Front-end Development

JavaScript

3

Fundamental Web Programming

Asst. Prof. Manop Phankokkruad, Ph.D.
School of Information Technology
King Mongkut's Institute of Technology Ladkrabang

KMITL

Outline

1. Introduction
2. JavaScript
3. Data types
4. Conditional Statements
5. Events
6. Accessing elements

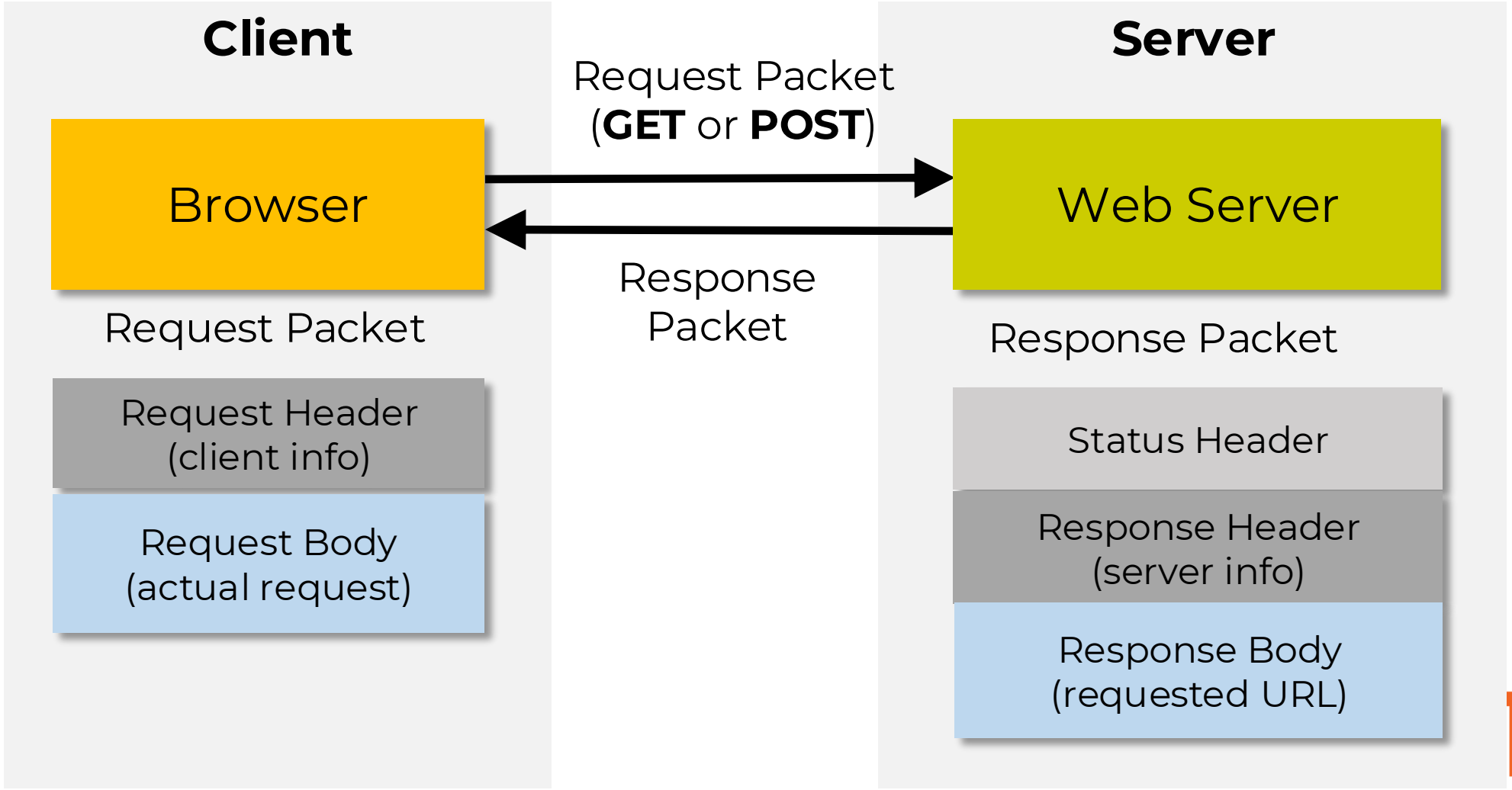
Introduction

The Three Layers of Web Design

1. HTML: The structure layer is the underlying HTML code of that page.
2. CSS: The styles layer dictates how a structured HTML document will look to a site's visitors and is defined by CSS (Cascading Style Sheets).
3. JavaScript: The behavior layer makes a website interactive, allowing the page to respond to user actions or to change based on a set of conditions.

Introduction

Request-Response Model



Introduction : GET and POST

In the HTTP protocol, data from the client is sent to the server in one of two primary ways:

GET

- attaches the forms data to the requested URL (URL encoded) and places in the body of request packet.
- the URL has the data attached it is visible in the Location box of the browser. Since this allow the data to be seen by any with in view of the screen it is not preferred.

