JavaScript

JavaScript

- Like PHP, JavaScript is a modern programming language that is derived from the syntax at C.
- It has been around just about as long as PHP, also having been invented in 1995.
- JavaScript, HTML, and CSS make up the three languages defining most of the user experience on the web.

JavaScript

- To start writing JavaScript, open up a file with the .js file extension.

- No need for any code delimiters like we had in PHP. Our website will know that our file is JavaScript because we'll explicitly tell it as much in an HTML tag.

- Unlike PHP which runs *server-side*, JavaScript applications run *client-side*, on your own machine.

JavaScript

- **Including JavaScript in your HTML**

- Just like CSS with `<style>` tags, you can directly write your JavaScript between `<script>` tags.

- Just like CSS with `<link>` tags, you can write your JavaScript in separate files and link them in by using the src attribute of the `<script>` tag.

JavaScript

- **Including JavaScript in your HTML**

- Just like CSS with `<style>` tags, you can directly write your JavaScript between `<script>` tags.

- Just like CSS with `<link>` tags, you can write your JavaScript in separate files and link them in by using the src attribute of the `<script>` tag.

JavaScript

- **Variables**

- JavaScript variables are similar to PHP variables.
  - No type specifier.
  - When a local variable is first declared, preface with the `var` keyword.

JavaScript

- **Variables**

- JavaScript variables are similar to PHP variables.
  - No type specifier.
  - When a local variable is first declared, preface with the `var` keyword.

```
$x = 44;
```

JavaScript

- **Variables**

- JavaScript variables are similar to PHP variables.
  - No type specifier.
  - When a local variable is first declared, preface with the `var` keyword.

~~`$x = 44;`~~

JavaScript

- **Variables**

- JavaScript variables are similar to PHP variables.
  - No type specifier.
  - When a local variable is first declared, preface with the `var` keyword.

```
var x = 44;
```

JavaScript

- **Conditionals**

- All of the old favorites from C are still available for you to use.

```
if
```

JavaScript

- **Conditionals**

- All of the old favorites from C are still available for you to use.

```
if
else if
```

JavaScript

- **Conditionals**

- All of the old favorites from C are still available for you to use.

```
if
else if
else
```

JavaScript

- **Conditionals**

- All of the old favorites from C are still available for you to use.

```
if
else if
else
switch
```

JavaScript

- **Conditionals**

- All of the old favorites from C are still available for you to use.

```
if
else if
else
switch
?:
```

JavaScript

- **Loops**

- All of the old favorites from C are still available for you to use.

JavaScript

- **Loops**

- All of the old favorites from C are still available for you to use.

```
while
```

JavaScript

- **Loops**

- All of the old favorites from C are still available for you to use.

```
while
do-while
```

JavaScript

- **Loops**

- All of the old favorites from C are still available for you to use.

`while`
`do-while`
`for`

JavaScript

- **Functions**

- All functions are introduced with the `function` keyword.

- JavaScript functions, particularly those bound specifically to HTML elements, can be *anonymous*—you don't have to give them a name!
  - We'll revisit anonymity a little later, and we'll revisit "binding to HTML elements" in the video on the Document Object Model.

JavaScript

- **Arrays**

- Declaring an array is pretty straightforward.

JavaScript

- **Arrays**

- Declaring an array is pretty straightforward.

```
var nums = [1, 2, 3, 4, 5];
```

JavaScript

- **Arrays**

- Declaring an array is pretty straightforward.

```
var nums = [1, 2, 3, 4, 5];
var mixed = [1,
             true,
             3.333,
             'five'];
```

JavaScript

- **Objects**

- JavaScript has the ability to behave as an *object-oriented* programming language.

- An object is sort of analogous to a C structure.

JavaScript

- **Objects**

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

JavaScript

- **Objects**

```
struct car
{
    int year;
    char model[10];
}
```

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

JavaScript

- **Objects**

```
struct car
{
    int year;
    char model[10];
}
```

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

```
struct car herbie;
```

JavaScript

- **Objects**

```
struct car
{
    int year;
    char model[10];
}
```

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

```
struct car herbie;
herbie.year = 1963;
herbie.model = "Beetle";
```

JavaScript

- **Objects**

```
struct car
{
    int year;
    char model[10];
}
```

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

```
struct car herbie;
year = 1963;
model = "Beetle";
```

JavaScript

- **Objects**

```
struct car
{
    int year;
    char model[10];
}
```

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

```
struct car herbie;
year = 1963;
model = "Beetle";
```

JavaScript

- **Objects**

- C structures contain a number of *fields*, which we might also call *properties*.
  - But the properties themselves can not ever stand on their own.

- Objects, meanwhile, have properties but also *methods*, or functions that are inherent to the object, and mean nothing outside of it.
  - Thus, like properties can methods not ever stand on their own.

JavaScript

- **Objects**

```
function(object);
```

JavaScript

- **Objects**

~~`function(object);`~~

JavaScript

- **Objects**

```
object.function();
```

JavaScript

- **Objects**

- The fields and methods of an object are similar in spirit to the idea of an associative array, with which we're familiar from PHP.

