



# FRONTEND

# JAVASCRIPT





# Javascript



- **Javascript** is the programming language of the web.
- Javascript allow access and control HTML element in webpage.
- Javascript allows making dynamic webpage from static webpage.
- One of the main reasons we use Javascript to build interactivity is because it allows us to manipulate the Document Object Model (or DOM) for short.





# Javascript



- Using Javascript, we can manipulate the DOM to do things like:
  - Modify existing HTML elements
  - Modify attributes on HTML elements
  - Add or modify the CSS associated with HTML elements
  - Add and delete HTML elements





# Variable



- The var keyword is used to declare a variable.
  - Ex: `var text="new text";`  
`var button = document.querySelector('button');`
- Variable always has a data type associated with it that tells the computer exactly how to handle the data that it's given. In Javascript, the data's type is automatically determined when the code is executed
- You can use method ***console.log()*** to output data to console webpage.





# JS Operator



## - Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Power Operator
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement





# JS Operator



## - Assignment Operators

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>
**=	<code>x **= y</code>	<code>x = x ** y</code>





# JS Operator



## - Comparison Operators

perator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator





# JS Operator



## - Logical Operators

Operator	Description
&&	logical and
	logical or
!	logical not





# JS Operator



## – Bitwise Operators

perato r	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5   1	0101   0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2
>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2





# JS Operator



## - Logical Operators

Operator	Description
&&	logical and
	logical or
!	logical not





# JS Condition



- Conditional statements are used to perform different actions based on different conditions.
- Syntax

```
if (condition) {  
    // execute if the condition is true  
}
```

```
if (condition) {  
    //execute if the condition is true  
} else {  
    // execute if the condition is false  
}
```



12



# JS Condition



## - Syntax

```
if (condition1) {  
    //execute if condition1 is true  
} else if (condition2) {  
    //execute if the condition1 is false and  
    condition2 is true  
} else {  
    //execute if the condition1 is false and  
    condition2 is false  
}
```





# JS Switch



```
switch(expression) {  
  case x:  
    // code block  
    break;  
  case y:  
    // code block  
    break;  
  default:  
    // code block  
}
```





# JS For Loop



- For loop execute a block of code a number of times.
- Syntax

```
for (for init; for condition; for update) {  
    // code block to be executed  
}
```

- For Each: loops through the values of an iterable objects (ex: array)

```
for (variable of iterable) {  
    // code block to be executed  
}
```



15



# JS While Loop



- While loop execute a block of code until condition is false.
- Syntax

```
while (condition) {  
    // code block to be executed  
}
```







# JS do while Loop



- Do while loop similar while loop but do while loop will execute the code block once, before checking condition.
- Syntax

```
do {  
    // code block to be executed  
}  
while (condition);
```





# JS Function (Method)



## - Define:

```
function name(parameter1, parameter2, ...) {  
    // code here  
}
```

```
var name = function (parameter1, parameter2, ...) {  
    // code here  
}
```





# Data Structures: Arrays



- Arrays are used to store multiple values in a single variable.

```
var phone = ["Apple", "Sam Sung", "Sony"];
```

- JS Arrays can store any valid data type.

```
var data = ["Tí", "Nguyễn Vannw", 20];
```

- JS Arrays don't limit to storing data.





# Arrays



- Declaration:
  - Way 1: Use [] instead.
  - Way 2: new Array().

```
var data = [];           // Good  
var data = new Array();  // Bad
```

- Access data: You access an array element by referring to the **index number**. Array indexes start with 0.

```
var cars = ["Saab", "Volvo", "BMW"];  
var myCar = cars[2];
```





# Arrays



- Add data:
  - Use the `Array.push()` method to add items to the end of an array.

```
var data = [];  
data.push("Lemon"); //or  
data[1]="Apple";
```





# Arrays



## - Remove data:

- Method `Array.pop()` to remove the last item in the array and returns the removed item.
- Method `Array.shift()` to remove the first item in the array and returns the removed item.
- `Array.unshift()` adds an item to the beginning of an array. `Array.unshift()` returns the new size of the array.

```
var data = [];  
data.push("Lemon"); //or  
data[1]="Apple";  
var item = data.shift();//remove Lemon, return lemon  
var size = data.unshift();//add Lemon, return 2  
var item1 = data.pop(); //remove Apple, return Apple
```



# Arrays



## - Edit data:

- To edit data we use index num to access.
- Index number: start: 0; end: n-1.

```
var data = [];  
data.push("Lemon"); //or  
data[1]="Apple";  
data[1]="Orange"; // change Apple → Orange
```

## - Array size:

- **Array.length** will return the number of items currently in the array.