KMITL

Introduction : GET and POST

In the HTTP protocol, data from the client is sent to the server in one of two primary ways: (cont.)

POST

- places the forms data into the body of the request packet,
- This way the data will not be seen by snooping eyes. It is more private than GET. This is the preferred way to send data to the server.

KMITL

Introduction

Client-side scripting benefits:

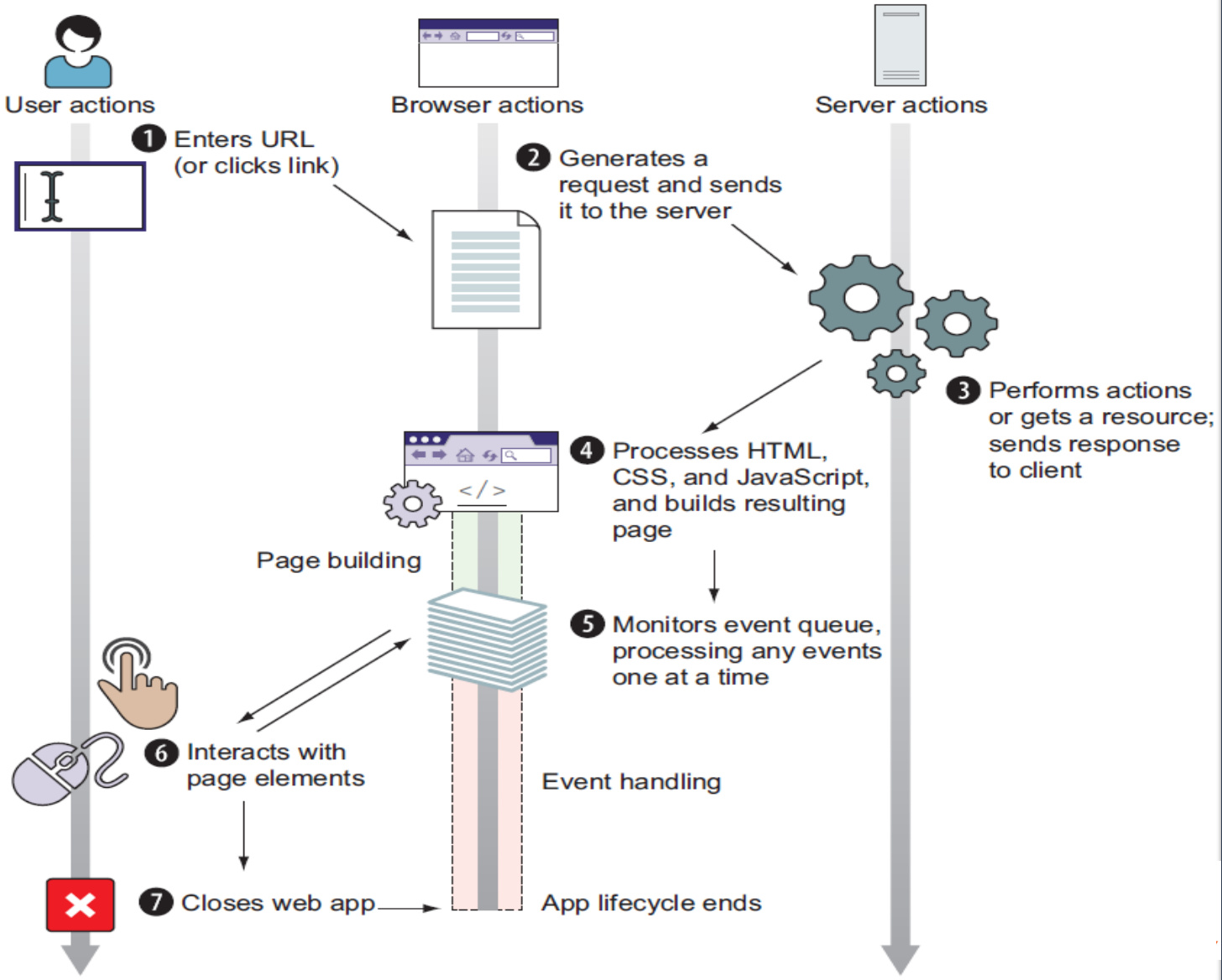
Scripts are embedded within and interact with the HTML of Website. **Scripts** interact with a cascading style sheet (CSS) file that styles the way the page looks. **Scripts** put less stress on the server because they don't require processing on the server.

- **Usability**: can modify a page without having to post back to the server (faster UI).
- **Efficiency**: can make small, quick changes to page without waiting for server.
- **Event-driven**: can respond to user actions like clicks and key presses.

KMITL

Introduction

The lifecycle of a client-side web application.



JavaScript

JavaScript is a high-level, interpreted programming language. JavaScript is a powerful programming language that can add interactivity to a website. **JavaScript** is one of the three core technologies of the World Wide Web.

