香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

LEARNING JAVASCRIPT

PROF. DAVID ROSSITER

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

LEARNING JAVASCRIPT

- Slides give the 'backbone'
- Practical demos
- All examples available on the web site
- Assessment at the end

IF YOU KNOW JAVASCRIPT

Check the *After this presentation...* message

Check the list of JavaScript covered

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

ABOUT JAVASCRIPT

PROF. DAVID ROSSITER

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

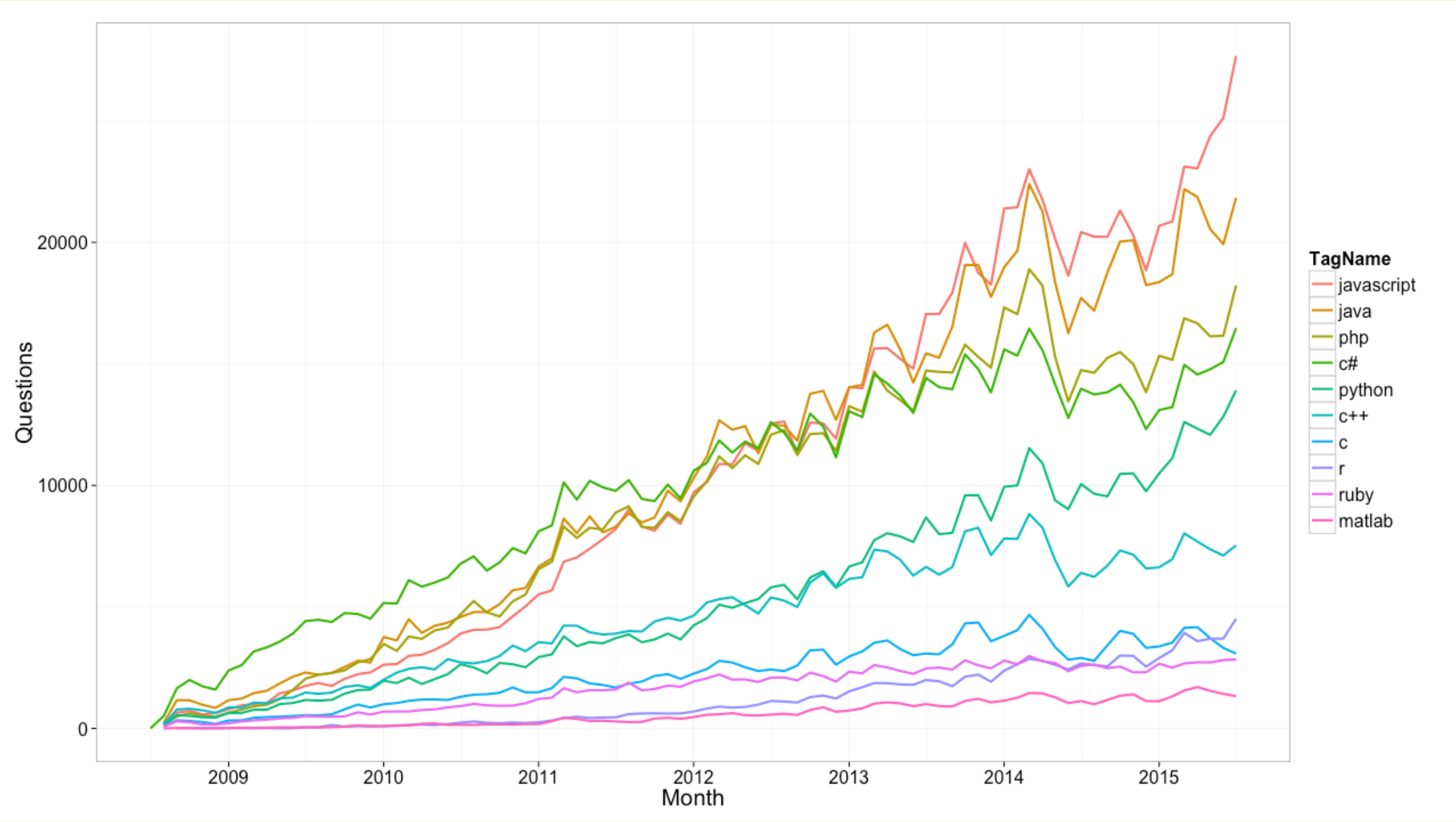
AFTER THIS PRESENTATION

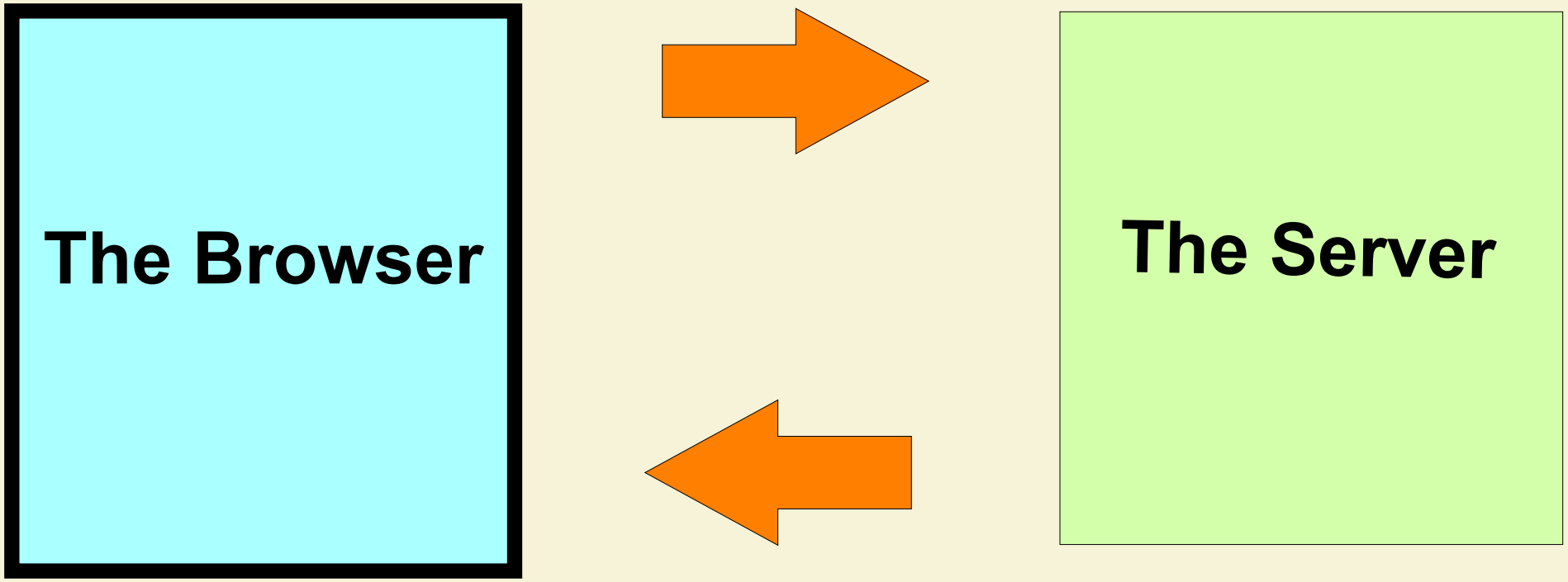
- You'll appreciate the popularity of JavaScript
- You'll understand where JavaScript can be used