# Arrays



## - Retrieve Arrays

```
var printArray = function(arrays){  
    for(var i = 0; i < arrays.length; i++) {  
        console.log(arrays[i]);  
    }  
}  
printArray(data);
```

```
var printArray = function(arrays){  
    for(var a of arrays){  
        console.log(a);  
    }  
}  
printArray(data);
```



24





# Arrays



- Print array or convert array to string
  - The **Array.toString()** method will take an array and convert it into a string, with items separated by commas.
  - The **console.log(array)** method will show all data of array to browser console;





# Data Structures: Object



- JS Object is a special data type that can contain multiple data type.
- Data of JS Object is called properties.
- Define:
  - We use "{}" to create Object.
  - A properties of the object have 2 parts: "name:value"

```
var person = {  
  firstName: "Tí",  
  lastName: "Nguyễn Văn",  
  age: 20,  
  class: "CNTT"  
};
```



# Data Structures: Object



## – Accessing Object Properties

- `object.properties`
- `object['properties']`

```
var fName = person.lastName;  
var lName = person["firstName"];
```

## – Add or modify properties:

```
person["firstName"] = Teo ;  
person.score        = 8.2;
```





# Data Structures: Object



- Remove properties: using the **delete** keyword to remove a properties.

```
delete person.score;
```

- Retrieve Arrays

```
for(var properties of person){  
    console.log(properties);// get properties name  
    console.log(person[properties]);// get  
        //properties value  
}
```





