1

Basic Technologies

- Hypertext Markup Language
- Cascading Style Sheets

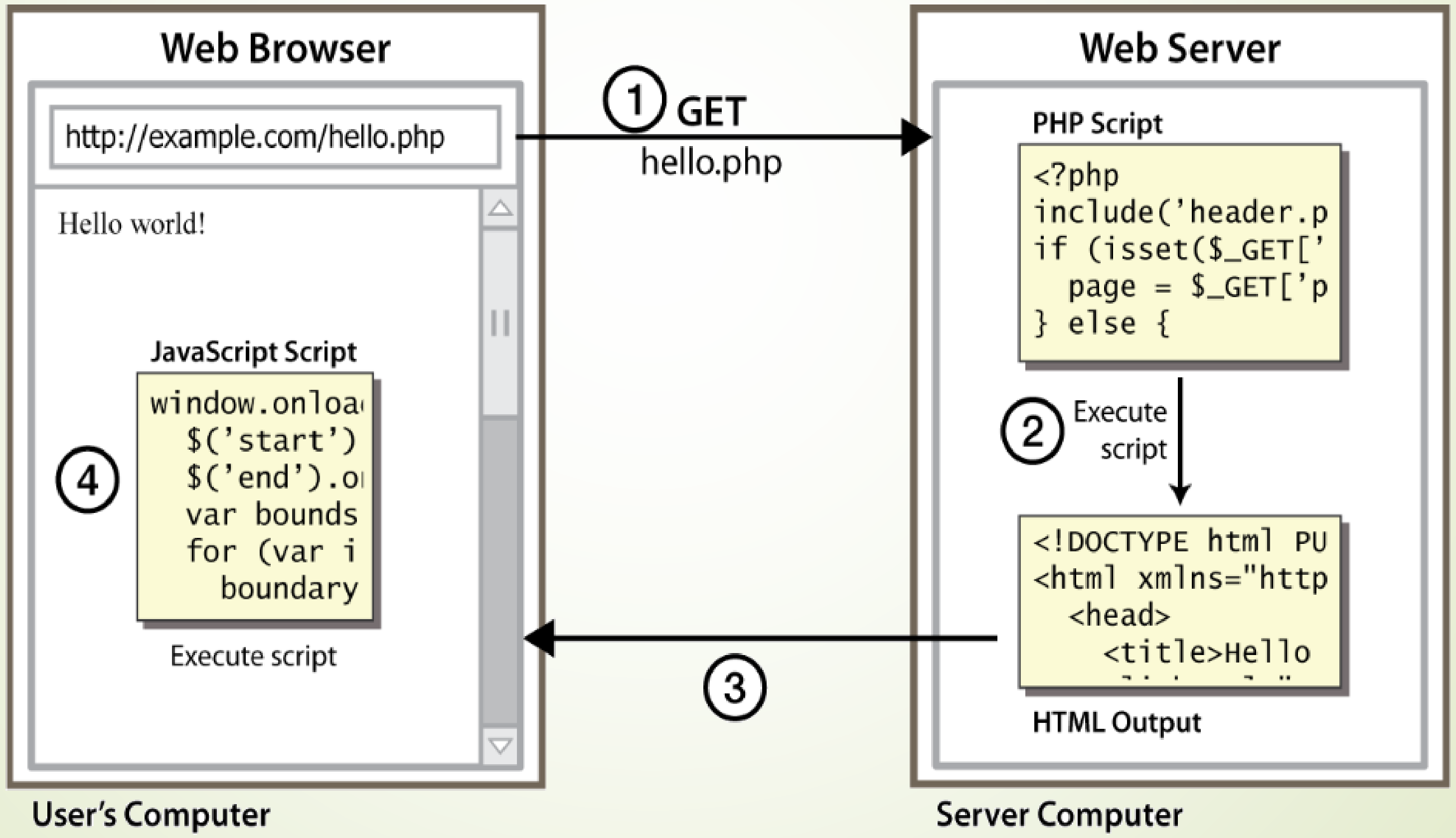
2

Interactive Web

- JavaScript
- Critical Rendering Path
- Json & Ajax
- jQuery
- React

JavaScript

Client-side Scripting



```
window.onloa
  $('start')
  $('end').o
  var bounds
  for (var i
    boundary
```

```
<?php
include('header.p
if (isset($_GET['
  page = $_GET['p
} else {
```

```
<!DOCTYPE html PU
<html xmlns="http
  <head>
    <title>Hello
```

