- **Bài 14:** Viết chương trình tính thuế thu nhập cá nhân.
- **Bài 15:** Viết chương trình tính lãi ngân hàng (lãi mẹ đẻ lại con) khi biết số tiền ban đầu, số tháng cho vay và lãi xuất hàng tháng.

Switch - Case Statements

- Java provides a **switch** statement to handle multiple conditions efficiently.

```
// switch-case-default
switch ( selector ) {
   case value-1:
      block-1; break;
   case value-2:
      block-2; break;
   case value-3:
      block-3; break;
   ......
   case value-n:
      block-n; break;
   default:
      default-block;
}
```



Example: Switch - case

- Write a program fragment that computes the appropriate weighted score for one test. The fragment should first read values of **testNumber** and **score** and the compute and print the appropriate value of **weightedScore = Score * Weight** using the weightings given in the following table

Example: A score of 27 on test 3 gives a weighted score of 5.4

Test Number	Weight
1	10%
2	20%
3	20%
4	15%
5	15%
6	20%

TEST – PART 1

Câu 1. Viết chương trình cho phép người dùng nhập vào **02 số nguyên và 01 phép tính (+, -, *, hoặc /).** Hãy thực hiện việc tính toán 02 số trên với phép tính tương ứng được nhập và hiển thị kết quả của phép tính hai số đó.

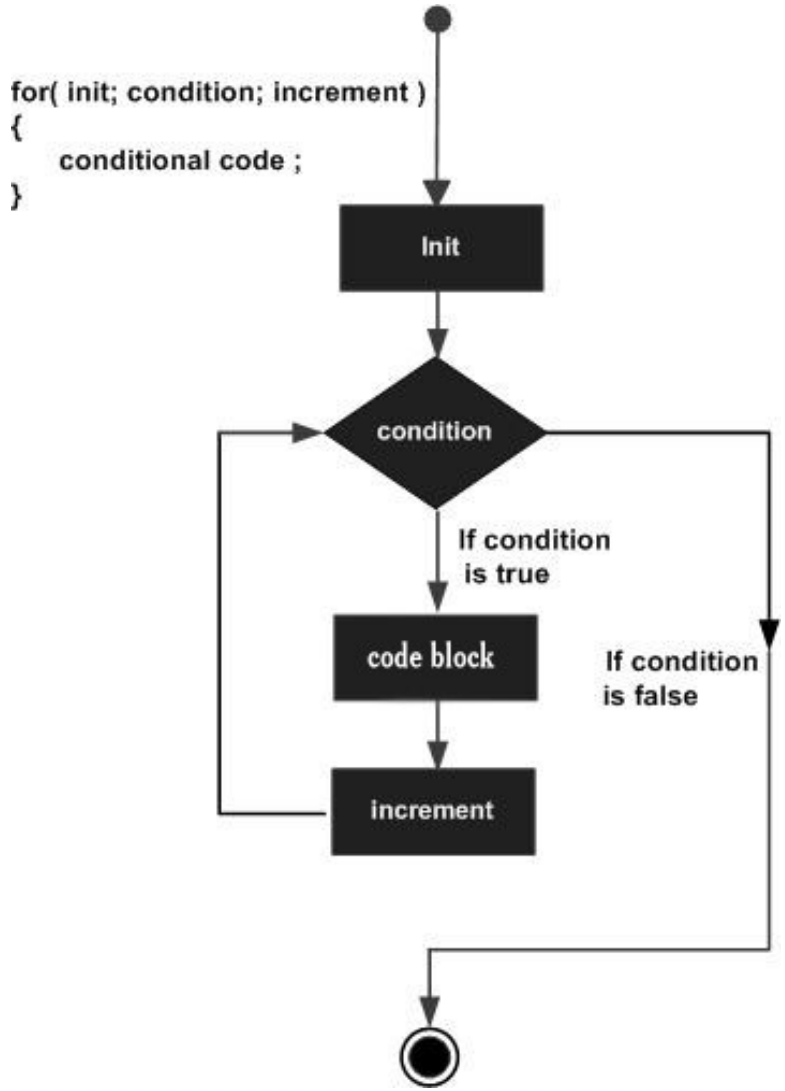
Câu 2. Có hai phương thức gửi tiền tiết kiệm:

Hình thức gửi	% lãi suất
1 tháng	1.5 %/tháng
3 tháng	4.5 %/tháng
6 tháng	5.0%/tháng
0 (Không kỳ hạn)	2.5%/tháng

Viết chương trình yêu cầu người dùng nhập vào số tiền gửi tiết kiệm và hính thức gửi thiết kiệm, hãy tính tiền lãi và tổng cộng số tiền nhận được tương ứng với hình thức gửi tiết kiệm

For Loop

- Loops can execute a block of code a number of times.
- for - loops through a block of code a number of times
- for/in - loops through the properties of an object



The For Loop

- Syntax:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

- **Statement 1** is executed (one time) before the execution of the code block.
- **Statement 2** defines the condition for executing the code block.
- **Statement 3** is executed (every time) after the code block has been executed.
- Example:

```
for (i = 0; i < 5; i++) {
  text += "The number is " + i + "<br>";
}
```