# JS HTML DOM

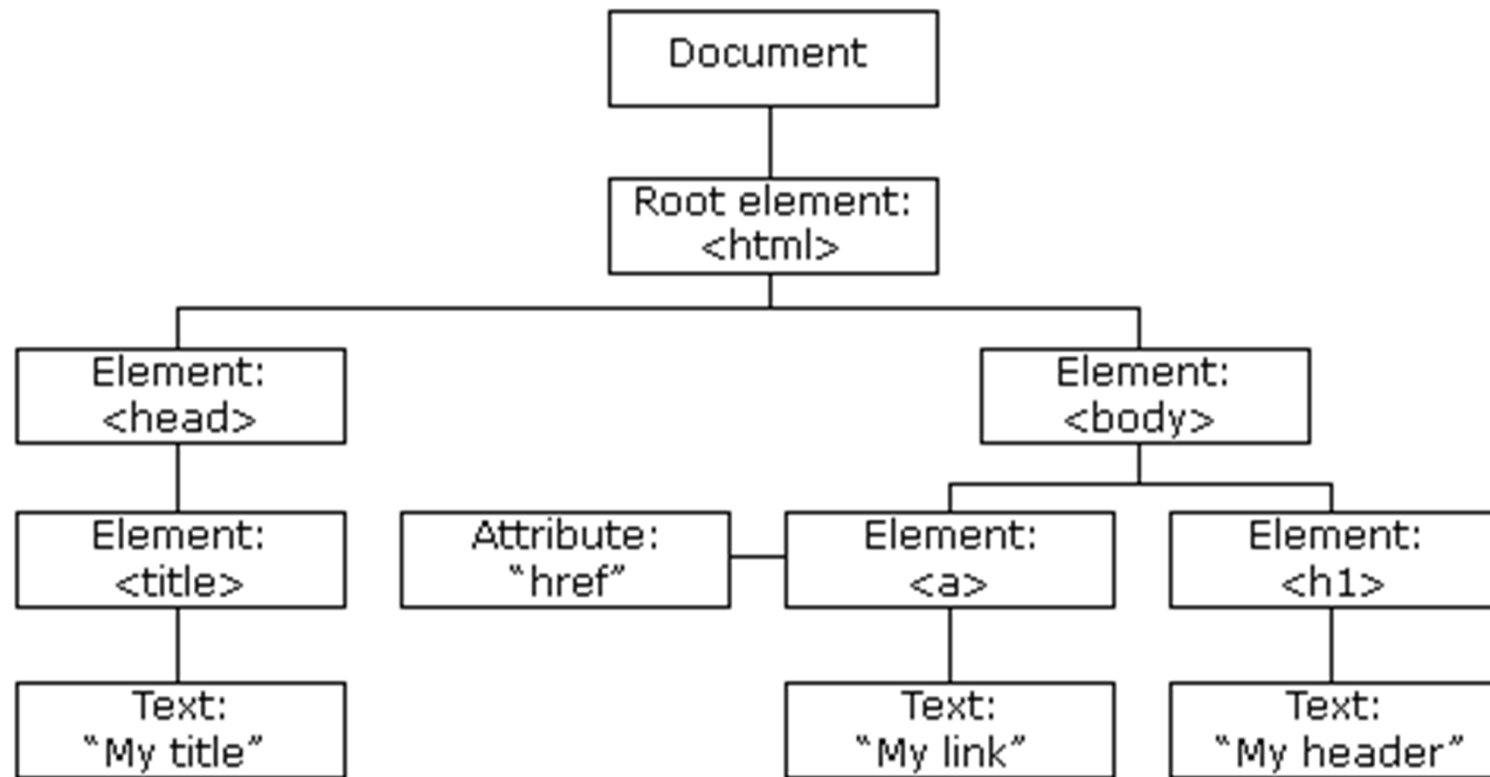


- **Working with Element Nodes:** Each element on your HTML page has an associated **element node** in the *Document Object Model (DOM)*.
- The DOM can be accessed using Javascript through the document object.
- ***document*** object behaves like any other Javascript object, and has many different properties and methods that are useful in accessing nodes in the DOM.





# JS HTML DOM





# JS HTML DOM



- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as objects
  - The properties of all HTML elements
  - The methods to access all HTML elements: add, get, modify, remove.
  - The events for all HTML elements





# DOM Methods



- Access element
  - `document.querySelector()`: return the **first** element it finds with the selector provided. If not found return null.
  - `document.getElementById(id)`: get element with an id attribute.
  - `document.getElementsByTagName(tag)`: get element with an html name tag.
  - `document.getElementsByClassName(class)`: get element with class attribute.







# DOM Methods



```
var app = document.querySelector("#app");  
var name = document.getElementById("name");  
var table = document.getElementsByTagName("table");  
var title = document.getElementsByClassName("title");
```





# DOM Methods



- Create Elements:

**`document.createElement(element)`**

- Insert Elements:

**`Element.appendChild(element_node)`**

```
var h1 = document.createElement('h1');  
h1.innerHTML = "A new HTML element!";  
  
var content = document.querySelector('#content');  
content.appendChild(h1);
```





# DOM Methods



- Remove Element: To remove an HTML element, you must know the parent of the element:

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

```
var child = document.getElementById("p1");
child.parentNode.removeChild(child);
```



# DOM Modifying Attributes



- Add, Remove, and View Element Attributes:
  - `Element.getAttributeNames()`
  - `Element.getAttribute(name)`
  - `Element.setAttribute(name, value)`
  - `Element.removeAttribute(name)`

```
var fName = person.lastName;  
var lName = person["firstName"];
```



36



# JS Mouse Event



Event	Description
<b>onclick</b>	The event occurs when the user clicks on an element
<b>oncontextmenu</b>	The event occurs when the user right-clicks on an element to open a context menu
<b>ondblclick</b>	The event occurs when the user double-clicks on an element
<b>onmousedown</b>	The event occurs when the user presses a mouse button over an element
<b>onmouseenter</b>	The event occurs when the pointer is moved onto an element
<b>onmouseleave</b>	The event occurs when the pointer is moved out of an element
<b>onmousemove</b>	The event occurs when the pointer is moving while it is over an element
<b>onmouseout</b>	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children
<b>onmouseover</b>	The event occurs when the pointer is moved onto an element, or onto one of its children
<b>onmouseup</b>	The event occurs when a user releases a mouse button over an element