JavaScript

Client-side Scripting

- benefits:
  - 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

JavaScript

JavaScript

- a lightweight programming language ("scripting language")
  - used to make web pages interactive
  - insert dynamic text into HTML (e.g. a date)
  - react to events (e.g. user clicks on a button)
  - get information about a user's computer (e.g. browser type)
  - perform calculations on user's computer (e.g. form validation)

JavaScript

Attaching in HTML

- can be included in HTML using the <script> tag. Unlike CSS JavaScript can be included anywhere in the HTML document:

```
<script type="text/javascript">
   alert('This is an alert triggered by JavaScript');
</script>
```

- can be triggered by event handler attributes added to HTML tags. Event handlers respond to actions the user makes on the web-page.

```
<div onclick="alert('JavaScript Alert!');">
   Click Me
</div>
```

II.2

JavaScript

Basics

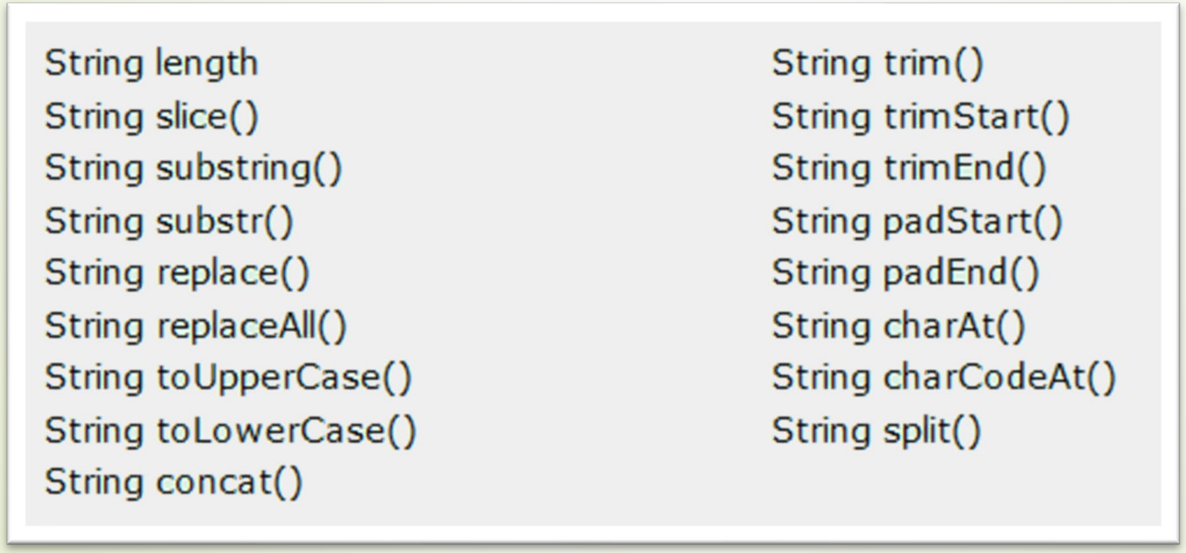
- variables, assignments
- special values: null and undefined
- concatenation and operators
- control structures