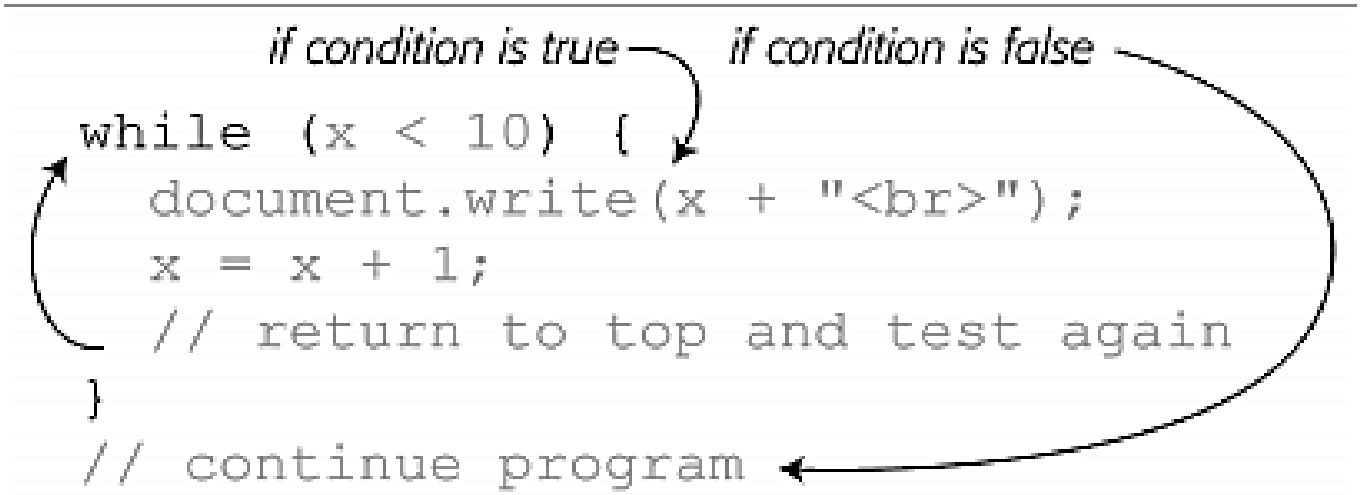
Bài tập vòng lặp For

- **Bài 1:** Hiển thị ra 20 số trong dãy fibonacci đầu tiên.
- **Bài 2:** Tìm số đầu tiên trong dãy fibonacci chia hết cho 5.
- **Bài 3:** Tính tổng của 20 số đầu tiên trong dãy fibonacci.
- **Bài 4:** Tính tổng của 30 số chia hết cho 7 đầu tiên trong các số tự nhiên.

While Loops

```
while (condition) {
 // javascript to repeat
}
```

Example:



BÀI TẬP

1. Tổng các giá trị từ 0 đến N (Nhập N từ bàn phím):

   Sum = 0 + 1 + 2 + 3 + 4 + … + N

2. Tổng các giá trị CHẴN từ 0 đến N

3. Tổng các giá trị LẺ từ 0 đến N

4. Tính giai thừa của N (với N > 0, N Nhập vào từ bàn phím)

5. Tìm (Hiển thị) các số Fibonacci từ 0 đến N (N < 50)

6. Viết chương trình thực hiện công thức sau (N nhập vào từ bàn phím, Hn là kết quả của phép tính):

$$H_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

Bài tập - Assignments

- Viết code thực hiện Menu lựa chọn, Lặp đi lặp lại việc thực hiện các lệnh Menu: Lựa chọn chức năng 1 và 2 (Nhập 0 để thoát).

1. Tính diện tích và chu vi hình chữ nhật:

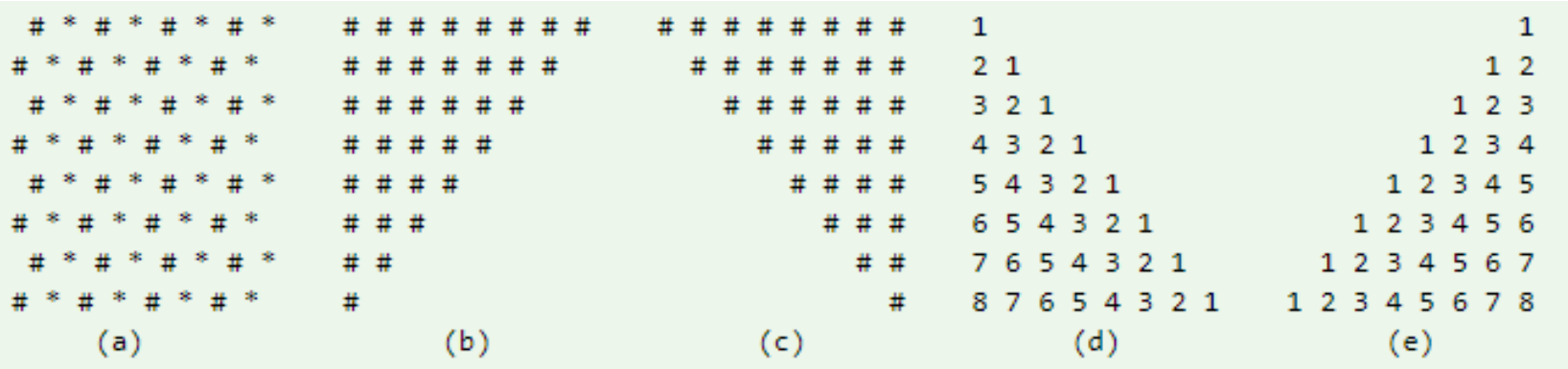
diện tích = c.dài x c.rộng,

chu vi = 2x(c.dài+ c.rộng)