 **jQuery**  
*write less, do more.*



# jQuery



- jQuery is JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- The jQuery library contains the following features:
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX
  - Utilities



39



# Advantages of Using jQuery



- **Save lots of time** — You can save lots of time and efforts by using the jQuery inbuilt effects and selectors and concentrate on other development work.
- **Simplify common JavaScript tasks** — jQuery considerably simplifies the common JavaScript tasks. Now you can easily create feature rich and interactive web pages with fewer lines of codes, a typical example is implementing Ajax to update the content of a page without refreshing it.



40





# Advantages of Using jQuery



- **Easy to use** — jQuery is very easy to use. Anybody with the basic working knowledge of HTML, CSS and JavaScript can start development with jQuery.
- **Compatible with browsers** — jQuery is created with modern browsers in mind and it is compatible with all major modern browsers such as Chrome, Firefox, Safari, Internet Explorer, etc.
- **Absolutely Free** — And the best part is, it is completely free to download and use.



41



# Getting Start



## – Downloading jQuery

- You can download jQuery from here:  
<https://jquery.com/download/>
- Once you've downloaded the jQuery file you can see it has .js extension, because the jQuery is just a JavaScript library, therefore you can include the jQuery file in your HTML document with the `<script>` element just like you include normal JavaScript files.



42



# Getting Start



- To run jQuery we must reference to jQuery lib.
- 2 way to reference:
  - jQuery CDN: include cdn link.

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

- jQuery local lib:
  - Download latest version from jQuery homepage <https://jquery.com/download/>.
  - Extract and link lib file to html page.

```
<script src="jquery-3.5.1.min.js"></script>
```



43



# Getting Start



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Simple HTML Document</title>
    <link rel="stylesheet href="css/style.css">
    <script src="js/jquery-3.5.1.min.js"> </script>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```



44



# Demo



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First jQuery Powered Web Page</title>
  <link rel="stylesheet" href="css/style.css">
  <script src="https://code.jquery.com/jquery-3.5.1.min.js">
  </script>
  <script>
    $(document).ready(function(){
      $("h1").css("color", "#0088ff");
    });
  </script>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```



# Demo



**Hello, World!**



46



# jQuery Syntax



- **`$(selector).action()`**
  - "\$": character to access jQuery.
  - **`$(selector)`**: to access to selector matching.
  - `action()`: method of jQuery.

`$(this).show()` - show the current element.

`$("a").hide()` - hides all `<a>` elements.

`$(".item").hide()` - hides all elements with `class="item"`.

`$("#item").hide()` - hides the element with `id="item"`.





# jQuery Running



- Document ready event: all jQuery methods must be declared inside document ready event
  - Way 1:

```
$(document).ready(function(){  
    // jQuery methods  
});
```

- Way 2:

```
$(function(){  
    // jQuery methods  
});
```



48





# jQuery Selectors



Syntax	Description
<code>\$("*")</code>	Selects all elements
<code>\$(this)</code>	Selects the current HTML element
<code>\$("p.intro")</code>	Selects all <code>&lt;p&gt;</code> elements with <code>class="intro"</code>
<code>\$("p:first")</code>	Selects the first <code>&lt;p&gt;</code> element
<code>\$("ul li:first")</code>	Selects the first <code>&lt;li&gt;</code> element of the first <code>&lt;ul&gt;</code>
<code>\$("ul li:first-child")</code>	Selects the first <code>&lt;li&gt;</code> element of every <code>&lt;ul&gt;</code>





# jQuery Selectors



## - Filtering Selectors:

- `first()`: return first element in all element selected.
- `last()`: return first element in all element selected.
- `eq(n)`: return  $n^{\text{th}}$  element in  $(n+1)$  element selected.
- `filter()`: filter all element want to select.
- `Not()`: filter all element want to remove.