JavaScript

- **Objects**

- The fields and methods of an object are similar in spirit to the idea of an associative array, with which we're familiar from PHP.

```
var herbie = {year : 1963, model: 'Beetle'};
```

JavaScript

- **Loops (redux)**

- How do we iterate across all of the key-value pairs of an object (or indeed, all of the elements of an array)?

JavaScript

- **Loops (redux)**

- How do we iterate across all of the key-value pairs of an object (or indeed, all of the elements of an array)?

```
foreach($array as $key)
{
   // use $key in here as a stand-in for $array[$i]
}
```

JavaScript

- **Loops (redux)**

- How do we iterate across all of the key-value pairs of an object (or indeed, all of the elements of an array)?

~~~~
foreach($array as $key)
{
    // use $key in here as a stand-in for $array[$i]
}
~~~~

JavaScript

- **Loops (redux)**

- How do we iterate across all of the key-value pairs of an object (or indeed, all of the elements of an array)?

```
for (var key in object)
{
   // use object[key] in here
}
```

JavaScript

- **Loops (redux)**

- How do we iterate across all of the key-value pairs of an object (or indeed, all of the elements of an array)?

```
for (var key of object)
{
   // use key in here
}
```

JavaScript

- **Loops (redux)**

```
var wkArray = ['Monday,
               'Tuesday',
               'Wednesday',
               'Thursday',
               'Friday',
               'Saturday',
               'Sunday'];
```

JavaScript

- **Loops (redux)**

```
var wkArray = ['Monday,
               'Tuesday',
               'Wednesday',
               'Thursday',
               'Friday',
               'Saturday',
               'Sunday'];
```

```
for (var day in wkArray)
{
   console.log(day);
}
```

JavaScript

- **Loops (redux)**

```
var wkArray = ['Monday,
               'Tuesday',
               'Wednesday',
               'Thursday',
               'Friday',
               'Saturday',
               'Sunday'];
```

```
for (var day of wkArray)
{
   console.log(day);
}
```

JavaScript

- **Printing and variable interpolation**

```
console.log(wkArray[day] . ' is day number '
                . (day + 1) . ' of the week!');
```

JavaScript

- **Printing and variable interpolation**

```
console.log(wkArray[day] + ' is day number '
            + (day + 1) + ' of the week!');
```

JavaScript

- **Printing and variable interpolation**

```
console.log(wkArray[day] + ' is day number '
                + (day + 1) + ' of the week!');
```

JavaScript

- **Printing and variable interpolation**

```
console.log(wkArray[day] + ' is day number '
            + (parseInt(day) + 1) +
            ' of the week!');
```

JavaScript

- **Functions (redux)**

- Arrays are a special case of an object (in fact, *everything* in JavaScript is a special case of an object), and has numerous methods that can applied to them:
  - `array.size(), array.pop(), array.push(x), array.shift();`

- There is also a method for arrays called `map()`, which can be used to apply a function to all elements of an array.
  - A great situation to use an *anonymous function*

JavaScript

- **Functions (redux)**

```
var nums = [1, 2, 3, 4, 5];
```

JavaScript

- **Functions (redux)**

```
var nums = [1, 2, 3, 4, 5];
```

```
nums = nums.map(function(num) {
    return num * 2;
});
```

JavaScript

- **Functions (redux)**

```
var nums = [1, 2, 3, 4, 5];
```

```
nums = nums.map(function(num) {
    return num * 2;
});
```

JavaScript

- **Functions (redux)**

```
var nums = [2, 4, 6, 8, 10];
```

```
nums = nums.map(function(num) {
    return num * 2;
});
```

JavaScript

- **Events**

- An *event* in HTML and JavaScript is a response to user interaction with the web page.
  - A user clicks a button, a page has finished loading, a user has hovered over a portion of the page, the user typed in an input field.

- JavaScript has support for *event handlers*, which are callback functions that respond to HTML events.
  - Many HTML elements have support for events as an attribute.

JavaScript

```
<html>
    <head>
        <title>Event Handlers</title>
    </head>
    <body>
        <button onclick="">Button 1</button>
        <button onclick="">Button 2</button>
    </body>
</html>
```

JavaScript

```
<html>
    <head>
        <title>Event Handlers</title>
    </head>
    <body>
        <button onclick="">Button 1</button>
        <button onclick="">Button 2</button>
    </body>
</html>
```

JavaScript

- **Events**

- We can write a generic event handler in JavaScript, creating an *event object*, that will tell us which of these two buttons was clicked.

JavaScript

```
<html>
	<head>
		<title>Event Handlers</title>
	</head>
	<body>
		<button onclick="alertName(event)">Button 1</button>
		<button onclick="alertName(event)">Button 2</button>
	</body>
</html>
```

JavaScript

```
function alertName(event)
{
    var trigger = event.srcElement;
    alert('You clicked on ' + trigger.innerHTML);
}
```

JavaScript

```
function alertName(event)
{
    var trigger = event.srcElement;
    alert('You clicked on ' + trigger.innerHTML);
}
```

JavaScript

```
function alertName(event)
{
    var trigger = event.srcElement;
    alert('You clicked on ' + trigger.innerHTML);
}
```