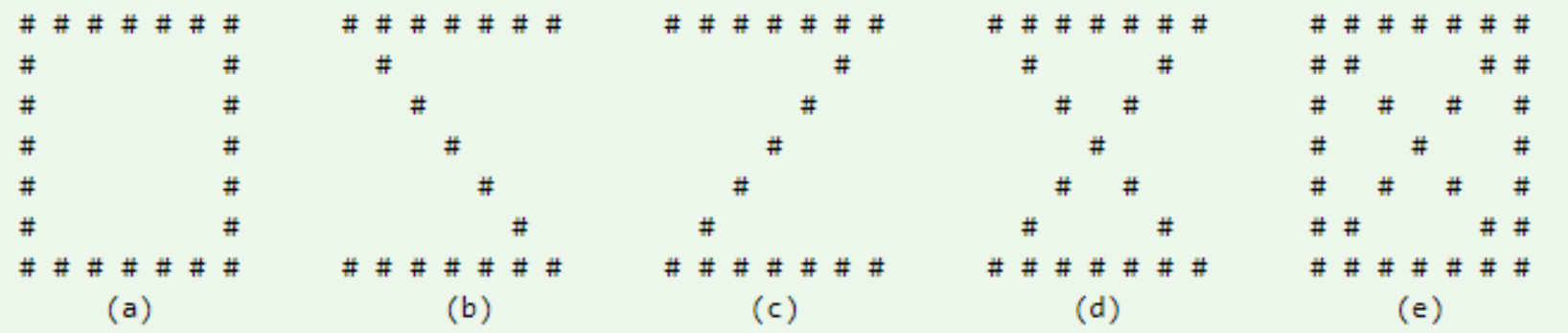
2. Tính diện tích và chu vi hình tròn: r là bán kính, $C = 2\pi r$, $A = \pi r^2$

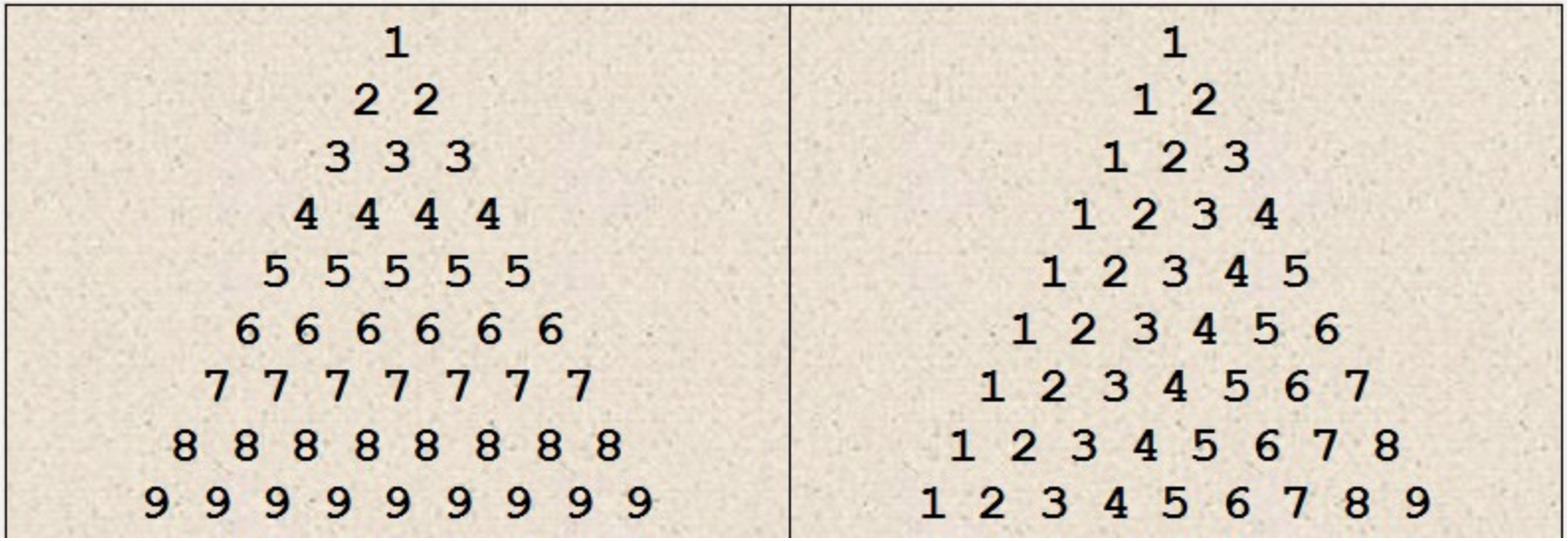
Exercises

1) Print these patterns using nested loop:



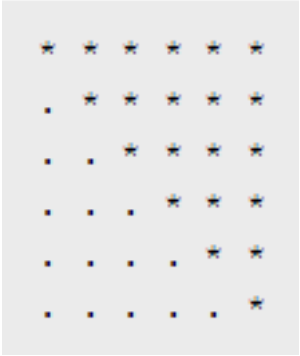
2) Print these patterns using nested loop:




1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9

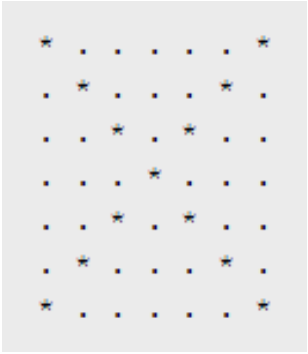
HOMEWORKS (1-2)