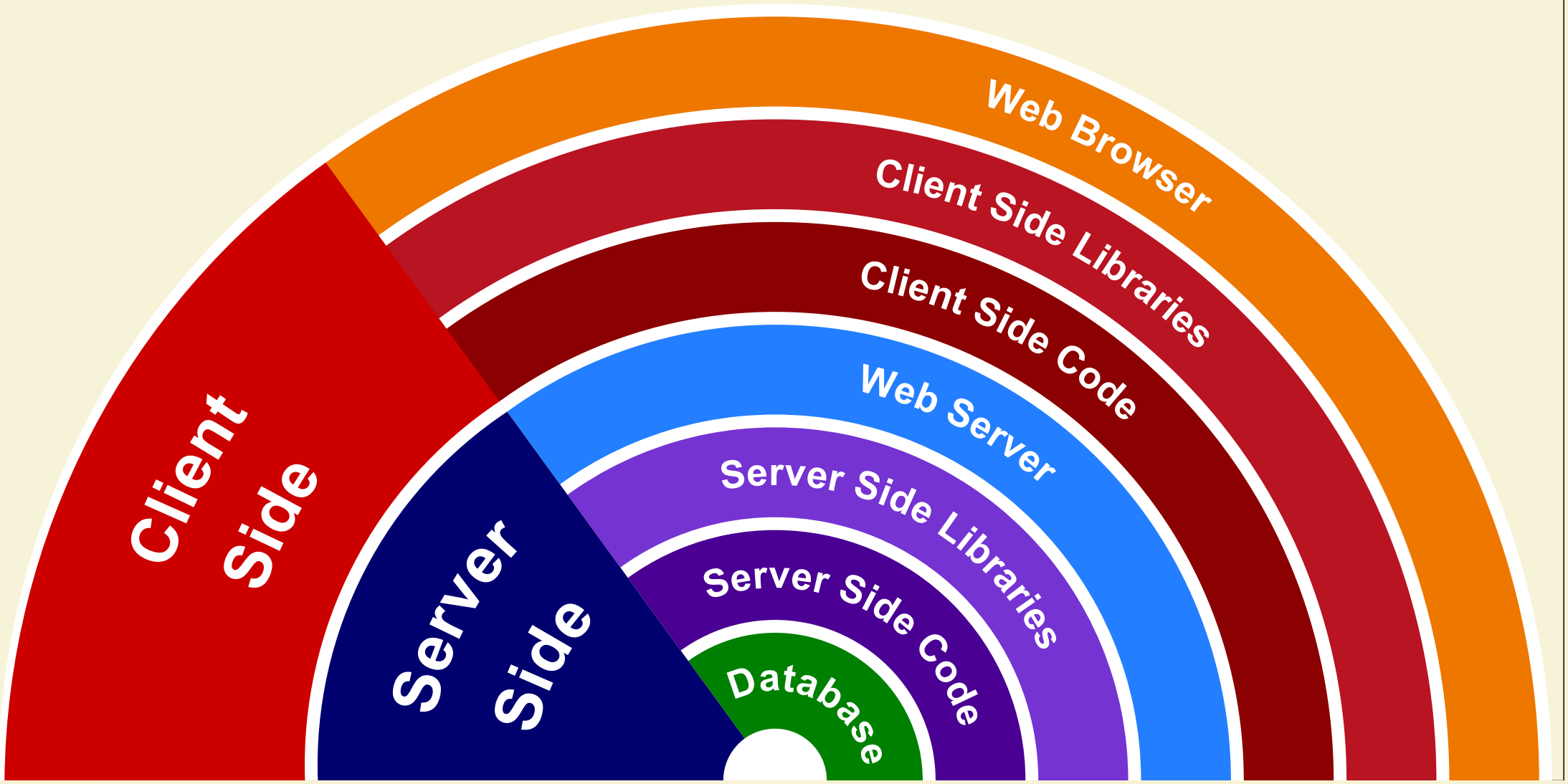
JAVASCRIPT POPULARITY

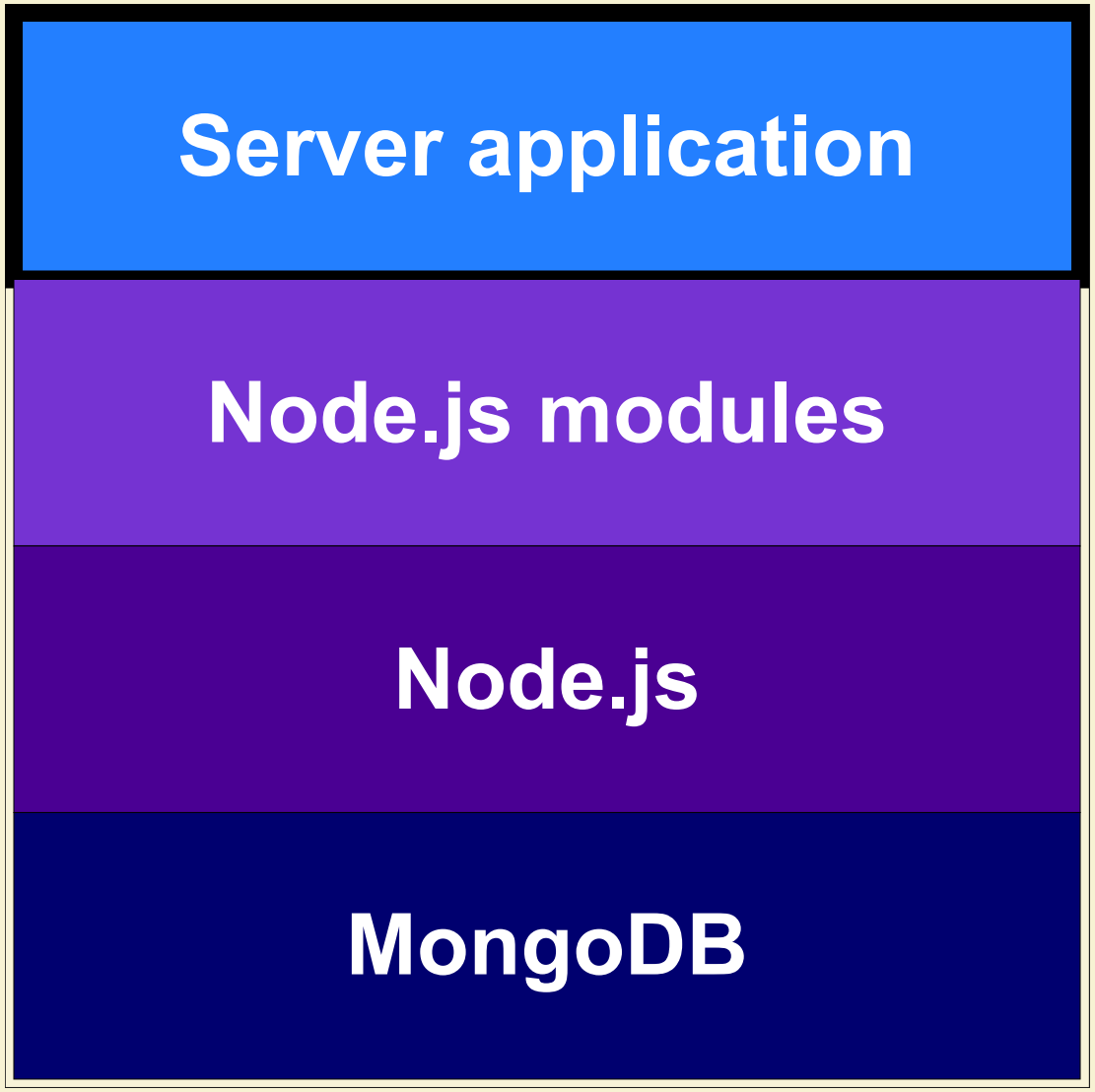
JavaScript (JS) is the dominant web programming language

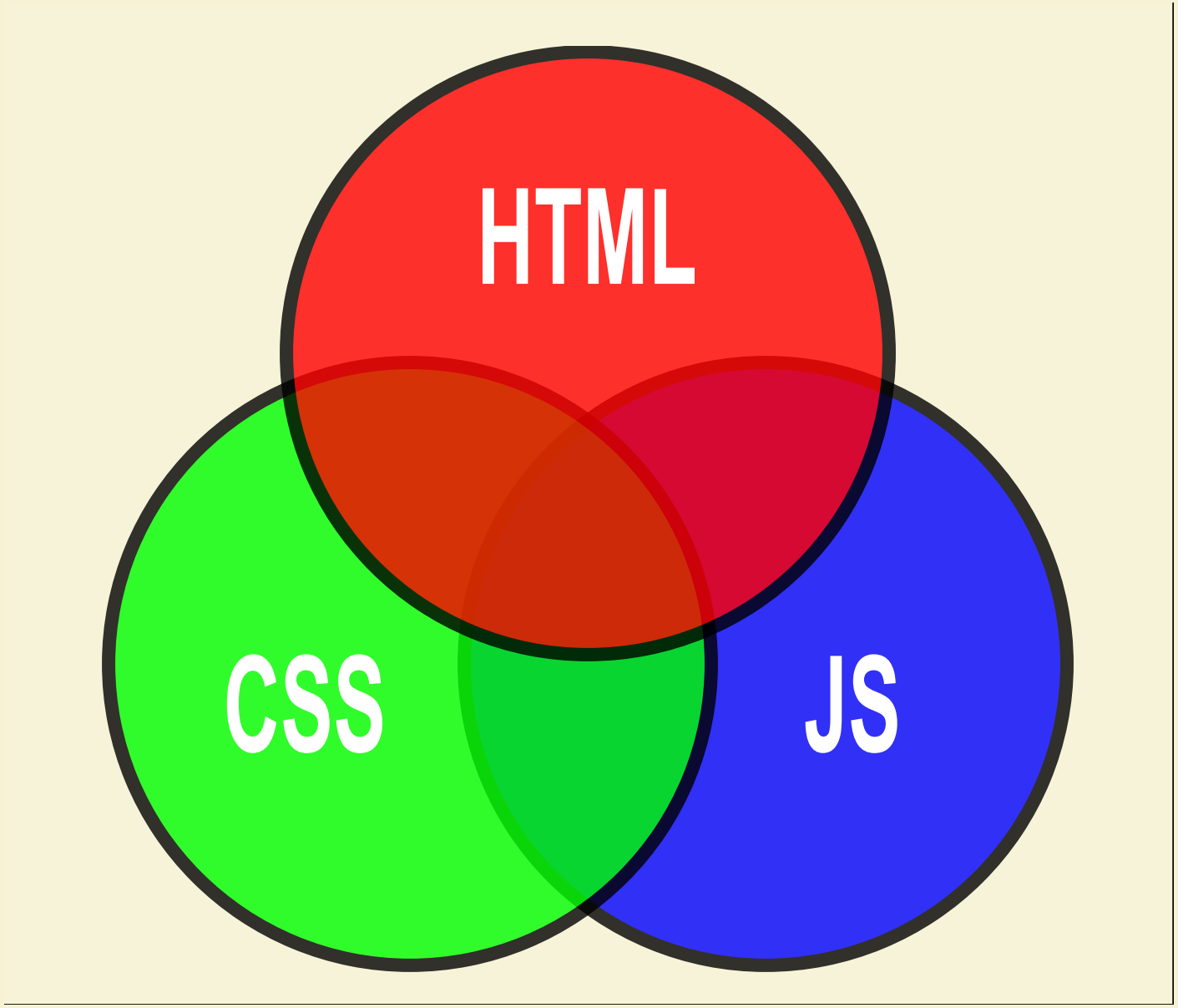
- Number of questions in Stack Overflow
  From http://blog.revolutionanalytics.com/2015/07/the-most-popular-programming-languages-on-stackoverflow.html

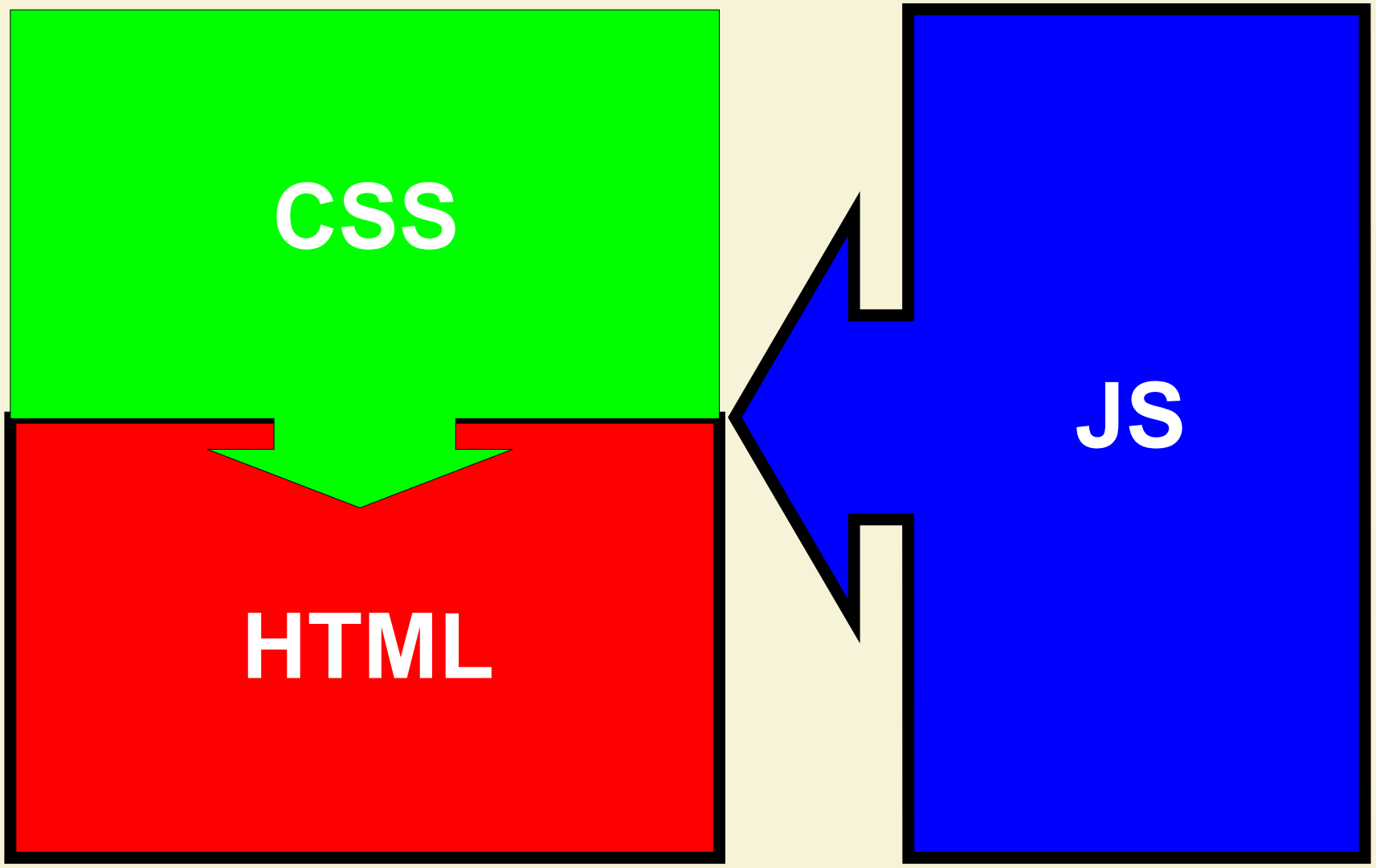

Questions
20000
10000
0
2009
2010
2011
2012
2013
2014
2015
Month
TagName
javascript
java
php
c#
python
c++
c
r
ruby
matlab