- Playgrounds/Sandboxes
  - https://playcode.io/
  - https://jsfiddle.net/

JavaScript

Basics

- string
  - usual functions



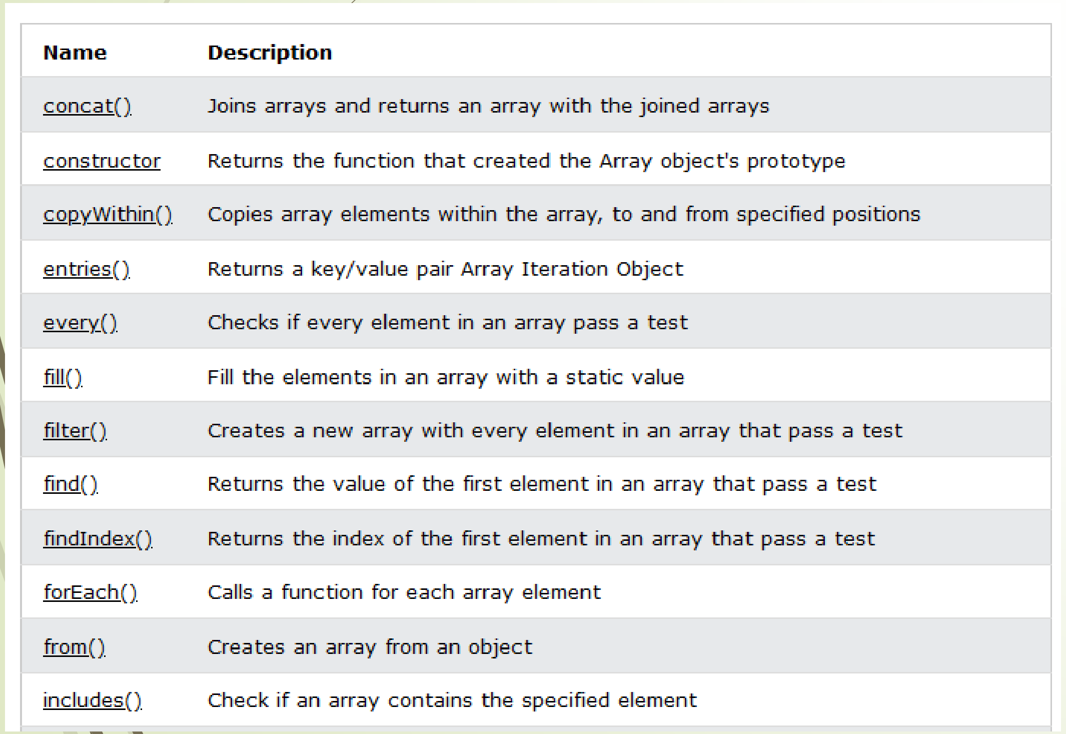
String length	String trim()
String slice()	String trimStart()
String substring()	String trimEnd()
String substr()	String padStart()
String replace()	String padEnd()
String replaceAll()	String charAt()
String toUpperCase()	String charCodeAt()
String toLowerCase()	String split()
String concat()	