- ECMA International - European association for standardizing information and communication systems") for standardization. The ECMA Specification is called "ECMA-262 ECMAScript Language Specification"

KMITL

JavaScript

Advantages of JavaScript

- A lightweight programming language.
- used to make web pages interactive.
- react to events (event-driven).
- get information about a user’s computer.
- perform calculations on user's computer.
- more relaxed syntax and rules.
- Increased interactivity.
- Less server interaction.

KMITL

JavaScript Frameworks

JavaScript frameworks are a type of tool that makes working with JavaScript easier and smoother. These frameworks also make it possible for the programmer to code the application as a device responsive. The popular JavaScript frameworks such as.

- Angular-JS
- React.JS
- Vue.js
- jQuery
- Angular
- Node.JS
- Ember.JS
- Ajax
- and more...

KMITL

JavaScript : Basic

- JavaScript code can be placed directly in the HTML files *<body>* or *<head>* tag.

```
<script ...>
   JavaScript code
</script>
```

```
<script language = "javascript" type = "text/javascript">
   JavaScript code
</script>
```

- The scripts inside an HTML document is interpreted in the order they appear in the document. Scripts in a function is interpreted when the function is called. So where you place the *<script>* tag matters.

JavaScript : Variables

JavaScript has three kinds of variable declarations. Always declare JavaScript variables with **var**, **let**, or **const**.

- **const** - declares a block-scoped, read-only named constant.
- **let** - declares a block-scoped, local variable, optionally initializing it to a value that the value of the variable can change.
- **var** - declares a variable, optionally initializing it to a value. declares a function-scoped or globally-scoped variable, optionally initializing it to a value.

KMITL

JavaScript : Data types

String	let myVariable = 'Bob'; let myVariable = "Bob";
Number	let myVariable = 10;
Boolean	let myVariable = true;
Array	let myVariable = [1,'Bob','Steve',10]; *Refer to each member of the array like this:* *myVariable[0], myVariable[1], etc.*
Object	let myVariable = { key : 'value' } ; = { key: [1, 'name', false] }

JavaScript : Functions

A **function** is simply a bunch of code bundled in a section. This bunch of code ONLY runs when the function is called. Functions allow for organizing code into sections and code reusability.

Two Ways in defining a function

As seen in the above example, there are two ways to define a function:

1. Use a function declaration statement in the form of:

```
function functionName ( parameters ) { ..... }
```

2. Use a function expression by assigning an anonymous function to a variable:

```
var functionVarName = function ( parameters ) { ..... }
```

JavaScript : Functions

For example, the following function take a function and an array as its arguments and apply the function to the array.

```
// Define a function, which takes a function and an array as its arguments,
// and apply the function to each item of the array.
function processArray(inFun, inArray) {
   let resultArray = [];
   for (let i in inArray) {
      // apply function to each item of the inArray
      resultArray[i] = inFun(inArray[i]);
   }
   return resultArray;
}
```

JavaScript : Functions

An anonymous function is a function without a name. The following shows how to define an anonymous function:

Syntax:

```
(function () {
   //...
});
```

For example, anonymous function that displays a message:

```
let show = function() {
    console.log('Anonymous function');
};
show();
```

JavaScript : Arrays

An array is an indexed collection. An array can be used to store a list of items (elements) under a single name with a running integer index. You can reference individual element via the integral index in the form of arrayName[index].

```
// Create an array by declaring a variable and assign an array literal
var weekdays = ["sun", "mon", "tue", "wed", "thu", "fri", "sat"];

// You can print out an array
console.log(weekdays);
//[ 'sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat' ]
```

KMITL

JavaScript : Arrays

Arrays also support these iterative methods that iterate through each item of the array, to support functional programming of filter-map-reduce pattern.

- **array.forEach(callback)**: takes a function with an argument which iterates through all the items in the array.

```
var fruits = ['apple', 'orange', 'banana'];
fruits.forEach( function (item) {
   console.log('processing item: ' + item);
});
```

JavaScript : Arrays

- **array.map(callback)**: return a new array, which contains all the return value from executing callback on each item. For example,

```
var fruits = ['apple', 'orange', 'banana'];
var results = fruits.map(function (item) {
   console.log('processing item: ' + item);
   return item.toUpperCase();   // map or transform
});
```

- **array.filter(callback)**: return a new array, containing the items for which callback returned true.

JavaScript : Object

An **object** is a data type in JavaScript that is used to store a combination of data in a simple key-value pair.

```
let user = {
        name : "Aziz Ali",
        yearOfBirth :  1988,
        calculateAge : function() {
        // some code to calculate age   }
}
```

Key
These are the keys in user object.

Value
These are the values of the respective keys in user object.