1. Write a program that prints an N-by-N triangular pattern like the one below:



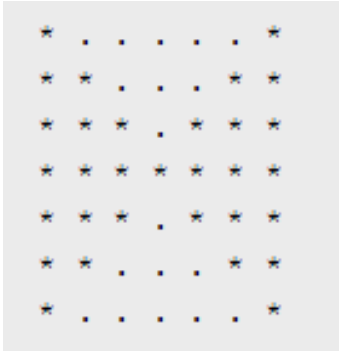
2. Write a program that prints a (2N + 1)-by-(2N + 1) like the one below

```
if ((i == -j) || (i == j)) System.out.print("* ");
```



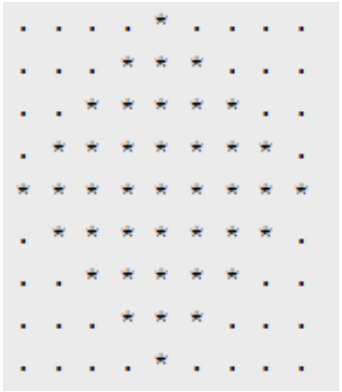
HOMEWORKS (3-4)

3. Write a program that prints a (2N + 1)-by-(2N + 1) like the one below:



```
if (i*i <= j*j) System.out.print("* ");
```

4. Write a program that prints a (2N + 1)-by-(2N + 1) like the one below:



BÀI TẬP TỔNG HỢP

- Viết code thực hiện Menu lựa chọn, Lặp đi lặp lại việc thực hiện các lệnh Menu: Lựa chọn chức năng 1, 2 và 3 (Nhập 0 để thoát).

1. Tính **n! = *n* * (*n* – 1) * (*n* – 2) * … * 1** (n nhập vào từ bàn phím)
2. Tính e: $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots$ (n nhập vào từ bàn phím)
3. Tính $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$ (n nhập vào từ bàn phím)

TEST – PART 2

Câu 1: Nhập vào hai số N1 và N2 (N1, N2 nhập vào từ bàn phím), hãy:

a) Cho biết **có bao nhiêu số chia hết cho 2 từ N1 đến N2?**

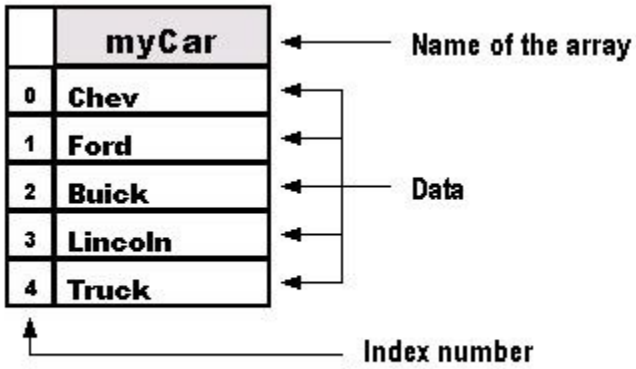
b) **Cho biết số lớn nhất từ N1 đến N2 chia hết cho 2?**

Câu 2: Nhập vào hai số M1 và M2 (M1, M2 nhập vào từ bàn phím), hãy cho biết giá trị lớn nhất mà M1 và M2 đều chia hết?

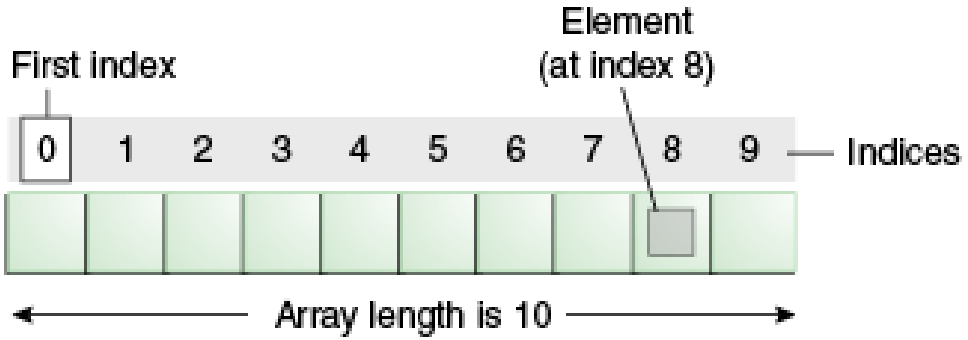
Câu 3: Write a program that displays all the numbers from 10 to 100, that are divisible by 5 but not 6?

JavaScript

Array



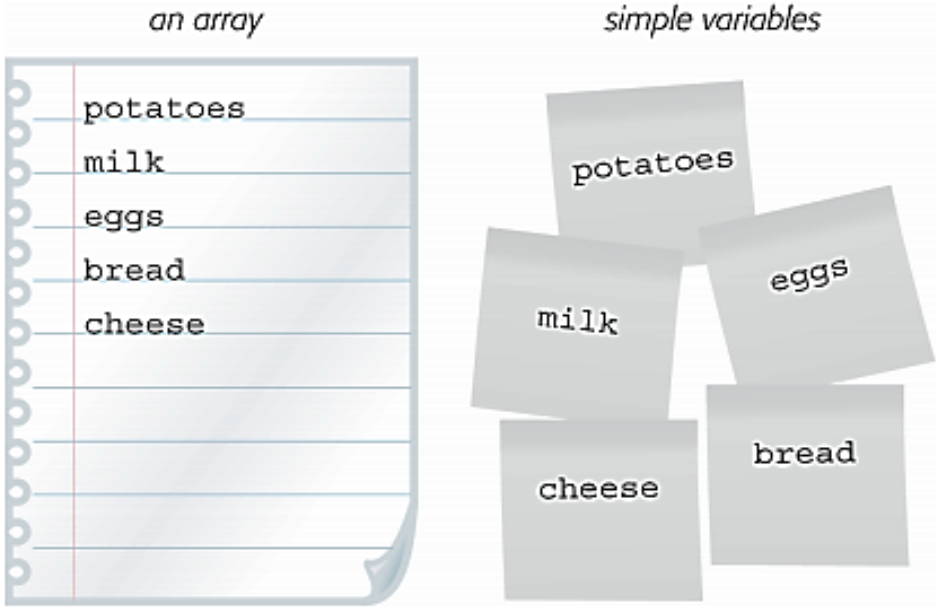
Comparison of an array to a column of data