II.2

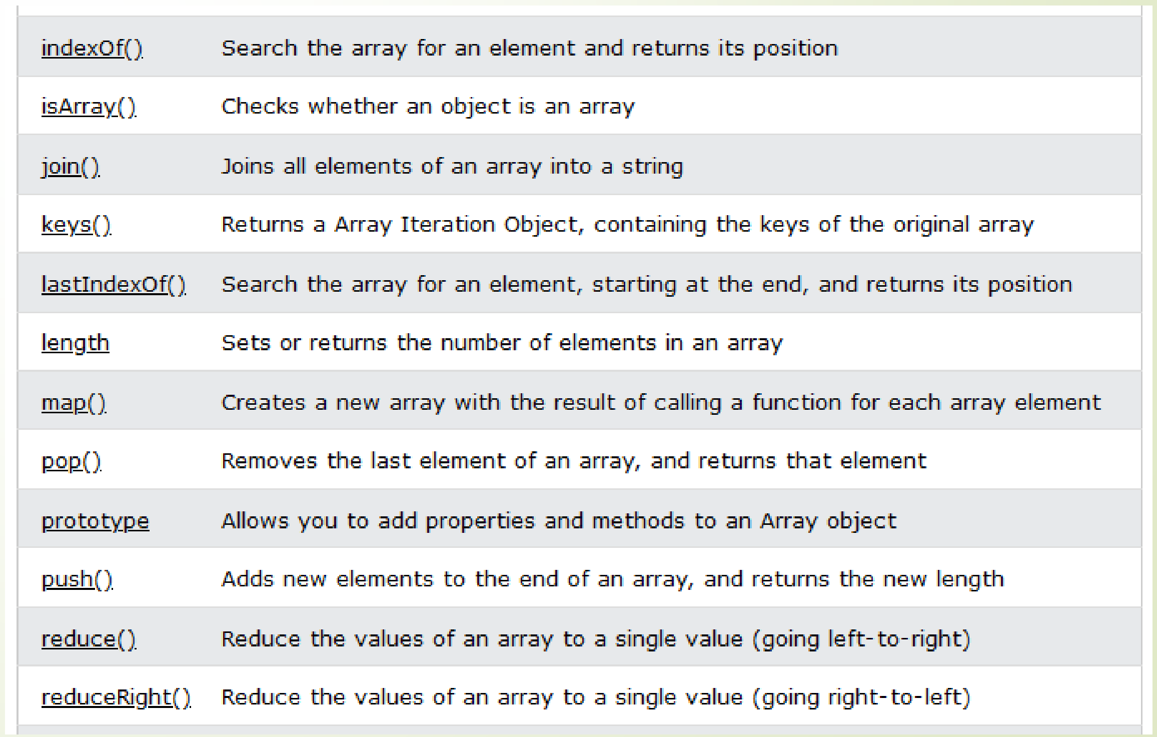
JavaScript

Basics

- arrays
  - creation   `var a = [10, 20, 30]`



Name	Description
concat()	Joins arrays and returns an array with the joined arrays
constructor	Returns the function that created the Array object's prototype
copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if every element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array that pass a test
findIndex()	Returns the index of the first element in an array that pass a test
forEach()	Calls a function for each array element
from()	Creates an array from an object
includes()	Check if an array contains the specified element



indexOf()	Search the array for an element and returns its position
isArray()	Checks whether an object is an array
join()	Joins all elements of an array into a string
keys()	Returns a Array Iteration Object, containing the keys of the original array
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
length	Sets or returns the number of elements in an array
map()	Creates a new array with the result of calling a function for each array element
pop()	Removes the last element of an array, and returns that element
prototype	Allows you to add properties and methods to an Array object
push()	Adds new elements to the end of an array, and returns the new length
reduce()	Reduce the values of an array to a single value (going left-to-right)
reduceRight()	Reduce the values of an array to a single value (going right-to-left)