# jQuery Selectors



Syntax	Description
<code>\$("[href]")</code>	Selects all elements with an href attribute
<code>\$("a[target='_blank']")</code>	Selects all <a> elements with a target attribute value equal to "_blank"
<code>\$("a[target!='_blank']")</code>	Selects all <a> elements with a target attribute value NOT equal to "_blank"
<code>\$(":button")</code>	Selects all <button> elements and <input> elements of type="button"
<code>\$("tr:even")</code>	Selects all even <tr> elements
<code>\$("tr:odd")</code>	Selects all odd <tr> elements





# Demo



```
<script>
  $(document).ready(function(){
    $("p.mark").css("background", "yellow");
    $("#mark span").css("background", "yellow");
    $("ul li").css("background", "yellow");
    $("ul#mark li").css("background", "red");
    $("ul.mark li").css("background", "green");
    $('a[target="_blank"]').css("background",
"yellow");
  });
</script>
```



# Demo



This is a paragraph.

This is another paragraph.

This is one more paragraph.

- List item one
- List item two
- List item three

- List item one
- List item two
- List item three

- List item one
- List item two
- List item three

Go to [Home page](#)





# Demo



```
<script>
  $(document).ready(function(){
    $("tr:odd").css("background", "yellow");
    $("tr:even").css("background", "orange");
    $("p:first").css("background", "red");
    $("p:last").css("background", "green");
    $("form :text").css("background", "purple");
    $("form :password").css("background", "blue");
    $("form :submit").css("background", "violet");
  });
</script>
```



# Demo



No.	Name	Email
1	John Carter	johncarter@mail.com
2	Peter Parker	peterparker@mail.com
3	John Rambo	johnrambo@mail.com

This is a paragraph.

This is another paragraph.

This is one more paragraph.

Name:

Password:

Sign In



55



# Demo



```
<script>
$(document).ready(function(){
    $("div").first().css("background", "yellow")
    $("div").eq(2).css("background", "blue");
    $("div").last().css("background", "red");
});
</script>
<div> First div</div>
<div> Second div </div>
<div> Thirt div </div>
<div> Last div </div>
```

First div

Second div

Thirt div

Last div







# jQuery event



Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

– Ex:

```
$(".btn").click(function(){  
    // action  
});
```



57



# Demo



58



## - Get Content:

- `text()`: Get the text content of selected elements
- `html()`: Get the content of selected elements (including HTML markup)
- `val()`: Get the value of form fields
- `attr("attributeName")`: Get value of html element attribute.

```
<p id="test">This is some <b>bold</b> text in a  
paragraph.</p>
```

---

```
$("#test").text() ➔ "This is some bold text in a  
paragraph."
```

```
$("#test").html() ➔ "This is some <b>bold</b> text in a  
paragraph."
```





# Demo





## - Get Content:

- `text("text")`: Set the text content of selected elements
- `html("html")`: Set the content of selected elements (including HTML markup)
- `val("value")`: Set the value of form fields
- `attr("attributeName", "value")`: Set value for html element attribute.

```
$("#test").text("This is some bold text in a paragraph. ")
```

```
$("#test").html("This is some <b>bold</b> text in a paragraph.")
```





## - Add Elements

- Append: Inserts content at the end of the selected elements
- prepend(): Inserts content at the beginning of the selected elements
- after(): Inserts content after the selected elements
- before(): Inserts content before the selected elements

```
$("#p").append("Appended text.");  
$("#p").prepend("Prepended text.");  
$("#p").after("Text after");  
$("#p").before("Text before");
```





## - Remove Elements

- `remove()`: Removes the selected element (and its child elements)
- `empty()`: Removes the child elements from the selected element

```
$("#content").remove();  
$("#content").empty();
```





## - Get, Set CSS Classes

- `addClass()`: Adds one or more classes to the selected elements
- `removeClass()`: Removes one or more classes from the selected elements
- `toggleClass()`: Toggles between adding/removing classes from the selected elements

```
$("h1, h2, p").addClass("bg-success border");  
$("h1, h2, p").removeClass("bg-success border");  
$("h1, h2, p").toggleClass("bg-success border");
```