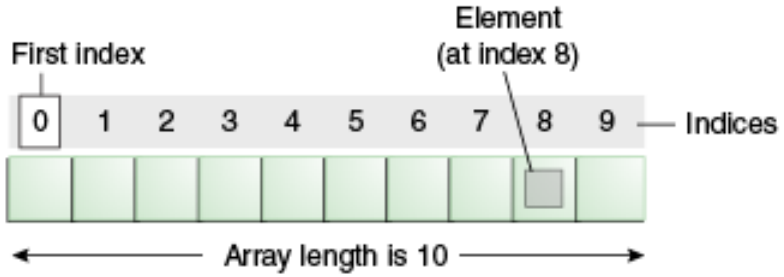


Arrays

```
var days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun'];
alert(days[0]);
```

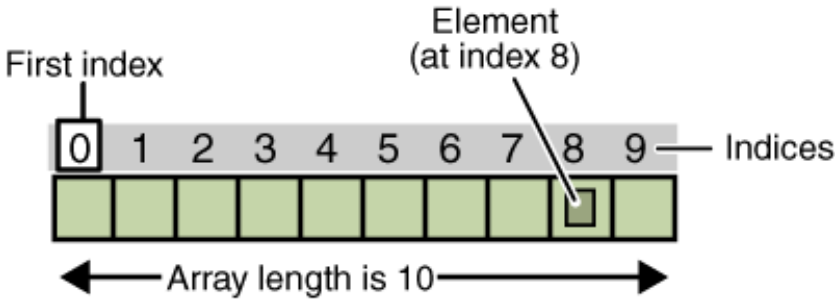
INDEX VALUE	ITEM	TO ACCESS ITEM
0	Mon	days[0]
1	Tues	days[1]
2	Wed	days[2]
3	Thurs	days[3]
4	Fri	days[4]
5	Sat	days[5]
6	Sun	days[6]

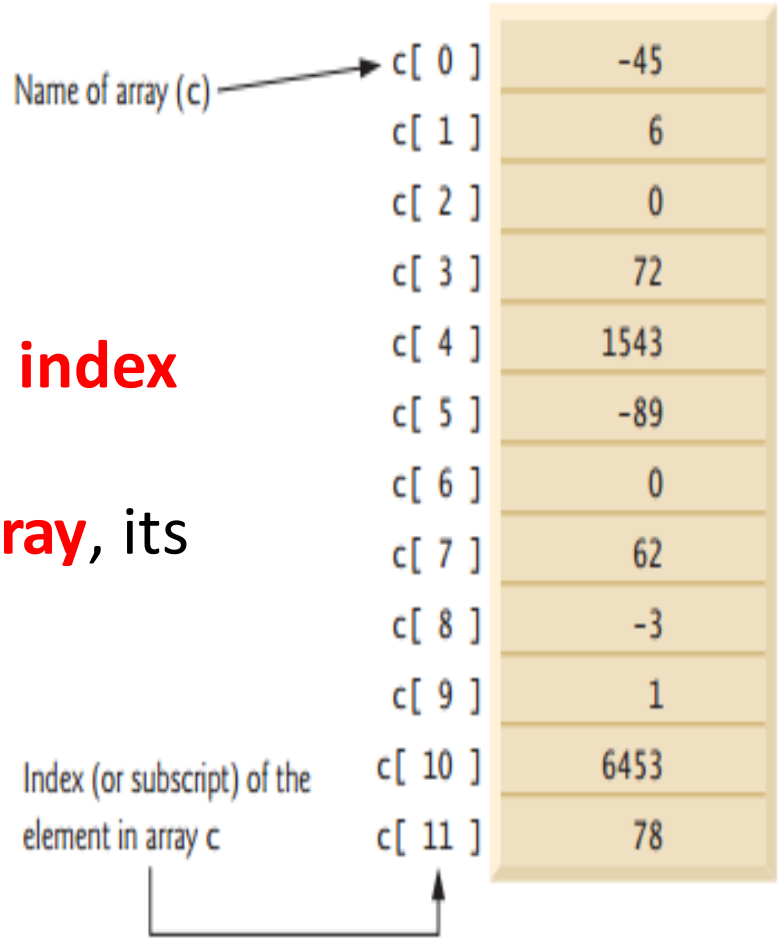




Arrays - Example

- a logical representation of **an integer array** called **c.**
  - ❑ This array contains 12 elements.
  - ❑ The first element in every array has **index zero.**
  - ❑ **Array length.** Once we create **an array**, its **length is fixed**.