II.2

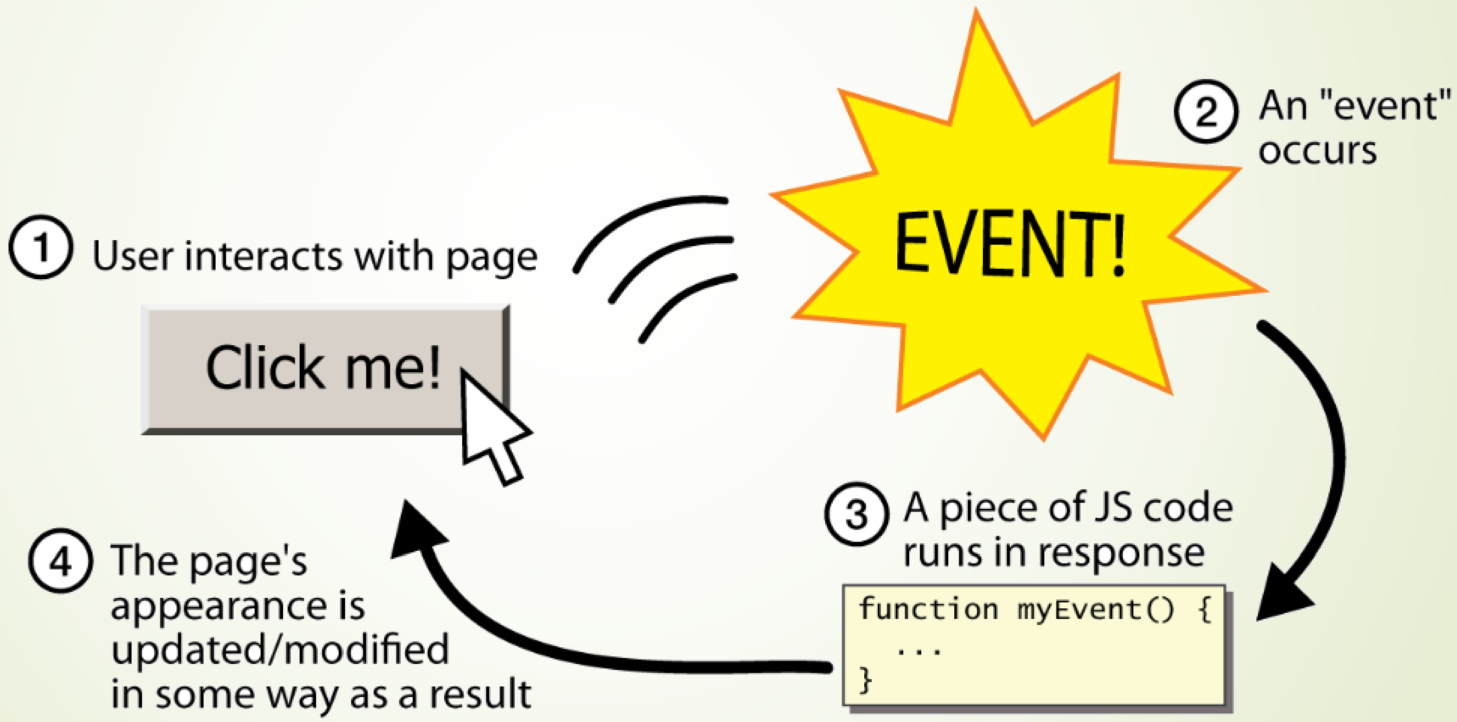
JavaScript

Functions

```
<html>
   <head>
      <script type = "text/javascript">
         function sayHello() {
            document.write ("Hello there!");
         }
      </script>
   </head>

   <body>
      <p>Click the following button to call the function</p>
      <form>
         <input type = "button" onclick = "sayHello()" value = "Say Hello">
      </form>
      <p>Use different text in write method and then try...</p>
   </body>
</html>
```

JavaScript

Event-driven programming


1 User interacts with page
Click me!
2 An "event" occurs
EVENT!
3 A piece of JS code runs in response
4 The page's appearance is updated/modified in some way as a result

```
function myEvent() {
  ...
}
```

JavaScript

Event-driven programming

- event handlers

  function that’s called when an event occurs

  - inline event handlers

```
<a href="site.com" onclick="dosomething();">A link</a>
```

  - DOM on-event handlers

```
window.onload = () => {
  //window loaded
}
```

II.2

JavaScript

Event-driven programming

- event handlers
  - using addEventListener

```
window.addEventListener('load', () => {
  //window loaded
})
```

JavaScript

Event-driven programming

- event object

```
link.addEventListener('click', event => {
  // link clicked
})
```

- ➢ target
  the DOM element that originated the event
- ➢ type
  the type of event
- ➢ stopPropagation()
  called to stop propagating the event in the DOM

JavaScript

BOM – Browser Object Model

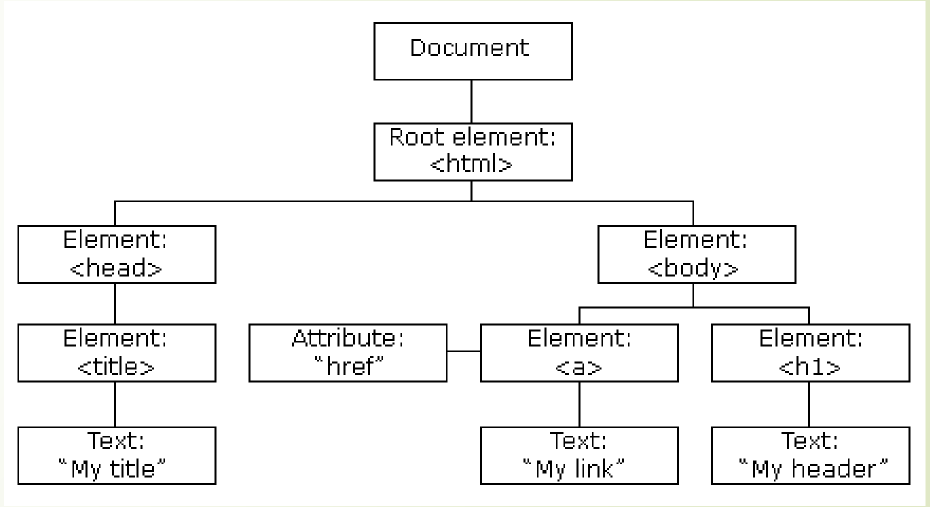
- objects in a Web document
  - window
    object of the browser, represents a window in a browser
    methods: alert, prompt, ...
  - window.screen
    contains information about the user's screen
  - window.location
    can be used to get the current page address (URL) and to redirect the browser to a new page
  - window.navigator
    contains information about the visitor's browser