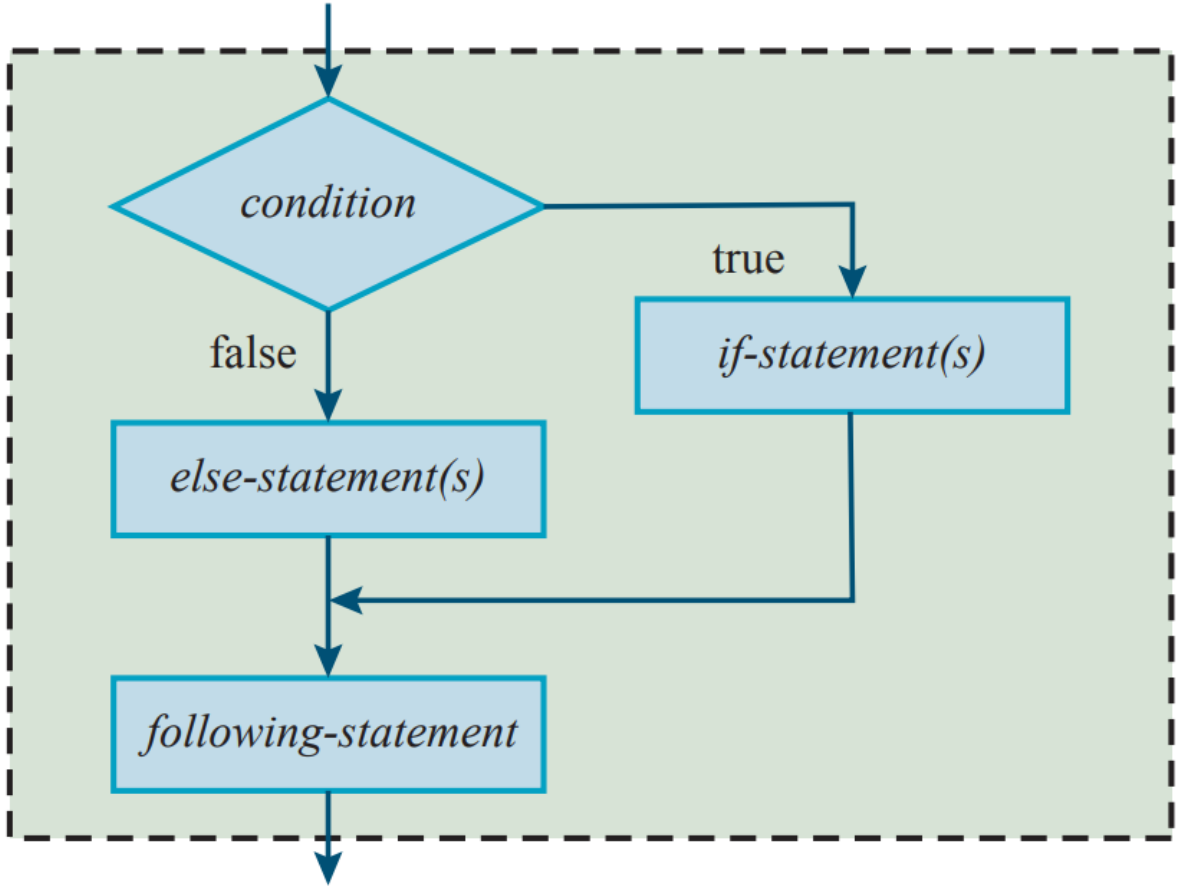
Method
If a key has a function as a value, its called a method.

KMITL

JavaScript : Conditional Statements

If - else statement: Run certain code, "if" a condition is met. If the condition is not met, the code in the "else" block is run (if available.)

```
if (condition) {
   if-statement(s)
}
else {
   else-statement(s)
}
```



