Whitespace is ...

... not significant.

What is the comment syntax?

What's the catch?

As in Java, in-line `//` and multiline `/* ... */`.

Unfortunately, the multiline delimiters can appear in string literals, making commenting out arbitrary blocks of codes unreliable.

What's strange about JavaScript's numerical types?

There are no integers, or floating point of any size other than 64-bits.
There are no characters either.

Create an exponential literal.

`xey` is the same as `x * pow(10, e)`.

Describe the string syntax.

How are they encoded?

As in Python, single or double quote delimiters can be used.

16-bit fixed Unicode is used.

Access the length of a string.

Concatenate two strings.

'string'.length

Use +.

What does the `<script>` tag do?

How do they relate to one another?

The browser executes the compilation unit referred to in the tag right away.

They share a global namespace.

What does `var` do?

It indicates variables private to a function.

Which statements can be labeled?

`switch`, `while`, `for`, `do`

What's different about JavaScript's blocks?

They don't define a new scope.

Give the syntax of both kinds of conditionals.

```
if (expr) then {
   ...
}
else {
   ...
}
```

To get a value:

```
if expr ? true-expr : false-expr
```

Conditionals accept what kind of values as "false"?

`false`, `null`, `undefined`, empty string, `0`, `NaN`

Give the `do-while` syntax.

```
do {
   ...
}
while (expr);
```

Give the `switch` syntax.

What's the classic `switch` mistake?

Same as Java.

```
switch (expr) {
  case expr: ...
  default: ...
}
```

It falls through.

Give the `while`/`do-while` syntax.

Identical to Java.

What are `for`'s two forms.

The first is as in C++/Java.

The second is a for-each:

```
for (x in y) {
   ...
}
```

What's the catch with for-each loops?

You can't know if the binding form was found in the iterable object's prototype chain. For that reason you often see the body of a for-each being a single condition of the form:

```
if (iterable.hasOwnProperty(bindingForm)) {
   ...
}
```

What happens if no `return` statement is given in a function.

The function's result is `undefined`.

How do you test for in/equality?

=== and !==

What unary operators are available?

`delete`, `new`, `typeof`, `+`, `-`, `!`

What can `typeof` return?

```
'number','string','boolean','undefined',
'function','object'
```

Create a named function literal.

Create an anonymous function literal.

```
function name(args) {
   ...
}

function(args) {
   ...
}
```

Give the regex literal syntax.

`/regex/`

It cam optionally be trailed by any of `g`, `i`, or `m`.

Give the array literal syntax.

As is common, a comma-separated list inside square brackets.