II.2

JavaScript

DOM – Document Object Model

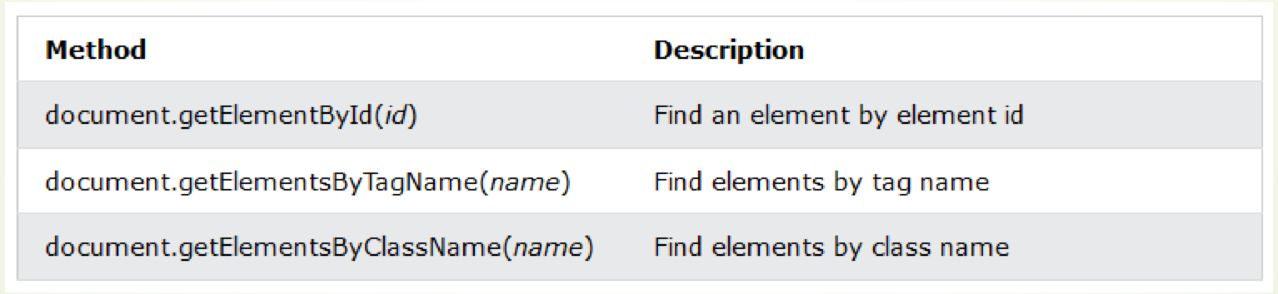
- JavaScript can
  - change all the HTML elements in the page
  - change all the HTML attributes in the page
  - change all the CSS styles in the page
  - remove existing HTML elements and attributes
  - add new HTML elements and attributes
  - react to all existing HTML events in the page
  - create new HTML events in the page



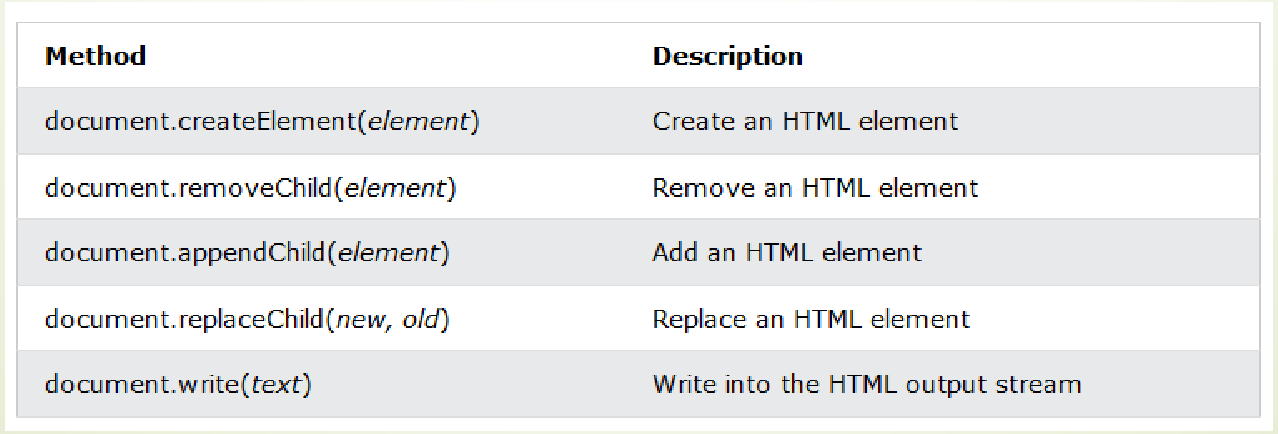
JavaScript

DOM – Document Object Model

- document
  is the owner of all other objects



Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name



Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

JavaScript

DOM – Document Object Model

- example: form validation

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

```
 <form name="myForm" action="/action_page.php"
    onsubmit="return validateForm()" method="post">

    Name:
        <input type="text" name="fname">
        <input type="submit" value="Submit">
</form>
```

JavaScript

Objects

- are used to store keyed collections of various data and more complex entities
- an be created with figure brackets {…} with an optional list of properties (“key: value” pairs)

```
let user = {
  name: "John",
  age: 30,
  "likes birds": true  // multiword property name must be quoted
};
```

- access via dot or square-bracket
- existence test: "in" operator
- "for ... in" loop

II.2

JavaScript

Objects

- methods in objects

```
let user = {
  name: "John",
  age: 30,
  "likes birds": true  // multiword property name must be quoted

 sayHi: function() {
    alert("Hello");
  }
};
```

- this = "self in Python"

JavaScript

Objects

- constructor, operator "new"
  - technically are regular functions. There are two conventions though:
    - they are named with capital letter first
    - they should be executed only with "new" operator