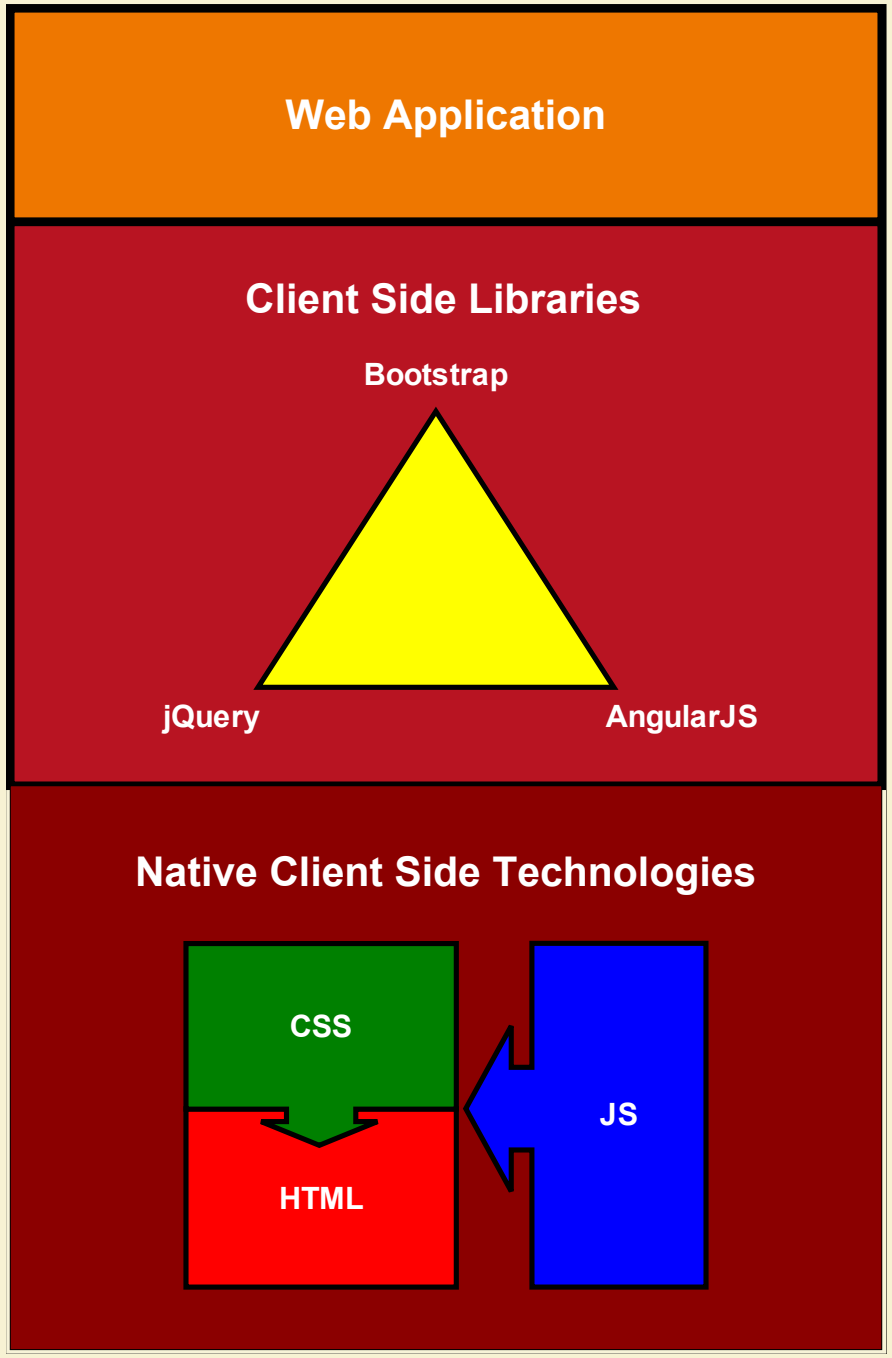

The Browser
The Server


Client Side
Server Side
Web Browser
Client Side Libraries
Client Side Code
Web Server
Server Side Libraries
Server Side Code
Database


Server application
Node.js modules
Node.js
MongoDB


HTML
CSS
JS


CSS
HTML
JS


Web Application
Client Side Libraries
Bootstrap
jQuery
AngularJS
Native Client Side Technologies
CSS
HTML
JS

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

GETTING TO KNOW JAVASCRIPT

PROF. DAVID ROSSITER

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to write simple JavaScript
- You'll be able to use variables and some simple functions

JAVASCRIPT FUNCTIONS WE WILL LOOK AT

```
alert()
```

```
prompt()
```

```
confirm()
```

WHERE TO PUT JAVASCRIPT?

- JavaScript code can go almost anywhere
- However, there is a common pattern

```html
<!DOCTYPE html>
<html>
```

```html
<head>
```

. . . load JavaScript libraries here . . .

```html
</head>
```

```html
<body>
```

. . . your JavaScript code typically goes at the end of body . . .

```html
</body>
```

```html
</html>
```

JAVASCRIPT IN THE SAME FILE

```
<script>
function surprise() {
    alert("Hello!");
}
</script>
```

JAVASCRIPT IN ANOTHER FILE

```
<script src="mycode.js"></script>
```

In *mycode.js*:

```
function surprise() {
    alert("Hello!");
}
```

SIMPLE INTERACTION

- There are 3 JavaScript popups:
  - `alert()`
  - `confirm()`
  - `prompt()`

SHOW A MESSAGE - ALERT()

- `alert()` shows text to the user e.g.

```
alert("Welcome!");
```

ALERT()

Click here to open the file

MAKING A DECISION - CONFIRM()

- `confirm()` displays a popup box with a message, along with an OK and a Cancel button

CONFIRM()

Click here to open the file

CONFIRM()

```
<!doctype html>
<html>
    <head>
        <title>Example of confirm()</title>
        <script>
            if (confirm("Want to go to Disneyland?"))
                document.location.href
                ="http://park.hongkongdisneyland.com";
        </script>
    </head>
</html>
```

VARIABLES

- A variable is like a box
- You can make a variable and put something in it e.g

```
var totalCost = 7000;
```

- Later, you can take it out of the box and use it
- You can change what is stored in the box any time

SIMPLE TEXT INPUT - PROMPT()

- For getting input from the user, you can use `prompt()`, e.g:

```
var user_name; // Create a variable
user_name=prompt("What is your name?");
```

- You don't have to create a variable before you use it
- However, it is good habit to get into

PROMPT()

Click here to open the file

PROMPT()

```
<!doctype html>
<html>
    <head>
        <title>Example of prompt()</title>
        <script>
            var user_name;
            user_name=prompt("What is your name?");
            document.write("Welcome to my page "
                + user_name + "!" );
        </script>
    </head>
</html>
```

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

VARIABLES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll understand different data types in JavaScript

WE WILL LOOK AT

`var`

`typeof`

DATA TYPES

- Number
- String
- Boolean
- Other e.g. Object

NUMBER

- JavaScript has only one type of number
- Can be written with or without a decimal place

```
var number1 = 34.289;
var number2 = 100;
```

- Can use scientific notation

```
var big_number = 123e5;     //12300000
var small_number = 123e-5; //0.00123
```

STRING

- A *string* simply means text
- You can use single or double quotes

```
var name = "David";
var title = 'Professor';
```

- You can use quotes inside a string, as long as they don't match the quotes surrounding the string

```
var message = "It's alright";
```

BOOLEAN

- A Boolean value can only be `true` or `false`

```
var condition1 = true;
var condition2 = false;
```

- Do not confuse Boolean values with String values

```
var myBool = true;      //Boolean type
var myString = "true"; //String type
```

A VARIABLE TYPE CAN CHANGE

- If you do this

```
var storage = "David";
```

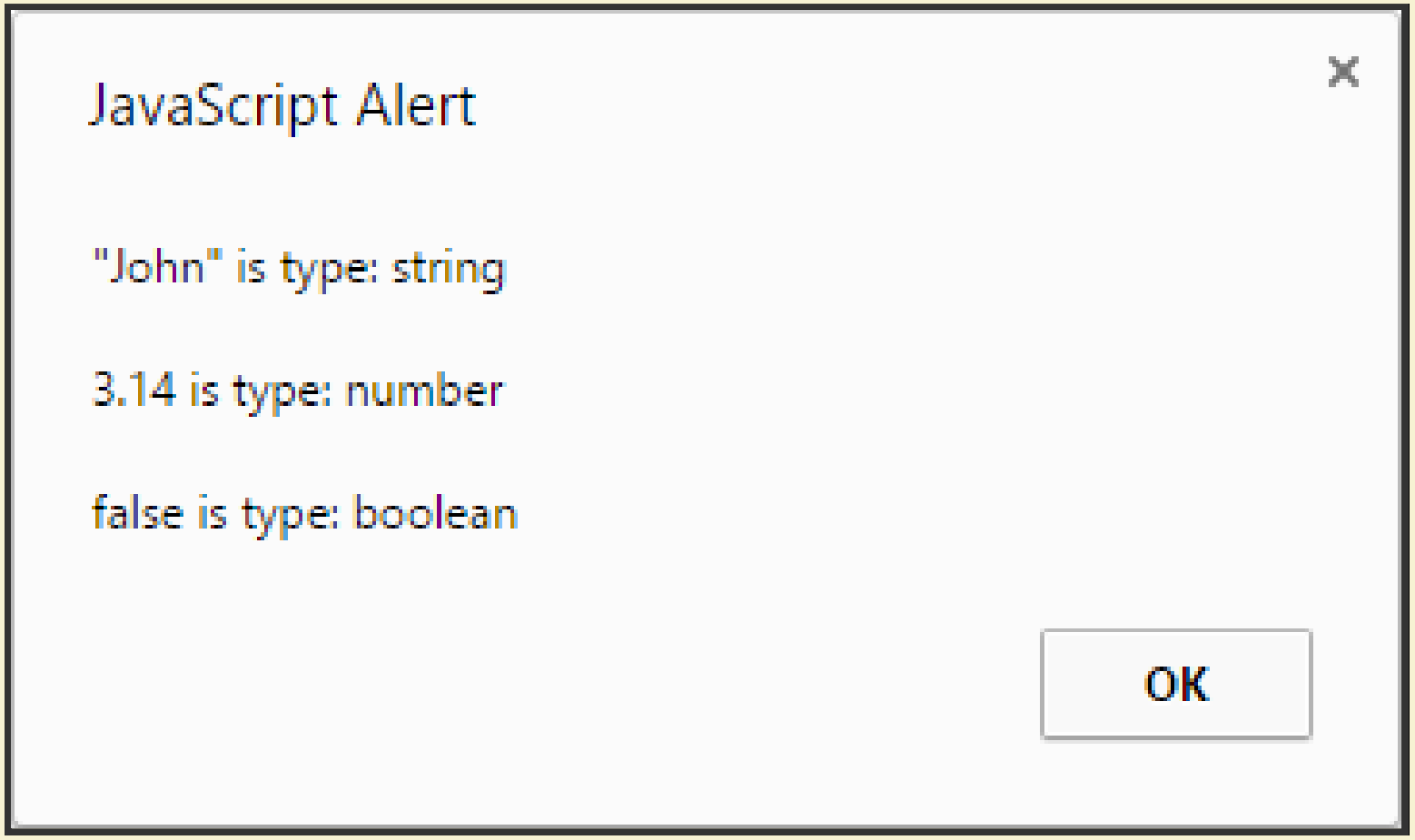
- And then this:

```
storage = 98;
```

- The type of the variable is immediately changed

USING TYPEOF

- You can use the `typeof` operator to check the type of a variable


JavaScript Alert
"John" is type: string
3.14 is type: number
false is type: boolean
OK

```html
<!doctype html>
<html>
<head>
  <title>Variable Type Example</title>
</head>
<body>
  <script>
  alert( '"John" is type: ' + typeof "John" + "\n\n"
        + "3.14 is type: " + typeof 3.14 + "\n\n"
        + "false is type: " + typeof false ) ;
  </script>
</body>
</html>
```