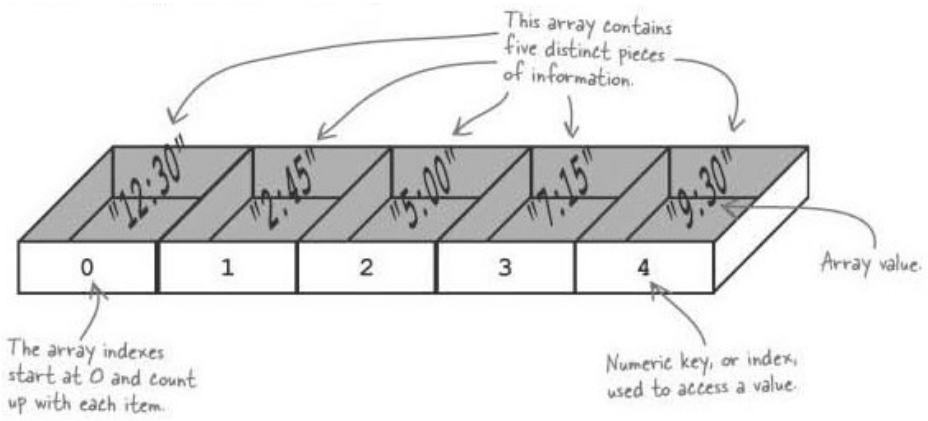
Declaring an array

- A literal is the best way to express an array:

```
var arr = [1, 2, 3, 4];
```

- The Array constructor:

```
var arr = new Array(1, 2, 3, 4);
```



HOW TO ADD ITEMS TO AN ARRAY?

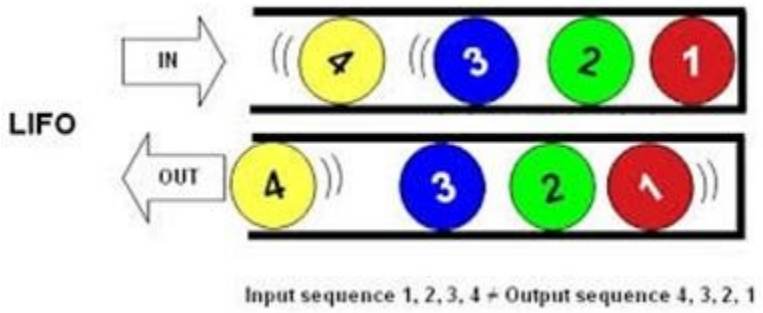
METHOD	ORIGINAL ARRAY	EXAMPLE CODE	RESULTING ARRAY	EXPLANATION
.length property	var p = [0,1,2,3]	p[p.length]=4	[0,1,2,3,4]	Adds one value to the end of an array.
push()	var p = [0,1,2,3]	p.push(4,5,6)	[0,1,2,3,4,5,6]	Adds one or more items to the end of an array.
unshift()	var p = [0,1,2,3]	p.unshift(4,5)	[4,5,0,1,2,3]	Adds one or more items to the beginning of an array.

METHOD	ORIGINAL ARRAY	EXAMPLE CODE	RESULTING ARRAY	EXPLANATION
pop()	var p = [0,1,2,3]	p.pop()	[0,1,2]	Removes the last item from the array.
shift()	var p = [0,1,2,3]	p.shift()	[1,2,3]	Removes the first item from the array.

Deleting Items from an Array

LIFO behavior (stack)

- LIFO stands for Last-In-First-Out and represents a stack.
- he last plate you added is the one on top of the pile and it's the only one you can take back.
- Functions that allow you to use the array as a stack:
  - **push(v)** -- adds an element v at the end of the array (at index length) --returns the new length of the array
  - **pop()** --removes the last element --returns the element that was deleted



LIFO behavior (stack)

```
var arr = [true, 2, 3, "4"];
arr.push(5);
console.log(arr);                // [true, 2, 3, "4", 5]

last_element = arr.pop();
console.log(arr);                // [true, 2, 3, "4"]
console.log(last_element);       // 5
```

```
[ true, 2, 3, '4', 5 ]
[ true, 2, 3, '4' ]
5
```

FIFO behavior (queue)

- FiFo stands for First-In-First-Out
- A queue in computer programming is similar: the **add** operation puts an element at the end of the queue and **remove** operation deletes elements from the start of the queue
  - **shift**() – returns the first item of the list and deletes the item
  - **push(v)** – adds an element v at the end of the array (at index length) --returns the new length of the array

FIFO behavior (queue)

```
var arr = [true, 2, 3, "4"];
first_item = arr.shift();
console.log(arr);                    // [2, 3, "4"]
console.log(first_item);             // true

arr.push(first_item);
```

```
[ 2, 3, '4' ]
true
```

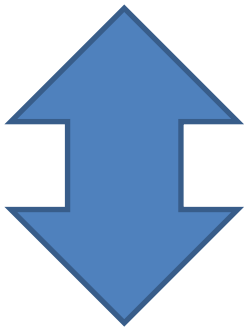
Loops and Arrays

```
var days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
      'Friday', 'Saturday', 'Sunday'];
 var counter = 0;
 while (counter < days.length) {
   document.write(days[counter] + ', ');
   counter++;
 }
```

counter *value before test*	*condition*	*loop?*	days[counter]	counter *value after* counter++
0	0 < 7	yes	days[0]	1
1	1 < 7	yes	days[1]	2
2	2 < 7	yes	days[2]	3
3	3 < 7	yes	days[3]	4
4	4 < 7	yes	days[4]	5
5	5 < 7	yes	days[5]	6
6	6 < 7	yes	days[6]	7
7	7 < 7	no		

For Loops

```
var num = 1;
while (num <= 100) {
  document.write('Number ' + num + ' <br>');
  num += 1;
}
```