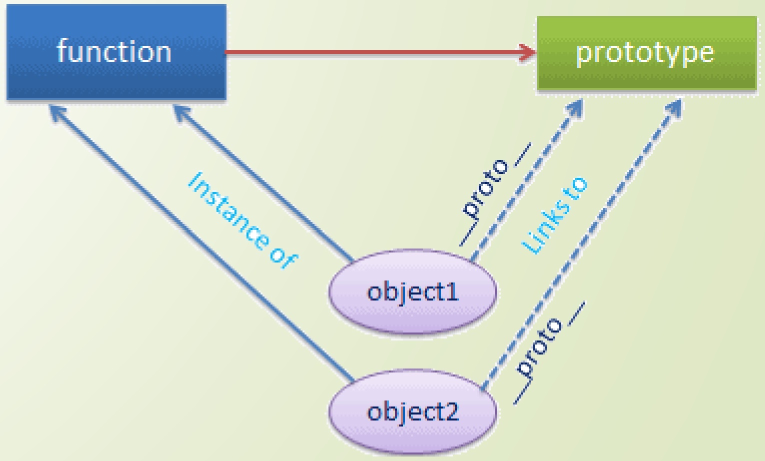
```
function User(name) {
  this.name = name;
  this.isAdmin = false;
}

let user = new User("Jack");
```

II.2

JavaScript

Objects

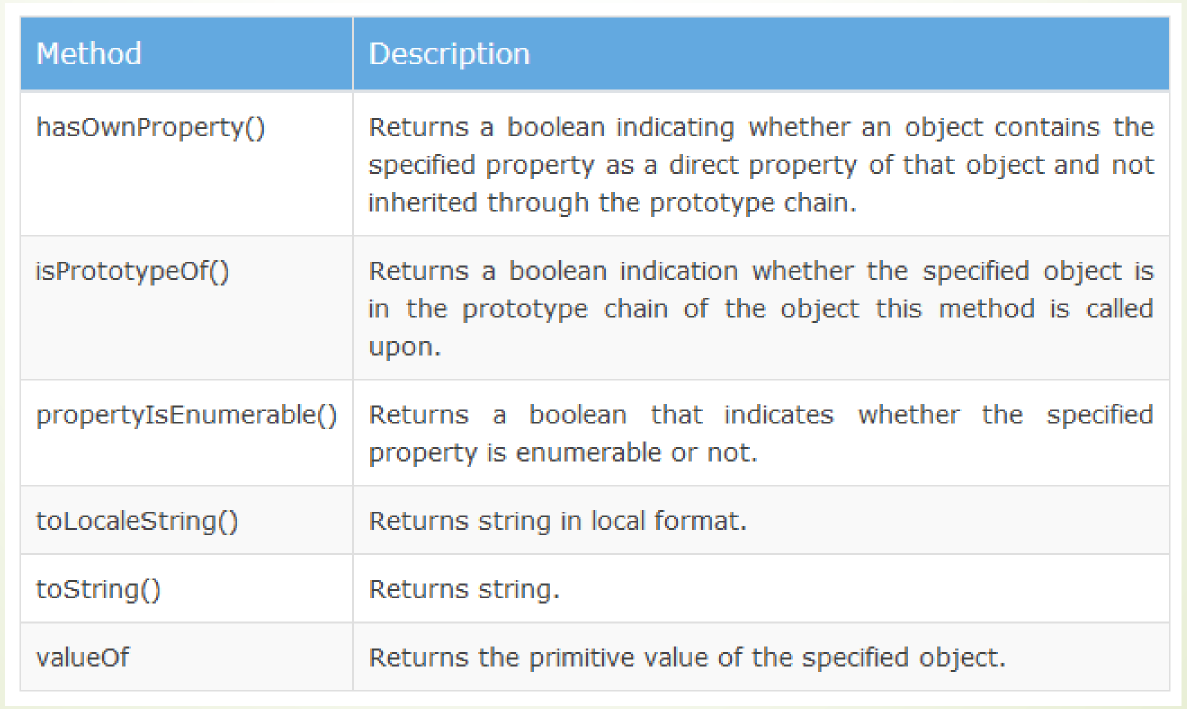
- prototype
  - an object that is associated with every functions and objects by default
    - function's prototype property is accessible and modifiable
    - object's prototype property is not visible
  - prototype object is special type of enumerable object to which additional properties can be attached to it which will be shared across all the instances of it's constructor function