COMMON CHANGES

Code	Quicker Typing
`count = count + 1`	`count++`
`count = count - 1`	`count--`
`count = count + 10`	`count += 10`
`hello = hello + "!"`	`hello += "!"`
`marks = marks - 20`	`marks -= 20`
`pigs = pigs * 5`	`pigs *= 5`
`cakes = cakes / students`	`cakes /= students`

FROM ONE TYPE TO ANOTHER

Function	Meaning
`parseInt()`	Converts to an integer
`parseFloat()`	Converts to a floating point number
`String()`	Converts the value of an object to a string

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

INTRODUCTION TO EVENTS AND FUNCTIONS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate the concept of events
- You'll understand how to use functions

WE WILL LOOK AT

Events	`onload`
Functions	`function`
	`return`

EVENTS

- An event is when something happens
- For example:
  - Click on something
  - Move the mouse
  - Press a key on the keyboard
- You can arrange for some code that you write to be executed when the event occurs

ONLOAD EVENT

- *onload* is triggered when the object has loaded

```
<body onload="alert('Hello!')">
```

. . . the main web page content goes here . . .

```
</body>
```

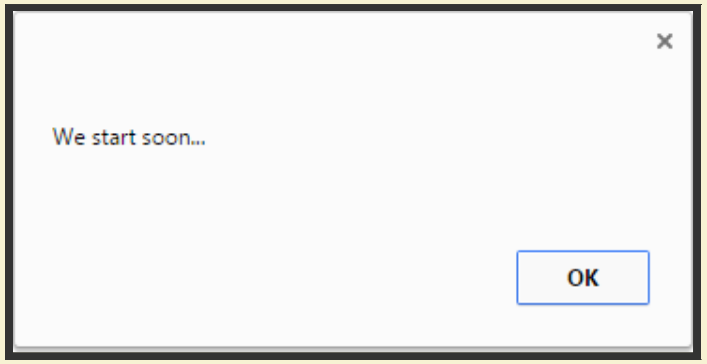
EXAMPLE

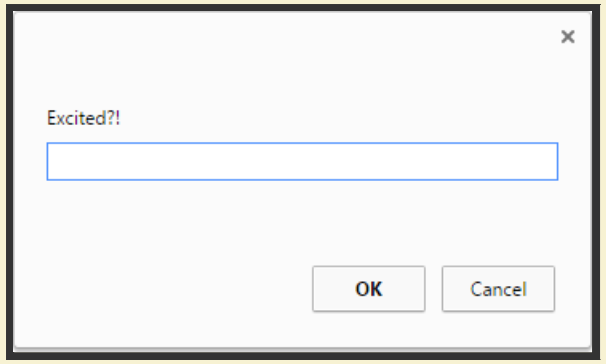
```
<!doctype html>
<html>
    <body onload="alert('Hello!')">
        <p>
            A message is shown as soon
            as the page is loaded.
        </p>
    </body>
</html>
```

You can execute as much code as you like

```
<!doctype html>
<html>
    <body onload="alert('Hello!');
            alert('We start soon...');
            prompt('Excited?!') ">
        <p>
            3 popup windows are shown as
            soon as the page is loaded.
        </p>
    </body>
</html>
```


Hello!
OK


We start soon...
OK


Excited?!
OK
Cancel

FUNCTIONS

- A function is a group of code:

```
function do_something() {
```

. . . code goes here . . .

```
}
```

- Run the function like this:

```
do_something();
```

```html
<!doctype html>
<html>
    <head>
        <title>Example of a function</title>
        <script>
            function greet_the_user(){
                alert('Hello!');
                alert('We start soon...');
                prompt('Excited?!')
            }
        </script>
    </head>
    <body onload="greet_the_user()">
    </body>
</html>
```

FUNCTION PARAMETERS

You can pass something to a function

```
function purchase( cats ) {
```

. . . *code here uses* cats . . .

```
}
```

- Run the function like this:

```
purchase( 10 );
```

FUNCTION RESPONSE

You can get a response from a function

```
function do_something() {
```

. . . *code here stores something in* answer . . .

```
    return answer; }
```

- Use the function like this:

```
result = do_something();
```

```
<!doctype html>
<html><body onload="check_user_age()" style="position:absolute">
    <h1>This is my naughty home page.</h1>
    <script>
        function check_user_age(){
            if (age_of_user() < 18)
                alert("Please go to another page.");
        }
        function age_of_user(){
            var age_text, age;
            age_text=prompt("What is your age?");
            age=parseInt(age_text);
            return age;
        }
</script></body></html>
```

A RECURSIVE FUNCTION

A function can call itself

```
function do_something( control_value ) {
```

. . . *code here calls* do_something(. . .)

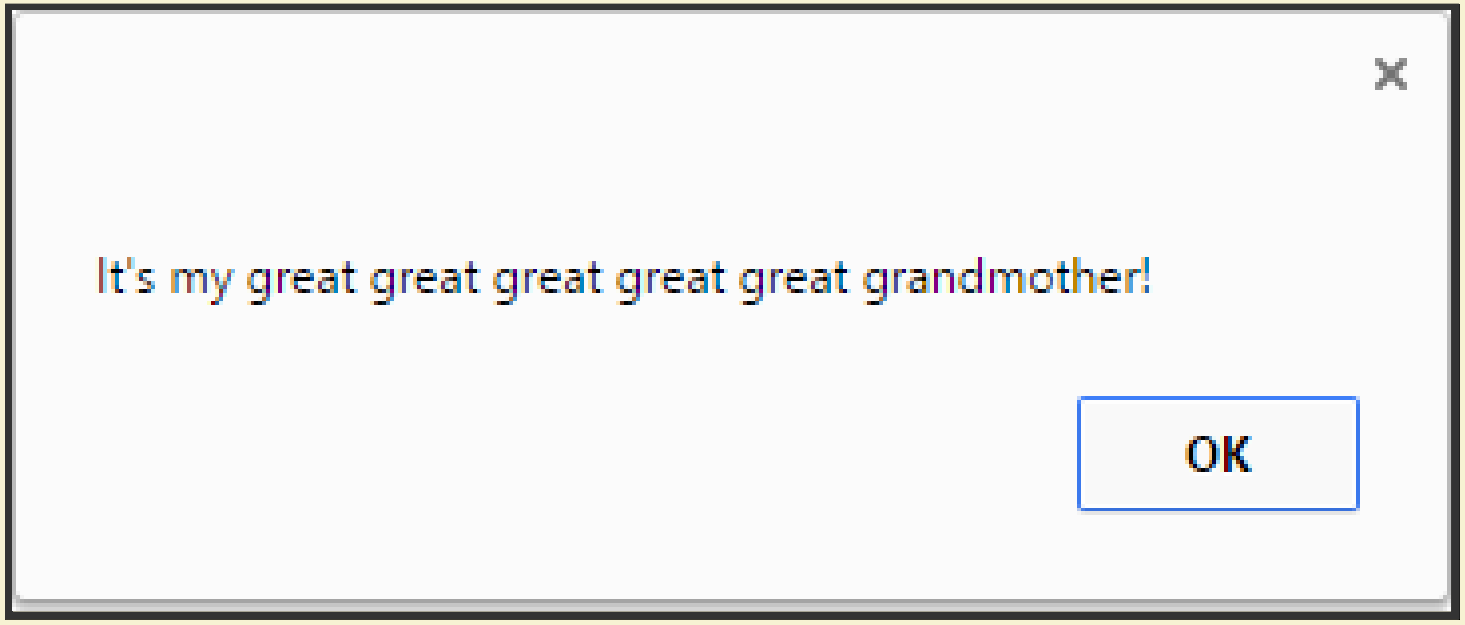
```
}
```

- Start the function like this:

```
result = do_something( 10 );
```

```
<!doctype html>
<html><body>
  <script>
    alert("It's my " + build_great(5) +
          "grandmother!");

    function build_great( depth ) {
      if (depth > 0)
        return "great " + build_great( depth - 1 );
      else
        return "";
    }
</script>
</body></html>
```


It's my great great great great great grandmother!
OK

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

MAKING DECISIONS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to make decisions using `if` statements
- You'll be able to make decisions using `switch` statements

WE WILL LOOK AT

`if`	`switch … case`
`if … else`	`default`
`if … else if …`	
`if … else if … else`	

MAKING DECISIONS

- `if` is used similar to regular English
- Lots of variations e.g.

```
if
if … else
if … else if … else
if … else if … else if … else
```

COMPARING THINGS

- `<` is less than
- `<=` is less than or equal to
- `>` is greater than
- `>=` is greater than or equal to
- `==` is equal to
- `!=` is not equal to

EXAMPLE

```
<!doctype html>
<html>
  <head><script>
    var user_name;

    user_name=prompt("What is your name?");
    if (user_name == "dave")
      alert("Great name!");
  </script></head>
</html>
```

USING BRACES

- You must use braces { } for more than 1 line of code:

```
if (user_name == "dave" ) {
    alert("Great name!");
    awesome_name=true;
}
```

- Braces are optional if there is only one line of code

IF ... ELSE

- `else` goes at the end of the `if`
- It handles any situation not already handled at that point

```
<!doctype html>
<html>
  <head><script>
    var user_name;

    user_name=prompt("What is your name?");
    if (user_name == "dave")
      alert("Great name!");
    else
      alert("Your name isn't great...");
  </script></head>
</html>
```

IF ... ELSE IF

- Use `else if` to add another test
- You can do this as many times as you like

```
<!doctype html>
<html>
  <head><script>
    var user_name;

    user_name=prompt("What is your name?");
    if (user_name == "dave")
      alert("Great name!");
    else if (user_name == "jogesh")
      alert("Pretty good name!");
  </script></head>
</html>
```