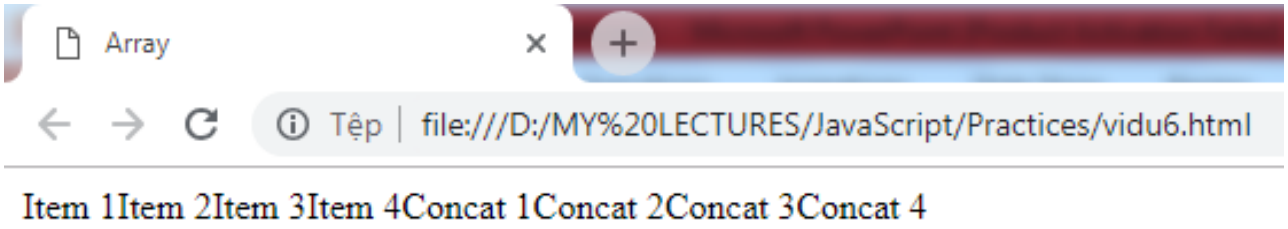
```
var days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
      'Friday', 'Saturday', 'Sunday'];
for (var i=0; i<days.length; i++) {
  document.write(days[i] + ', ');
}
```

```
for (var num = 1; num <= 100; num++) {
  document.write('Number ' + num + ' <br>');
}
```

Merging Arrays

- The **concat()** method can be used **to merge an array with one or more arrays**:

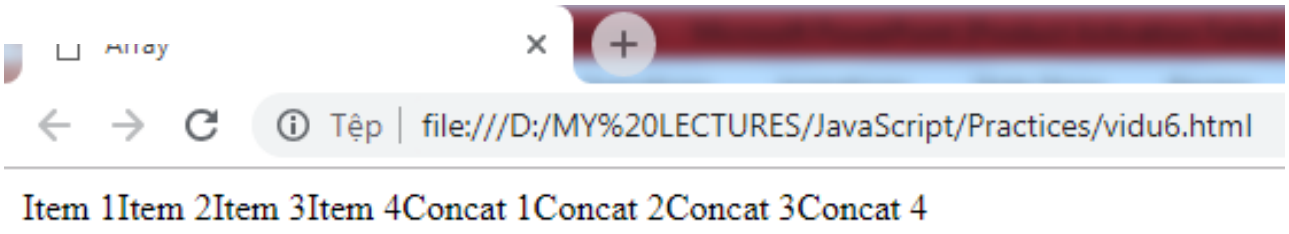
```
var arr_string = ['Item 1','Item 2', 'Item 3', 'Item 4'];
//Merging Arrays
arr_string = arr_string.concat(['Concat 1','Concat 2','Concat 3','Concat 4']);
for (let i = 0; i < arr_string.length; i++) {
    document.write(arr_string[i]);
}
```



The join() Method

- The join() method can be used to turn the array into a string that comprises all the items in the array, separated by commas

```
//Join method
var result = arr_string.join('-');
//Display
 for (var i = 0; i < result.length; i++) {
    document.write(result[i]);
}
```



Item 1-Item 2-Item 3-Item 4-Concat 1-Concat 2-Concat 3-Concat 4

Code Đếm số lượng phần tử có giá trị Chẵn/Lẻ

```
//1. Dem Chan/Le
var chan =0, le=0;
for(var k =0; k < arr_num.length; k++){
    if ((arr_num[k] % 2) === 0) {//Chia het
        chan++;
    } else {//Ko chia het
        le++;
    }
}
```

Đếm trong mảng có bao nhiêu giá trị cộng lại bằng Z

```
var Z = prompt("Nhap vao gia tri Z",'Gia tri mac dinh');
var dem = 0;
for(var i =0; i< arr_num.length; i++){
    for(var j = i+1; j< arr_num.length; j++){
        if(arr_num[i]+arr_num[j] == Z.valueOf()){
            dem++;
            console.log('Num 1:' + arr_num[i] + ' Num 2:' + arr_num[j])
        }
    }
}
console.log("Dem " + dem);
```

Exercies

- Write a program to **create an array** that is comprised of elements of **double type**. **The size** of the array and the array **elements** will be defined by **the user input via Keyboard.**
  1) print all elements in the array
  2) Calculate and print the sum, the average of all elements in the array.
  3) the minimum and the maximum value in the array,
  4) and count how many elements are smaller or larger than the average

Bài tập

- Cho mảng số nguyên A (Giá trị tùy ý), anh/chị hãy:
  - Cho biết mảng A có bao nhiêu phần tử Chẵn
  - Cho biết mảng A có bao nhiêu phần tử Lẻ
  - Đưa các giá trị chẵn của mảng A vào mảng C
  - Đưa các giá trị lẻ của mảng A vào mảng L
  - Hiển thị các giá trị trong mảng C và L
  - Cho biết giá trị lớn nhất trong mảng C, L và trong A
  - ** Hãy hiển thị các giá trị trong mảng C thỏa mãn: Hai giá trị trong mảng L cộng lại bằng giá trị lớn nhất trong mảng C

HOMEWORK EXERCISES

- **Exercise 1:** Write a program to read a series of words from the keyboard, and **find the index** of the first **match** of a **given word (from the keyboard).**
- **Exercise 2:** Write a JavaScript program to **find** the **duplicate values** of an **array of integer** values.
- **Exercise 3:** Write a JavaScript program to find the common elements between two arrays of integers
- **Exercise 4:** Write a JavaScript program to find the second largest element in an array

HOMEWORK EXERCISES (Cont …)

- **Exercise 5:** Write a JavaScript program to find the second smallest element in an array.
- **Exercise 6:** Write a JavaScript program to find the number of even and odd integers in a given array of integers