KMITL

JavaScript : Conditional Statements

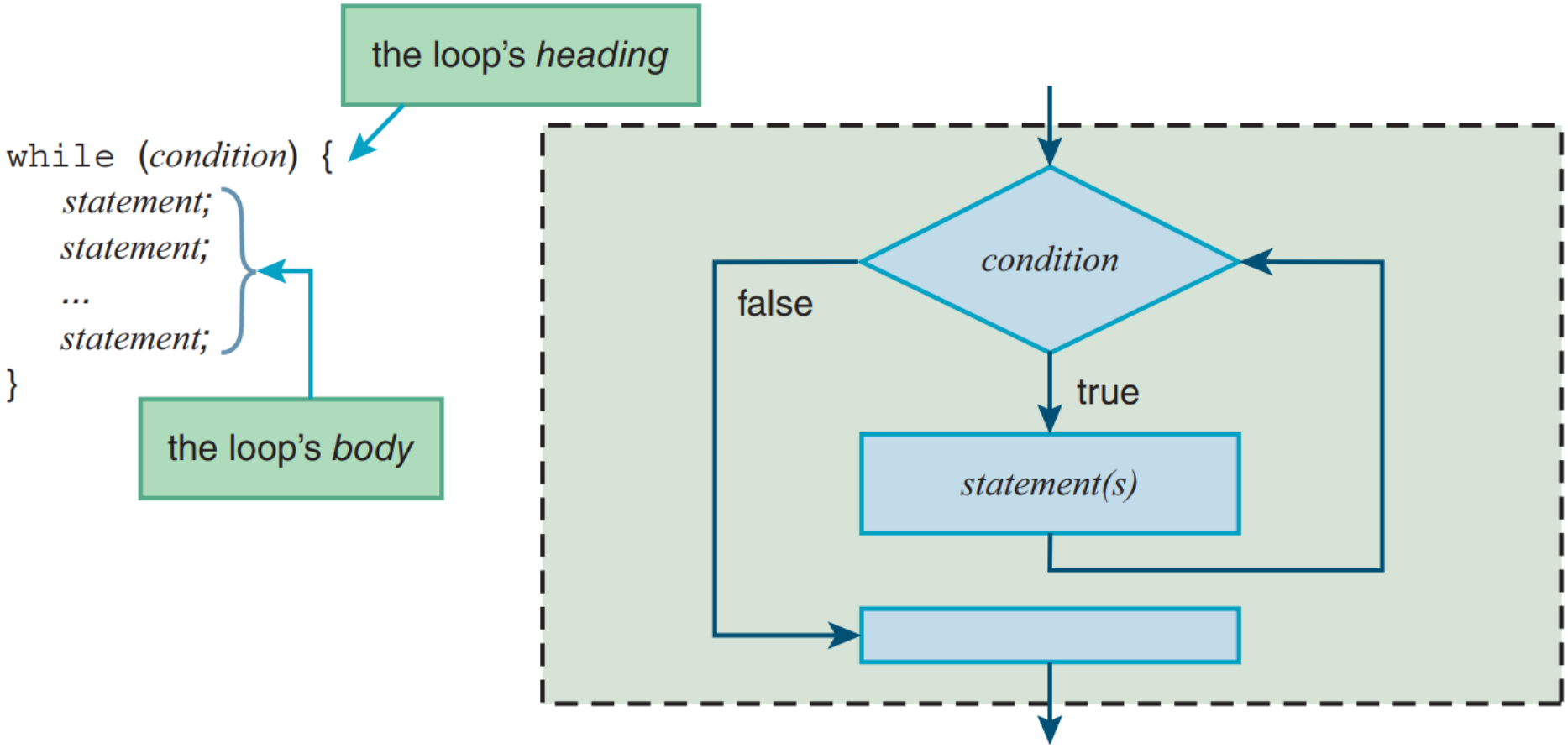
switch statement: Takes a single expression and runs the code of the "case" where the expression matches. The "break" keyword is used to end the switch statement.

```
switch (controlling-expression)
{
   case constant1 :
      statement(s);
      break;
   case constant2 :
      statement(s);
      break;
   .
   .
   .
   default :
      statement(s);
} // end switch
```

optional

JavaScript : Loops and Additional Controls

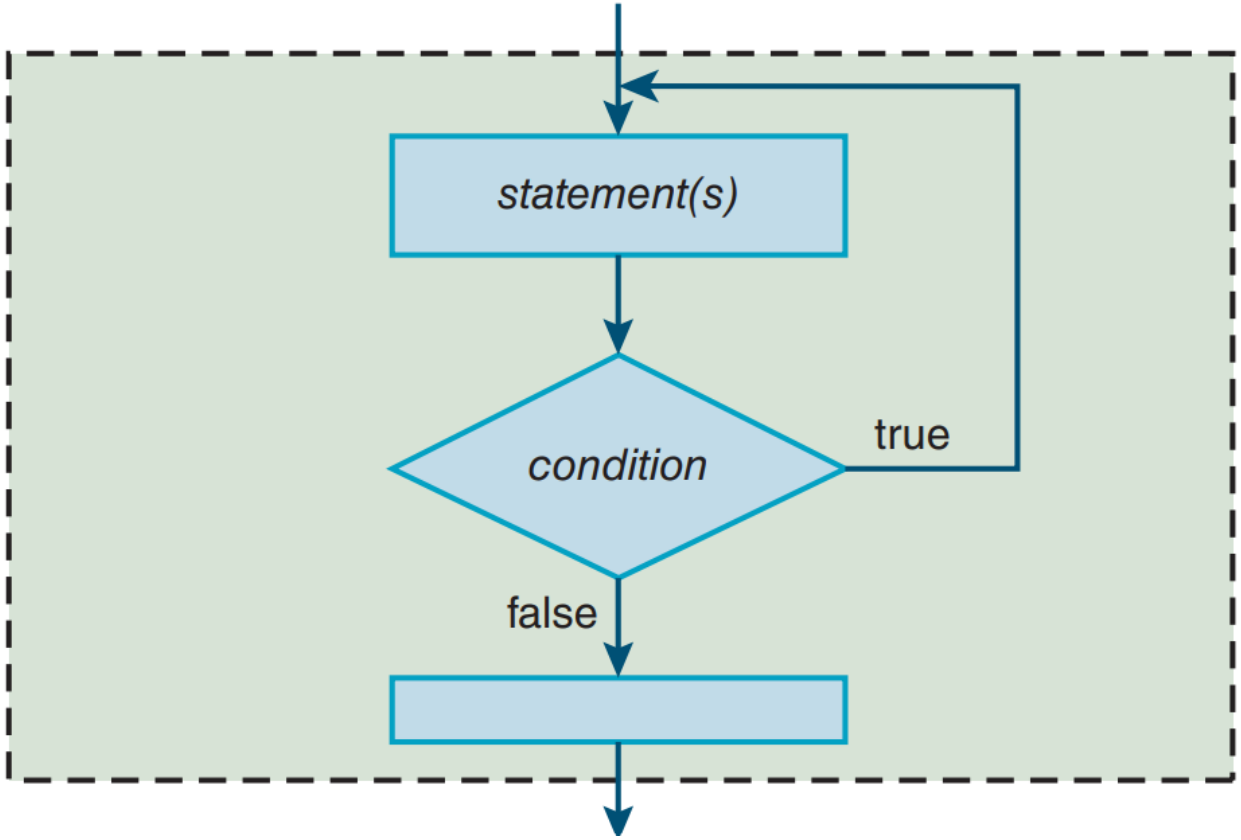
while statement is the most flexible of the three types of loops. You can use it for any task that needs repetitive operations.