IF ... ELSE IF ... ELSE

- Here's an example of everything working together

```
<!doctype html>
<html><head><script>
  var user_name;

  user_name=prompt("What is your name?");
  if (user_name == "dave")
    alert("Great name!");
  else if (user_name == "jogesh")
    alert("Pretty good name!");
  else if (user_name == "oz")
    alert("Excellent name!");
  else
    alert("Your name isn't great, never mind...");
</script></head></html>
```

SWITCH

- Used for a series of comparisons:

```
switch(variable_name) {
    case "option_1": do_something_1();
                     break;

            . . .   :   . . .

    case "option_n": do_something_n();
                     break;

            default:  do_something_default();
}
```

```html
<!doctype html>
<html>
    <head>
        <script>
        var user_name=prompt("What is your name?");

        switch(user_name) {
            case "dave":
                alert("Great name!");
                break;
            case "jogesh":
                alert("Pretty good name!");
                break;
            default:
                alert("Your name isn't great, never mind...");
        }
        </script>
    </head>
</html>
```

SWITCH

- `break` is used to stop any more `case` comparisons
- Sometimes `break` is appropriate, sometimes it isn't

```html
<!doctype html>
<html>
    <head>
        <script>
        var user_name=prompt("What country would you like to visit?");
        switch(user_name) {
            case "Canada":
            case "France":
                alert("Take me also!");
                break;
            case "Japan":
            case "Philippines":
                alert("Great! Have fun!");
                break;
            case "North Korea":
                alert("Oh! Good luck!");
                break;
            default:
                alert("I am sure you will have a great time");
        }
        </script>
    </head>
</html>
```

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

WHILE LOOPS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to create `while` loops
- You'll be able to create `do ... while` loops

WE WILL LOOK AT

`while`

`do … while`

LOOPS

- A loop repeats some code again and again
- Here we will look at:
  - `while`
  - `do … while`

WHILE LOOPS

- A while loop is the simplest loop

```
while (condition) {
```

. . . code goes here . . .

```
};
```

- Each time the loop content is executed we call it an *iteration*

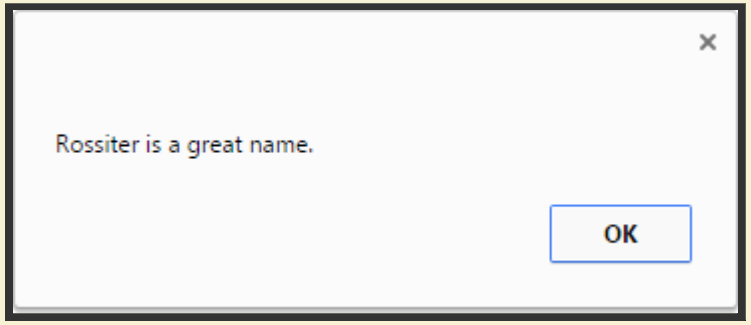
INDEXOF()

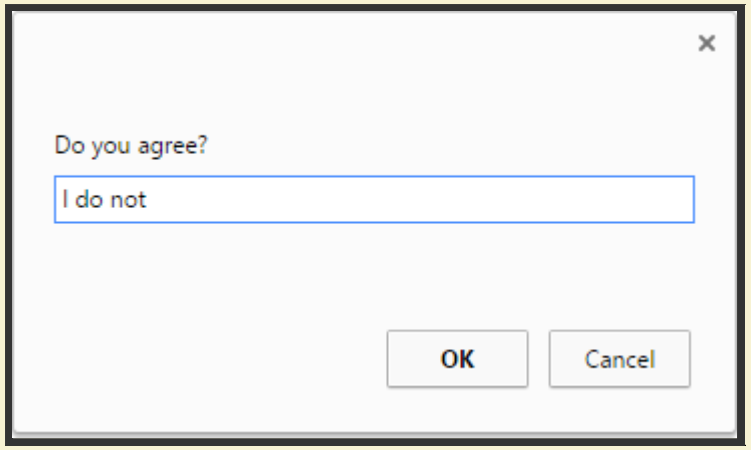
- *string*`.indexOf("text")`
  gives you the location of the first "text" in the string
- For example:

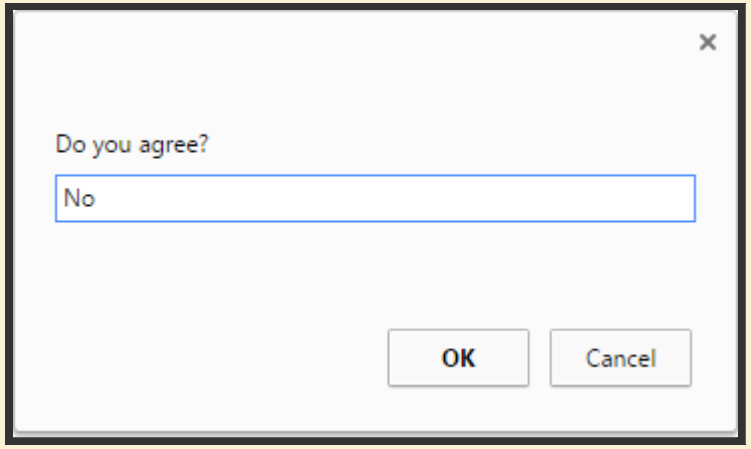
```
var text = "The cat's hat was wet";
result = text.indexOf("at");
```

- *result* will be 5

```
<!doctype html>
<html><head>
    <title>Example of while()</title>
    <script>
        var response, finished;
        finished=false;
        alert("Rossiter is a great name.");
        while (!finished){
            response=prompt("Do you agree?");
            if (response.indexOf("y")==0)
                finished=true;
        }
    </script>
</head></html>
```


Rossiter is a great name.
OK


Do you agree?
I do not
OK
Cancel


Do you agree?
No
OK
Cancel


Do you agree?
yea
OK
Cancel

DO ... WHILE LOOPS

- do ... `while` is an 'upside-down' version of `while`

```
do {
```

. . . code goes here . . .

```
} while (condition);
```

- A do ... `while` loop will be executed at least once

```
<!doctype html>
<html><head>
    <title>Example of do .. while()</title>
    <script>
        var response, finished;
        finished=false;
        alert("Rossiter is a great name.");
        do {
            response=prompt("Do you agree?");
            if (response.indexOf("y")==0)
                finished=true;
        } while (!finished);

    </script>
</head></html>
```

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

MORE ON VARIABLES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate the concept of local variables
- You'll appreciate the concept of global variables

LOCAL VARIABLES

- Variables declared within a function can only be accessed within the function
- They are *local* to the function, and so are called local variables

```html
<!doctype html>
<html><body>
  <script>
    function show_money() {
      var money = 2;
      alert("In the function, the value is: "+ money);
    }
    money = 99;
    alert("In the main part, the value is: "+ money);
    show_money();
    alert("In the main part, the value is: "+ money);
  </script>
</body></html>
```

Click here to open the example

GLOBAL VARIABLES

- The opposite of a local variable is a *global variable*
- Global variables are created in the main part
- They can work inside or outside functions

```html
<!doctype html>
<html><body>
  <script>
    function show_money() {
      alert("In the function, the value is: "+ money);
    }
    var money = 99;
    alert("In the main part, the value is: "+ money);
    show_money();
    alert("In the main part, the value is: "+ money);
  </script>
</body></html>
```

Click here to open the example

LOCAL AND GLOBAL VARIABLES SHARING THE SAME NAME

- JavaScript will give priority to the local variable inside the function

CREATING GLOBAL VARIABLES INSIDE FUNCTIONS

- If you assign a value to a variable that has not been *declared*, it will automatically become a global variable

```
<!doctype html>
<html><body>
  <script>
    function show_money() {
      money = 2;
      alert("In the function, the value is: "+ money);
    }
    show_money();
    alert("In the main part, the value is: "+ money);
  </script>
</body></html>
```

Click here to open the example

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

LOGICAL OPERATORS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate Boolean values
- You'll know more about logical operators and how to use them

WE WILL LOOK AT

Boolean values	`true`
	`false`
Logical operators	`&&`
	`||`
	`!`

BOOLEAN

- A *Boolean* value is either `true` or `false`
- A variable which has a Boolean value is called a Boolean variable

LOGICAL OPERATORS

- Logical operators work with Boolean values
- JavaScript has these logical operators:
  - Logical And - the && operator
  - Logical Or - the || operator
  - Logical Not - the ! operator

AND - &&

- `&&` - the result is `true` if both inputs are `true`, otherwise the result is `false`

AND - &&

a	b	a && b
false	false	false
false	true	false
true	false	false
true	true	true

```
<html><body><script>
  var you_are_rich = false;
  var you_have_partner = true;
  var you_have_flat = true;
  var life_is_fantastic = you_are_rich
   && you_have_partner && you_have_flat;
  alert("life is fantastic is " +
              life_is_fantastic);
  you_are_rich = true;
  life_is_fantastic = you_are_rich
   && you_have_partner && you_have_flat;
  alert("life is fantastic is now " +
                 life_is_fantastic);
</script></body></html>
```

SHORT-CIRCUIT IN AND

- JavaScript is clever
- When it evaluates an And it checks the first input
- If the value is `false` it knows the result must be `false`
- So it doesn't bother checking the next input

```html
<!doctype html>
<html>
  <body><script>
    function first_function() {
      alert("first_function() is running!");
      return true;
    }
    function second_function() {
      alert("second_function() is running!");
      return false;
    }
    var test_function =
       first_function() && second_function();
  </script></body>
</html>
```

AFTER SWAPPING THE FUNCTIONS

```
<html><body><script>
    function first_function() {
      alert("first_function() is running!");
      return true;
    }
    function second_function() {
      alert("second_function() is running!");
      return false;
    }
    var test_function_swapped =
       second_function() && first_function();
</script></body></html>
```

OR - ||

- | | - the result is `false` if both inputs are `false`, otherwise the result is `true`

OR - ||

a	b	a || b
false	false	false
false	true	true
true	false	true
true	true	true

```html
<!doctype html>
<html>
  <body><script>
    var you_are_rich = false;
    var you_have_partner = true;
    var you_have_flat = false;
    var life_is_good = you_are_rich
      || you_have_partner || you_have_flat;
    alert("life is good is " + life_is_good);
    you_have_partner = false;
    life_is_good = you_are_rich
      || you_have_partner || you_have_flat;
    alert("life is good is now " + life_is_good);
  </script></body>
</html>
```

SHORT-CIRCUIT IN OR

- If JavaScript is evaluating Or and the first input is true, it knows the result must be `true`
- So it doesn't bother checking the second input

```html
<!doctype html>
<html>
  <body><script>
    function first_function() {
      alert("first_function() is running!");
      return true;
    }
    function second_function() {
      alert("second_function() is running!");
      return false;
    }
    var test_function =
       first_function() || second_function();
  </script></body>
</html>
```

NOT - !