JavaScript

Objects

- prototype methods

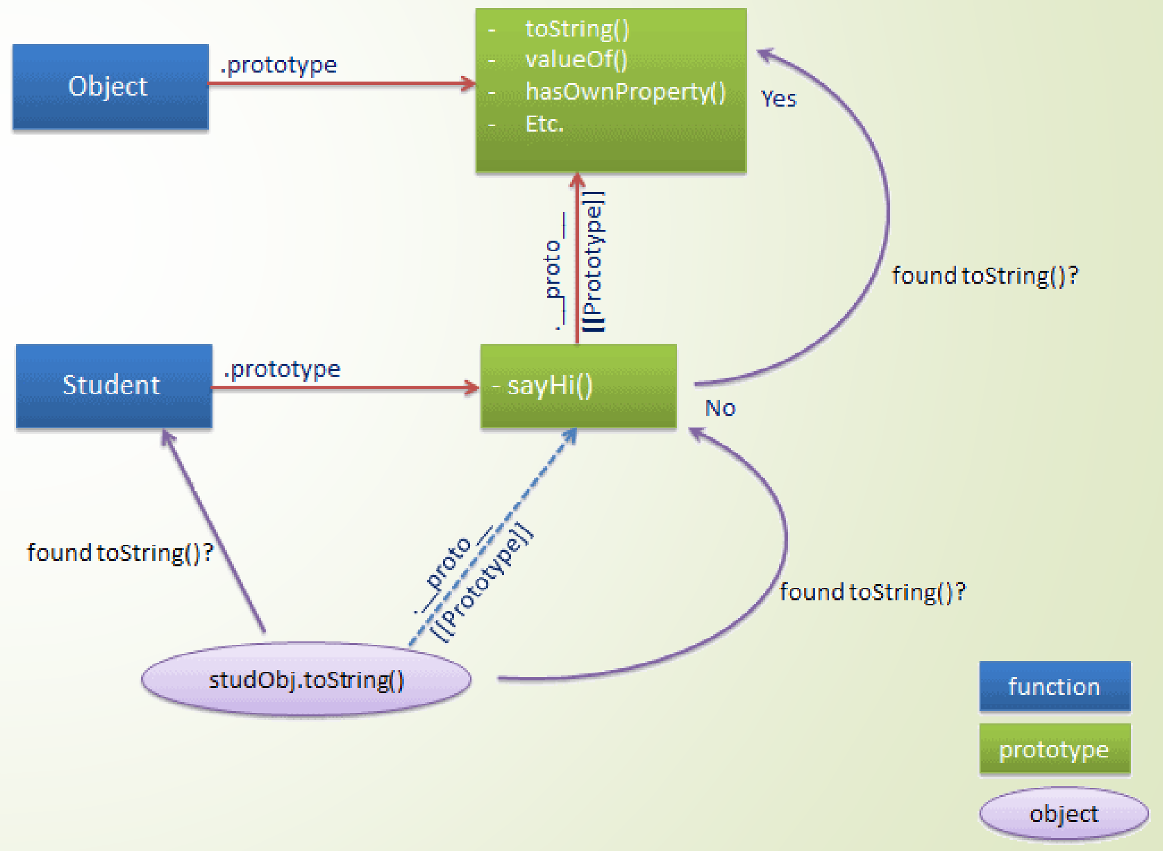


Method	Description
hasOwnProperty()	Returns a boolean indicating whether an object contains the specified property as a direct property of that object and not inherited through the prototype chain.
isPrototypeOf()	Returns a boolean indication whether the specified object is in the prototype chain of the object this method is called upon.
propertyIsEnumerable()	Returns a boolean that indicates whether the specified property is enumerable or not.
toLocaleString()	Returns string in local format.
toString()	Returns string.
valueOf	Returns the primitive value of the specified object.

JavaScript

Objects

- prototype usage
  - to find properties and methods of an object
  - to implement inheritance



JavaScript

Objects

- inheritance

```
function Person(firstName, lastName) {
    this.FirstName = firstName;
    this.LastName = lastName;
};

Person.prototype.getFullName = function () {
    return this.FirstName + " " + this.LastName;
}
```

```
function Student(firstName, lastName, schoolNamee) {
    Person.call(this, firstName, lastName);

    this.SchoolName = schoolName;
}

Student.prototype = new Person();
Student.prototype.constructor = Student;

var std = new Student("James","Bond", "XYZ");

std.getFullName();
```