JavaScript : Loops and Additional Controls

do-while statement is a control flow statement that executes a block of code at least once, and then either repeatedly executes the block, or stops executing it, depending on a given boolean condition at the end of the block.

```
do {
    statement;
    statement;
    ...
    statement;
} while (condition) ;
```

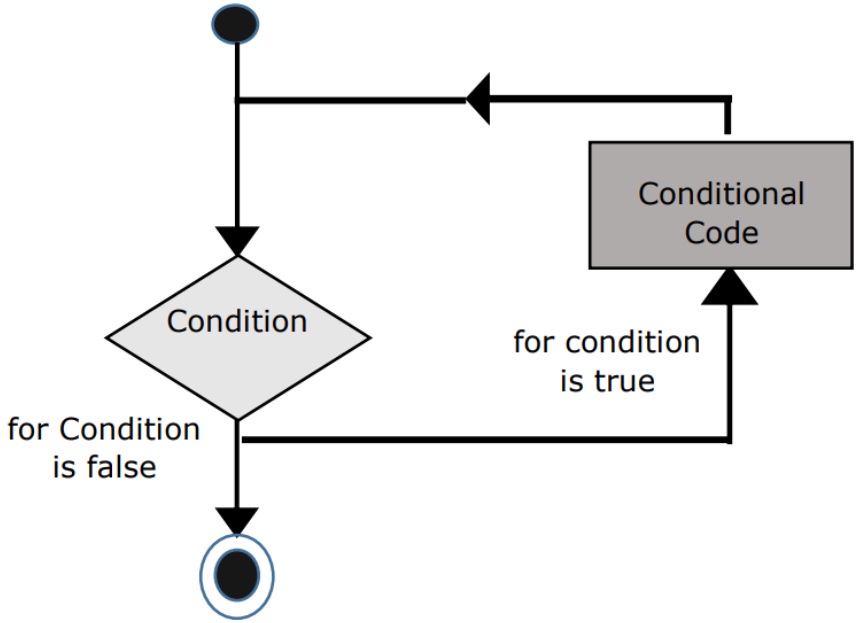


JavaScript : Loops and Additional Controls

The **for** statement is the most compact form of looping. It includes the following three important parts:

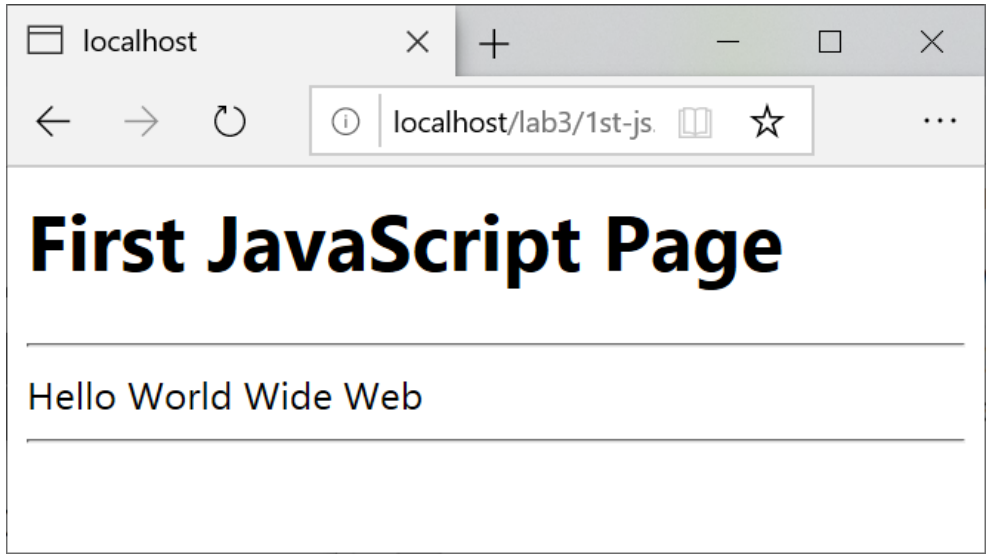
```
for (initialization; test condition; iteration statement){
     Statement(s) to be executed if test condition is true
}
```