- ! - the result is the opposite of the input

NOT - !

a	!a
false	true
true	false

```html
<!doctype html>
<html>
  <head>
    <title>Not Operator Example</title>
  </head>
  <body>
    <script>
      var you_are_male = true;
      var you_are_female = !you_are_male;
      alert("you_are_male is " + you_are_male);
      alert("you_are_female is " + you_are_female);
    </script>
  </body>
</html>
```

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

ARRAYS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

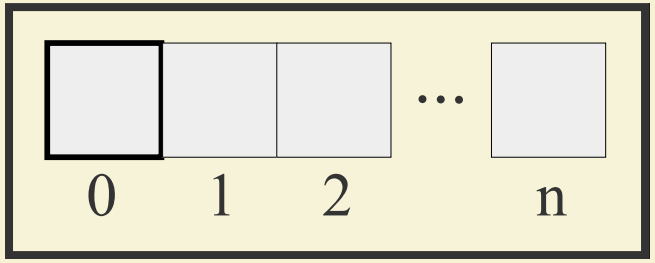
- You'll understand and use the array data structure
- You'll be able use some common array functions

ARRAY FUNCTIONS

`[]`	`push()`	`concat()`
`length`	`shift()`	
`join()`	`pop()`	
	`unshift()`	

ARRAY

- An array is a linear continuous storage



- You can think array as a group of boxes
- Each box has a unique identity, which is called an *index*
- The *index* of the first box is **0**

CREATING AN ARRAY

- Here is how you create a new array with 3 boxes:

```
var pets = ["Dog", "Cat", "Rabbit"];
```

- You can create a new array with 10 boxes without any element inside the boxes like this:

```
var pets = new Array(10);
```

- You can put anything in an array
- Any element can be any data type

JOIN()

- Use *array*.`join(`*separator*`)` to convert *array* into string:

```
var pets = ["Dog", "Cat", "Rabbit"];
alert(pets.join(" and "));
// This shows "Dog and Cat and Rabbit"
```

- *separator* is by default `", "`

```
var pets = ["Dog", "Cat", "Rabbit"];
alert(pets.join());
// This shows "Dog,Cat,Rabbit"
```

GETTING SOMETHING

- With this array:

```
var pets = ["Dog", "Cat", "Rabbit"];
```

- You can retrieve something like this:

```
alert(pets[2]); // This shows "Rabbit"
```

CHANGING SOMETHING

- With this array:

```
var pets = ["Dog", "Cat", "Hamster"];
```

- You can change something stored in the array like this:

```
pets[2] = "Rabbit";
// Now pets is ["Dog", "Cat", "Rabbit"]
```

ARRAY SIZE

- You can know the size of an array (i.e. how many boxes it has) using *array*`.length`:

```
var pets = ["Dog", "Cat", "Rabbit"];
alert(pets.length); // This shows 3
```

ADDING TO THE END

- Add a new element to the end of an array with *array*.push( ):

```
var pets = ["Dog", "Cat", "Rabbit"];
pets.push("Hamster");
// Now pets is
// ["Dog", "Cat", "Rabbit", "Hamster"]
```

- The *index* are automatically updated

ADDING TO THE FRONT

- Add a new element to the front with *array*.`unshift()`:

```
var pets = ["Dog", "Cat", "Rabbit"];
pets.unshift("Hamster");
// Now pets is
// ["Hamster", "Dog", "Cat", "Rabbit"]
```

- The *index* are automatically updated

REMOVING FROM THE BACK

- To remove an element from the end, use *array*.`pop()`:

```
var pets = ["Dog", "Cat", "Rabbit"];
var result = pets.pop();
// Now pets is [Dog", "Cat"]
```

- `pop()` returns the removed element, so *result* is `"Rabbit"`

REMOVING FROM THE FRONT

- *array*.`shift()` removes an element from the front:

```
var pets = ["Dog", "Cat", "Rabbit"];
var result = pets.shift();
// Now pets is ["Cat", "Rabbit"]
```

- `shift()` returns the removed element, so *result* is `"Dog"`
- The *index* are automatically updated

COMBINING TWO ARRAYS

- Use *array1*.`concat(`*array2*`)` to combine two arrays into one:

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var primes = [2, 3, 5, 7, 11];
var result = pets.concat(primes);
// result is ["Dog", "Cat", "Rabbit", "Hamster",
//            2, 3, 5, 7, 11]
```

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

GENERATING RANDOM NUMBERS

PROF. DAVID ROSSITER

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to generate and manipulate random numbers

WE WILL LOOK AT

```
Math.random()
```

```
Math.floor()
```

OVERVIEW

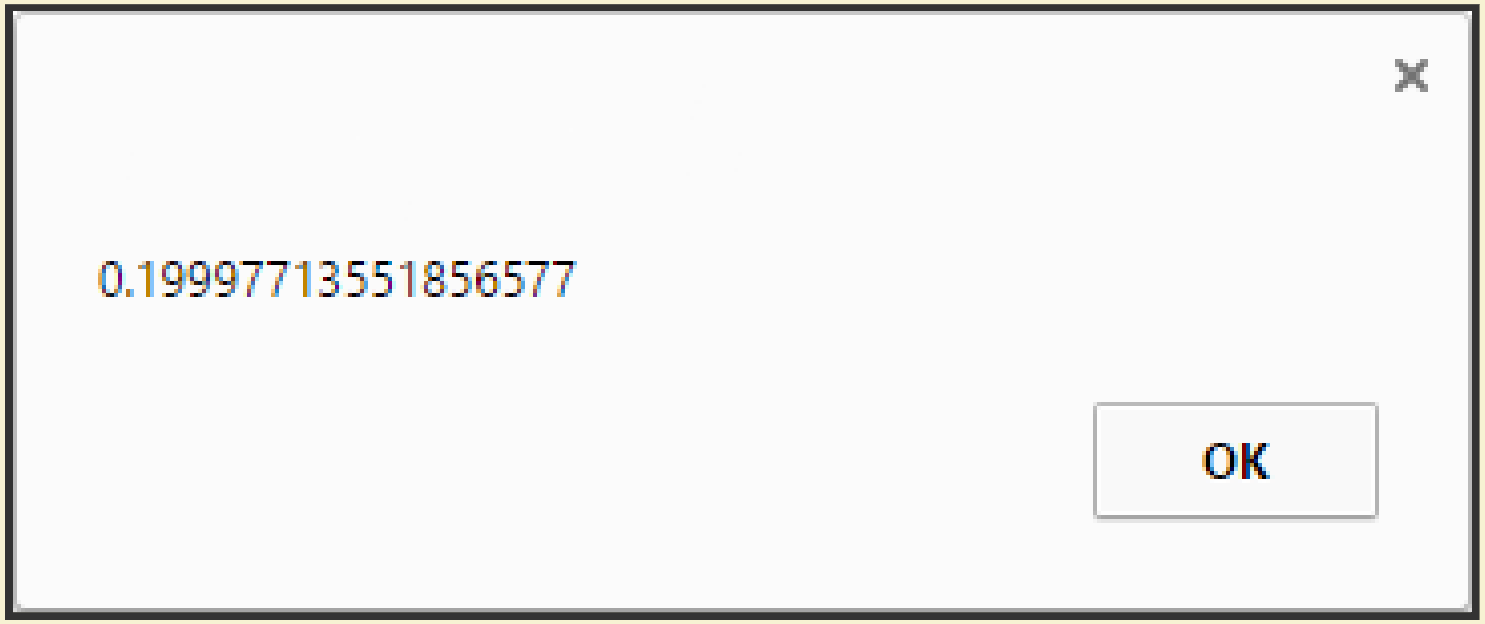
- Generate a random number
- Set up the range
- Throw away the decimal place

GENERATING A RANDOM NUMBER

- You can generate a random number like this:

```
var random_number = Math.random();
```

- The resulting range is [0, 1)
- 1 will not be generated



Click here to open the example

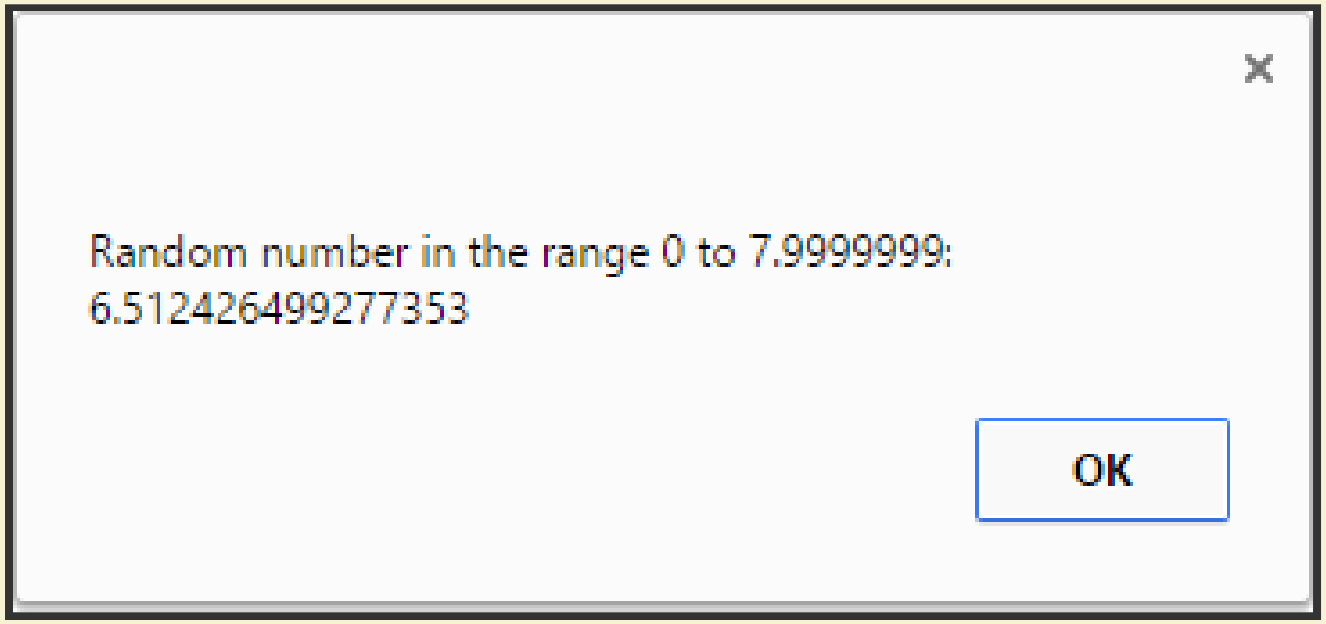
```html
<!doctype html>
<html>
    <body>
        <script>
            var random_number;
            random_number = Math.random();
            alert( random_number );
        </script>
    </body>
</html>
```

SETTING UP THE RANGE

- So far the random number is in the range 0 up to 1
- Multiply in order to get the range you want, i.e.