## - Get, Set CSS Classes

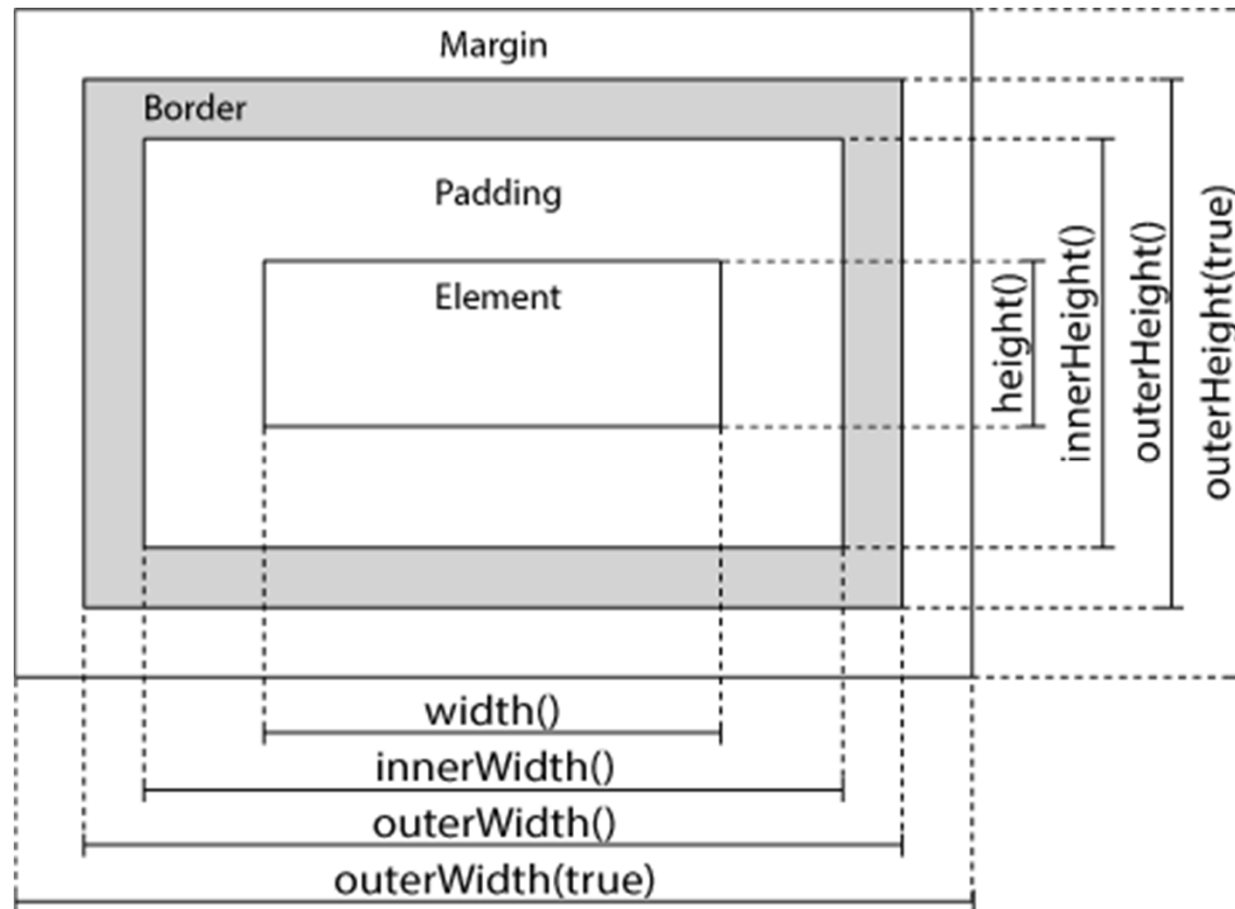
- `css("propertyname")`: return the value of a CSS property
- `css("propertyname","value")`: set a CSS property

```
$( "p" ).css( "background-color" );  
$( "p" ).css( "background-color", "yellow" );
```