- The loop initialization where we initialize our counter to a starting value.
- The test statement which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed.
- The iteration statement where you can increase or decrease your counter.



JavaScript : A Simple Script

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
</head>
<body>
    <h1>First JavaScript Page</h1>
    <script>
        document.write("<hr>");
        document.write("Hello World Wide Web");
        document.write("<hr>");
    </script>
</body>
</html>
```



KMITL

JavaScript : A Simple Script

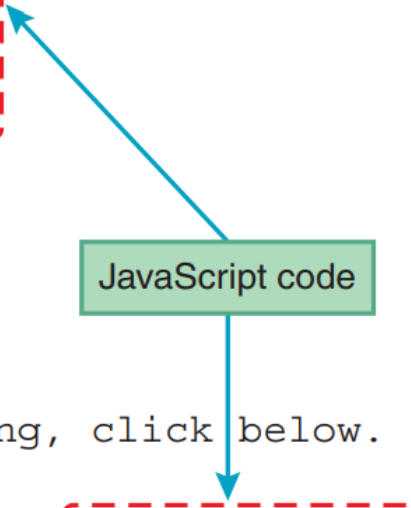
```
  <script>
    alert("Hello, world!");
  </script>
</head>
<body>
  <h1>My first JavaScript says:</h1>
  <script>
    document.write("<h2><em>Hello world, again!</em></h2>");
    document.write("<p>This document was last modified on "
        + document.lastModified + ".</p>");
  </script>
</body>
</html>
```

JavaScript : A Simple Script

Figure: Source code for simple web page

```
<head>
<meta charset="utf-8">
<meta name="author" content="John Dean">
<title>Hello</title>
<script>
 function displayHello() {
   var msg;
   msg = document.getElementById("message");
   msg.outerHTML = "<h1>Hello, world!</h1>";
 }
</script>
</head>