{set}

Sets

- Sets were introduced to the specification in ES6.
- A set is a data structure that represents a collection of unique values, so it cannot include any duplicate values ;
- Sets offer a useful way to keep track of data without having to check if any values have been duplicated.
- It’s also quick and easy to check if a particular value is in a set, which can be a slow operation if an array is used

The available **methods in set** are

1. *add()*
2. *size →property*
3. *has()*
4. *forEach()*
5. *delete()*
6. *clear()*

```
animals.add('🐯');
animals.add('🐼');
animals.add('🦡');
animals.add('□');
console.log(animals.size); // 4
animals.add('🐼');
console.log(animals.size); // 4

console.log(animals.has('🐯')); // true
animals.delete('🐯');
console.log(animals.has('🐯')); // false

animals.forEach(animal => {
  console.log(`Hey ${animal}!`);
});

// Hey 🐼!
// Hey 🦡!
// Hey □!

animals.clear();
console.log(animals.size); // 0
```

Creating Sets

- An empty set is created using the new operator and Set() constructor:

  let list = new Set(); hoặc var list = new Set()

```
//Khai báo tập hợp - list
let list = new Set();
//Thêm giá trị vào tập hợp - list
list.add(4);
list.add(5);
list.add(9);
list.add(8).add(7).add(3); //Add nhiều
//Hiển thị
list.forEach(list_item => {
    document.write(list_item + ' , ');
});
```

Functions

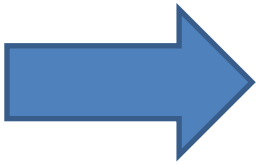
The basic structure of a function looks like this:

```
function functionName() {
 // the JavaScript you want to run
}
```

- Example:

```
function printToday() {
  var today = new Date();
  document.write(today.toDateString());
}
```

- For example, to make our printToday function run, you'd simply type:
  printToday();



```
<p>Today is <strong>
<script>
 printToday();
</script>
</strong></p>
```

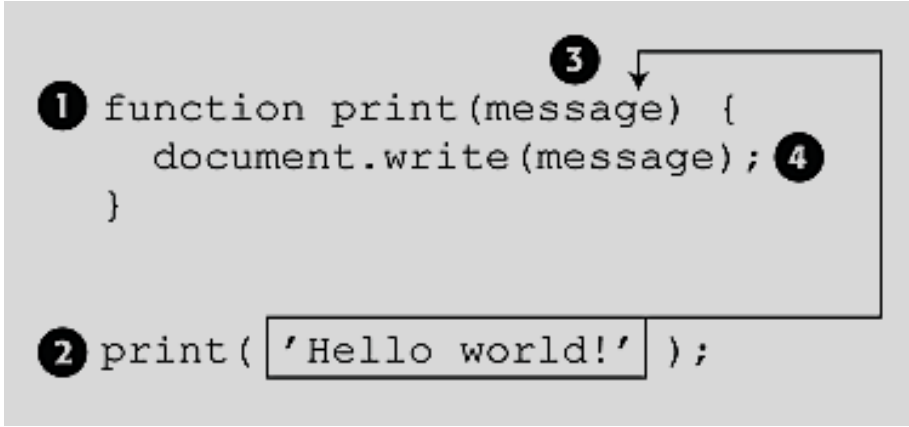
Functions (Cont..)

Function have one paramater:

```
function functionName(parameter) {
  // the JavaScript you want to run
}
```

Pass any number of arguments to a function

```
function functionName(parameter1, parameter2, parameter3) {
 // the JavaScript you want to run
}
```



Retrieving Information from Functions

```
function functionName(parameter1, parameter2) {
  // the JavaScript you want to run
  return value;
}
```

- **Example:**

```
var TAX = .08; // 8% sales tax
function calculateTotal(quantity, price) {
  var total = quantity * price * (1 + TAX);
  var formattedTotal = total.toFixed(2);
  return formattedTotal;
}

var saleTotal = calculateTotal(2, 16.95);
document.write('Total cost is: $' + saleTotal);
```

Viết hàm nhập và hiển thị các giá trị trong mảng (Số nguyên)

```
//Nhập giá trị vào Prompt, các giá trị ngăn cách nhau bằng dấu phẩy
function Nhap_mang() {
    //Tách chuỗi bằng hàm split('Ký hiệu để phân tách')
    var A = prompt("Nhap cac gia tri cho mang (ngan cach dau ,):").split(',');
    return A;
}
//Hàm hiển thị
function HienThi(arr) {
    for(var i =0; i< arr.length; i++){
        document.write("<br>"+arr[i]);
    }
}
```

```
//Gọi hàm
var arr = Nhap_mang();
HienThi(arr);
```

Bài tập

1. Viết hàm nhập vào một mảng các số nguyên
2. Viết hàm hiển thị tất cả các giá trị đã nhập vào mảng
3. Viết hàm tính Tổng tất cả các giá trị trong mảng
4. Viết hàm tính tổng tất cả các giá trị Chẵn/Lẻ trong mảng
6. Viết hàm đếm xem có bao nhiêu giá trị chẵn/Lẻ trong mảng
7. Viết hàm tách các giá trị chẵn/Lẽ ra mảng Chẵn/Lẻ tương ứng
8. Viết hàm đếm xem có tồn tại giá trị X trong mảng hay ko?
9. Viết hàm đếm xem có bao nhiêu giá trị trong mảng có giá trị lớn hơn X nhưng nhỏ hơn Y.
10. Viết hàm đếm xem có bao nhiêu cặp giá trị trong mảng cộng lại với nhau bằng Z.