```
random_number = Math.random() * max_value;
```

- We now have a number in the range [0, max_value)

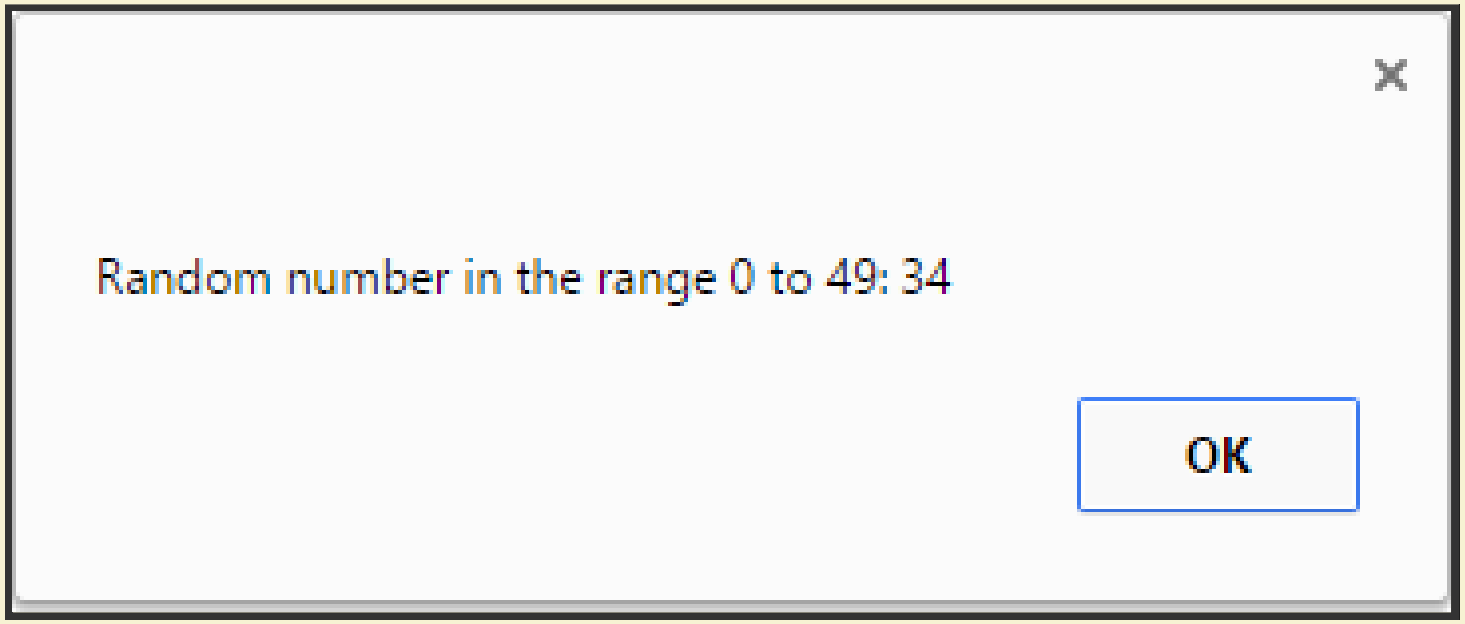


Click here to open the example

```html
<!doctype html>
<html>
  <body>
    <script>
      var random_number;
      random_number = Math.random() * 8;
      alert("Random number in the range 0 to " +
            "7.9999999:\n" + random_number );
    </script>
  </body>
</html>
```

THROW AWAY THE DECIMAL PLACE

- There is still a decimal place
- `Math.floor()` dumps the decimal place
- For example, 2.82248 becomes 2



Click here to open the example

```html
<!doctype html>
<html>
  <body>
    <script>
      var random_number;
      random_number = Math.random() * 50;
      random_number = Math.floor( random_number );
      alert("Random number in the range 0 to 49: " +
                                  random_number);
    </script>
  </body>
</html>
```

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

AN EXAMPLE JAVASCRIPT PROJECT

PROF. DAVID ROSSITER

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

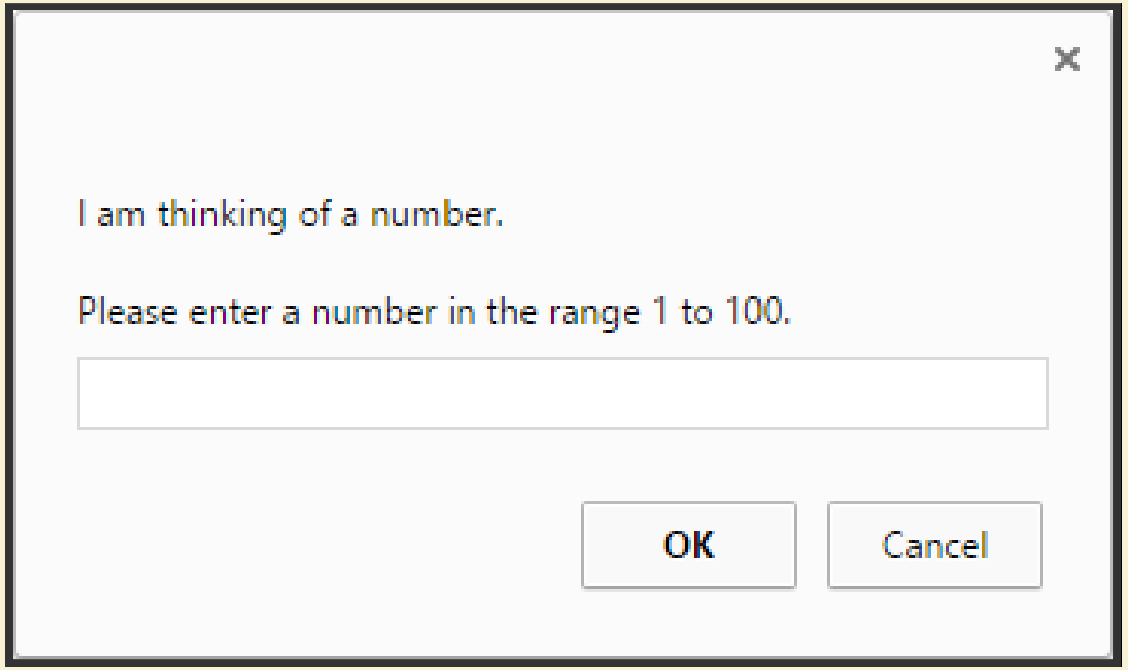
- You'll have stronger JavaScript skills!

THIS PROJECT USES

function	while	alert()	Math.random()
return	if	prompt()	Math.floor()
onload()			parseInt()
			isNaN()

STRENGTHENING YOUR UNDERSTANDING

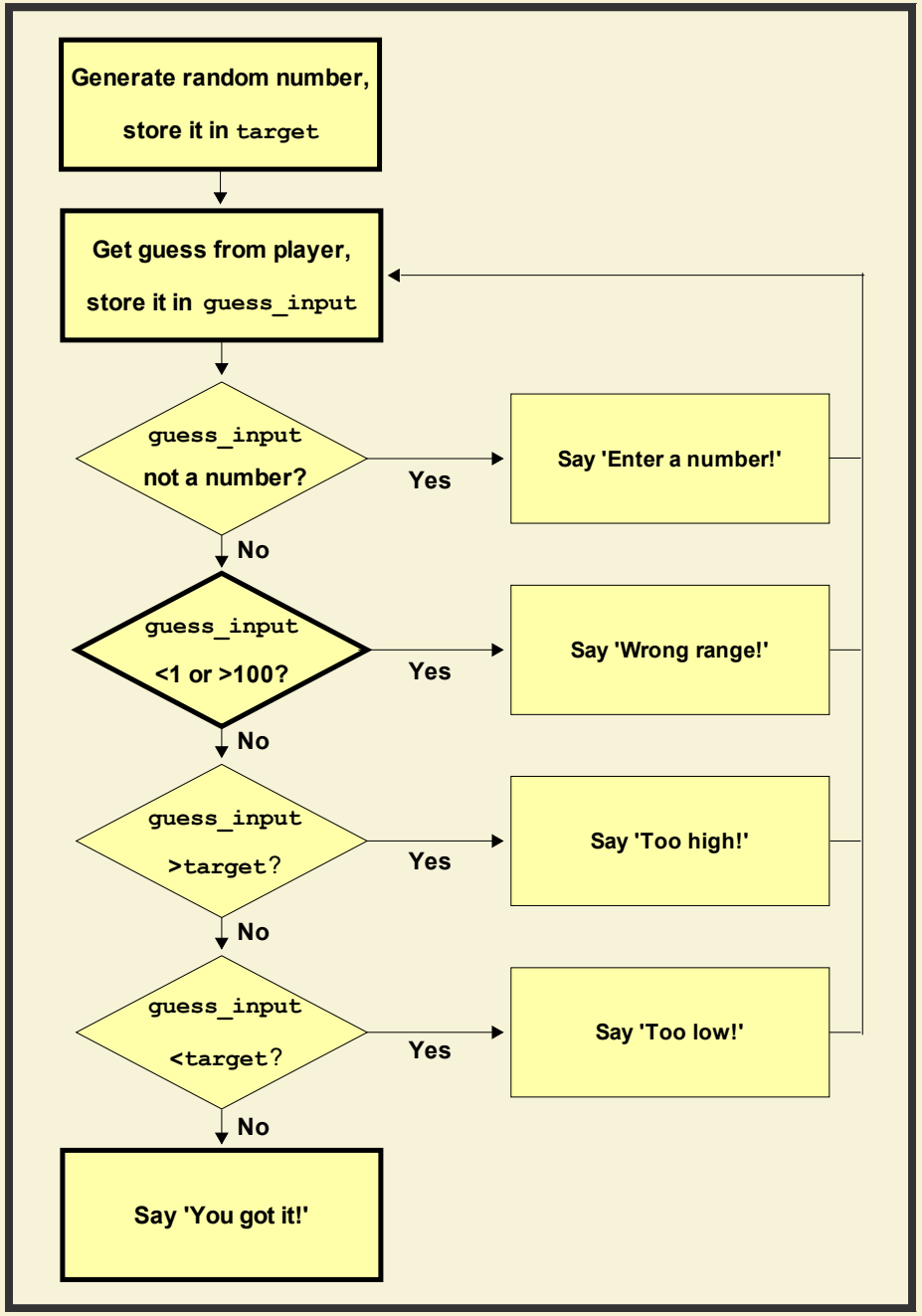
- Let's use some of the techniques you have learned
- We will make a simple guessing game



Click here to play the game

HOW IT WORKS

- The computer thinks of a number in the range [1, 100]
- The player has to guess what it is
- The computer tells the player if
  if answer is right or wrong
- When the game is over, the player is told how many times they guessed


Generate random number, store it in target
Get guess from player, store it in guess_input
guess_input not a number?
Yes
Say 'Enter a number!'
No
guess_input <1 or >100?
Yes
Say 'Wrong range!'
No
guess_input >target?
Yes
Say 'Too high!'
No
guess_input <target?
Yes
Say 'Too low!'
No
Say 'You got it!'

HTML PART

```
<html>
<head>
    <title>JavaScript Guessing Game</title>
</head>
    <body onload="do_game()">
        <script src="js_guessing_game.js">
        </script>
    </body>
</html>
```

- The main function is triggered when the web page is loaded:

```
<body onload="do_game()">
```

- The actual code is stored in another file:

```
<script src="js_guessing_game.js">
</script>
```

JAVASCRIPT COMPONENTS

- 1. The global variables
- 2. The main game function - `do_game()`
  - 2.1. Generate a random number in the range [1,100]
  - 2.2. A `while` loop
- 3. Check the input function - check_guess()
  - To check whether the player's guess is:
    3.1. not a number, 3.2. out of range, 3.3. too large,
    3.4. too small, or 3.5. correct
  - 3.5. Give feedback to the user

1. THE GLOBAL VARIABLES

```
var target;
var guess_input_text;
var guess_input;
var finished = false;
var guesses = 0;
```

2. MAIN GAME FUNCTION

- 2.1. Generate a random number in the range 1 to 100

```
var random_number = Math.random() * 100;
var random_number_integer = Math.floor(random_number);
target = random_number_integer + 1;
```

- 2.2. Use a `while` loop

```
while (!finished) {
```

. . . code goes here . . .