<body>
<h3 id="message">
  To see the traditional first-program greeting, click below.
</h3>
<input type="button" value="Click Me!" onclick="displayHello();">
</body>
</html>
```



JavaScript : Events

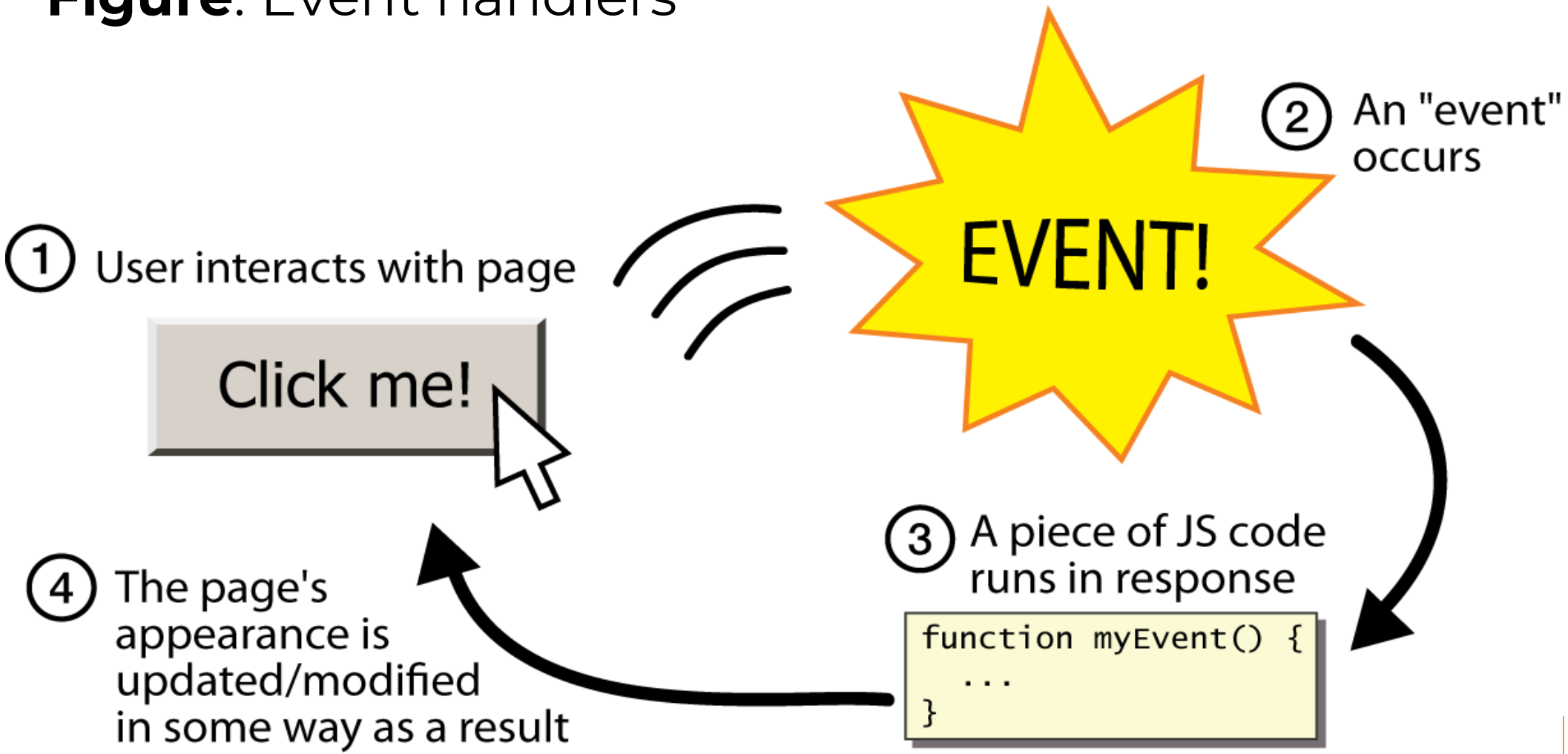
JavaScript in the browser uses an event-driven programming model. JavaScript programs instead wait for user actions called **events** and respond to them.

- Event-driven programming is a writing programs driven by **user events**.
- The event could be the DOM is loaded, or an asynchronous request that finishes fetching, or a user clicking an element or scrolling the page, or the user types on the keyboard.
- There are a lot of different kind of events.

KMITL

JavaScript : Events

Figure: Event handlers



JavaScript : Buttons

There are different types of buttons, each with its own syntax. To keep things simple, we'll start with just one type of button, and here's its syntax:

```
<input type="button"
  value="button-label"
  onclick="click-event-handler">
```

JavaScript : Buttons

```
window.onload = init;

function init() {
  document.getElementById("btn1").onclick = changeHeading1;
  document.getElementById("btn2").onclick = changeHeading2;
  document.getElementById("btn3").onclick = changeParagraph;
}

function changeHeading1() {
  var elm = document.getElementById("heading1");  // One element
  console.log(elm.innerHTML);  // Print current value before modificat
  elm.innerHTML = "Hello";
}
```

KMITL

JavaScript : Accessing elements

Accessing elements:

```
document.getElementById("id")
document.getElementByClassname("Classname")
document.getElementByName("Name")
document.getElementByTagName("TagName")
document.forms["Form_name"]["ElementId"]
```

document.getElementById returns the DOM object for an element with a given id. It can change the text inside most elements by setting the **innerHTML** property. Also, It can change the text in form controls by setting the value property.

KMITL

Accessing elements

HTML

```html
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```

JavaScript

```javascript
function changeText() {
    let span = document.getElementById("output");
    let textbox = document.getElementById("textbox");
    textbox.style.color = "red";
}
function getValue() {
    let fname = document.forms["myForm"]["FirstName"].value;
}
```

Accessing elements

Figure: body container for Email Address Generator web page

```
<body>
<h3>
  Enter your first and last names and then click the button.
</h3>
<form>
  First Name:
  <input type="text" id="first" size="15" autofocus>
  <br>
  Last Name:
  <input type="text" id="last" size="15">
  <br><br>
  <input type="button" value="Generate Email"
    onclick="generateEmail(this.form);">
</form>
<p id="email"></p>
</body>
</html>
```

Use `autofocus` for the first-name text control.

The `this` keyword refers to the object that contains the JavaScript in which `this` appears. In this example, the enclosing object is the button element's object.

Accessing elements

Figure: head container for Email Address Generator web page

```
<meta name="author" content="John Dean">
<title>Email Address Generator</title>
<script>
  // This function generates an email address.

  function generateEmail(form) {
    document.getElementById("email").innerHTML =
      form.elements["first"].value + "." +
      form.elements["last"].value + "@park.edu";
    form.reset();
    form.elements["first"].focus();
  } // end generateEmail
</script>
```

Parameter that holds the form object.

More Information

- JavaScript Tutorial
  https://www.w3schools.com/js/default.asp
- JavaScript Tutorial
  https://www.javascripttutorial.net/
- Javascript Tutorial
  https://www.tutorialspoint.com/javascript/

KMITL