## - Dimensions





# jQuery - HTML



- **width():** set or get the width of an element (**without** padding, border and margin).
- **height():** set or get the height of an element (**without** padding, border and margin).
- **innerWidth():** get returns the width of an element (**includes** padding)
- **innerHeight():** get returns the width of an element (**includes** padding)





# jQuery - HTML



- **outerWidth():** get the width of an element (**includes** padding and border)
- **outerHeight():** get the height of an element (**includes** padding and border)
- **outerWidth(true):** get the width of an element (**includes** padding, border, and margin)
- **outerHeight(true):** get the height of an element (**includes** padding, border, and margin)

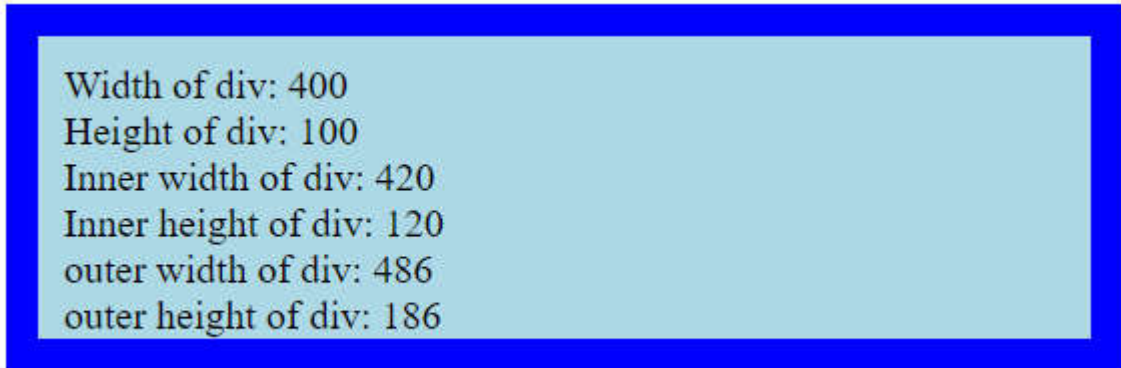




# jQuery - HTML



```
#div1 {  
  height: 100px;  
  width: 400px;  
  margin: 20px;  
  padding: 10px;  
  border: 13px solid blue;  
  background-color: lightblue;  
}
```



Width of div: 400  
Height of div: 100  
Inner width of div: 420  
Inner height of div: 120  
outer width of div: 486  
outer height of div: 186





# jQuery - Traversing



- HTML DOM is designed like a tree in data structure. Every HTML elements has a parent element and childrent element. Traversing is understood as moving through each element (parent and child) .





# jQuery - Traversing



## - Traversing method:

- `parent()`: returns a direct parent of this element.
- `children()`: returns all direct children of this element.
- `find()`: returns descendant elements of the selected element (search in all children).
- `next()`: return element next to this element.
- `nextAll()`: return all next element.
- `prev()` return element previous to this element.
- `prevAll()`: return all previous element.





# jQuery No-Conflict Mode



- jQuery uses the dollar sign (\$) as a shortcut or alias for jQuery. Thus, if you use another JavaScript library that also uses the \$ sign as a shortcut, along with the jQuery library on the same page, conflicts could occur. Fortunately, jQuery provides a special method named `noConflict()` to deal with such situation.







# jQuery No-Conflict Mode



- The `jQuery.noConflict()` method return the control of the `$` identifier back to other libraries. The jQuery code in the following example (line no-10) will put the jQuery into no-conflict mode immediately after it is loaded onto the page and assign a new variable name `$j` to replace the `$` alias in order to avoid conflicts with the prototype framework.



73



```
var $j = jQuery.noConflict();
$j(document).ready(function(){
    // Display an alert message when the element with ID foo is clicked
    $j("#foo").click(function(){
        alert("jQuery is working well with prototype.");
    });
});

// Some prototype framework code
document.observe("dom:loaded", function(){
    // Display an alert message when the element with ID bar is clicked
    $(bar).observe('click', function(event){
        alert("Prototype is working well with jQuery.");
    });
});
```





# Q & A



75