```
};
```

2.2. INSIDE THE WHILE LOOP

1. Get the player's input

```
guess_input_text = prompt("Please enter a number " +
                          "in the range 1 to 100.");
```

2. Convert the input to an integer

```
guess_input = parseInt(guess_input_text);
```

3. Increment the number of guesses

```
guesses += 1
```

4. Check the player's answer

```
finished = check_guess();
```

3. CHECK_GUESS()

- Checks whether the player's guess is:
  - 3.1. Not a number
  - 3.2. Out of range
  - 3.3. Too large
  - 3.4. Too small
  - 3.5. Correct

ISNAN() FUNCTION

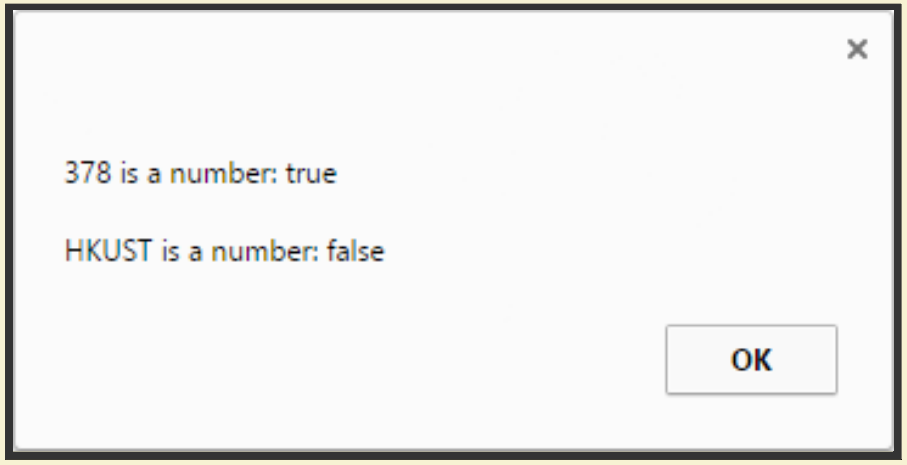
- Returns true if the input parameter is NOT a number and vice versa
- We will make use of this function to check whether the player has entered a number

ISNAN() EXAMPLE

```
<html>
<head>
    <title>isNaN() Example</title>
</head>
<body><script>
    alert("378 is a number: " + !isNaN(378) +"\n\n" +
          "HKUST is a number: " + !isNaN("HKUST"));
</script></body>
</html>
```

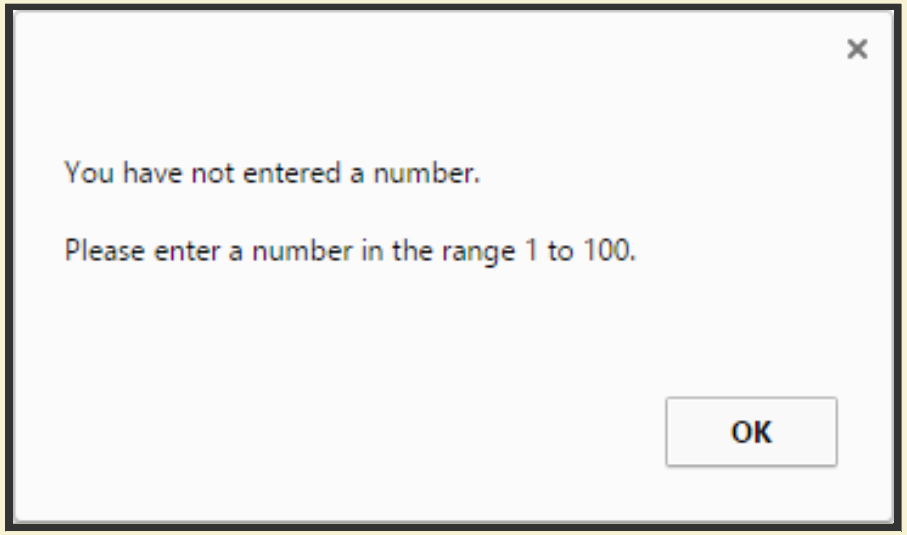
Click here to see the example

ISNAN() EXAMPLE



IF THE PLAYER'S GUESS IS :
3.1. NOT A NUMBER

```
if (isNaN(guess_input)) {
    alert("You have not entered a number.\n\n" +
          "Please enter a number in the range 1 to 100.");
    return false;
}
```

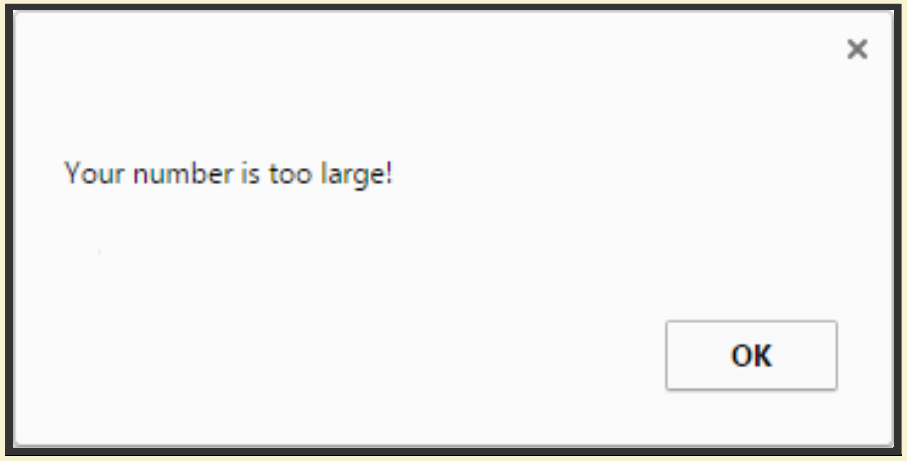


3.2. OUT OF RANGE

```
if ((guess_input < 1) || (guess_input > 100)) {
    alert("Please enter an integer number" +
          "in the range 1 to 100.");
    return false;
}
```

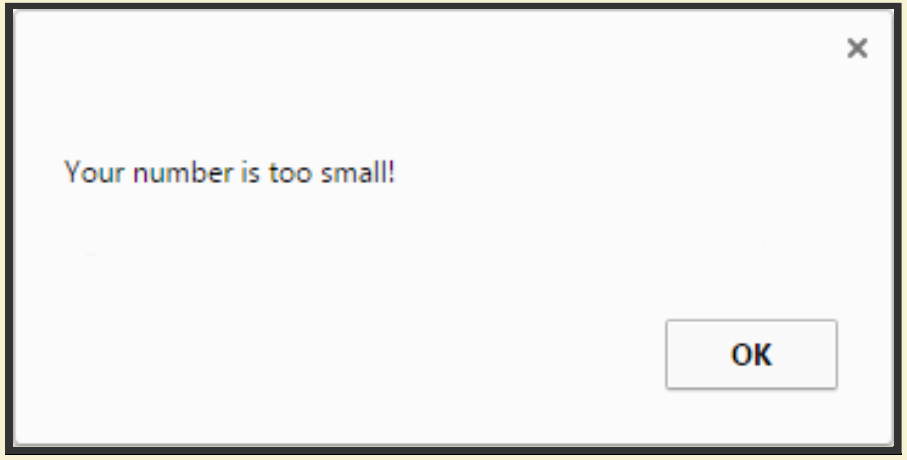
3.3. TOO LARGE

```
if (guess_input > target) {
    alert("Your number is too large!");
    return false;
}
```



3.4. TOO SMALL

```
if (guess_input < target) {
    alert("Your number is too small!");
    return false;
}
```



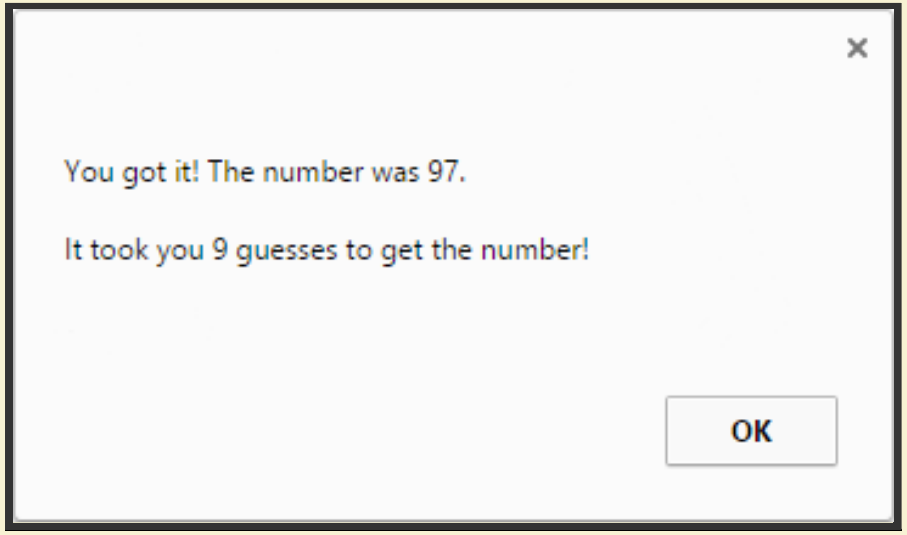
3.5. CORRECT

1. Congratulate the player and show the number of guesses

```
alert("You got it! The number was " + target +
      ". \n It took you " + guesses +
      "guesses to get the number!");
```

2. Return a `true` value to the main function

```
return true;
```


You got it! The number was 97.
It took you 9 guesses to get the number!
OK

```
var target;
var guess_input_text;
var guess_input;
var finished = false;
var guesses = 0;

function do_game(){
    var random_number = Math.random() * 100;
    var random_number_integer = Math.floor(random_number);
    target = random_number_integer + 1;

    while (!finished) {
        guess_input_text = prompt("I am thinking of a number "+
                                  "in the range 1 to 100.\n\n"+
                                  "What is the number? ");
        guess_input = parseInt(guess_input_text);
        guesses += 1;
        finished = check_guess();
    }
}
```

```
function check_guess() {
    if (isNaN(guess_input)) {
        alert("You have not entered a number.\n\n" +
              "Please enter a number in the range 1 to 100.");
        return false;
    }
    if ((guess_input < 1) || (guess_input > 100)) {
        alert("Please enter an integer number in the range 1 to 100.");
        return false;
    }
    if (guess_input > target) {
        alert("Your number is too large!");
        return false;
    }
    if (guess_input < target) {
        alert("Your number is too small!");
        return false;
    }
    alert("You got it! The number was " + target +
          ".\n\nIt took you " + guesses +
          " guesses to get the number!");
    return true;
}
```